Few-shot Learning for Trajectory-based Mobile Game Cheating Detection

Yueyang Su
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences, China
suyueyang19b@ict.ac.cn

Di Yao*
Institute of Computing Technology, Chinese Academy of Sciences, China
yaodi@ict.ac.cn

Xiaokai Chu
Wenbin Li
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences, China
chuxiaokai@ict.ac.cn
liwenbin20z@ict.ac.cn

Jingping Bi*
Institute of Computing Technology, Chinese Academy of Sciences, China
bjp@ict.ac.cn

Shiwei Zhao
NetEase Fuxi AI Lab, China
zhaoshiwei@corp.netease.com

Runze Wu
NetEase Fuxi AI Lab, China
wurunze1@corp.netease.com

Shize Zhang
NetEase Fuxi AI Lab, China
zhangshize@corp.netease.com

Jianrong Tao
NetEase Fuxi AI Lab, China
hztaojianrong@corp.netease.com

Hao Deng
NetEase Fuxi AI Lab, China
denghao02@corp.netease.com

ABSTRACT

With the emerging of smartphones, mobile games have attracted billions of players and occupied most of the share for game companies. On the other hand, mobile game cheating, aiming to gain improper advantages by using programs that simulate the players' inputs, severely damages the game's fairness and harms the user experience. Therefore, detecting mobile game cheating is of great importance for mobile game companies. Many PC game-oriented cheating detection methods have been proposed in the past decades, however, they can not be directly adopted in mobile games due to the concern of privacy, power, and memory limitations of mobile devices. Even worse, in practice, the cheating programs are quickly updated, leading to the label scarcity for novel cheating patterns. To handle such issues, we in this paper introduce a mobile game cheating detection framework, namely FCDGame, to detect the cheats under the few-shot learning framework. FCDGame only consumes the screen sensor data, recording users' touch trajectories, which is less sensitive and more general for almost all mobile games. Moreover, a Hierarchical Trajectory Encoder and a Cross-pattern Meta Learner are designed in FCDGame to capture the intrinsic characters of mobile games and solve the label scarcity problem, respectively. Extensive experiments on two real online games show that FCDGame achieves almost 10% improvements in detection accuracy with only few fine-tuned samples.

*Corresponding authors.

This work is licensed under a Creative Commons Attribution International 4.0 License.

KDD '22, August 14–18, 2022, Washington, DC, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539157

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Anomaly detection**.

KEYWORDS

Mobile Game, Cheating Detection, Few-shot Learning

ACM Reference Format:
Yueyang Su, Di Yao, Xiaokai Chu, Wenbin Li, Jingping Bi, Shiwei Zhao, Runze Wu, Shize Zhang, Jianrong Tao, and Hao Deng. 2022. Few-shot Learning for Trajectory-based Mobile Game Cheating Detection. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, Washington, DC, USA, 9 pages. https://doi.org/10.1145/3534678.3539157

1 INTRODUCTION

With the popularity of the commonly-used smart phones, recent years have witnessed a prosperous development of the online mobile games. Nowadays, mobile games represent one of the largest and fastest-growing Internet business sectors. For example, according to the research report released by Newzoo [1], the total number of mobile game players has exceeded two billion across the world and the revenues will reach 116.4 billion dollars in 2024.

Despite the glories, the game companies always have to face the attacks from the game cheating programs. The currency and items acquired in games can be sold to other players for real profits. Thus many cheating activities generated by automated game bots are observed in almost all online games, bringing in incalculable loss for the game companies. Therefore, how to detect such cheating activities and building a healthy game environment have always been a vital and essential problem for many companies.

Many research attentions have been attracted to game cheating detection [4, 13] in the past decades. Existing works can be roughly

[1] https://newzoo.com/products/reports/global-mobile-market-report/

categorized into two groups, *i.e.*, traditional approaches [10, 26] and machine learning-based approaches [12, 21]. Traditional approaches usually maintain a blacklist of game bot processes by aggregating knowledge from experts, where players along with the blocked processes would be detected as the cheats. However, these methods heavily rely on the prior knowledge of experts and can not be generalized to different games. Recent years have also witnessed a trend in machine learning-based cheating detection. Sadly, most of them are designed for PC games and rely on sufficient labeled samples for (semi)-supervised model learning.

