Human-centric Computing and Information Sciences

August 2025 | Volume 15

KCIA
Korea Computer Industry Association

www.hcisjournai.com

HCIS Human-centric Computing and Information Sciences

RESEARCH (Research Manuscript) Open Access

Human-centric Computing and Information Sciences (2025) 15:45
DOI: https://doi.org/10.22967/HCIS.2025.15.045
Received: April 26, 2024; Accepted: June 20, 2024; Published: August 15, 2025

RobTC: Robust Spatio-temporal Trajectory Classification via Collaborative Learning in IoT

Jia Jia[1], Linghui Li[1,*], Pengfei Qiu[1], Xu Zhang[2], Binsi Cai[1], Ximing Li[1], and Xiaoyong Li[1]

Abstract

With the explosion of crowd mobility data generated by universal mobile devices equipped with spatial positioning modules, deep neural networks (DNNs) have been extensively applied to trajectory data mining and modeling. However, recent studies have shown that DNNs are vulnerable to certain adversarial examples, which are ingeniously crafted by introducing minute and imperceptible perturbations to original examples, but can fool classifiers with high confidence. To enhance the robustness of DNN-based trajectory classification, we propose a novel collaborative learning method for robust spatio-temporal trajectory classification, named RobTC, which consists of an autoencoder-based self-representation network (SRN) for robust latent feature learning and a gated recurrent unit (GRU)-based classification network by sharing parameters with the SRN to safeguard against various adversarial attacks. Furthermore, we introduce feature-level constraints between the original input and the corresponding adversarial examples instead of the point-level denoising strategies to effectively suppress the potential "error amplification effect." Extensive experiments on the Geolife and Beijing taxi traces datasets demonstrate that our method yields significant improvements (white-box 15% and black-box 13%) over the state-of-the-art methods, suggesting that our proposed method can significantly enhance the model's robustness against various adversarial attacks while preserving the model's prediction accuracy on original examples.

Keywords

Collaborative Learning, Gated Recurrent Unit, Spatio-temporal Trajectory, Adversarial Examples

1. Introduction

Driven by the proliferation of the Internet of Things (IoT) [1], an array of mobile devices equipped with spatial positioning modules, including the global positioning system (GPS), have ubiquitously dominated the consumer market, generating and accumulating massive spatio-temporal trajectory data. Given this wealth of trajectory data, numerous studies centered around urban contexts have spearheaded a myriad of practical applications, such as urban management, analyses of epidemic propagation, and traffic control, highlight the crucial significance of trajectory data mining and modeling within the IoT

*Corresponding Author: Linghui Li (lilinghui@bupt.edu.cn)
[1]Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education, Beijing University of Posts & Telecommunications, Beijing, China
[2]School of Electronic and Engineering, Nanjing University, China
Jia Jia and Linghui Li contributed equally to this work.

scenarios through such achievements.

Initially, spatio-temporal trajectory classification mainly relied on experts' prior knowledge and custom-crafted features derived from their insights. For instance, their solutions commonly utilize traditional machine learning algorithms [2], like a decision tree, K-nearest neighbor, and support vector machine (SVM), which involve the selection of trajectory attributes such as velocity, acceleration, and direction to formulate models. With their recent rapid development, deep learning methods have been applied to classification tasks, and demonstrated superior performance on raw data without the need for feature engineering procedures. Since they can be easily applied to trajectories of arbitrary length and are more flexible than traditional methods, increasingly more studies on trajectory classification are shifting towards deep neural network (DNN)-based techniques [3–7], such as convolutional neural network (CNN) and recurrent neural network (RNN) for self-adaptive feature extraction and classification in a data-driven approach.

However, recent studies [8–10] have exposed the DNNs' vulnerability to adversarial examples that are generally imperceptible to humans but can fool classifiers with high confidence. In Fig. 1, some systems like this one engaged in trajectory classification tasks have been successfully compromised by meticulously crafted adversarial examples. Moreover, these adversarial examples possess the ability to transfer between different models, greatly increasing security risks, such as serious traffic accidents [11]. This highlights the necessity to enhance the robustness of DNNs against adversarial examples to enable their effective implementation across diverse scenarios.

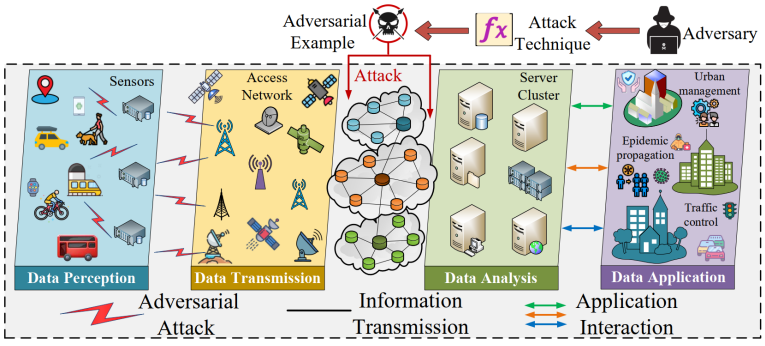


Fig. 1. Process of trajectory modeling under the IoT. Trajectory data may be struck by an adversarial attacks before it enters an analysis model.

Previous attempts [12, 13] to restore the resilience of DNNs, including adversarial training and gradient masking, have primarily focused on adjusting the parameters of the target model to enhance its robustness. While these approaches have achieved a degree of success against adversarial attacks, they are accompanied by limitations, like expensive calculation, vulnerable to black-box attacks, performance penalty on original examples, and insufficient generalization. Considering that an adversarial example is generated by introducing noise to original example, a logical idea is to denoise such an example before feeding it into the target model (Fig. 2). In comparison to adversarial training, this approach is more straightforward and reasonable. Subsequent exploration [14] validates the feasibility of this idea, showing its effectiveness in reducing point-level noise. Nevertheless, none of the denoisers successfully remove all noise in adversarial examples, allowing small residues to be amplified into significant perturbations in the feature-level representation, a phenomenon known as the "error amplification effect," which still persists and leads to incorrect prediction.

To address this issue, we propose a robust spatio-temporal trajectory classification (RobTC) method that imposes the loss function of the denoiser to minimize the difference in feature-level representation of the target model induced by the adversarial and original examples by departing from setting the loss function at the point level (Fig. 2). In particular, we leverage an autoencoder self-representation network

for robust latent feature learning and a gated recurrent unit (GRU)-based classification network, with a collaborative manner to enhance the robustness of the model. In comparison to the point-level denoising method, our method excels in suppressing the impact of adversarial perturbations. Moreover, when compared to adversarial training, which is the current state-of-the-art safeguard method, our method provides the following advantages:

- We propose a novel collaborative learning method to autonomously learn the robust latent feature of spatio-temporal trajectories through an integration of a self-representation network and a classification network for robust trajectory classification to safeguard against both white and black box adversarial attacks.
- Our proposed method effectively suppresses the "error amplification effect" by imposing feature-level constraints within the collaborative learning framework, which significantly enhances the robustness of trajectory classification against various adversarial attacks in a simplified manner.
- Extensive experiments on two publicly available datasets demonstrate that the advantages of our proposed method can resist various adversarial attacks without significant performance penalties on the original examples.

The rest of the article is organized as follows. In Section 2, we offer a brief review of the related work, while in Section 3, we provide a detailed explanation of the overall structure of our proposed method. We then offer an in-depth presentation of the empirical insights gathered through our experiments in Section 4, and finally, Section 5 summarizes our findings and articulates their implications for future work.

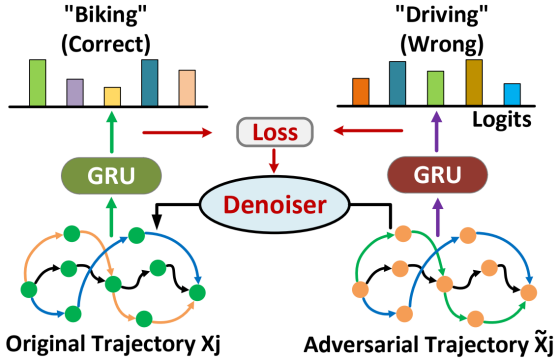


Fig. 2. Idea of feature-level representation guided denoising. Although the initial difference between the original and adversarial trajectory is small, it undergoes amplification in the feature-level representation of the GRU. Consequently, we have employed the distance in feature-level representation to guide the training procedure of the trajectory denoiser, aiming to suppress the impact of perturbations.

2. Related Work

We provide a brief overview encompassing spatio-temporal trajectory classification, adversarial attack techniques, and defensive strategies, by organizing the explanation into three main parts.

