ST-iFGSM: Enhancing Robustness of Human Mobility Signature Identification Model via Spatial-Temporal Iterative FGSM

Mingzhi Hu
Worcester Polytechnic Institute
mhu3@wpi.edu

Xin Zhang
Worcester Polytechnic Institute
xzhang17@wpi.edu

Yanhua Li
Worcester Polytechnic Institute
yli15@wpi.edu

Xun Zhou
University of Iowa
xun-zhou@uiowa.edu

Jun Luo
Lenovo Group Limited
jluo1@lenovo.com

ABSTRACT

The Human Mobility Signature Identification (HuMID) problem aims at determining whether the incoming trajectories were generated by a claimed agent from the historical movement trajectories of a set of individual human agents such as pedestrians and taxi drivers. The HuMID problem is significant, and its solutions have a wide range of real-world applications, such as criminal identification for police departments, risk assessment for auto insurance providers, driver verification in ride-sharing services, and so on. Though Deep neural networks (DNN) based HuMID models on spatial-temporal mobility fingerprint similarity demonstrate remarkable performance in effectively identifying human agents' mobility signatures, it is vulnerable to adversarial attacks as other DNN-based models. Therefore, in this paper, we propose a Spatial-Temporal iterative Fast Gradient Sign Method with L_0 regularization – ST-iFGSM – to detect the vulnerability and enhance the robustness of HuMID models. Extensive experiments with real-world taxi trajectory data demonstrate the efficiency and effectiveness of our ST-iFGSM algorithm. We tested our method on both the ST-SiameseNet and an LSTM-based HuMID classification model. It shows that ST-iFGSM can generate successful attacks to fool the HuMID models with only a few steps of attack in a small portion of the trajectories. The generated attacks can be used as augmented data to update and improve the HuMID model accuracy significantly from 47.36% to 76.18% on testing samples after the attack (86.25% on the original testing samples).

CCS CONCEPTS

• **Computing methodologies → Supervised learning by classification**; Neural networks; Anomaly detection; Adversarial learning.

KEYWORDS

Spatial-temporal data mining; Driver identification; Adversarial attack; Adversarial training

ACM Reference Format:
Mingzhi Hu, Xin Zhang, Yanhua Li, Xun Zhou, and Jun Luo. 2023. ST-iFGSM: Enhancing Robustness of Human Mobility Signature Identification Model via Spatial-Temporal Iterative FGSM. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599513

1 INTRODUCTION

CNN NEWS
Uber now lets users text with a safety agent to monitor rides
Uber
Authorized driver
Unauthorized driver
INSURANCE CLAIM FORM
DENIED
5 Reasons Why Your Car Insurance Claim Was Denied
Real insurant
Not the real insurant

Figure 1: Ride safety on public news.

Advancements in mobile sensing and information technologies have enabled a massive amount of human mobility data to be collected every day, embedding unique human decision strategies and preferences. Given a trajectory corpus consisting of historical mobility data from various human decision-makers (*e.g.*, pedestrians, taxi drivers, and gig-economy workers), and a set of new trajectories claimed to be generated by a specific agent, the HuMID problem aims to determine whether the incoming trajectory was indeed generated by the claimed agent or not. The HuMID problem is significant and leads to numerous practical applications, especially in business and public safety. For example, automatic driver identification systems for taxi and ride-sharing service providers such as Uber [46] and Lyft [24] are key applications to prevent unauthorized operations of services. The upper figure in Figure 1 shows companies like Uber have identified several cases of pretended drivers offending passengers and has enabled on-trip report from the application for passenger's safety [28, 32, 41]. When authorized and unauthorized drivers exhibit different driving habits (*e.g.*, familiar driving areas, comfortable working time, *etc.*), HuMID models can tell the differences and trigger an alarm. HuMID's role extends beyond ride-sharing services, playing a critical part in automobile insurance by verifying the identity of insured individuals through their historical driving behavior. This ensures that the vehicle was operated by a driver covered under the policy. For example, in the lower figure in Figure 1, an unauthorized driver (whose trajectories are colored blue) has a different driving pattern from the insurant (whose trajectories are colored green). HuMID models detect and report such differences to the insurance provider. Hence, HuMID models offer the potential for early detection of unauthorized driver substitutions, thereby enhancing passenger safety.

Many previous studies on HuMID are based on DNNs [4, 13, 17, 33, 37]. Though DNN-based models have demonstrated remarkable performance in classification, regression, and many other tasks, they are vulnerable to carefully crafted adversarial examples (or attacks) [12, 18, 30, 31, 44, 45]. HuMID models are no exception.

In reality, HuMID systems are not infallible. One unauthorized Uber driver can deceive the HuMID model by imitating an authorized driver's behaviors in terms of his/her seeking and serving areas. As depicted in the top right figure of Figure 1, an unauthorized Uber driver (who is familiar with the grey area) attempts to mimic the authorized driver (who is familiar with the green and blue areas) and goes to the blue area to serve. The HuMID model may incorrectly recognize both trajectories as valid and from the authorized driver. This poses a concern for automobile insurance providers who need to ensure that the insured individual was the one operating the vehicle. However, as demonstrated in the lower-right figure in Figure 1, if the insured driver (with green trajectories) always goes through a school area in his daily commute, an uninsured driver can avoid being detected by also going through the same school area. That is, altering uninsured driver's driving route to follow the red arrow instead of the original blue trajectory can successfully deceive the HuMID model. The vulnerabilities of HuMID models present a significant threat to public safety.

Several works try to mitigate the vulnerability of DNNs to adversarial attacks by leveraging the attack samples as augmented data [12, 30, 44, 45]. Goodfellow et al. [12] discovered a simple and fast way named the fast gradient sign methods (FGSM) to generate perturbations on images. Moosavi-Dezfooli et al. [30] showed that the existence of "universal perturbations" can fool a network classifier on image data. Su et al. [44] further focused on non-identifiable attacks and claimed that altering just one pixel in the image is able to fool three backbones all convolution network[43], Network in Network[22] and VGG16 network[42] on 70.97% of evaluated images. All these studies utilize adversarial training and leverage attacked samples as augmented data to enhance the robustness of DNN models. Madry et al. [27] further introduced an adversarial training approach that requires creating adversarial examples for the entire training data in each iteration. However, this design has limitations in terms of its effectiveness. Subsequently, Shafahi et al. [39] proposed free adversarial training to accelerate adversarial training. Wong et al. [48] discovered fast adversarial training, which combines FGSM adversarial training with random initialization to further accelerate computation. While many attacks and training methods have been focused on the image domain, it is crucial to consider spatial-temporal data mining and address the HuMID challenges in order to enhance the robustness of HuMID models against adversarial attacks.

In this paper, our goal is to improve the robustness of the HuMID model by generating useful adversarial trajectories for further training the model. To accomplish this, we design a Spatial-Temporal iterative Fast Gradient Sign Method with L_0 regularization – ST-iFGSM – to generate adversarial attacks on state-of-the-art (SOTA) HuMID models. ST-iFGSM can produce imperceptible attacks with minimal editions on original human mobility data, and the attacking method is able to improve HuMID model performance via clarifying decision boundaries as shown in Figure 2. In this figure, the blue and grey triangles are malicious trajectories, the black line is the original HuMID model's decision boundary, and the red line is the decision boundary of the enhanced HuMID model. This shows that the enhanced HuMID model can detect the malicious trajectories (*i.e.*, grey triangles) that try to fool it.

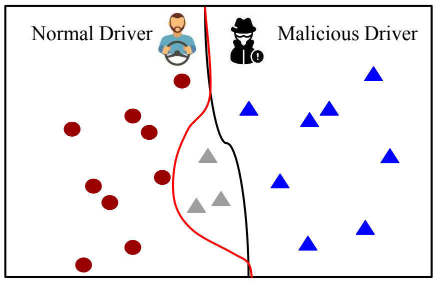

Figure 2: Decision boundary before (colored black) and after (colored red) enhancing the HuMID model with adversarial attacks (grey triangles).