Different from PC games, due to the temptation of huge profit, cheats in mobile games are changing fast, and the newly-captured cheating samples usually never appear in the cheating list before. That is to say, we usually have to face a problem of **few-shot** mobile game cheating detection, *i.e.*, only few labeled samples available for the model adaptation when detecting novel cheating samples. Thus, existing methods can not be directly translated into mobile games. It is non-trivial to handle the cheating detection problem in mobile games. The reasons lie in the following two aspects:

- *Data Privacy.* Unlike PC games, mobile devices contain much sensitive information including locations, communications, and photographs. Requesting authorities for accessing such data is unacceptable for mobile game players. Therefore, in most cases, only the screen sensor data [14], which records the finger positions and events(*e.g.*, click,drag), is feasible in mobile games for cheating detection.
- *Label Scarcity.* New cheating patterns are emerging quickly in mobile games while the detection should be conducted timely. It is intractable to obtain sufficient labeled samples for model training. That is to say, we usually have to face a **few-shot learning** problem. Therefore, how to detect cheating players in few-shot settings is a new problem and has not been studied.

In this paper, we proposed a *F*ew-shot *C*heating *D*etection method for mobile *Game*s, namely *FCDGame*, to systemically solve the problems mentioned above. In brief, FCDGame is a meta-learning based framework, which has the following attractive characters:

- **Secure.** We only study the data collected by the screen sensors, which only records the touch coordinates and event types on the screen. It is the passive data in mobile games and would not offend the data privacy of players.
- **Low Demand.** FCDGame is designed for few-shot cheating detection. It extracts the common characteristics of previously detected cheating samples and can adapt to novel cheating patterns with only 1~5 labeled samples.
- **General.** As the screen sensor data can be collected in almost all kinds of mobile games, the proposed model is general for various games.

To obtain such characters, we model the screen sensor data as touch trajectories of fingers under a hierarchical deep learning architecture. The potential patterns of the trajectories relate to the users' operations and are important for cheating detection. However, unlike traditional trajectory data, *e.g.*, road networks, the touch trajectories are organized in a more complex hierarchical structure which makes it difficult to extract patterns. Therefore, inspired by the natural hierarchical structure of touch trajectories, we design a meta-learning based model FCDGame with hierarchical architecture. In detail, FCDGame contains two novel modules, *i.e.*, the Hierarchical Trajectory Encoder and the Cross-patterns Meta Learner. The Hierarchical Trajectory Encoder models the sequential information of coordinates in each event as a representation vector and aggregates the vectors to generate the trajectory embedding. The parameters of the encoder are optimized with the Cross-patterns Meta Learner which is motivated by the meta-learning framework [8]. Specifically, We adopt the cheating detection on known patterns as the meta tasks and utilize the labeled samples of these patterns as supervised information to train a general model which can quickly adapt to novel cheating detection tasks. The main contributions are summarized as follows:

- **New Task**. We define a new task, *i.e.*, few-shot mobile game cheating detection, which generally exists in almost all mobile games.
- **Hierarchical Architecture**. Based on the touch trajectory, we propose a novel hierarchical model FCDGame. It models the spatio-temporal information in coordinate-wise and fuses in event-wise to generate trajectory embedding. Our model not only protects data privacy but also extracts the common characters from different cheating patterns.
- **Real Commercial Data**. Extensive experiments on two real Netease games show that FCDGame achieve state-of-the-art performance and is general for different games. For example, we achieve over 80% accuracy in detecting novel cheating patterns even though the labeled samples available are limited in 1~5.

2 PRELIMINARY

In this section, we first detail the studied data collected from two real online games. After that, we give some formal definitions of cheating detection and translate it into a few-shot learning problem.