2.1 Spatio-temporal Trajectory Classification

Existing spatio-temporal trajectory classification methods can be categorized into two groups, namely traditional machine learning and deep learning approaches. Traditional machine learning approaches typically require manual formulation of decision rules and reasoning mechanisms to solve classification tasks. Lee et al. [15] first developed a hierarchical feature structure by combining region-based and trajectory-based clustering techniques, with a primary focus on spatial dimensions. Patel [16] combined

the spatial distribution, duration and regional association information of trajectory to generate trajectory features and improve classification accuracy. Dodge et al. [17] extracted attributes such as velocity, acceleration, rotation angle, displacement, and deviation rate of change from the trajectories, to optimize the estimation models based on them. Zheng et al. [18] proposed a segmentation method based on change points to better complete the trajectory semantic segmentation and a graph-based post-processing algorithm to further improve the classification performance. Biljecki et al. [9] proposed a new fully automatic trajectory segmentation method that better deals with signal deficiency and noise effects in raw data. Xiao et al. [19] propose a method based on ensemble learning to extract local features from sub-trajectories and then combine these features. Saini et al. [20] proposed a trajectory classification method combining fuzzy C-means clustering SVM, resulting in improved accuracy compared with the traditional classification method. However, these methods heavily rely on human expert knowledge and manually customized features, making it difficult to process large amounts of trajectory data.

To address certain challenges of manually extracting features from massive datasets, recent research has increasingly turned to deep learning methods. Dabiri and Heaslip [21] employed CNNs to predict the travel modes of raw trajectories, in order to address the vulnerability of manual features to traffic and environmental conditions, as well as human bias in feature creation. Zeng et al. [22] proposed a seq2seq model consisting of a convolutional encoder and a cyclic conditional random field to output an accurate and reasonable travel mode label. Gao et al. [23] used RNN to capture the spatio-temporal semantics of user movement patterns to solve the trajectory-user linking (TUL) problem. Zhou et al. [24] proposed TULVAE model to represent the hierarchical and structural semantics of trajectories through high-dimensional latent variables and to mitigate the problem of data sparsity by utilizing large amounts of unlabeled data. Liu et al. [25] introduced a new piecewise convolution mechanism for spatio-temporal GRU that better considers spatial and temporal interval information. Liang et al. [26] further advanced the RNN by transforming it into a continuous-time model, in which the continuous state adheres to ordinary differential equations between successive points. Liu et al. [27] introduced a parallel model that combines a statistics-based approach with a spatio-temporal relations based approach to obtain a more robust representation. Kim et al. [28] integrate the CNN model with the LSTM network into an overall structure to better extract sequential features from the trajectories. Yao et al. [29] embedded spatio-temporal and pattern dimension features together to provide more comprehensive information for pattern recognition. Wen et al. [30] proposed social ordinary differential equation (ODE), incorporating temporal agent dynamics and agent interactions into modeling. By applying a pre-processing location sequence, Park et al. [31] used a causal location embedding model to capture the time dependence of the location. In practical applications, deep learning models have demonstrated excellence in capturing intricate spatio-temporal semantic relationships, resulting in steadily improved their accuracy over iterations of versions. Nevertheless, a common trend in these methods has been their emphasis on optimizing model accuracy while overlooking the robustness. While recent study [32] has shown that deep learning models are vulnerable to adversarial attacks in contrast to existing methods, our proposed method mainly emphasizes the approach to enhance the robustness of trajectory classification while preserving the accuracy.

2.2 Adversarial Attack Techniques

Adversarial examples X^* essentially means that adding a small perturbation ϵ to the given input X, the output $y_{X^*} \neq y_X$ holds true. In practice, an array of techniques has emerged for defending against adversarial examples, and can be broadly categorized as optimization-based and gradient-based methods. Optimization-based methods approach the generation of adversarial examples as an optimization problem and employ optimizers such as box-constrained L-BFGS [33] or Adam for solving it, which are powerful but quite slow. In contrast, Kurakin et al. [34] first proposed the gradient based method called fast gradient sign method (FGSM) that efficiently finds adversarial examples by summing the attack strength across every feature dimension, which is significantly more efficient than L-BFGS by only computing gradients once. Kurakin et al. [35] further refined FGSM by compressing the perturbation strength ϵ and introducing

multiple iterations FGSM (I-FGSM) with a small step size α to achieve better performance than FGSM. Dong et al. [36] took a different approach with the momentum iterative FGSM (MI-FGSM) by integrating a momentum term into the generation process of adversarial examples, resulting in more severe damage to the target. It's bears mention that while iterative methods like I-FGSM and MI-FGSM are formidable as white-box adversaries, they tend to suffer from decreased transferability[37], making them less effective in black-box attack scenarios.

2.3 Defensive Strategies

2.3.1 Defensive techniques via network modification

The existing defensive strategies are mainly divided into two categories, which we have summarized and denoted in Table 1 [14, 38–40, 42–50]. This defensive strategy-based network modification is geared towards enhancing the resilience of the target model against adversarial examples, and the most extensively investigated technique is adversarial training [38], which aims to augment the training data by injecting adversarial examples into the training process. While these methods can indeed learn how to enhance the model's capacity in order to handle hybrid examples and recover its resilience against adversarial attacks, they exhibit poor generalizability to unknown attacks. Several other methods revolve around the concept of gradient masking [39], where specific regularized or smoothing labels are incorporated during the training procedure to reduce the target model's sensitivity to input perturbations, such as feature compression [40], network distillation [50], region-based classifiers [42], and saturated networks [43]. However, they are still vulnerable to black-box attacks, even leading to a performance penalty on original examples.

Table 1. Comparison of different defensive strategies

Model	Generalization	White-box attack	Black-box attack	Computation	Performance penalty
Adversarial training [38]	×	√	×	×	×
Gradient masking [39]	×	√	×	×	×
Feature compression [40]	×	×	√	×	×
Network distillation [50]	√	×	√	√	×
Region-based [42]	×	√	×	√	×
Saturated networks [43]	√	×	√	√	√
MaungMuang et al. [44]	√	√	×	√	×
Shah et al. [45]	√	√	×	√	×
Song et al. [46]	√	×	√	×	×
Zari et al. [47]	√	√	×	×	×
Zhao et al. [48]	√	√	×	√	×
Han et al. [49]	√	×	√	√	×
Liao et al. [14]	√	√	×	√	√
This work	√	√	√	√	√

"√" means that the factor is taken into account and "×" means not taking this factor into account.

2.3.2 Defensive techniques via input transformation

Input transformation defenses seek to eliminate adversarial perturbations by transforming inputs before they are being fed into the target network. Certain prior methods treat adversarial perturbations as high-frequency noise and consequently employ traditional denoising techniques to mitigate these minor perturbations. For instance, the study [44] examined the impact of data compression on eliminating adversarial noise. Alternatively, Shah et al. [45] have applied a range of filters including the median filter and averaging filter to mitigate perturbations. Song et al. [46] experimented with five different trans-

formations and found that total variation minimization and data quilting both demonstrate effective defensive performance. However, these denoising methods primarily address small perturbations and are susceptible to information loss.

More recently, there have been efforts to rectify adversarial examples by utilizing generative models. Among these efforts, Zari et al. [47] employed dimensionality reduction as a defensive strategy, such as principal component analysis (PCA). Zhao et al. [48] employed a denoising auto-encoder to eliminate adversarial perturbations in MNIST digits. Similarly, Han et al. [49] utilized PixelCNNs to transform adversarial examples into clean ones. While these methods demonstrate strong performance on small datasets, they encounter scalability issues when applied to higher-resolution or larger datasets. Last but not least, Liao et al. [14] employed a high-level denoising method to reduce noise, with a certain level of effectiveness being achieved. However, they encountered issues such as “error amplification effect” and vulnerability to adversarial examples. We suspect that the root of these problems is limited to the point level, and ignore the feature level, which directly determines the final result.

3. Collaborative Learning Framework

We introduce a collaborative learning framework, named RobTC, which aims to address the inherent limitations of DNN-based trajectory classification methods when dealing with adversarial examples, so that the robustness of trajectory classification methods is effectively enhanced. We then detail our proposed method called RobTC, including the motivation, over the entire framework, self-representation network, classification network, and loss function in this section.

3.1 Motivation

We know that adversarial training is a delicate balancing strategy, as it involves coupling two processes: one for classifying the original examples and the other for defending against adversarial examples. This coupling, while intensifying computational demands, can also result in unforeseen alterations to the target model, ultimately causing unwarranted performance penalty on original examples. Therefore, we consider decoupling these two processes, which entails denoising adversarial examples before they enter into the target model. Previous work [14] has shown that adversarial noise can indeed be removed through point-level denoising. However, this method cannot completely remove all noise in adversarial examples and residual noise can be amplified in high-level hidden representations, known as the “error amplification effect,” which still results in incorrect predictions. To address this issue, we abandon the constraint at the point level, but at the feature level. Specifically, we achieve this by joining a feature-level loss function into the denoising process that involves minimizing the difference in high-level representations between the original and perturbed examples. Recognizing the necessity for an effective learning mechanism to express the robust inherent features of the original example, we employ an autoencoder to achieve this. Simultaneously, we utilize GRU to capture sequential information present in these temporal data.