Our contributions are summarized as follows:

- To the best of our knowledge, we are the first to propose to improve the HuMID model robustness via generating adversarial attacks from human-generated spatial-temporal data. (See Section 2).
- We give definitions of imperceptible attacks in the spatial-temporal domain, and develop a novel L_0-constrained iterative FGSM technique, *i.e.*, ST-iFGSM, to generate imperceptible spatial-temporal attacks with a limited number of trajectory editions. ST-iFGSM generated attacks can successfully fool SOTA HuMID models (See Section 3).
- We validate our framework using real-world human-generated spatial-temporal data and two different SOTA HuMID models, *i.e.*, ST-SiameseNet [37] and a multi-class HuMID model using LSTM. Extensive results show a drastic accuracy drop on ST-iFGSM generated imperceptible perturbations, and a significant performance increase when using ST-iFGSM-generated attacks to do the adversarial training compared with baselines (See Section 4).

We have released our code and data publicly at our Github page[1].

[1]ST-iFGSM project page: https://github.com/mhu3/ST-Siamese-Attack

2 OVERVIEW

In this section, we define the spatial-temporal HuMID adversarial attack problem and highlight the research challenges. For brevity, we present a table of notations in Table 1.

Table 1: Notations.

Notations	Descriptions
$p = \langle lat, lng, t\rangle$	GPS record.
$\tau = \{a, \langle p_1, p_2, ..., p_n\rangle\}$	Trajectory.
$\mathcal{T} = \{\tau\}, \mathcal{T}^{adv} = \{\tau'\}$	Trajectory set and related adversaries.
δ	Perturbations added on the trajectory.
f, f^{enh}	Original and enhanced HuMID model.
$\hat{y} = f(\tau, \tau')$	HuMID model prediction.
$\ell(f(\tau, \tau'), y)$	Loss for prediction on adversaries.
θ	Parameters of HuMID model f.

2.1 Human-Generated Spatial-Temporal Data for HuMID Attacks

Thanks to the rapid development of mobile sensing technologies, GPS devices are everywhere on *e.g.*, vehicles, smartphones, smart watches, *etc.* Spatial-temporal data in the form of GPS traces are easily accessible from these devices.

Definition 1: Human-generated spatial-temporal trajectory τ**.** Each GPS record p consists of a location in latitude lat, longitude lng, and a time stamp t, *i.e.*, $p = \langle lat, lng, t\rangle$. A trajectory τ is a sequence of GPS records with a label of the agent a who generated the data denoted as $\tau = \{a, \langle p_1, p_2, \cdots, p_n\rangle\}$. The set of trajectories is denoted as $\mathcal{T}$ containing trajectories from m human agents $\mathcal{T} = \{\tau_{a_1}, \cdots, \tau_{a_m}\}$ where each agent a_i might have multiple trajectories. Moreover, we denote $\mathcal{T}^{adv}$ as the generated adversarial attacks from $\mathcal{T}$. For each $\tau \in \mathcal{T}$, there is its counterpart adversarial trajectory τ' from the same agent.

Definition 2: Spatial-temporal HuMID model f**.** A spatial-temporal HuMID model f is the target model to be attacked and enhanced. For brevity, we say HuMID models to indicate the spatial-temporal ones. Given a set of human-generated spatial-temporal trajectories $\mathcal{T}$ from agents $a_1, \cdots, a_m$, a HuMID model f verifies if an incoming/test trajectory $\tau^c_{a_i}$ claimed to be generated by agent a_i indeed matches agent a_i's behavior. For example, given a pair of trajectories τ_{a_i} from historical human trajectory dataset $\mathcal{T}$ and $\tau^c_{a_i}$ claimed to be from agent a_i, the HuMID model f evaluates the similarity between the two input trajectories as $f(\tau_{a_i}, \tau^c_{a_i})$, and returns 1 if the claimed trajectory $\tau^c_{a_i}$ resembles agent a_i's behavior, and returns 0 if otherwise. We use $\hat{y}$ to denote the prediction from the HuMID model f. Further, we denote f^{enh} as the enhanced HuMID model when training with the adversarial attacks $\mathcal{T}^{adv}$. Therefore, if we write the loss function applied in the HuMID model as ℓ, the loss between a pair of true trajectory τ and its adversary τ' is $\ell(f(\tau, \tau'), y)$ where y is the true label for agent identity. We denote δ as the perturbation added on the true trajectory τ. Here we list the binary classification case as an example. However, we also target other HuMID models that involve multi-class classification or handle trajectory data differently, without loss of generality.

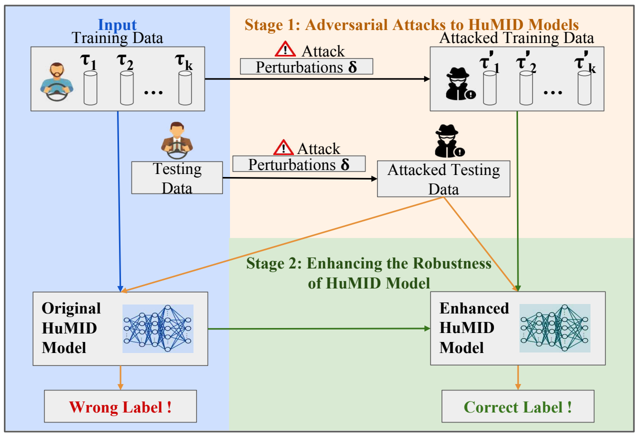
Input
Training Data
τ_1 τ_2 … τ_k
Stage 1: Adversarial Attacks to HuMID Models
Attack Perturbations δ
Attacked Training Data
τ'_1 τ'_2 … τ'_k
Testing Data
Attack Perturbations δ
Attacked Testing Data
Stage 2: Enhancing the Robustness of HuMID Model
Original HuMID Model
Enhanced HuMID Model
Wrong Label !
Correct Label !

Figure 3: Enhancing the robustness of Spatial-temporal HuMID model solution framework.

2.2 Spatial-Temporal HuMID Adversarial Attack Problem

Problem definition. Given a set of human-generated spatial-temporal trajectories $\mathcal{T}$ and a trained HuMID model f on this dataset, we aim to generate a set of adversarial trajectories $\mathcal{T}^{adv}$ that is able to *i.)* lower the classification accuracy of the HuMID model f on adversarial trajectories $\mathcal{T}^{adv}$, and *ii.)* increase the enhanced HuMID model f^{adv} accuracy against the attacking strategy after adversarial training with attacked data $\mathcal{T}^{adv}$.

Challenges. The proposed spatial-temporal HuMID adversarial attack problem is challenging in two aspects: (**C1**) Considering the complicated spatial-temporal scenario and human decisions, how to design and generate an invisible attack that can fool HuMID models towards making wrong decisions with minimal attacks in terms of edition range and size? (**C2**) How to use the generated adversarial spatial-temporal data to enhance the robustness of the HuMID model to detect abnormal data and make the correct decisions?

2.3 Solution Framework

To solve the spatial-temporal HuMID adversarial attack problem, we propose the solution framework in Figure 3. It takes the human mobility dataset and a target HuMID model as inputs and contains two processing stages: *Stage 1.* iteratively generating and selecting adversarial attack samples which could fool the target HuMID model (See Section 3.1), and *Stage 2.* training the HuMID model with the adversarial attack to improve the model robustness (See Section 3.2).

3 METHODOLOGY

In this section, we solve the spatial-temporal HuMID adversarial attack problem by defining spatial-temporal adversarial attacks on the HuMID models, designing a gradient-based approach to identify perturbations on the original dataset to fool a target HuMID model towards wrong predictions, and developing an iterative approach to eliminate redundant and unnecessary perturbations for imperceptible attacks (*i.e.*, tackling the challenge **C1**, See Section 3.1).

We further demonstrate our solutions to enhance the robustness of HuMID models (*i.e.*, tackling the challenge **C2**, See Section 3.2).