2.1 Data description

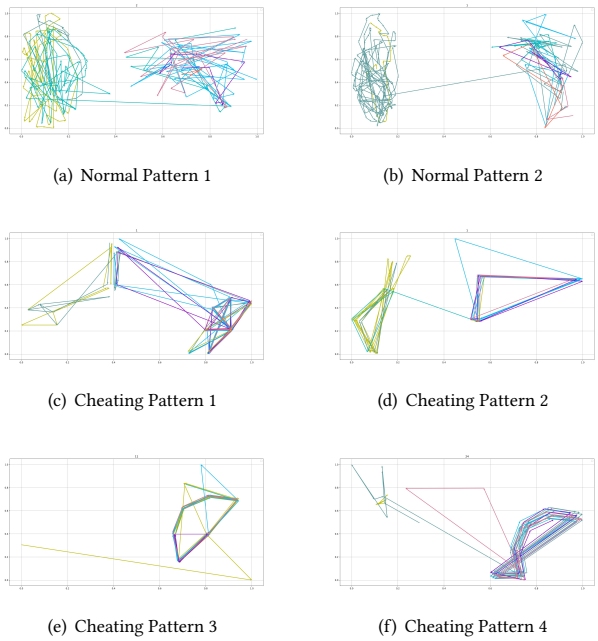
We collect one-week data (from April 21 to April 27, 2021) from two mobile games of Netease Inc. [2], and the sampling frequency of successive coordinates is 400ms. For privacy concerns, we denote them as Battle and Rookie.

- **Battle**: There are 507,544 touch trajectories of players, 15,669 of which are labeled as cheats by experts.
- **Rookie**: The rookie consists of 322,899 touch trajectories, including 10,305 cheating samples.

Data structure. The touch trajectories are organized as a sequence of events which are basic operation units, *i.e.*, click, down-up, down-click-up, drag, and drag-click. Each event is a sequence of coordinates with spatial and time information. In order to model the touch trajectories, it is necessary to model the potential characters of different events, and a hierarchical model is needed.

Cheating Patterns. The cheating samples in our datasets are labeled by the experts in Netease product teams. However, the original labels are either positive (cheating) or negative (normal). It is hard to figure out the sub-classes cheating patterns without the sub-classes of game cheating trajectories. To achieve the detection

[2] https://game.163.com/



(a) Normal Pattern 1 (b) Normal Pattern 2

(c) Cheating Pattern 1 (d) Cheating Pattern 2

(e) Cheating Pattern 3 (f) Cheating Pattern 4

Figure 1: Illustration of several normal and cheating examples in game Battle.

of novel cheating patterns, we adopt the unsupervised learning technique, *i.e.*, seq2seq auto-encoder [32] to represent the cheating trajectories into embedding vectors and employ DBSCAN [3] to obtain the clusters of the cheating samples. After that, the experts check the samples in each cluster and find five cheating patterns which are different from each other. We provide some representative samples in the game Battle to show the extracted cheating patterns in Figure 1. As shown in the figure, compared with the normal patterns in (a) and (b), the cheating patterns are more regular. However,due to the limited labeled samples and the complexity of trajectories, it is hard to directly detect by traditional methods, *e.g.*, angle-based statistics methods [29]. Among the cheating patterns, besides the common characters, there are also some differences which are not only in the easily disturbed features, *e.g.*, angle and density, but also in the potential features. However, the angle and density are sensitive to the selected labeled samples and lead to poor performances of traditional methods in the few-shot settings. Therefore the potential features are more useful, but they are difficult to be extracted.

2.2 Problem Definition

Given a dataset of touch trajectories, there usually exist some trajectories which are not likely generated by normal players. The mobile game cheating detection task is to separate game cheating trajectories from normal data.