3.2 Overall Architecture

As demonstrated in Fig. 3, the proposed model comprises two components, namely the upper SRN and bottom CN. The role of SRN corresponds to that of a “teacher” guiding the training procedure of CN, so that the SRN needs to possess the capability of extracting the robust inherent feature from the original trajectory. The SRN comprises an encoder and a decoder, both seamlessly integrated with GRU. Conversely, the CN comprises an encoder and a target network stacked in sequence. The encoder acts like a “student,” learning robust features from the SRN and feeding them to the target network, effectively suppressing the “error amplification effect.”

Our task involves modeling transportation modes from a spatio-temporal trajectory, which is essentially a trajectory classification problem. A set of trajectory segments X is defined as $\{X_1, \dots, X_N\}$, where N is the number of trajectory segments. A trajectory segment $X_j = \{x_1, x_2, x_3, \dots, x_n\}$ consists of n trajectory points. A trajectory point x_i is defined as a tuple $\{l_i, d_i, t_i\}$, where $l_i, d_i \in \mathbb{R}^2$, $t_i \in \mathbb{R}^+$ for $i \in (1, n)$, the l_i represents the longitude, d_i represents the latitude, and t_i represents the time stamp. Finally, we estimate the probability distribution $P(Y_j = m \mid X_j)$ that belongs to the transportation mode m based on the input trajectory segment X_j.

Throughout the training stage, we feed an original trajectory segment X_j and an adversarial trajectory segment X_j^* derived from X_j into the SRN and CN, respectively, so that they are encoded into the high-level representations. Note that these two components are trained simultaneously in a collaborative manner by sharing parameters with each other. This collaborative approach ensures the transmission of robust features by using the high-level representation learned from original examples to guide the high-level representation obtained from adversarial examples. This critical step helps suppress the amplification of adversarial perturbations in the high-level representation before feeding them into the target model. Subsequently, we have removed the SRN, leaving only the CN responsible for modeling task. Essentially, this approach to collaboratively denoise adversarial noise can be regarded as a preprocessing step in the overall workflow.

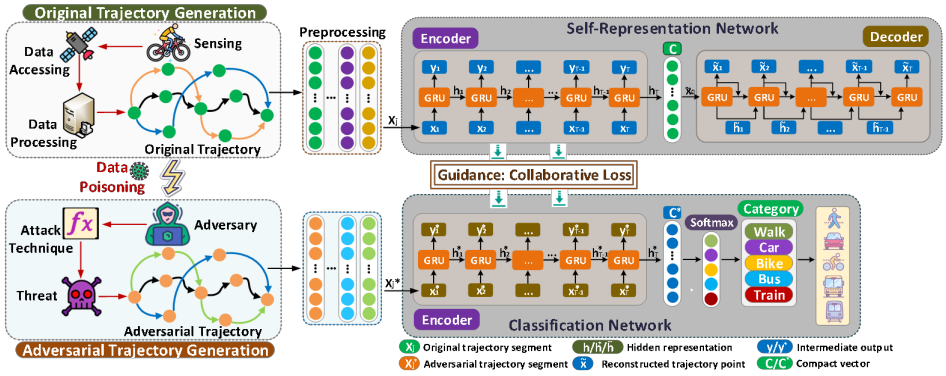


Fig. 3. Illustration of RobTC. The framework mainly consists of two components, specifically the upper SRN that automatically learns the robust features of the original trajectories to guide the bottom CN in effectively defending against the adversarial trajectories.

3.3 Self-representation Network

The SRN consists of an encoder and a decoder, where the encoder takes the original trajectory as input while the decoder produces a reconstructed trajectory sequence as output. The encoder integrated by GRU is responsible for extracting the robust high-level features from the original trajectory segment $X_j = \{x_1, x_2, x_3, \dots, x_n\}$. The model learns to encode a fixed-length sequence into a fixed-length vector representation, and conversely decodes the fixed-length vector representation back into a fixed-length sequence. From a probabilistic standpoint, the model serves as a general method for learning conditional distributions over a fixed-length sequence conditioned on another fixed-length sequence, e.g., $p(\tilde{x}_1, \dots, \tilde{x}_n | x_1, \dots, x_n)$. Specifically, the autoencoder first compresses X_j into a compact vector representation c and then reconstructs it to $\tilde{X}_j = \{\tilde{x}_1, \dots, \tilde{x}_n\}$ based on c.

3.3.1 Encoder

The encoder is a GRU that sequentially processes each point information of the input sequence X_j. The computational formula of the sequence X_j at step t is formalized as:

$$f = \begin{cases} z_t = \sigma(W_z \cdot [h_{t-1}, x_t]), \\ r_t = \sigma(W_r \cdot [h_{t-1}, x_t]), \\ h'_t = tanh(W_h \cdot [r_t \odot h_{t-1}, x_t]), \\ h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h'_t, \end{cases} \tag{1}$$

where the z_t signifies the update gate, r_t denotes the reset gate and h'_t represents the memory state. The W_z, W_r, and W_h are learned parameters. The σ and $tanh$ denote the activation function, the symbol $\odot$ denotes the element-wise product. As it reads each point information, the hidden state of the GRU evolves according to Equation (1). Until reaching the ending time-step T, we get the final representation c of the input trajectory sequence as shown in the equation below:

$$c = h_T. \tag{2}$$

3.3.2 Decoder

We have employed the GRU of the decode to reconstruct trajectory point $\tilde{X}_j$ based on the compact vector c. The decoder sequentially generates each trajectory point and integrates them with the hidden representation previously shown by the encoder, advancing the prediction until it concludes with the ending time step T, which can be written as:

$$\tilde{x}_0 = c, \quad \tilde{h}_t = f(\tilde{x}_{t-1}, \tilde{h}_{t-1}), \tag{3}$$

where $\tilde{x}_{t-1}, \tilde{h}_{t-1}$represent the reconstructed trajectory point and hidden representation in the $t-1$ time step, respectively. The reconstructed trajectory point for t time step is denoted as the equation below:

$$\tilde{x}_t = W_t \cdot \tilde{h}_t, \tag{4}$$

where W_t denotes weight matrices.

3.4 Classification Network

The encoder employed in CN is identical to the one utilized in the SRN, and the significant difference is that the decoder in CN is replaced by the target model. In contrast to SRN, we feed the adversarial trajectory segment X_j^* (generated by Equations 11 and 12) into CN to generate compact vector c^* as expressed by the equation below:

$$c^* = h_T^*, \tag{5}$$

where c^* represents the compact vector that summarizes the entire adversarial trajectory segment. Finally, we can mathematically express the probability distribution p of the transportation mode as the equation below:

$$p = \text{softmax}(W_c \cdot c^*), \tag{6}$$

where W_c denotes weight matrices.

3.5 Loss Function

In the construction of the RobTC framework, we have primarily employed three essential loss functions, namely self-representation network loss, collaborative loss, and classification network loss.

3.5.1 Self-representation network loss

We know that self-representation network loss is particularly important in construction of the self-

representation learning network. Not only does it directly affect the robustness of features extracted from the original examples but also exert an indirect influence on the decision boundaries of the classification network trained in collaboration with it. Specifically, the central component of the network involves the compression and subsequent reconstruction of the original example. The reconstruction process aims to make it as closely aligned as possible with the original input. To achieve this, we opt for the mean squared error as the function $\mathcal{L}^{SRN}$, which is denoted as the equation below:

$$\mathcal{L}^{SRN} = \sum_{j}^{N} \left\| X_j - \tilde{X}_j \right\|_2^2, \quad j \in \{1, \dots, N\}, \tag{7}$$

where N denotes the number of trajectory segments, X_j signifies the j-th original trajectory segment, and $\tilde{X}_j$ denotes the j-th reconstructed trajectory segment.

3.5.2 Collaborative loss

In this section, we introduce a collaborative training module that involves both SRN and CN networks. This module corrects the decision boundary of CN by enforcing a feature-level constraint between the two networks through a loss function. The loss aligns the high-level features extracted by the two networks, where the robust features constrain non-robust ones. This alignment aims to prevent the backward propagation and amplification of errors affixed to high-level features. Specifically, we feed the original examples into the SRN and the corresponding adversarial examples into the CN, respectively, both running simultaneously in a collaborative manner. This approach is employed to minimize the difference between the feature-level representations of the two networks, enabling the difference between two feature vectors to be essentially minimized as well. To achieve this, we use the L_2 norm as a collaborative loss $\mathcal{L}_j^{SC}$, which can be expressed as:

$$\mathcal{L}_j^{SC} = \sum_{t}^{T} \| h_t - h_t^* \|_2, \quad t \in \{1, \dots, T\}, \tag{8}$$

where h_t and h_t^* represent the feature-level representation of the original and adversarial trajectory points at time step t, respectively, and T denotes the ending time step. The entire expression for $\mathcal{L}^{SC}$ is given as the equation below:

$$\mathcal{L}^{SC} = \sum_{j=1}^{N} \mathcal{L}_j^{SC}, \quad j \in \{1,2, \dots, N\}, \tag{9}$$

where j denotes the j-th trajectory segment and N denotes the number of trajectory segments.