3.1 Stage 1: Adversarial Attacks to HuMID Models

In this part, we show how adversarial attacks perform on the spatial-temporal HuMID data, design and generate imperceptible attacks that can fool HuMID models towards making wrong decisions. We first show some SOTA adversarial attack works below.

3.1.1 Limitations of the state-of-the-art works. Most of the adversarial attack approaches focus on the image domain [12, 18, 30, 44, 45]. Among them, FGSM [12] is a one-step algorithm that perturbs images through a single, large step, increasing the loss of the target model. In spatial-temporal cases, suppose τ denotes a trajectory to be perturbed and τ' is the perturbed trajectory, FGSM was designed based on the L_∞-norm to maximize the loss function, *i.e.*,

$$\max_{\tau'} \ell(f(\tau'), y)$$
$$s.t. \|\tau - \tau'\|_\infty \leq \epsilon.$$

To accelerate the attack generation speed, Goodfellow et al. [12] computes the sign of the gradient using the *sign* function instead of calculating the gradient values of the loss function, where the attacks are generated as below:

$$\tau' = \tau + \epsilon \cdot sign\,[\nabla_\tau \ell(f(\tau), y)].$$

The $\ell(f(\tau'), y)$ is the loss function used to classify τ' with its true label y, and ϵ denotes the noise/perturbation level. Here, *sign* is the sign function that returns 1 for positive gradients and -1 otherwise.

However, in HuMID problems, directly applying FGSM faces difficulties, *i.e.*, *i).* perturbations with the L_∞-norm constraint are likely to result in perturbed GPS points off the road or inaccessible, and *ii).* perturbations on the temporal features likely produce unrealistic trajectories with dramatic spatial changes that are obvious to detect.

3.1.2 HuMID attacks via spatial-temporal perturbations. In FGSM, there is no constraint for the pixels to be changed. However, to fit in the spatial-temporal HuMID problem and generate realistic attacks, we define the objective for a HuMID attack on a trajectory τ as below:

$$\max_{\tau'} \ell(f(\tau'), y)$$
$$s.t. \|\tau - \tau'\|_\infty \leq \epsilon, \ \|\tau - \tau'\|_0 \leq \eta. \tag{1}$$

Here, each generated adversary τ' aims to mislead the HuMID model into making incorrect decisions that deviate from the true label y. Meanwhile, in an adversary, the number of GPS point edits is limited by the L_0-norm to be less than or equal to η. This allows the adversary to be imperceptible as it follows most of the original trajectory's patterns except for the η modified steps in the trajectory. In addition, the change distance in terms of the L_∞-norm is constrained to be at most ϵ in order to make reasonable perturbations on the road close to the original GPS points. For example, in a taxi trajectory, the objective in Eq. (1) says that an attacker tries to mislead the HuMID model by maximizing the model loss. At the same time, the attack wants to have a maximum of η GPS points to be edited given the original trajectory, where each edit should be within ϵ km from the original GPS point. However, since the length of each trajectory is different, it is difficult to find a fixed η to solve the problem. Therefore, in practice, we determine the minimum number of steps (*i.e.*, GPS points) in trajectories to be attacked to promise attack success. The objective function thus becomes:

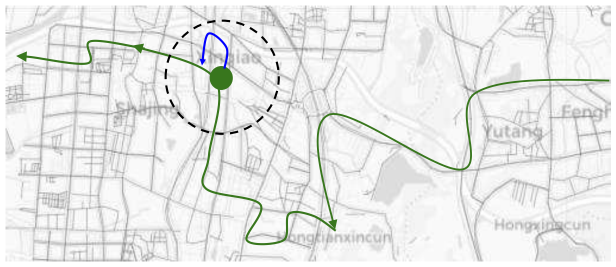

Figure 4: An illustration on one attacked trajectory. The green line with the arrow is the original trajectory, and the blue arrow demonstrates where the attack happens.

$$\min_{\tau'} \|\tau - \tau'\|_0$$
$$s.t. \|\tau - \tau'\|_\infty \leq \epsilon, \ f(\tau') \neq y. \tag{2}$$

Eq. (2) says that we minimize the number of steps to be attacked when the editions are close to original GPS points and the HuMID model is successfully fooled towards a random wrong output. [2]

An illustrative example. Figure 4 shows how the GPS records in a trajectory are attacked. The green line with the arrow shows one real trajectory, where the attack happens following the blue arrow. At a GPS point (*i.e.*, the green circle), the dashed black circle demonstrates the edition spatial range constrained by the L_∞-norm to be below ϵ km. Therefore, the blue attack happens within ϵ km from the original GPS point. In this figure, only one GPS point edition is allowed, *i.e.*, $\eta = 1$. Note that we do not alter the temporal information as it would lead to a drastic change in trajectories and uncommon back-and-forth behaviors in human mobility data.

Based on the spatial-temporal adversarial attacks on HuMID models, we develop an FGSM-based approach to generate adversarial perturbations with iterative searching and L_0-guided redundancy elimination.

3.1.3 ST-iFGSM Attack with L_0 Constraint. Now, we formally introduce our spatial-temporal iterative FGSM, in short, ST-iFGSM, based on FGSM. ST-iFGSM has two processing steps, *i.e.*, applying an iterative approach to enforce attack success, and utilizing an elimination strategy to satisfy the L_0-norm constraint.

Step 1: Iterative FGSM for a successful attack. Though fast in implementation with a single-step attack, FGSM alone does not always produce a successful attack that is able to fool a target model [18]. Therefore, we follow the iterative FGSM (I-FGSM) [18] to apply an iterative approach, that takes a series of small attack steps and adjusts the attack directions constantly to fulfill an attack task. In other words, to generate a successful attack and force the HuMID model to give a wrong prediction, we repeatedly update

[2]Note that we do not consider attacking a trajectory by changing its lengths. This is because changing trajectory length likely leads to a large trajectory mismatch in L_∞-norm, and we follow adversarial attack works to maintain original data dimension [12, 18, 30, 44, 45].

Algorithm 1 Adversarial Attack with ST-iFGSM

```
Require: A target trajectory $\tau$; a HuMID model $f$ with loss function $\ell$; a "to-be-attacked" GPS indices vector $\boldsymbol{v}$; perturbation bound $\epsilon$ and $\eta$, learning rate $\alpha$.
Ensure: An adversarial trajectory $\tau'$.
1: Initialize the "to-be-attacked" attack vector $\boldsymbol{v} = (1, 1, \cdots, 1)$ of $n$ dimension for GPS points in the target trajectory $\tau$, and initialize $\tau' = \tau$.
2: repeat
3:     for $j = 0, 1, \cdots, N-1$ do
4:         Calculate the perturbation $\delta$ based on Eq. (3), i.e., $\delta = clip(\alpha \cdot sign[\nabla_{\tau'}\ell(f(\tau'), y)], -\epsilon, \epsilon)$.
5:         Assign perturbations based on the perturbation $\delta$ and the perturbation vector $v$, i.e., $\tau' = \tau' + \delta \odot \boldsymbol{v}$.
6:     end for
7:     Record the perturbations after $N$ loops of FGSM, i.e., $\Delta = \tau' - \tau$.
8:     Calculate the gradient for the attack $\boldsymbol{g} = \nabla_{\tau'}\ell(f(\tau'), y)$.
9:     Select the indices $i = \arg\min_i |g_i|$ that contribute least to the attack, and update the $i$th values in the vector $\boldsymbol{v}$ as 0.
10:    Update the attacked trajectory as $\tau' = \tau + \Delta \odot \boldsymbol{v}$.
11: until $f(\tau') = f(\tau)$.
12: Return the previous $\tau'$ which satisfies $f(\tau') \neq f(\tau)$ with minimized attacked steps.
```

our adversary sample τ' for N iterations following:

$$\begin{aligned} &\tau'_0 = \tau, \\ &\tau'_{j+1} = clip\left[\tau'_j + sign[\nabla_{\tau'_j}\ell(f(\tau'_j), y)]\right], \\ &\text{for } j = 0, 1, \cdots, N-1. \end{aligned} \tag{3}$$

Specially, a one-time attack indicates the case when only one iteration happens, *i.e.*, $N = 1$. In the equation, the *clip* term represents the clipping function to enforce the L_∞-norm constraint. It performs a GPS-level clipping of the trajectory so that all perturbed GPS points are within ϵ distances from the original GPS points.