However, the cheating patterns change fast in mobile games. The labeled samples of novel cheating patterns are hard to obtain. Therefore, we have to face a few-shot learning problem. The general few-shot learning [31] is to transfer the knowledge from an auxiliary dataset that is different from the task and learn a model for the target task with a small amount of supervised information. In our task, we aim to discover cheating samples of novel patterns with few labeled samples and adopt the labeled samples of known cheating patterns as the auxiliary dataset.

Formally, we denote a touch trajectory as $E = [(T_1, e_1), \cdots, (T_M, e_M)]$, where each tuple consists of a coordinates sequence and it's event type. The coordinates sequence is denoted as $T_i = (p_1, p_2, \cdots, p_N)$, and each coordinate $p_j = (x_j, y_j, t_j)$ is a tuple, in which x_j and y_j are the pixel coordinates and t_j is the timestamp of the position p_j.

Given the samples of K_p ground-truth cheating patterns $\mathcal{T} = \{\mathcal{T}_1, \cdots, \mathcal{T}_{K_p}\}$, where K_p is the number of known cheat patterns and $\mathcal{T}_i \in \mathcal{T}$ is a set of labeled touch trajectories E with labels 0 (normal) or 1 (cheating). We assume that samples in $\mathcal{T}$ are sufficient. Then few labeled samples of K_n novel cheating patterns(for example, less than 10 samples per class) are used as supervised information for target tasks, denoted as $\mathcal{T}^{new} = \{\mathcal{T}_1^{new}, \cdots, \mathcal{T}_{K_n}^{new}\}$. our task is to detect samples of the K_n novel cheating patterns with the information of $\mathcal{T}^{new}$ and the knowledge of ground-truth cheating patterns $\mathcal{T}$.

3 METHODOLOGY

In this section, we detail the proposed FCDGame. As shown in Figure 2, FCDGame consists of two key components: Hierarchical Trajectory Encoder and Cross-pattern Meta Learner. The Hierarchical Trajectory Encoder is proposed to model the spatio-temporal relations of touch trajectories as well as the hierarchical relationship between coordinates sequences and events sequences. The Cross-pattern Meta Learner is an optimization strategy that transfers knowledge from ground-truth cheating patterns and guides the learning of the general meta-model. The meta-model can adapt to the novel cheating detection task quickly with few labeled samples of the target pattern.

3.1 Hierarchical Trajectory Encoder

As shown in Figure 3, we propose a hierarchical encoder to capture both the spatio-temporal information in coordinates sequences and the correlation of events sequences, namely Hierarchical Trajectory Encoder. Our Hierarchical Trajectory Encoder contains two procedures: (1) Coordinate-wise trajectory encoding, which models the spatial and temporal relations of coordinates in each event; (2) Event-wise trajectory encoding, which takes the output of Coordinate-wise trajectory encoding as input and captures the relations in event pairs. Next, we specify the operations in the two procedures.

3.1.1 Coordinate-wise trajectory encoding. As described in Section 2.1, the original data in touch trajectories are organized by events. Within each event, the coordinates sequence is captured to determine the event type, and the correlation of coordinates in the same event is stronger than those cross events. For example, the coordinates in a drag event are intended to complete a directional move, and are more relevant compared to the coordinate of the following click. Thus, it is necessary to mode the coordinate-wise relations of coordinates in each event.

Suppose the length of the coordinates sequence is N, we denote it as $T = (p_1, p_2, \cdots, p_N)$. Each coordinate p_i consists of a triad,