3.5.3 Classification network loss

The CN network comprises two components, namely an encoder and a target network. The encoder is responsible for denoising the adversarial examples, while the subsequent target network classifies them. We have chosen the cross-entropy loss here because its primary objective is to maximize the probability of correct class predictions while minimizing the probabilities of other classes, which enhances the classification network's ability to differentiate between various classes. When dealing with transportation mode category, the loss for CN can be expressed as follows:

$$\mathcal{L}^{CN} = -\sum_{j} \sum_{m=1}^{M} Y_{j,m} \, log\left(Y_{j,m}^*\right), \tag{10}$$

where $Y_{j,m}$ represents ground truth, $Y_{j,m}^*$ represents the predicted category, j denotes the j-th trajectory segment, N denotes the number of trajectory segments, m denotes the index class, and M denotes the

total category of transportation modes. The final loss is denoted as the equation below:

$$\mathcal{L} = \mathcal{L}^{SRN} + \beta\mathcal{L}^{SC} + \lambda\mathcal{L}^{CN}, \tag{11}$$

where the β and λ denote the hyper-parameters for balancing different parts of the final loss.

4. Experiments and Analysis

To explore the effectiveness and robustness of RobTC, we have conducted extensive experiments on two publicly available spatiotemporal trajectory datasets, namely Geolife and Beijing Taxi Traces. This part is mainly divided into experimental preliminary, ablation study, hyperparameter evaluation, performance evaluation, denoising loss & performance evaluation, and error amplification effect.

4.1 Preliminary Experiment

4.1.1 Datasets

The Geolife dataset [51] was collected within the Geolife project by Microsoft Research Asia from 2007 to 2012 and has since been widely used in the field of spatiotemporal trajectory classification. Specifically, it comprises the unit of km and approximately 8,000 various modes of transport labeled into one of the five categories such as bike, bus, car, train, and walk. Beijing Taxi Traces dataset [50] collected GPS trajectories of 10,357 taxis in Beijing from February 2–8, 2008. The dataset comprises approximately 15 million data points, covering a total trajectory distance of 9 million kilometers, of which 2000 sub-trajectory segments are truncated to perform a binary classification task, for example one is available, the other is occupied. The two datasets record the longitude, latitude, and timestamp in different trajectories, providing a reliable basis for our classification tasks. As part of the preliminary experiment setup, we have split them into training (80%), validation (10%), and testing sets (10%).

4.1.2 Preprocessing

Before conducting the experiments, it is essential to preprocess the raw data, which includes removing duplicated records and redundant information such as zero values and altitude. We have normalized the longitude and latitude to ensure they would fall within the range of 0 to 1. Timestamps are converted into hours to extract the relative time information. Furthermore, each trajectory segment is standardized to consist of 100 points. Insufficient points are filled with zeros, while excess points are truncated to maintain uniform length across all trajectories. Subsequently, we have employed single-step and multi-step adversarial attacks, including FGSM, I-FGSM, and MI-FGSM, to target models such as CNN, LSTM, GRU, and RobTC. The perturbation factor ϵ is set to 16 to generate the requisite adversarial examples.

$$x_1^* = x_1, \quad x_{t+1}^* = x_t^* + \alpha \cdot sign\big(\nabla_x \mathcal{L}(x_t^*, Y)\big). \tag{12}$$

To ensure that the crafted adversarial examples would satisfy the L_∞ or L_2 constraint, we could either limit x_t^* within an ϵ range of x or establish $\alpha = \epsilon/T$ as α, where T represents the number of iterations. Iterative approaches [36] have been shown to outperform single-step ones in white-box attacks, but the opposite is true for transferability. On the other hand, the MI-FGSM formula can be written as:

$$\begin{aligned} g_{t+1} &= \mu \cdot g_t + \frac{\nabla_x \mathcal{L}(x_t^*, Y)}{\|\nabla_x \mathcal{L}(x_t^*, Y)\|_1}, \\ x_{t+1}^* &= x_t^* + \alpha \cdot sign(g_{t+1}), \end{aligned} \tag{13}$$

where μ serves as the decay factor of the momentum term and g_t is accumulated gradient at iteration t.

4.1.3 Experiment settings

Given that existing spatiotemporal trajectory classification tasks have already achieved outstanding performance using both the CNN and RNN approaches [52–54], we can consider some of them as our baseline. More importantly, we aim to improve the robustness of the model while simultaneously preserving its accuracy.

CNN: VGG16 network serves as a representative based on the CNNs.

LSTM: The Social LSTM [53] can automatically learn the typical interactions occurring between trajectories that coincide in time by connecting the LSTM corresponding to nearby sequences, and achieve a better modeling result.

GRU: TrajGRU [54] is excellent at spatio-temporal trajectory classification.

ENSEMBLE: We have employed VGG16, Social LSTM, and TrajGRU as trained models for adversarial training. Then FGSM, I-FGSM, and MI-FGSM attacks are performed on the three generated models, with $\epsilon = 16$ for each attack, to generate corresponding adversarial examples. For simplicity, CNN represents VGG16, LSTM represents Social LSTM, and GRU represents TrajGRU. In each batch of training, we alternate the source of adversarial examples between the model currently being trained and one of the generated models.

Evaluation of data: In addition to a robustness evaluation, the amount of training data and time is also an important metric. During the training stage, RobTC utilizes only a small portion of trajectory data, making it highly efficient in terms of training. Specifically, only 4K original trajectory segments are employed in constructing our training dataset, whereas all 8K original trajectory segments from the Geolife dataset are used to train GRUens. Furthermore, RobTC is trained on 4K trajectory segments for less than 50 epochs, while CNNens, LSTMens, and GRUens all need to train on 8K trajectory segments for approximately 200 epochs. In other words, with fewer training data and less time, RobTC significantly outperforms adversarial training in defending against various adversarial attacks. The findings suggest that addressing the decoupled task of denoising is much easier than tackling the coupled tasks of classification and defense. During the testing stage, we utilized a dataset consisting of 2K original trajectory segments, which were divided into 20 batches.

4.2 Ablation Study

To systematically evaluate the effectiveness of the SRN component within the overall architecture of the proposed method, we have conducted ablation studies by comparing the model's performance under two contrasting conditions, namely one with the presence of the SRN component and another without it (Table 2,Table 3).

In white-box attacks, the accuracy of CNN, LSTM, and GRU decreases by up to 80% compared to the same ones in original examples, while RobTC experiences a more modest decrease of 18%, which highlights the vulnerability of the three deep models and the enhanced robustness of the proposed model. Besides, when comparing the GRU with the RobTC, the accuracy of the former is only 26.86%, whereas the latter achieves 71.27% in accuracy, showing the effectiveness of the self-representation learning framework in the RobTC model. In contrast to white-box attacks, black-box attacks are more likely to transfer between models, making them more challenging to defend against. In black-box settings, the accuracy of CNN, LSTM and GRU decreases by up to 58%, whereas RobTC decreases by only 9%, indicating that these three deep models are vulnerable to black-box attacks, while the proposed model still shows an excellent robustness. Additionally, in the comparison between GRU and RobTC, the accuracy of GRU is only 25.86%, while RobTC achieves the highest accuracy of 78.27%, with a significant improvement by nearly 53%.

Granted that the adversarial example essentially adds noise to the original sample to maximize feature distortion, we intend to fix this distortion to ensure that the latent features of the adversarial example are aligned with that of the original example. In fact, the effectiveness of our method can be attributed to its use of SRN to extract the robust representation from the original example and use it to guide the CN

network for denoising the adversarial example. As a strict constraint, this guidance corrects the decision boundaries of the network, thus effectively suppressing the amplification of perturbations carried by adversarial examples. Essentially, we build an end-to-end mapping network that maps the source domain (adversarial example) to the target domain (original example), with the noise ultimately removed the noise from the adversarial example.

Table 2. Various deep models under different attack techniques (no defensive protection)

	Attack	Accuracy (%)			
		CNN	**LSTM**	**GRU**	**RobTC**
CNN	FGSM	21.82*	42.25	43.38	**77.42**
	I-FGSM	6.62*	46.65	46.19	**78.17**
	MI-FGSM	4.36*	27.12	26.96	**74.26**
LSTM	FGSM	43.27	22.23*	41.28	**77.15**
	I-FGSM	46.63	8.63*	44.52	**78.27**
	MI-FGSM	27.78	5.75*	26.94	**75.19**
GRU	FGSM	42.98	40.16	20.86*	**74.87**
	I-FGSM	46.84	44.74	7.23*	**76.97**
	MI-FGSM	28.76	26.19	4.35*	**73.83**
RobTC	FGSM	43.38	42.12	40.76	**71.27***
	I-FGSM	46.65	45.68	43.19	**73.36***
	MI-FGSM	25.94	26.14	25.86	**69.18***
ORIGINAL		81.16	82.96	**83.24**	83.21

The top row corresponds to undefended models, while the leftmost column designates the models responsible for generating adversarial examples. "∗" represents white-box attacks, while data without "*" label represents black-box attacks, and "ORIGINAL" represents models of original examples (without perturbation). Note that the deep models are all in an undefended state, with RobTC being the only model equipped with a defense mechanism. The bold font indicates the best performance in each test.