Step 2: Redundancy elimination for L_0-norm constraint. To minimize the number of perturbed GPS points in a trajectory, *i.e.*, to enforce the L_0-norm constraint, we compute the gradient of the loss in Eq. (3) evaluated at the adversarial instance τ'_N, *i.e.*, $\boldsymbol{g} = \nabla\ell(f(\tau'_N), y)$ after we get the Nth update from the previous iterative FGSM step. Considering that there are n GPS points in an attacked trajectory, *i.e.*, $\tau'_N = \{a, \langle p'_1, p'_2, \cdots, p'_n\rangle\}$, the gradient $\boldsymbol{g}$ is a vector of n dimensions, *i.e.*, $\boldsymbol{g} = (g_1, g_2, \cdots, g_n)$ with each dimension evaluated at the ith GPS point in the trajectory with $i = 1, 2, \cdots, n$. Here each g_i is a scalar. We then select some perturbed GPS points that contribute the least to the attack with the smallest gradient, *i.e.*, $\arg\min_{i=1,\cdots,n}|g_i|$ and remove these i's from the "to-be-attacked" GPS set. Therefore, the GPS points that contribute the least to the attack will be left unchanged. To remove all unnecessary perturbations, we repeat this process several times until the ST-iFGSM perturbed trajectory fails to attack the HuMID model f.

ST-iFGSM Algorithm. We present our detailed algorithm in Algorithm 1. The input consists of a target trajectory τ to be attacked which is a sequence of GPS points, a HuMID model f, the perturbation L_∞-norm bound ϵ, the perturbation L_0-norm bound η, and a learning rate α. We also maintain a vector $\boldsymbol{v}$ that has n dimensions representing n GPS points in the trajectory τ. The ith dimension in the vector $\boldsymbol{v}$ corresponds to the ith GPS point, and it bears a binary value with 1 indicating a GPS point that should be perturbed, and 0 otherwise, *i.e.*, $\boldsymbol{v} \in \{0, 1\}^n$. The vector $\boldsymbol{v}$ is initialized as $(1, 1, \cdots, 1)$ to allow attacks at all GPS points. The algorithm works by iteratively updating perturbations δ to the input trajectory τ until ST-iFGSM fails to attack the HuMID model, *i.e.*, $f(\tau') = f(\tau)$ (line 2-11). Specifically, in each iteration, we iterate over N loops of FGSM to generate an accumulated perturbation Δ (line 7). In each iterative FGSM loop, we first calculate the perturbation specific to the loop based on Eq. (3) (line 4) and update the attack trajectory τ' accordingly for the next FGSM loop. Note that we enforce the L_∞-norm constraint via clipping (line 4). After the iterative FGSM stage with the accumulated perturbation Δ, we start selecting GPS points to remain unchanged (line 8-10). For this, we first calculate the gradients of loss $\boldsymbol{g}$ based on the current attack τ' (line 8). We calculate the gradients and select the perturbed GPS points that contribute the least to the attack, *i.e.*, gradient values with the smallest absolute values, and update their values in the attack vector $\boldsymbol{v}$ to be zero (line 9). We get the perturbation at this step by updating the attack as $\tau' = \tau + \Delta \odot \boldsymbol{v}$ (line 10). Until $f(\tau') = f(\tau)$, we return the previous τ' which satisfies $f(\tau') \neq f(\tau)$ with minimized attacked steps (line 11-12).

3.2 Stage 2: Enhancing the Robustness of HuMID model

To enhance the HuMID model and protect it against adversarial attacks, we introduce the ST-FGSM fast adversarial training with the proposed L_0-constrained ST-iFGSM attack (when iteration number is 1), based on the fast adversarial training [48].

ST-FGSM fast adversarial training. Unlike projected gradient descent (PGD) based training [27] and free adversarial training [39] which need more computational cost, our ST-FGSM fast adversarial training follows fast adversarial training [48], utilizing random initialization and performing a one-time HuMID model update on perturbation. Fast adversarial training [48] has been shown to be as effective as PGD-based methods but much more efficient. Besides, we combine fast adversarial training with L_0-norm constraint ST-FGSM attack (*i.e.*, iteration equals 1). Before finding the sample with the least L_0-norm, successful adversarial samples would be generated each time we minimize the number of steps to be attacked. We leverage all these samples to update the parameter θ of the HuMID model, which improves the robustness of the model.

Algorithm 2 shows how our ST-FGSM fast adversarial training works, the input includes the training data $\mathcal{T}$, the number of training epochs E, the HuMID model f parameterized by θ with loss function ℓ, and learning rate α_θ and α_δ. In each epoch e, we get $|\mathcal{T}|/B$ batches of data. For each batch B, trajectories are sampled from $\mathcal{T}$ and are denoted as $\mathcal{T}_{e,b}$ (line 4). For each $\tau \in \mathcal{T}_{e,b}$, we initialize related perturbation from a uniform distribution with maximum noise level ϵ, *i.e.*, $\delta = Uniform(-\epsilon, \epsilon)$ and an attack vector $\boldsymbol{v}$ (line 5-6). With these trajectories, we calculate the perturbations based on Eq. (3) to generate the attacked samples τ' (line 7-9). We then use the attacked samples to update the model parameter based on gradient descent by minimizing the loss $\mathbb{E}_{\mathcal{T}'_{e,b}}[\ell(f(\tau'), y)]$ (line 10).

Algorithm 2 Adversarial Training with ST-iFGSM Attacks

```
Require: Training data $\mathcal{T}$; a HuMID model $f_\theta$ to be enhanced with loss function $\ell$ and parameter $\theta$; perturbation bound $\epsilon$ and $\eta$; learning rate $\alpha_\delta$, learning rate $\alpha$ and batch size $B$.
Ensure: An enhanced HuMID model $f^{enh}_{\theta'}$.
1:  Initialize the HuMID model's parameter as $\theta_{0,0} = \theta$.
2:  for $e = 1, 2, \cdots, E$ do
3:    for $b = 1, 2, \cdots, |\mathcal{T}|/B$ do
4:      Sample a batch of $B$ samples from $\mathcal{T}$ and denote as $\mathcal{T}_{e,b}$.
5:      Initialize $\delta = Uniform(-\epsilon, \epsilon)$ and the "to-be-attacked" attack vector $\boldsymbol{v} = (1, 1, \cdots, 1)$ of $n$ dimension for GPS points for each sample $\tau$ in $\mathcal{T}_{e,b}$.
6:      Impose perturbation $\delta$ and generate the perturbation $\tau' = \tau + \delta \odot \boldsymbol{v}$ for each $\tau \in \mathcal{T}_{e,b}$.
7:      repeat
8:        For each $\tau'$, update the perturbation $\delta$ based on Eq. (3), i.e., $\delta = clip(\delta + \alpha_\delta \cdot sign(\nabla_{\tau'} \ell(f_{\theta_{e,b-1}}(\tau'), y), -\epsilon, \epsilon)$.
9:        Assign perturbations based on $\delta$ and the perturbation vector $v$, i.e., $\tau' = \tau + \delta \odot \boldsymbol{v}$.
10:       Update the HuMID model's parameter based on samples in $\mathcal{T}'_{e,b}$, i.e.,
            $\theta_{e,b} = \theta_{e,b-1} - \alpha_\theta \nabla_{\theta_{e,b-1}} \mathbb{E}_{\mathcal{T}'_{e,b}} [\ell(f_{\theta_{e,b-1}}(\tau'), y)]$.
11:       For each $\tau'$, calculate the corresponding gradient $\boldsymbol{g} = \nabla_{\tau'} \ell(f_{\theta_{e,b}}(\tau'), y)$.
12:       Select the indices $i = \arg\min_i |g_i|$ that contribute least to the attack, and update the $i$th values in the vector $\boldsymbol{v}$ as 0.
13:     until All $B$ samples are correctly predicted, i.e. satisfy $f(\tau') = f(\tau)$.
14:   end for
15: end for
16: Return the enhanced HuMID model $f^{enh}_{\theta'}$ with updated parameter $\theta' = \theta_{E,|\mathcal{T}|/B}$.
```

The loss function calculates the HuMID model's loss on the original label and prediction on attacked data. Then for each sample, we start selecting GPS points to remain unchanged until all samples in B can not be successfully attacked (line 11-12). After E epochs, an enhanced model $f^{enh}_{\theta'}$ is obtained.