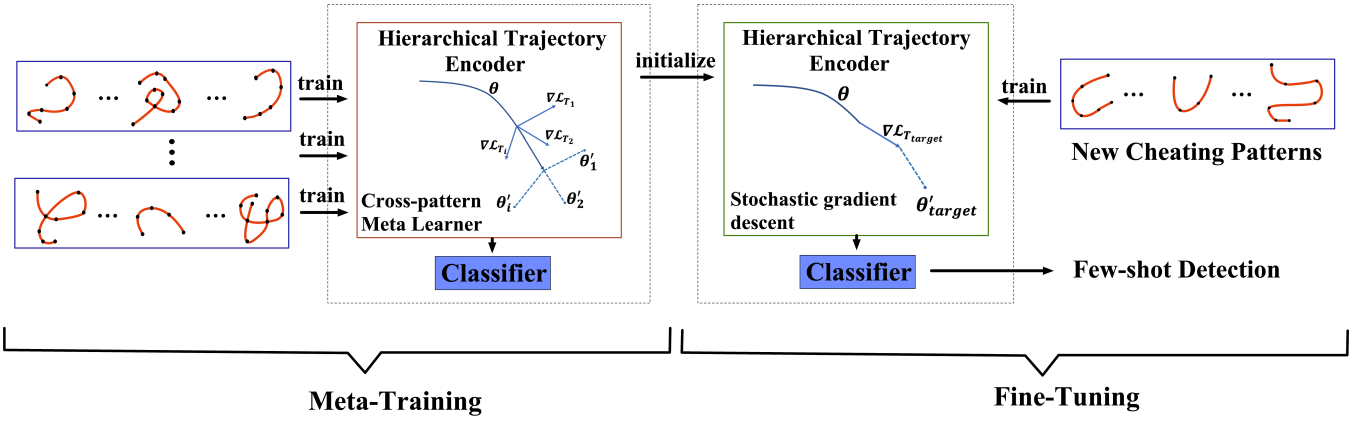


Figure 2: The illustration of FCDGame. FCDGame is trained across multiple known cheating patterns and can be well adapted to the novel(target) cheating patterns with few labeled samples. Specifically, the Hierarchical Trajectory Encoder is optimized with Cross-pattern Meta Learner in Meta-Training and the learned parameters are used to initialize the model in Fine-tuning.

i.e., $p_i = (x_i, y_i, t_i)$, which represents the x coordinate, y coordinate, and recording time respectively. Since the value of the triad is in the discrete space, we need to first process it into vectors.

process coordinate. In original data, the coordinate is recorded as the location of the pixel, which could be various in different device types with different resolutions. Thus, we employ a min-max normalization to transform the pixel location into a normalized range:

$$x_{max} = max(x_1, \cdots, x_N); \quad x_{min} = min(x_1, \cdots, x_N)$$
$$y_{max} = max(y_1, \cdots, y_N); \quad y_{min} = min(y_1, \cdots, y_N)$$
$$f_i^x = \frac{x_i - x_{min}}{x_{max} - x_{min}}; \quad f_i^y = \frac{y_i - y_{min}}{y_{max} - y_{min}}$$

process time. The time t_i in the triad denotes the absolute time when sampled, which, however, shows weak relatedness in the context. On the opposite, the time interval between different coordinates usually contains much useful information that can reflect the abnormal behaving patterns. For example, in first-person shooter games (FPS), the time interval between coordinates describes the reaction time of players. The operations with less reaction time are impossible to be generated by real players and can be considered as cheating activities.

Thus, instead of using the absolute time, we employ the time gaps as the input feature in FCDGame.

$$f_i^t = t_i - t_{i-1}; \quad f_1^t = 0$$

Similar to the coordinate features, we also normalize time features with min-max normalization as:

$$f_i^t = \text{Normalize}(f_i^t, max(f^t), min(f^t))$$

After that, We concatenate both coordinate features and time feature as $\mathbf{f}_i = \text{concat}(f_i^x, f_i^y, f_i^t)$, and take it as the input of Coordinate-wise trajectory encoding.

Then we employ BiLSTM to refine a expressive latent representation for each trajectory. Formally, a generic BiLSTM hidden layer updates with the following formulas:

$$\overrightarrow{\mathbf{h}}_i = \overrightarrow{f}(\overrightarrow{\mathbf{h}}_{t-1}, \mathbf{f}_i)$$
$$\overleftarrow{\mathbf{h}}_i = \overleftarrow{f}(\overleftarrow{\mathbf{h}}_{t+1}, \mathbf{f}_i)$$

where $\overrightarrow{f}$ and $\overleftarrow{f}$ are the forward and backward LSTMs, $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ are the forward and backward hidden states. Then the hidden state can be concatenated as $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i, \overleftarrow{\mathbf{h}}_i]$, which captures both the past and future information in the sequence. In detail, the hidden state at the last timestamp is used to be the representation of the current event, denoted by $\mathbf{v} = f(\mathbf{h}_{N-1}, f_N)$.