Table 3. Various deep models under different attack techniques (adversarial training protection)

	Attack	Accuracy (%)			
		CNNens	**LSTMens**	**GRUens**	**RobTC**
CNN	FGSM	69.74	68.64	68.96	77.42
	I-FGSM	71.17	70.38	65.27	78.17
	MI-FGSM	68.69	67.67	68.21	74.26
LSTM	FGSM	66.65	68.58	68.51	77.15
	I-FGSM	69.36	71.21	70.22	78.27
	MI-FGSM	65.61	68.15	64.13	75.19
GRU	FGSM	67.96	69.69	66.18	76.87
	I-FGSM	70.97	71.61	69.29	77.97
	MI-FGSM	66.12	66.21	64.36	75.83
RobTC	FGSM	67.34	68.26	69.18	71.27
	I-FGSM	71.27	72.37	71.29	73.36
	MI-FGSM	66.25	67.24	67.89	69.18
ORIGINAL		79.34	80.89	81.57	83.21

The top row corresponds to undefended models, while the leftmost column designates the models responsible for generating adversarial examples. "ORIGINAL" represents models of original examples (without perturbation). Note that the deep models are all in an undefended state, with RobTC being the only model equipped with a defense mechanism.

4.3 Hyperparameter Evaluation

To verify the effective of the hyper-parameters of the proposed model, we have investigated the number

of iterations t in multi-step attacks, decay factor μ in momentum iteration, and the setting of perturbation factor ϵ in adversarial attacks. The primary objective of these experiments is to get a more suitable hyperparameter to achieve better attack effects for testing the robustness of our proposed model.

4.3.1 Decay factor μ

We know that the decay factor μ is particularly important in increasing the success rate of attacks, where the success rate is defined as the misclassification rate of the corresponding models with adversarial examples as inputs. When $\mu = 0$, the momentum-based iterative method typically reverts to the standard iterative method, namely I-FGSM. We have conducted MI-FGSM attacks based on CNN with a perturbation $\epsilon = 16$ (referencing to the ϵ research below), 10 iterations (inefficiency with too many iterations), and a range of decay factors from 0.0 to 2.0 in increments of 0.1.

As shown in Fig. 4, the success rate against adversarial examples on the CNN, LSTM, GRU, and RobTC models, are adjusted for the decay factor μ range. Notably, the success rate of white-box attacks on CNN is close to 80%, which remains consistent across various decay factor magnitudes. This result shows the effectiveness of the iterative method in white-box attacks, even without the influence of decay factors. The success rate curves for LSTM and GRU in black-box attacks exhibit a unimodal shape, with the peak occurring around $\mu = 1.0$, which we have adopted for our subsequent experiments. When $\mu = 1.0$, an alternate interpretation of the g_t defined in Equation (13) is that it simply accumulates all previous gradients to undertake the current update. For our proposed method, regardless of the variations in the decay factor, the attack success rate consistently remains below 20%. This result serves as additional confirmation of the robustness of our proposed model against black-box attacks.

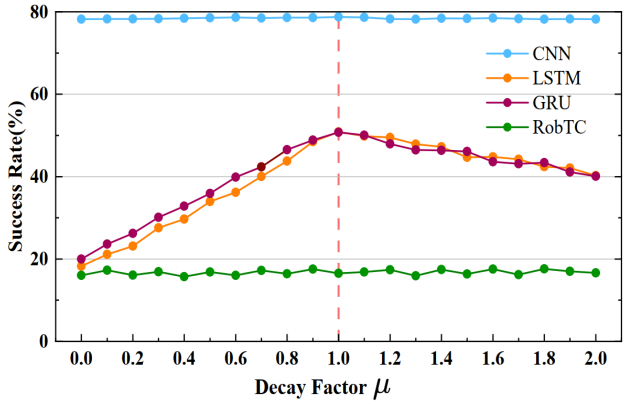


Fig. 4. Success rate (%) of adversarial examples generated for CNNs against CNN (white-box), LSTM, GRU, and RobTC (black-box), while varying μ from 0.0 to 2.0.

4.3.2 Number of iterations

We delve into the impact of the number of iterations on the success rate when using I-FGSM and MI-FGSM. Employing consistent hyperparameters (i.e., $\epsilon = 16$, $\mu = 1.0$), we have attacked the CNN model with varying iteration counts ranging from 1 to 10. Subsequently, we evaluate the success rate of adversarial examples against CNN, LSTM, GRU, and RobTC models.

As shown in Fig. 5, with an increase in the number of iterations, the success rate of I-FGSM against the black-box model gradually decreases, whereas MI-FGSM preserves a relatively high-success rate. This suggests that adversarial examples generated through the iterative approach are prone to overfitting to white-box models and have limited transferability across different models. However, momentum-based iterative methods effectively mitigate the trade-off between a white-box attack and transferability, thus performing well in both white and black box attack settings. To achieve a better attack performance, we set 2 as the number of iterations, as seen in the red dashed line in Fig. 5.

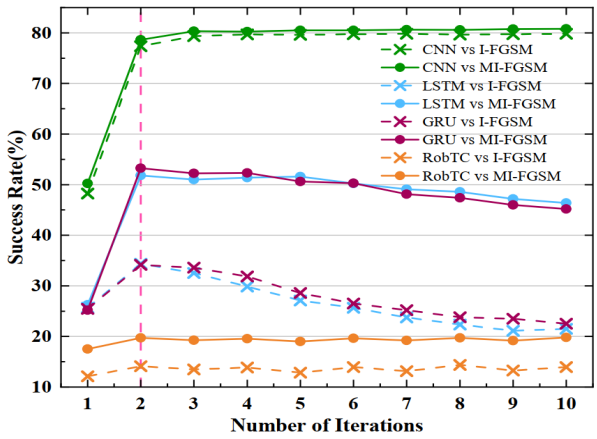


Fig. 5. Success rate (%) of adversarial examples generated for the CNNs against CNN (white-box), LSTM, GRU, and RobTC (black-box).

4.3.3 Perturbation factor ϵ

Since adversarial examples are crafted by adding small perturbations to original examples, and the strength of these perturbations depends on the magnitude of the perturbation factor ϵ, it holds significant practical significance to study the impact of the perturbation factor on the effectiveness of attacks, which helps to optimize the subsequent defensive strategy research. Then, we have conducted extensive experiments on two publicly available datasets, which includes comparing the accuracy of three models (GRU, GRUens, RobTC) against FGSM, I-FGSM, and MI-FGSM attacks with varying ϵ. The experimental results are illustrated in Fig. 6. Fig. 6(a)–6(c) represent the results on the Geolife dataset, while Fig. 6(d)–6(f) represent the results on the Beijing taxi trajectory dataset. It is evident from these curves that the accuracy of the GRU model sharply decreases as ϵ increases from 0 to 2, and then slightly decreases within a range of 2 to 16, stabilizing at a lower level. In contrast, the accuracy of GRUens and RobTC shows a mild decrease as ϵ increases from 0 to 2, then stabilizing at higher levels within a range of 2 to 16. Overall, when subjected to the same perturbation strength, the accuracy of GRUens decreases more compared to RobTC. This suggests that GRUens is relatively sensitive to perturbations but that RobTC is not. Upon observing each curve (ranging from 2 to 16), we can see that the attack itself slowly intensifies and gradually approaches saturation point.

4.4 Performance Evaluation

4.4.1 Evaluation of ensemble adversarial training models

Performance evaluation is mainly divided into two parts, specifically one is to compare methods based on adversarial training (Section 4.4), and the other is to compare methods based on denoising method (Section 4.5). Here, ensemble adversarial training is an enhanced version of adversarial training, as it significantly improves the model's robustness against black-box attacks. This prompts us to conduct comparative experiments using it alongside our proposed model. We set ϵ to 16, set the number of iterations to 2 for both I-FGSM and MI-FGSM, and set μ to 1.0 for MIFGSM. As shown in Table 3, the accuracy of all ensemble adversarial training models experiences a decline under the adversarial settings. However, this decline, although reaching as high as 17% in some cases, is significantly smaller than the corresponding deep models, which indicates that ensemble adversarial training models exhibit a good robustness. Notably, the RobTC model displays the highest accuracy drop of only 14%, surpassing the other three ensemble adversarial training models (CNNens, LSTMens, and GRUens). These findings suggest that the RobTC model exhibits a stronger robustness compared with ensemble adversarial training models.