4 EXPERIMENTAL EVALUATION

In this section, we evaluate the performances of ST-iFGSM with L_0 using the taxi GPS dataset collected in Shenzhen, China in July 2016, and one SOTA HuMID model *i.e.*, ST-SiameseNet [37] and a multi-class classification model to attack against. We compare with other baselines to demonstrate that *i).* Our proposed ST-iFGSM is able to generate useful attacks to fool the HuMID model and decrease its evaluation accuracy, *ii).* ST-iFGSM generated attacks are imperceptible with minimal edits to the original human mobility data, and *iii).* adversarial training enhances the robustness of the HuMID model and defend against attacks generated from ST-iFGSM. The experiment setups and results are described in detail below.

4.1 Data Description and Preparation

Our work takes two urban data sources as input, including (1) taxi GPS trajectory data and (2) road map data. For consistency, both datasets were collected in Shenzhen, China in July 2016.

Taxi trajectory data includes GPS data collected in Shenzhen, China during July 2016 from taxis. There are 17,877 unique taxis in the dataset each equipped with a GPS unit, and each GPS unit produces one GPS point in roughly every 40 seconds. Each day, around 51 million GPS records are gathered in total, and each record has five essential data fields: latitude, longitude, unique taxi identifier, time stamp, as well as a passenger indicator. The passenger indicator demonstrates if a passenger is aboard or not with 1 representing passenger onboard, and 0 otherwise.

Road map data of Shenzhen covers the area defined between 22.44° to 22.87° in latitude and 113.75° to 114.63° in longitude. The data is from OpenStreetMap [34] and has 21,000 roads with six levels.

Data preprocessing We employ the taxi trajectory data and the road map data to extract the taxi driver passenger seeking and serving trajectories for training the ST-SiameseNet [37] and LSTM-based classification model.

Map gridding and time quantization. To mitigate the disclosure of sensitive information, we employ data anonymization techniques by discretizing the trajectories, thereby safeguarding the identities of individuals and minimizing the potential for re-identification. Specifically, we use a standard quantization trick and partition the Shenzhen area into grid cells with equal side-length 0.01° in latitude and longitude [20, 21, 37]. Figure 5 shows the gridding result in Shenzhen, China, where grids colored in dark grey are inaccessible on the sea.

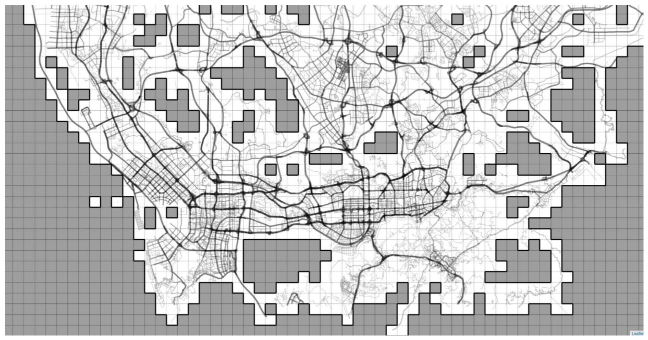

Figure 5: Map gridding demonstration.

Eliminating cells in the ocean, those unreachable from the city, and other irrelevant cells give a total of 1,934 valid cells. We further divide each day into five-minute intervals for a total of 288 intervals per day, denoted as $I = \{\tilde{t}_k\}$, with $1 \leqslant k \leqslant 288$. A spatial-temporal region r is a pair of a grid cell g and a time interval $\tilde{t}_k$. Each GPS record is $p = \langle lat, lng, t \rangle$ and can be represented as an aggregated state $s = \langle g, \tilde{t}_k \rangle$. A trajectory of agent a then can be mapped to sequences of spatial-temporal regions, $\tau = \{a, \langle r_1, r_2, ..., r_n \rangle\}$.

Transit Modes Extraction. Different transit modes can show different patterns of driving behavior. Seeking and serving trajectories in the taxi driving scenario reflect different characteristics of each taxi driver. Thus, we split the trajectories into seeking τ^s and driving/serving trajectories τ^d based on the status of the vehicle and whether there are passengers on board.

4.2 Experiment Setups

Training preparation. To identify different human agents' mobility signatures on our dataset, we train ST-SiameseNet [37] and a multi-class classification HuMID model, which verifies human mobility identity respectively by implementing ST-SiameseNet and an LSTM multi-class classification model respectively.
For ST-SiamaseNet, we randomly select 500 drivers from July 4th to July 15th (10 workdays) and use their 5 seeking and 5 serving trajectories for training and validation on the first 8 days. We randomly select a pair of seeking and driving trajectories from driver 1 and driver 2, *i.e.*, $(\tau^{s,1}, \tau^{d,1})$ and $(\tau^{s,2}, \tau^{d,2})$. We randomly generate 500,000 pairs of trajectories, half of which are from the same driver and the other half from different drivers. ST-SiameseNet learns driving behavior from seeking and serving trajectories and shows if the pair of trajectories are from the same driver or not.
For multi-class classification model, we select 4 drivers from July to December in 2016 from the same dataset and use their 5 seeking and 5 serving trajectories *i.e.*, $(\tau^{s,1}, \tau^{d,1})$ for modeling. There are 111 days of trajectories for these four drivers, which we divided into 90, 5, and 16 days for the training dataset, validation dataset, and testing dataset, respectively.
Testing preparation. We apply our model ST-iFGSM with L_0-norm constraint to attack the trajectories in the testing data and lead the ST-SiameseNet and multi-class classification model towards wrong predictions. Fig. 6 demonstrates the performance of ST-SiameseNet [37] to identify if the trajectories come from the same driver or not. Giving a pair of trajectories from two different authorized drivers, ST-SiameseNet will give the prediction that the drivers are different. However, if we attack driver 2's trajectories to mimic driver 1's trajectories, then ST-SiameseNet will give the prediction that the trajectories are coming from the same driver. It demonstrates after modification or attack on the original human trajectory, the ST-SiameseNet could be fooled and give the opposite prediction.
For ST-SiameseNet, to test how our ST-iFGSM works, we split the testing data into two parts. Specifically, one part of the data contains trajectories from seen drivers on their unseen days (the 9th and 10th days), and another includes trajectories from new drivers who are previously unseen from the training pool on unseen days (the 9th and 10th days). In the unseen driver part of the testing data, there are an extra 197 unique drivers besides those in the training data. In total in each testing data part, there are 5,000 pairs of trajectories from the same driver and 5,000 pairs of trajectories from different drivers.
For multi-class classification model, the testing dataset contains 16 days of trajectories of the same 4 drivers in the training dataset.
Evaluation metrics. We evaluate the performance of ST-iFGSM based on the prediction accuracy from the ST-SiameseNet [37] and LSTM multi-class classification model, *i.e.*, the ratio between the number of correctly predicted samples over the number of all testing samples.

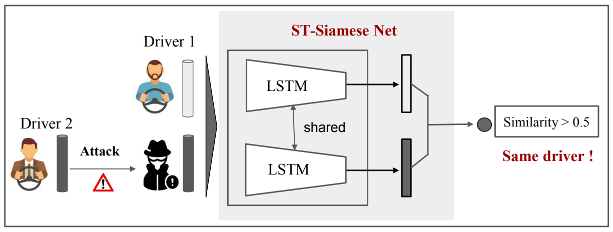

Figure 6: A malicious driver fools the ST-SiameseNet to show a wrong result.