3.1.2 Event-wise trajectory encoding. For an events sequence $E = [(T_1, e_1), \cdots, (T_M, e_M)]$, the event type information in each tuple (T_i, e_i) is also essential to study the patterns of touch trajectories from the perspective of player's operation. We encode the event type information with an embedding matrix. Specifically, for each event type, we construct a randomly initialized embedding matrix $\mathbf{E} \in \mathbb{R}^{d \times O}$, in which O is the amount of event types and d is the dimension of the embedding vector.

To refine the representation of the whole touch trajectory, we conduct the Coordinate-wise trajectory encoding to obtain the representation of each coordinates sequence and concatenate them with the event type embedding as:

$$\mathbf{v}_i = \text{Encoding}_c(T_i)$$
$$\mathbf{e}_i = \text{Embedding}(e_i, \mathbf{E})$$
$$\mathbf{r}_i = \text{concat}(\mathbf{v}_i, \mathbf{e}_i)$$

where Encoding_c is the Coordinate-wise trajectory encoding. After that, we employ the Transformer [30] as the trajectory encoder, which has shown outstanding performance in modeling sequential data, such as nlp, video recognition *etc.* [20, 43]. As shown in Figure 3, we employ a L-layer Transformer to generate the final embedding of the touch trajectory. The input is the aggregation of $\mathbf{r}_i$, which can be denoted as:

$$\mathbf{X} = [\mathbf{r}_1; \cdots; \mathbf{r}_M]$$

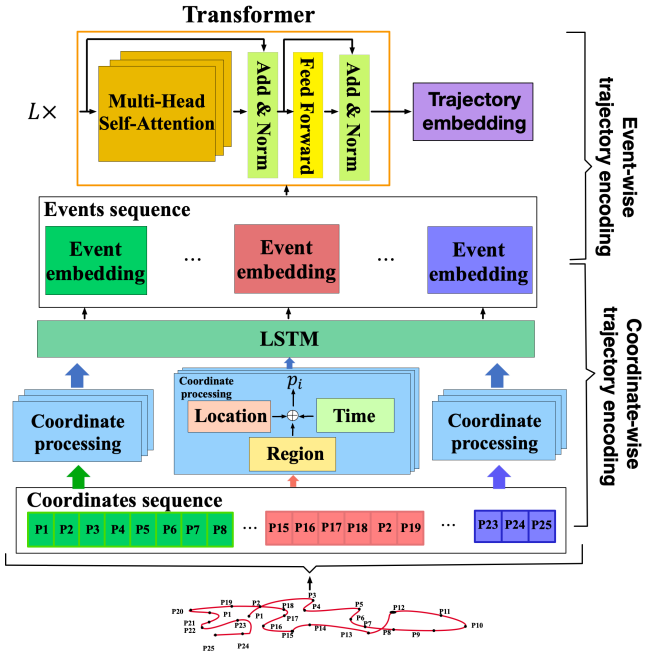


Figure 3: The model architecture of Hierarchical Trajectory Encoder. The encoder contains two modules, where a Coordinate-wise trajectory encoding module takes the coordinates as input and studies the spatio-temporal information for each event, while the Event-wise trajectory encoding module fuses the events information and generates the touch trajectory embedding.