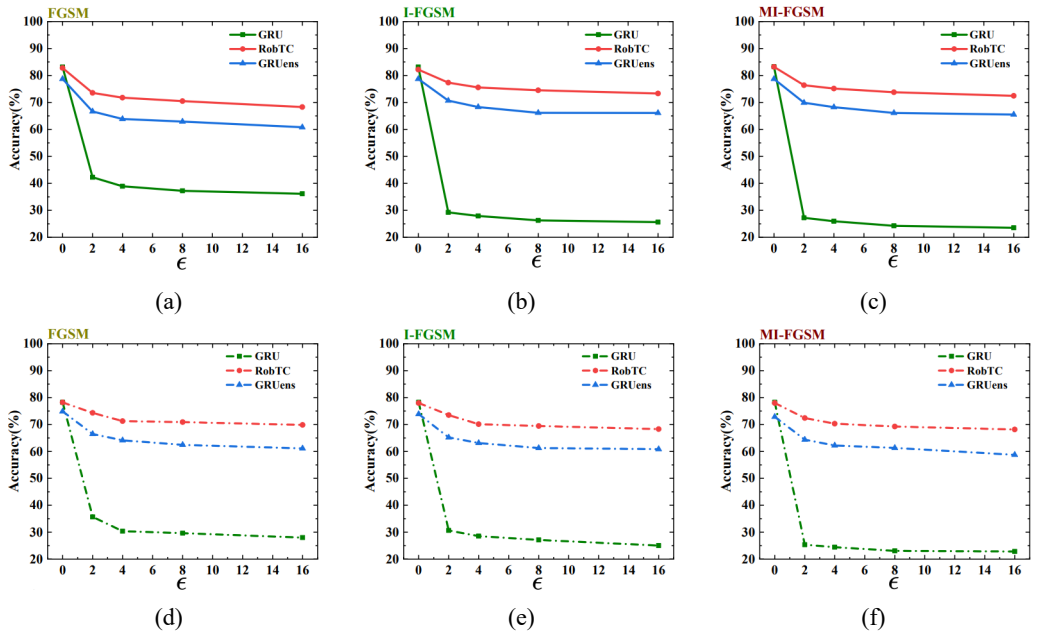


Fig. 6. Accuracy of various models when subjected to FGSM, I-FGSM, and MI-FGSM adversarial attacks (ϵ ranging from 0 to 16). (a)–(c) depict the results obtained from the Geolife datasets, and (d)–(f) show the results from the Beijing taxi traces datasets.

4.4.2 Evaluation of all models on original examples

In fact, we should increase the model robustness without sacrificing its performance. At present, although many defense methods improve the robustness of the model, they are accompanied by the side effect of performance penalty. So, we have conducted experiments using their original examples, rather than adversarial examples. Tables 2 and 3 reveal that ensemble adversarial training models have a slight decline in prediction accuracy compared to deep models, inferring a certain level of performance penalty. However, RobTC can significantly enhance the model's robustness against various adversarial attacks, while preserving the model's prediction accuracy on original examples. Our hypothesis is that injecting adversarial examples during training amounts to introducing a regularization term, which dampens the model's fitting process and inevitably leads to a certain degree of performance loss. In practice, our proposed method adopts a preprocessing manner, discarding this regularization operation, which involves removing the attached noise from adversarial examples before feeding them into the target model. Consequently, even when operating on the original examples, our model achieves a higher prediction accuracy.

4.5 Denoising Loss & Performance Evaluation

Denoising loss is a metric used to evaluate the effectiveness of models in denoising original input. It quantifies the difference between the original input and the denoised output, indicating the level of distortion in the input. We have compared RobTC with the previous methods: pixel guided denoiser (PGD) and high-level representation guided denoiser (LGD) [14] and then repeated the experiment 10 times to take an average of the final results. Tables 4 and 5 reveals that RobTC and LGD exhibit similar denoising loss, both of which are inferior to that of PGD. We hypothesize that this difference may arise from the presence of not only vertical information transfer but also a lateral one within the DUNET structure of PGD, and the fusion of these two types of information mitigates information loss to some extent. In terms of accuracy, RobTC preserves the model's prediction accuracy compared with ND (no

defence) in the original examples. Additionally, it significantly outperforms the other three models in both white and black box attack settings, fully demonstrating that our proposed model still maintains a superior robustness.

Table 4. Denoising loss of different denoising methods in the experiment

Defense	Original	White-box		Black-box	
		$\epsilon = 4$	$\epsilon = 16$	$\epsilon = 4$	$\epsilon = 16$
ND	0.0000	0.0185	0.0528	0.0187	0.0536
PGD	**0.0142**	**0.0145**	**0.0161**	**0.0143**	**0.0169**
LGD	0.0372	0.0369	0.0372	0.037	0.0378
RobTC	0.0354	0.0362	0.0368	0.0371	0.0374

Denoising loss is the L_1 distance between the input trajectory segments & the denoised ones, ND means no defense, and original represents the original trajectory. The bold font indicates the best performance in each test.

Table 5. Robust accuracy (%) of different denoising methods in the experiment

Defense	Original	White-box		Black-box	
		$\epsilon = 4$	$\epsilon = 16$	$\epsilon = 4$	$\epsilon = 16$
ND	**83.24**	35.42	28.23	42.12	47.52
PGD	73.63	48.12	44.61	58.08	52.23
LGD	75.21	55.12	52.61	61.08	60.23
RobTC	83.21	**69.26**	**67.27**	**74.33**	**73.42**

ND means no defense, and original represents the original trajectory. The bold font indicates the best performance in each test.

4.6 Error Amplification Effect

Previous experiments have revealed an inconsistency between the denoising loss of PGD and its corresponding classification accuracy, which appears in different attack settings. Whether for the white- or black-box, the denoising loss of PGD outperforms that of LGD, but its accuracy is lower than that of LGD. Also, the denoising loss of the RobTC model is comparable to LGD to that of LGD, yet its accuracy surpasses that of LGD, which fully demonstrates the inconsistency. To explore this inconsistency, we measured the hierarchical perturbation of the target model caused by the input. Specifically, we quantified the perturbation intensity at layer l by defining T_p as the perturbed trajectory and then used the equation to calculate the perturbation intensity:

$$I_l(T_p, x) = \left|f_l(T_p) - f_l(x)\right| / |f_l(x)|. \tag{14}$$

As shown in Fig. 7, the perturbation intensity I_l curves of the five models tend to expand with the growth of the number of model layers, and the adversarial examples here are generated by "FGSM × CNN/LSTM/GRU ($\epsilon = 16$)." When the denoised example is at the 0-th layer of the target model, meaning it has just been fed into the target model, the perturbation intensity of PGD, adversarial, and Gaussian noise are similar and significantly lower than LGD and RobTC. The perturbation intensity values of PGD, adversarial, and Gaussian noise are around 0.15, while LGD and RobTC are around 0.2, which shows the initial state of different perturbations carried by the denoised examples when inserted into the target model. Although the initial state perturbation intensity of adversarial and PGD is weak, with an increase in the layers of the target model increase, the tiny perturbation carried by them gradually amplifies, becoming greater than LGD, RobTC, and Gaussian noise. It is clear that both LGD and RobTC exhibit a great suppression of this "error amplification effect," but RobTC performs better as it approximates our baseline, Gaussian noise.

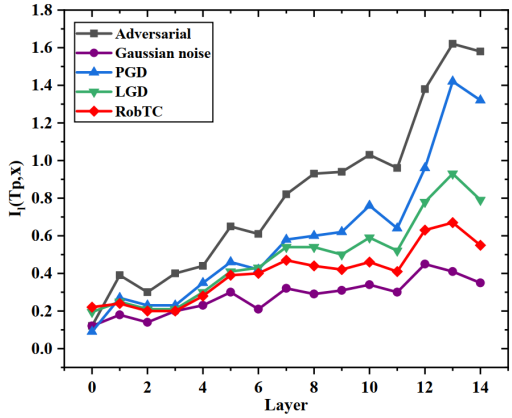
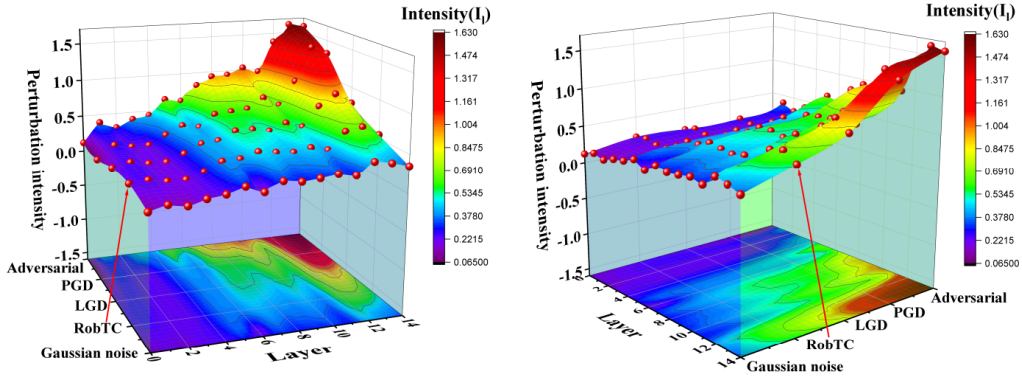


Fig. 7. Visualization of layerwise perturbation level of the target model. Adversarial, Gaussian noise, PGD, LGD, and RobTC correspond to the I_l for adversarial examples, Gaussian noise perturbed examples, PGD denoised examples, LGD denoised examples, and RobTC denoised examples, respectively.