4.3 Baseline Algorithms

We compare our ST-iFGSM with baselines and analyze the generalization of our approach to show that ST-iFGSM is able to generate constrained and imperceptible attacks faster. To generate imperceptible attacks on trajectories, we constrain that each attacked unit (*i.e.*, a spatial-temporal grid g) in the trajectory only changes to a neighboring grid. Note that we do not add perturbations on the time stamps t in the trajectories. In order to mimic the trajectories that could happen in real life (*e.g.*, unauthorized service and identity theft), all approaches only attack the seeking trajectories of driver 2 ($\tau_{s,2}$) in each pair input of for ST-siameseNet. For LSTM multi-class classification model, we only attack seeking trajectories of the drivers and allow modification on the original steps to be two grid cells next to them. Below, we detail the baselines we used:

- **Fast Gradient Sign Method (FGSM) [12]** performs a one-step perturbation on images to increase the classifier's loss with the L_∞-norm. It is a white box attack that has complete access to the target model.
- **Iterative Fast Gradient Sign Method (I-FGSM) [18]** is an extension of FGSM that repeatedly applies FGSM on pixel-level attacks on an image. The resulting adversaries are clipped to limit the maximum perturbation for each pixel on an image.
- **Iterative Fast Gradient Method (I-FGM) [29]** ignores the *sign* function in I-FGSM and utilizes the L_2-norm to normalise gradients to iteratively generate an attack. Despite the fact that the sign operator can be viewed as a normalization process, it also changes the most effective direction.
- **Carlini & Wagner Attack with L_0-norm constraint (C&W L_0) [2]** is a SOTA attack approach. It generates a set of attacks that compute norm-restricted additive perturbations[35]. C&W L_0 represents the C&W approach with the L_0-norm constraint. To avoid local optimization, they use multiple starting-point gradient descents in the ball of the adversarial range. C&W generated adversarial examples have been shown to have a low distortion in the L_0-norm, effectively minimizing of changed pixels.

4.4 Evaluation Results

In this section, we first present the evaluation results of ST-iFGSM with other baselines in terms of the attack performance, *i.e.*, the ratio between the number of correctly predicted samples of demonstrations over the number of all testing samples after the attack. In the second part, we assess the performance of our proposed ST-FGSM adversarial training in enhancing the robustness of the HuMID model, *i.e.* the ratio of correctly predicted samples over all testing samples after the attack.

Table 2: Attack methods comparison on ST-SiameseNet. For all evaluation values, the lower the better.

Methods	Acc. on Adv. Seen	Acc. on Adv. Unseen	Time on Seen(s)	Time on Unseen(s)	#Attacks
ST-iFGSM L_0 (i=2)	46.67%	34.85%	122.68	110.08	**35**
ST-iFGSM L_0 (i=1)	47.66%	36.25%	107.97	112.97	42
FGSM	57.59%	47.93%	3.85	2.47	118
I-FGSM	45.49%	33.12%	84.04	82.12	118
I-FGM	45.68%	33.56%	84.96	84.40	117
C&W L_0	46.74%	34.74%	198.03	199.60	51

Table 3: Robustness comparison on ST-SiameseNet. All values above 70% indicate good models.

Methods	Acc. on Seen	Acc. on Unseen Unseen
ST-iFGSM (i=2) L_0	76.15%	70.98%
ST-iFGSM (i=1) L_0	76.18%	71.02%
FGSM	76.39%	71.37%
I-FGSM	76.16%	70.98%
I-FGM	76.43%	71.30%
C&W L_0	80.96%	75.93%

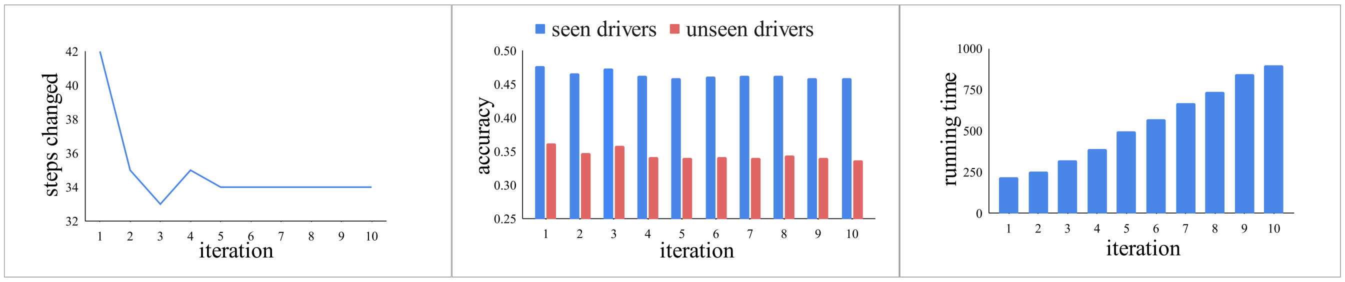

Figure 7: Ablation study on how iteration affects the ST-iFGSM performance.

4.4.1 *Results with ST-Siamese Model.*

Adversarial Attack Performance. To compare our method with baseline approaches, we measure the accuracy of the ST-SiameseNet on adversarial samples and the number of edits on test data in Table 2. A lower accuracy with fewer number attacks (*i.e.*, grid-level) on both test datasets indicates a stronger attacking method that generates more powerful attacks. Before the attacks, the prediction accuracy of ST-SiameseNet on the seen and unseen testing datasets are 86.85% and 82.92% respectively. The table shows that our ST-iFGSM with L_0-norm (iteration = 2, learning rate α = 0.01) achieves the best performance since all these methods have similar attacking accuracy, but our ST-iFGSM with L_0-norm attacks the fewest steps in the trajectories attacked. Specifically, it attacked only an average of 35 steps on driver 2's five seeking trajectories. On the other hand, while FGSM (iteration = 1, learning rate = 0.02) is the most efficient approach with the shortest attack time, as it is a one-time attack, the attack may fail, resulting in the lowest successful attacking rate. I-FGSM (iteration = 50, learning rate = 0.001) and I-FGM (iteration = 50, learning rate = 0.05) are two efficient methods for attacking our HuMID model, but they never constrain the steps in the trajectories being attacked, so the number of steps being attacked is large, which could not easily happen in the real-life attack. C&W methods with L_0-norm constraint (iteration = 2, learning rate α = 0.01) show a similar attack successful rate as our method. However, its method for finding the constant in the objective function [2] is more computationally expensive, furthermore, its method for reducing the steps to attack is not as powerful as ours, as it attacks 51 steps in driver 2's seeking trajectories.

Ablation Studies. We have also learned how iteration affects the results of our ST-iFGSM results in Figure 7. In our case, increasing iteration has little effect on attacking accuracy on both seen and unseen testing datasets (We keep the product of learning rate and iteration the same). When increasing iterations from 1 to 2, the steps in the trajectories decrease, but the decrease is not significant with further iterations. Consequently, adding more iterations leads to increased running time. In our case, 2 iterations yield the best results for our ST-iFGSM to attack ST-SiameseNet.

Adversarial Training Performance. To enhance the robustness of HuMID model and make it immune to ST-iFGSM, we apply fast adversarial training [48] on the training dataset and use it to do adversarial training (2 epochs). Table 3 shows we improve the model robustness with the ST-FGSM fast adversarial training against the different attack methods. For instance, the HuMID model improves prediction accuracy on seen testing data attacked by l_0-constrained ST-iFGSM (i=2) from 46.67% to 76.15%. However, after adversarial training, ST-SiameseNet's prediction accuracy slightly decreases on the original dataset. It achieves 80.56% accuracy on the seen testing dataset and 75.93% on the unseen dataset, potentially due to added noise during adversarial training.