Specifically, each layer is composed of two sub-layers, namely multi-head self-attention mechanism and position-wise fully connected feed-forward network. The multi-head self-attention mechanism is adopted to study the spatio-temporal relationship between different events, and the ith self-attention can be denoted as follows:

$$\mathbf{Q}^i = \mathbf{H}^i\mathbf{W}_i^Q, \mathbf{K}^i = \mathbf{H}^i\mathbf{W}_i^K, \mathbf{V}^i = \mathbf{H}^i\mathbf{W}_i^V$$

$$\mathbf{A}^i = softmax(\frac{\mathbf{Q}^i(\mathbf{K}^i)^T}{\sqrt{d_k}})\mathbf{V}^i$$

where $\mathbf{H}$ is the input of each attention layer and $\mathbf{H}^0 = \mathbf{X}$ is the input of the first attention layer. $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$ are the learnable parameters. Finally, the output $\mathbf{t}$ of the Transformer is the final embedding of the whole events sequence.

Then we feed it to a classifier and optimize it with Cross-pattern Meta Learner for few-shot cheating detection.

3.2 Cross-pattern Meta Learner

In reality, novel cheating types of mobile games are constantly emerging, and it is impractical to obtain a large number of labeled samples. Therefore, a few-shot cheating detection method is urgently needed.

The few-shot cheating detection is rather challenging due to the fact that the supervised information in the labeled dataset of novel cheating patterns $\mathcal{T}^{new}$ is insufficient to train a cheating detector.

We observe that cheating patterns of mobile games usually have similar underlying common characters in touch trajectories, which can be generalized for both seen and unseen cheating patterns. Therefore, we introduce a meta-learning based optimization strategy, namely Cross-pattern Meta Learner. The training procedure is two folds, *i.e.*, meta-training and fine-tuning.

Inspired by the MAML [8], we define the following meta tasks for mobile game cheating detection. In meta-training, FCDGame treats different cheating patterns as different meta tasks. For data in $\mathcal{T}$, we have K_p meta tasks, and we randomly sample normal samples to construct train data with cheating samples.

Specifically, we first divide the dataset of each cheating pattern into support set and query set, *i.e.*, $\mathcal{T}_i = \mathcal{T}_i^{support} \bigcup \mathcal{T}_i^{query}$. In each training round, we sample k touch trajectories for each cheating pattern in $\mathcal{T}_i^{support}$ and organize them as the training data for the i-th meta task.

We denote the parameters of FCDGame as $\theta = \{\theta^E, \theta^C\}$, where θ^E, θ^C represent the trajectory encoder's and the classifier's respectively. As described in [8], the meta training contains two sub-procedures, *i.e.*, customized model update and meta learner update. In the first one, the parameters $\theta_i^{'}$ of i-th task are initialed by θ, and then they are updated by the loss of the samples from $\mathcal{T}_i^{support}$, which is expressed as:

$$\theta_i^{'} = \theta - \alpha\nabla_\theta\mathcal{L}_{\mathcal{T}_i^{support}}(f(\theta))$$

In the meta learner update, the $\mathcal{T}_i^{query}$ is adopted to the customized model with $\theta_i^{'}$ to get the loss of each task which is used to update the meta learner parameters θ:

$$\theta = \theta - \beta\nabla\theta \sum_{i\in[1,...K_p]} L_{\mathcal{T}_i^{query}}(f(\theta))$$

With the above procedure, FCDGame will be optimized for several sampling iterations until it converges.

Fine-tuning. In the fine-tuning procedure, we employ $\mathcal{T}^{new}$ to optimize the model in meta-training to detect novel cheating patterns. Then we evaluate the model on the dataset which has the same cheating types as $\mathcal{T}^{new}$ and denoted as $\mathcal{E}^{new} = \{\mathcal{E}_1^{new}, ...\mathcal{E}_{K_n}^{new}\}$. To detect the samples of the i-th cheating pattern in $\mathcal{E}_i^{new}$, we use the samples in $\mathcal{T}_i^{new}$ as the fine-tuned data. Note that the parameters of θ^E are shared in tasks while the parameters of the classifier are trained from scratch. Owing to the training mechanism of meta learner, FCDGame can fast adapt with few labeled samples.

Table 1: The statistics of the datasets.