To better demonstrate the varying suppression effects of different denoising models on this "error amplification effect," we plotted a 3D visualization. As shown in Fig. 8, the underlying coordinates in both subfigures represent the number of layers of the target model and the denoised examples using different denoising models. The ordinate represents the corresponding perturbation intensities. Fig. 8(a) and 8(b) are identical but presented from different angles. These two views collectively provide a comprehensive depiction of the change in perturbation intensity as the denoised example enters the target model.



(a) (b)

Fig. 8. A 3D visualization of the "error amplification effect" suppression through the RobTC model: (a) front view and (b) back view.

From the Fig. 8(a), we can observe that as five different types of examples carrying noise are fed into the target model, the intensity of their respective perturbation is amplified layer by layer, but this also manifests intensity differences. From the Fig. 8(b), we can clearly see the perturbation intensity of the five types of noisy examples when they reach the final layer of the target model. This view also confirms the results observed in the front view Fig. 8(a), with particularly apparent intensity in the last layer of the target model. Through these experiments presented above, we can observe that the RobTC model effectively suppresses the impact of the "error amplification effect" throughout the target model. Compared with the PGD denoising method, the initial denoising effect of RobTC is not optimal. However, when the feature propagates back layer by layer, this "error amplification effect" is suppressed, and the

error parasitic in the feature is gradually weakened during the transmission of the feature to the top layer. It is well known that the top feature pair basically determines the final classification result, thus accounting for the inconsistency between its denoising loss and classification accuracy. As mentioned above (Section 4.2), we have guided the CN by using the robust features extracted from the SRN to minimize the feature distortion of the adversarial example compared to the original one. In this process, SRN constrains the CN layer by layer and inhibits an error amplification layer by layer.

5. Summary & Future Work

To enhance the robustness of DNN-based trajectory classification, we propose a novel collaborative learning method for robust spatio-temporal trajectory classification named RobTC, which consists of an autoencoder-based self-representation network for robust latent feature learning and a GRU-based classification network by sharing information with each other to defend against various adversarial attacks. We introduce feature-level constraints between the original input and the corresponding adversarial examples instead of the point-level denoising strategies to effectively suppress the potential "error amplification effect". Extensive experiments on the Geolife and Beijing taxi traces datasets demonstrate that our proposed method yields significant improvements (white-box 15% and black-box 13%) over the state-of-the-art methods, suggesting that ours can significantly enhance the model's robustness against various adversarial attacks while preserving its prediction accuracy on original examples. In practice, the proposed method is characterized by a simpler training procedure with fewer training data and less time.

In future work, we intend to extend the application of the proposed method to other practical domains to evaluate its feasibility and scalability and anticipate that the creation of adversarial defense models in diverse domains, such as images and text, may follow a similar approach to our model, with only minor modifications to the feature extraction phase. Given the limited availability of labeled adversarial examples in spatio-temporal trajectory modeling tasks, we continue to explore the potential of pre-training approaches to address this challenge. Specifically, we seek to incorporate pre-training mechanisms like few-shot learning into our future work.

Author's Contributions

Conceptualization, JJ, LL; Investigation and methodology, JJ, LL; Project administration, JJ, PQ; Resources, JJ, XZ; Supervision, JJ, BC; Writing of the original draft, XL, BC; Writing of the review and editing, XZ; Software, BC, XZ; Validation, XZ, PQ; Formal analysis, JJ, BC; Data curation, JJ, XL; Visualization, JJ, LL.

Funding

This work is supported in part by the National Natural Science Foundation of China (Grant No. 62202066 and 62302056), the Fundamental Research Funds for the Central Universities (No. 2023RC71 and 2023RC85), and the Beijing Natural Science Foundation (Grant No. 4242026).

Competing Interests

The authors declare that they have no competing interests.

References

[1] M. Zancanaro, G. Gallitto, D. Yem, and B. Treccani, "Improving mental models in IoT end-user development," *Human-Centric Computing and Information Sciences*, vol. 12, article no. 48, 2022. https://doi.org/10.22967/HCIS.2022.12.048

[2] Q. Wang and Z. Mu, "Feature selection and SVM parameter synchronous optimization based on a hybrid intelligent optimization algorithm," *Human-Centric Computing and Information Sciences*, vol. 13, article no. 12, 2023. https://doi.org/10.22967/HCIS.2023.13.012

[3] X. Guo, M. Gao, J. Pan, J. Shang, A. Souri, Q. Li, and A. Bruno, "Crowd counting via attention and multi-feature fused network," *Human-centric Computing and Information Sciences*, vol. 13, article no. 50, 2023. https://doi.org/10.22967/HCIS.2023.13.050

[4] L. Xiong, X. Xiong, F. Zhang, and H. Chen, "Unsupervised deep embedding clustering for AIS trajectory," in *Proceedings of 2022 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Kuala Lumpur, Malaysia, 2022, pp. 2283-2286. https://doi.org/10.1109/IGARSS46834.2022.9884800

[5] D. Al-Fraihat, Y. Sharrab, F. Alzyoud, A. Qahmash, M. Tarawneh, and A. Maaita, "Speech recognition utilizing deep learning: a systematic review of the latest developments," *Human-centric Computing and Information Sciences*, vol. 14, article no. 15, 2024. https://doi.org/10.22967/HCIS.2024.14.015

[6] H. Min, X. Xiong, P. Wang, and Z. Zhang, "A hierarchical LSTM-based vehicle trajectory prediction method considering interaction information," *Automotive Innovation*, vol. 7, no. 1, pp. 71-81, 2024. https://doi.org/10.1007/s42154-023-00261-0

[7] Y. Wang, H. Wu, J. Zhang, Z. Gao, J. Wang, P. S. Yu, and M. Long, "PredRNN: a recurrent neural network for spatiotemporal predictive learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, pp. 2208-2225, 2023. https://doi.org/10.1109/TPAMI.2022.3165153

[8] G. Tao, W. Sun, T. Han, C. Fang, and X. Zhang, "RULER: discriminative and iterative adversarial training for deep neural network fairness," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Singappore, 2022, pp. 1173-1184. https://doi.org/10.1145/3540250.3549169

[9] F. Biljecki, H. Ledoux, and P. Van Oosterom, "Transportation mode-based segmentation and classification of movement trajectories," *International Journal of Geographical Information Science*, vol. 27, no. 2, pp. 385-407, 2013. https://doi.org/10.1080/13658816.2012.692791

[10] J. Zhang, J. T. Huang, W. Wang, Y. Li, W. Wu, X. Wang, Y. Su, and M. R. Lyu, "Improving the transferability of adversarial samples by path-augmented method," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Vancouver, Canada, 2023, pp. 8173-8182. https://doi.org/10.1109/CVPR52729.2023.00790

[11] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018 pp. 1625-1634. https://doi.org/10.1109/CVPR.2018.00175

[12] Y. Liu, W. Zhang, and J. Wang, "Zero-shot adversarial quantization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, TN, USA, 2021, pp. 1512-1521. https://doi.org/10.1109/CVPR46437.2021.00156

[13] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, "Recent advances in adversarial training for adversarial robustness," 2021 [Online]. Available: https://arxiv.org/abs/2102.01356.

[14] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 1778-1787. https://doi.org/10.1109/CVPR.2018.00191

[15] J. G. Lee, J. Han, X. Li, and H. Gonzalez, "TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 1081-1094, 2008. https://doi.org/10.14778/1453856.1453972

[16] D. Patel, "Incorporating duration and region association information in trajectory classification," *Journal of Location Based Services*, vol. 7, no. 4, pp. 246-271, 2013. https://doi.org/10.1080/17489725.2013.805828

[17] S. Dodge, R. Weibel, and E. Forootan, "Revealing the physics of movement: comparing the similarity of movement characteristics of different types of moving objects," *Computers, Environment and Urban Systems*, vol. 33, no. 6, pp. 419-434, 2009. https://doi.org/10.1016/j.compenvurbsys.2009.07.008

[18] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W. Y. Ma, "Understanding transportation modes based on GPS data for web applications," *ACM Transactions on the Web (TWEB)*, vol. 4, no. 1, article no. 1, 2010. https://doi.org/10.1145/1658373.1658374

[19] Z. Xiao, Y. Wang, K. Fu, and F. Wu, "Identifying different transportation modes from trajectory data using tree-based ensemble classifiers," *ISPRS International Journal of Geo-Information*, vol. 6, no. 2, article no. 57, 2017. https://doi.org/10.3390/ijgi6020057

[20] R. Saini, P. Kumar, P. P. Roy, and D. P. Dogra, "An efficient approach for trajectory classification using FCM and SVM," in *Proceedings of 2017 IEEE Region 10 Symposium (TENSYMP)*, Cochin, India, 2017, pp. 1-4. https://doi.org/10.1109/TENCONSpring.2017.8070076

[21] S. Dabiri and K. Heaslip, "Inferring transportation modes from GPS trajectories using a convolutional neural network," *Transportation Research Part C: Emerging Technologies*, vol. 86, pp. 360-371, 2018. https://doi.org/10.1016/j.trc.2017.11.021

[22] J. Zeng, Y. Yu, Y. Chen, D. Yang, L. Zhang, and D. Wang, "Trajectory-as-a-sequence: a novel travel mode identification framework," *Transportation Research Part C: Emerging Technologies*, vol. 146, article no. 103957, 2023. https://doi.org/10.1016/j.trc.2022.103957

[23] Q. Gao, F. Zhou, K. Zhang, G. Trajcevski, X. Luo, and F. Zhang, "Identifying human mobility via trajectory embeddings," *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, Melbourne, Australia, 2017, pp. 1689-1695.