4.4.2 *Results with LSTM multi-class classification model.*

Adversarial Attack Performance. Table 4 shows the accuracy on the testing dataset after attacking, the original accuracy on the testing dataset is 89.06%. After attacking, our ST-iFGSM(iteration = 2, learning rate = 0.01) attacks the least steps of the seeking trajectories of driver 2 on average, it only attacks 40 steps. FGSM (iteration = 1, learning rate = 0.05), I-FGSM (iteration = 20, learning rate = 0.01), and I-FGM (iteration = 50, learning rate = 0.05) are the three most efficient methods for the attack, but they attack 145, 143, and 130 steps. C&W l_0 (iteration = 8, learning rate = 0.01) attacks 57 steps, and it is also slower than our method.

Table 4: Attack comparison for LSTM multi-class classification model. For all evaluation values, the lower the better.

Methods	Acc. on Adv.	Time(s)	#Attacks
ST-iFGSM L_0 (i=2)	59.38%	10.58	**40**
ST-iFGSM L_0 (i=1)	59.38%	8.25	**41**
FGSM	60.94%	0.32	145
I-FGSM	59.38%	8.93	143
I-FGM	65.62%	8.90	130
C&W L_0	60.94%	79.20	57

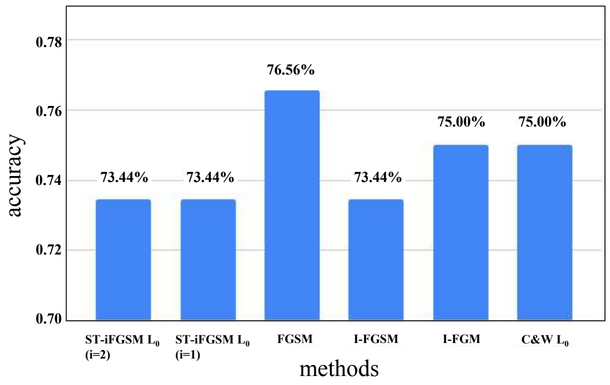

Figure 8: Enhanced LSTM multi-class classification model performance with different attack methods.

Adversarial Training performance. Fig. 8 shows we improve the robust performance of the LSTM-based classification model by ST-FGSM fast adversarial training (2 epochs). The enhanced HuMID model shows that the accuracy of the attacked testing data generated by different methods is around 75%, indicating that the proposed method could improve its robustness against attacks. Moreover, its prediction accuracy on the original testing dataset drops only from 89.06% to 85.94%.

5 RELATED WORK

HuMID has been extensively studied in recent years due to the rise of the ride-sharing business model and urban intelligence[5, 20, 37, 50, 53]. However, our research represents the first attempt known to us in improving the robustness of the HuMID model through adversarial attacks on spatial-temporal data. Related work is summarized below.

Urban computing. Urban computing is a broad research field that combines urban sensing, data management, and data analytics on urban data [16, 23, 25, 36, 52]. A group of works focuses on taxi operation and passenger-seeking problems [10, 11, 26, 38, 51]. They seek to identify the best actionable solution for improving the performance of taxi drivers. Besides, a group considers the benefits of passengers. Ren et al. [3, 17, 37] focus on driver identification, they attempted to match the identities of human agents only from the observed trajectory data and detect the abnormal driver behaviors which enhance the safety of passengers. However, We discovered that the trajectories within the HuMID model are susceptible to adversarial attacks, implying that if a malicious driver intentionally alters the trajectories in real-life scenarios, the trained HuMID DNN models can be deceived. Therefore, our primary focus lies in enhancing the robustness of these HuMID DNN models.

Adversarial attack. Many previous studies found that DNN-based models perform exceptionally well in classification, regression, and many other tasks, but they are vulnerable to carefully crafted adversarial examples [12, 18, 30, 31, 44, 45]. Most of the attacking methods were applied to the image classification problem. Li et al. found Deep Learning-based Text Understanding (DLTU) is inherently vulnerable to adversarial text attacks [19]. There is also some related work that focuses on generating adversarial examples for text classification problems [1, 6, 9, 49] and video recognition [15, 47]. As far as we know, we are the first to explore minor perturbations that can easily disrupt human mobility trajectories.

Adversarial training. Adversarial training is one of the most effective methods for DNN-based models to defend against adversarial attacks. Unlike other defense strategies, it seeks to improve model robustness on an intrinsic level [12, 14, 27, 40, 45]. However, it is much slower than vanilla neural network training because it must construct adversarial examples for the entire training data at each iteration. So standard adversarial training is impractical on large-scale problems due to the high computational cost. Shsfahi et al. [39] proposed free adversarial training to accelerate adversarial training, achieving comparable robustness to PGD adversarial training[27]. The main point is that instead of performing individual gradient computations for each update step, both model parameters and perturbations are updated in a single backward pass. Wong et al. [48] discovered fast adversarial training, which combines FGSM adversarial training with random initialization to further accelerate computations. While adversarial training models are typically employed for image classification tasks, we believe we are the first to use fast adversarial training to improve the robustness of DNN-based HuMID models.

6 CONCLUSION

In this paper, we propose to enhance the robustness of the HuMID model against adversarial attacks by training with attacked spatial-temporal data. Our proposed L_0-constrained ST-iFGSM technique generates subtle changes to the trajectories that effectively fool HuMID models. We evaluate the performance of the attack on both the ST-SiameseNet and our method outperforms the baselines in terms of efficiency and the number of steps required to deceive the models. To defend against such attacks, we use ST-FGSM fast adversarial training which leads to an improvement in the accuracy of the SOTA HuMID models in predicting the identification of attacked data, demonstrating that these models can be made more resilient to adversarial samples.

7 ACKNOWLEDGEMENTS

Mingzhi Hu, Xin Zhang, and Yanhua Li were supported in part by NSF grants IIS-1942680 (CAREER), CNS-1952085, CMMI-1831140, and DGE-2021871. Xun Zhou was funded in part by the SAFER-SIM University Transportation Center through a grant from the U.S. Department of Transportation (69A3551747131). Jun Luo was partially supported by ARC Discovery Project (Grant DB210100743).

REFERENCES

[1] Melika Behjati, Seyed-Mohsen Moosavi-Dezfooli, Mahdieh Soleymani Baghshah, and Pascal Frossard. 2019. Universal adversarial attacks on text classifiers. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 7345–7349.

[2] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*. Ieee, 39–57.

[3] Ernest Cheung, Aniket Bera, Emily Kubin, Kurt Gray, and Dinesh Manocha. 2018. Identifying driver behaviors using trajectory features for vehicle navigation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3445–3452.

[4] Arijit Chowdhury, Tapas Chakravarty, Avik Ghose, Tanushree Banerjee, and P Balamuralidhar. 2018. Investigations on driver unique identification from smartphone's GPS data alone. *Journal of Advanced Transportation* 2018 (2018).

[5] Ye Ding, Yanhua Li, Ke Deng, Haoyu Tan, Mingxuan Yuan, and Lionel M Ni. 2017. Detecting and analyzing urban regions with high impact of weather change on transport. *IEEE Transactions on Big Data* 3, 2 (2017), 126–139.

[6] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751* (2017).

[7] Daniel Embarcadero-Ruiz, Helena Gómez-Adorno, Alberto Embarcadero-Ruiz, and Gerardo Sierra. 2022. Graph-based Siamese network for authorship verification. *Mathematics* 10, 2 (2022), 277.

[8] Chenhui Feng, Tao Wang, Yue Yu, Yang Zhang, Yanzhi Zhang, and Huaimin Wang. 2020. Sia-RAE: A Siamese Network based on Recursive AutoEncoder for Effective Clone Detection. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 238–246.

[9] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 50–56.

[10] Yong Ge, Chuanren Liu, Hui Xiong, and Jian Chen. 2011. A taxi business intelligence system. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 735–738.

[11] Yong Ge, Hui Xiong, Alexander Tuzhilin, Keli Xiao, Marco Gruteser, and Michael Pazzani. 2010. An energy-efficient mobile recommender system. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 899–908.

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[13] David Hallac, Abhijit Sharang, Rainer Stahlmann, Andreas Lamprecht, Markus Huber, Martin Roehder, Jure Leskovec, et al. 2016. Driver identification using automobile sensor data from a single turn. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 953–958.