[24] F. Zhou, Q. Gao, G. Trajcevski, K. Zhang, T. Zhong, and F. Zhang, "Trajectory-user linking via variational autoencoder," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, Stockholm, Sweden, 2018, pp. 3212-3218.

[25] H. Liu, H. Wu, W. Sun, and I. Lee, "Spatio-temporal GRU for trajectory classification," in *Proceedings of 2019 IEEE International Conference on Data Mining (ICDM)*, Beijing, China, 2019, pp. 1228-1233. https://doi.org/10.1109/ICDM.2019.00152

[26] Y. Liang, K. Ouyang, H. Yan, Y. Wang, Z. Tong, and R. Zimmermann, "Modeling trajectories with neural ordinary differential equation," in *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, Virtual Event, 2021, pp. 1498-1504.

[27] J. Liu, Y. Liu, W. Zhu, X. Zhu, and L. Song, "Distributional and spatial-temporal robust representation learning for transportation activity recognition," *Pattern Recognition*, vol. 140, article no. 109568, 2023. https://doi.org/10.1016/j.patcog.2023.109568

[28] J. Kim, J. H. Kim, and G. Lee, "GPS data-based mobility mode inference model using long-term recurrent convolutional networks," *Transportation Research Part C: Emerging Technologies*, vol. 135, article no. 103523, 2022. https://doi.org/10.1016/j.trc.2021.103523

[29] Y. Yao, H. Zhang, X. Shi, J. Chen, W. Li, X. Song, and R. Shibasaki, "LTP-Net: life-travel pattern based human mobility signature identification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 14306-14319, 2023. https://doi.org/10.1109/TITS.2023.3303835

[30] S. Wen, H. Wang, and D. Metaxas, "Social ode: multi-agent trajectory forecasting with neural ordinary differential equations," in *Computer Vision – ECCV 2022*. Cham, Switzerland: Springer, 2022, pp. 217-233. https://doi.org/10.1007/978-3-031-20047-2_13

[31] C. Park, T. Kim, J. Hong, M. Choi, and J. Choo, "Pre-training contextual location embeddings in personal trajectories via efficient hierarchical location representations," in *Machine Learning and Knowledge Discovery in Databases: Applied Data Science and Demo Track*. Cham, Switzerland: Springer, 2023, pp. 125-140. https://doi.org/10.1007/978-3-031-43430-3_8

[32] Z. Wei, J. Chen, Z. Wu, and Y. G. Jiang, "Boosting the transferability of video adversarial examples via temporal translation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 3, pp. 2659-2667, 2022. https://doi.org/10.1609/aaai.v36i3.20168

[33] M. Tayyab, M. Marjani, N. Z. Jhanjhi, I. A. T. Hashem, R. S. A. Usmani, and F. Qamar, "A comprehensive review on deep learning algorithms: security and privacy issues," *Computers & Security*, vol. 131, article no. 103297, 2023. https://doi.org/10.1016/j.cose.2023.103297

[34] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain generalization: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 4396-4415, 2023. https://doi.org/10.1109/TPAMI.2022.3195549

[35] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial Intelligence Safety and Security*. Boca Raton, FL: Chapman and Hall/CRC, 2018, pp. 99-112.

[36] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 9185-9193. https://doi.org/10.1109/CVPR.2018.00957

[37] J. Zhang, W. Wu, J. T. Huang, Y. Huang, W. Wang, Y. Su, and M. R. Lyu, "Improving adversarial transferability via neuron attribution-based attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, New Orleans, LA, USA, 2022, pp. 14993-15002. https://doi.org/10.1109/CVPR52688.2022.01457

[38] X. Jia, Y, Zhang, X. Wei, B. Wu, K. Ma, J. Wang, and X. Cao, "Prior-guided adversarial initialization for fast adversarial training," in *Computer Vision – ECCV 2022*. Cham, Switzerland: Springer, 2022, pp. 567-584. https://doi.org/10.1007/978-3-031-19772-7_33

[39] Y. Yu and C. Z. Xu, "Efficient loss function by minimizing the detrimental effect of floating-point errors on gradient-based attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Vancouver, Canada, 2023, pp. 4056-4066. https://doi.org/10.1109/CVPR52729.2023.00395

[40] F. Y. Wang, D. W. Zhou, H. J. Ye, and D. C. Zhan, "Foster: feature boosting and compression for class-incremental learning," in *Computer Vision – ECCV 2022*. Cham, Switzerland: Springer, 2022, pp. 398-414. https://doi.org/10.1007/978-3-031-19806-9_23

[41] J. Song, Y. Chen, J. Ye, and M. Song, "Spot-adaptive knowledge distillation," *IEEE Transactions on Image Processing*, vol. 31, pp. 3359-3370, 2022. https://doi.org/10.1109/TIP.2022.3170728

[42] K. ElHaj, D. Alshamsi, and A. Aldahan, "GeoZ: a region-based visualization of clustering algorithms," *Journal of Geovisualization and Spatial Analysis*, vol. 7, no. 1, article no. 15, 2023. https://doi.org/10.1007/s41651-023-00146-0

[43] H. Zhang, X. Zhao, H. Wang, B. Niu, and N. Xu, "Adaptive tracking control for output-constrained switched MIMO pure-feedback nonlinear systems with input saturation," *Journal of Systems Science and Complexity*, vol. 36, no. 3, pp. 960-984, 2023. https://doi.org/10.1007/s11424-023-1455-y

[44] A. MaungMaung, I. Echizen, and H. Kiya, "Hindering adversarial attacks with multiple encrypted patch embeddings," in *Proceedings of 2023 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Taipei, Taiwan, 2023, pp. 1398-1404. https://doi.org/10.1109/APSIPAASC58517.2023.10317501

[45] A. Shah, J. I. Bangash, A. W. Khan, I. Ahmed, A. Khan, A. Khan, and A. Khan, "Comparative analysis of median filter and its variants for removal of impulse noise from gray scale images," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 3, pp. 505-519, 2022. https://doi.org/10.1016/j.jksuci.2020.03.007

[46] S. Song, Y. Chen, N. M. Cheung, and C. C. J. Kuo, "Defense against adversarial attacks with SaaK transform," 2018 [Online]. Available: https://arxiv.org/abs/1808.01785.

[47] O. Zari, J. Parra-Arnau, A. Unsal, T. Strufe, and M. Onen, "Membership inference attack against principal component analysis," in *Privacy in Statistical Databases*. Cham, Switzerland: Springer, 2022, pp. 269-282. https://doi.org/10.1007/978-3-031-13945-1_19

[48] W. Zhao, L. Shang, Y. Yu, L. Zhang, C. Wang, and J. Chen, "Personalized tag recommendation via denoising auto-encoder," *World Wide Web*, vol. 26, no. 1, pp. 95-114, 2023. https://doi.org/10.1007/s11280-021-00967-3

[49] X. Han, H. Zheng, and M. Zhou, "CARD: classification and regression diffusion models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 18100-18115, 2022.

[50] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, San Jose, CA, USA, 2010, pp. 99-108. https://doi.org/10.1145/1869790.1869807

[51] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W. Y. Ma, "Mining user similarity based on location history," in *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Irvine, CA, USA, 2008, pp. 1-10. https://doi.org/10.1145/1463434.1463477

[52] I. Kontopoulos, A. Makris, and K. Tserpes, "A real-time trajectory classification module," in *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Methods for Enriched Mobility Data: Emerging Issues and Ethical Perspectives 2023*, Hamburg, Germany, 2023, pp. 11-14. https://doi.org/10.1145/3615885.3628005

[53] A. lahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social LSTM: human trajectory prediction in crowded spaces,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 961-971. https://doi.org/10.1109/CVPR.2016.110

[54] X. Shi, Z. Gao, L. Lausen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, “Deep learning for precipitation nowcasting: a benchmark and a new model,” *Advances in Neural Information Processing Systems*, vol. 30, pp. 5617-5627, 2017.