[14] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. 2015. Learning with a strong adversary. *arXiv preprint arXiv:1511.03034* (2015).

[15] Linxi Jiang, Xingjun Ma, Shaoxiang Chen, James Bailey, and Yu-Gang Jiang. 2019. Black-box adversarial attacks on video recognition models. In *Proceedings of the 27th ACM International Conference on Multimedia*. 864–872.

[16] Amin Vahedian Khezerlou, Xun Zhou, Lufan Li, Zubair Shafiq, Alex X Liu, and Fan Zhang. 2017. A traffic flow approach to early detection of gathering events: Comprehensive results. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 6 (2017), 1–24.

[17] Tung Kieu, Bin Yang, Chenjuan Guo, and Christian S Jensen. 2018. Distinguishing trajectories from different drivers using incompletely labeled trajectories. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 863–872.

[18] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. 2018. Adversarial examples in the physical world. In *Artificial intelligence safety and security*. Chapman and Hall/CRC, 99–112.

[19] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271* (2018).

[20] Yanhua Li, Jun Luo, Chi-Yin Chow, Kam-Lam Chan, Ye Ding, and Fan Zhang. 2015. Growing the charging station network for electric vehicles with trajectory data analytics. In *2015 IEEE 31st international conference on data engineering*. IEEE, 1376–1387.

[21] Yanhua Li, Moritz Steiner, Jie Bao, Limin Wang, and Ting Zhu. 2014. Region sampling and estimation of geosocial data with dynamic range calibration. In *2014 IEEE 30th International Conference on Data Engineering*. IEEE, 1096–1107.

[22] Min Lin, Qiang Chen, and Shuicheng Yan. 2013. Network in network. *arXiv preprint arXiv:1312.4400* (2013).

[23] Chen Liu, Ke Deng, Chaojie Li, Jianxin Li, Yanhua Li, and Jun Luo. 2016. The optimal distribution of electric-vehicle chargers across a city. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 261–270.

[24] Lyft. 2023. Lyft Services.

[25] Bo Lyu, Shijian Li, Yanhua Li, Jie Fu, Andrew C Trapp, Haiyong Xie, and Yong Liao. 2016. Scalable user assignment in power grids: a data driven approach. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 1–4.

[26] Shuo Ma, Yu Zheng, and Ouri Wolfson. 2013. T-share: A large-scale dynamic taxi ridesharing service. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 410–421.

[27] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[28] Aarian Marshall. 2020. Uber's New Features Put a Focus on Rider Safety.

[29] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence* 41, 8 (2018), 1979–1993.

[30] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1765–1773.

[31] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2574–2582.

[32] O'Brien. 2022. Uber now lets users text with a safety agent to monitor rides. https://www.cnn.com/2022/08/30/tech/uber-safety-update/index.html.

[33] Min-hwan Oh and Garud Iyengar. 2019. Sequential anomaly detection using inverse reinforcement learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & data mining*. 1480–1490.

[34] OpenStreetMap contributors. 2017. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org.

[35] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*. IEEE, 582–597.

[36] Meng Qu, Hengshu Zhu, Junming Liu, Guannan Liu, and Hui Xiong. 2014. A cost-effective recommender system for taxi drivers. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 45–54.

[37] Huimin Ren, Menghai Pan, Yanhua Li, Xun Zhou, and Jun Luo. 2020. St-siamesenet: Spatio-temporal siamese networks for human mobility signature identification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1306–1315.

[38] Huigui Rong, Xun Zhou, Chang Yang, Zubair Shafiq, and Alex Liu. 2016. The rich and the poor: A Markov decision process approach to optimizing taxi driver revenue efficiency. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2329–2334.

[39] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. 2019. Adversarial training for free! *Advances in Neural Information Processing Systems* 32 (2019).

[40] Uri Shaham, Yutaro Yamada, and Sahand Negahban. 2018. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing* 307 (2018), 195–204.

[41] Faiz Siddiqui. 2019. Uber makes changes amid swarm of criticism over rider safety. https://www.washingtonpost.com/technology/2019/09/26/uber-makes-safety-changes-amid-swarm-criticism-over-protection-riders/.

[42] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[43] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* (2014).

[44] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* 23, 5 (2019), 828–841.

[45] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[46] Uber. 2023. Uber Services.

[47] Zhipeng Wei, Jingjing Chen, Xingxing Wei, Linxi Jiang, Tat-Seng Chua, Fengfeng Zhou, and Yu-Gang Jiang. 2020. Heuristic black-box adversarial attacks on video recognition models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 12338–12345.

[48] Eric Wong, Leslie Rice, and J Zico Kolter. 2020. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994* (2020).

[49] Jincheng Xu and Qingfeng Du. 2020. Texttricker: Loss-based and gradient-based adversarial attacks on text classification models. *Engineering Applications of Artificial Intelligence* 92 (2020), 103641.

[50] Tong Xu, Hengshu Zhu, Xiangyu Zhao, Qi Liu, Hao Zhong, Enhong Chen, and Hui Xiong. 2016. Taxi driving behavior analysis in latent vehicle-to-vehicle networks: A social influence perspective. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1285–1294.

[51] Nicholas Jing Yuan, Yu Zheng, Liuhang Zhang, and Xing Xie. 2012. T-finder: A recommender system for finding passengers and vacant taxis. *IEEE Transactions*

on knowledge and data engineering 25, 10 (2012), 2390–2403.

[52] 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.

[53] Xingjian Zhang, Xiaohua Zhao, and Jian Rong. 2014. A study of individual characteristics of driving behavior based on hidden markov model. *Sensors & Transducers* 167, 3 (2014), 194.

A APPENDIX FOR REPRODUCIBILITY

In this section, we provide detailed information to support the reproducibility of the results in this paper. We implement our deep neural network on Python 3.9.12 with the machine learning library Keras, version 2.10.0, and Tensorflow version 2.10.1. Our experiments run on a virtual machine running Linux-ubuntu 20.04 − $x86\_64$ with 1 GPU, NVIDIA A100-SXM4-80GB.

A.1 Settings of ST-SiameseNet

We train our model on the Shenzhen taxi dataset as a balanced binary class classification problem, since we randomly select a pair of trajectories with equal probability in each iteration. To predict whether the trajectories from two time periods belong to the same driver, we implement the standard back-propagation on feed-forward networks by adaptive moment estimation (Adam) with first momentum (set to 0.9) and second momentum (set to 0.999). Our mini-batch size is 32, and each trajectory has paddings of 60. The learning rate is 0.0001. We trained the network for 30 epochs. The following is the structure of ST-SiameseNet for human mobility signature identification:

- LSTMS. The trajectories are embedded in two hidden layers containing 200 and 100 units, respectively.
- LSTMD. The LSTMD has the same components of neurons as LSTMS
- Similarity-learner. It's a three-layer fully-connected network with hidden units [64,32,8,1]. We use ReLU activation functions for all hidden layers and sigmoid activation at the output layer.

For classification implementation, we set the similarity score threshold to 0.5 which is influenced by the sigmoid activation function utilized in the output layer. A value below 0.5 indicates a lower probability of similarity, while a value above 0.5 suggests a higher probability[7, 8]. Trajectories with similarity scores above 0.5 are classified as belonging to the same agent while trajectories with scores below 0.5 are classified as belonging to different agents. If an attack drives the ST-SiameseNet similarity score from below 0.5 to above 0.5, the classifier predicts the paired trajectories from two different drivers to be from the same one after the attack, which demonstrates a successful attack. Note that the similarity threshold differs based on different HuMID tasks for different datasets.

A.2 Settings of LSTM multi-class classification model

The dataset preprocessing is the same as ST-SiameseNet. The learning rate is 0.000015. We trained the network for 100 epochs, with a batch size equal to 16.

- LSTMS. The trajectories are embedded in two hidden layers containing 200 and 100 units, respectively.
- LSTMD. The LSTMD has the same components of neurons as LSTMS