Dataset	Normal and Cheating Patterns					
	Normal	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_1^{new}$	$\mathcal{T}_2^{new}$
Battle	30000	1195	1912	1200	1650	1483
Rookie	30000	1356	1505	899	1306	942

4 EXPERIMENT

In this section, we detail the experiment setup and conduct extensive experiments to demonstrate the effectiveness of our proposed framework on two real game datasets, *i.e.*, Battle and Rookie.

Table 2: Performance comparison results of Precision, Recall and F1-score

Dataset	Model	1-shot			5-shot			10-shot		
		precision	recall	F1-score	precision	recall	F1-score	precision	recall	F1-score
Battle	Rule-based method	0.3216	0.4524	0.3759	0.3753	0.4985	0.4282	0.3819	0.5960	0.4655
	BiLSTM	0.5729	0.8539	0.6857	0.6758	0.8114	0.7374	0.71	0.7972	0.7511
	ABLSTM	0.5194	0.5071	0.4980	0.7017	0.8833	0.78054	0.7286	0.8598	0.7852
	H-LSTM	0.6073	0.5885	0.5825	0.7443	0.8950	0.8127	0.7746	0.9254	0.8426
	MAML	0.5792	0.7940	0.6698	0.7304	0.9202	0.8144	0.8099	0.9002	0.8527
	EncDec-AD	0.5593	0.7456	0.6392	0.3107	0.8054	0.4484	0.3887	0.8812	0.5395
	FCDGame	**0.6481**	**0.8913**	**0.7475**	**0.7471**	**0.9923**	**0.8493**	**0.85**	**0.9444**	**0.8947**
Rookie	Rule-based method	0.2613	0.3269	0.2904	0.3958	0.4370	0.4154	0.4176	0.4620	0.4387
	BiLSTM	0.3178	0.4556	0.3744	0.6012	0.715	0.6531	0.4133	0.7716	0.5384
	ABLSTM	0.4042	0.5466	0.4647	0.601	0.6667	0.6321	0.4642	0.5728	0.7001
	H-LSTM	0.4618	0.6776	0.5493	0.6509	0.8076	0.7208	0.7286	0.8598	0.7852
	MAML	0.2407	0.6297	0.3483	0.6626	0.7667	0.7108	0.5899	0.8806	0.7066
	EncDec-AD	0.2457	0.6313	0.3537	0.3922	0.72	0.5078	0.2002	0.8802	0.3262
	FCDGame	**0.6257**	**0.7963**	**0.6116**	**0.7007**	**0.9999**	**0.8694**	**0.82**	**0.9836**	**0.8944**

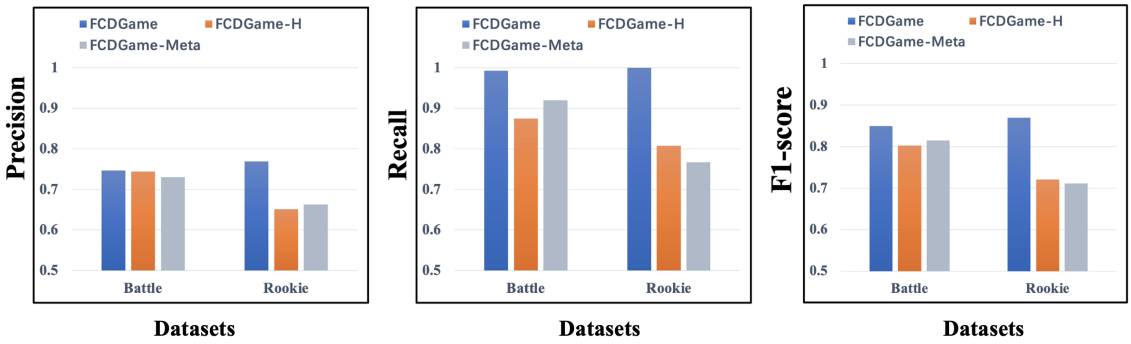


Figure 4: The result of ablation study.

Table 3: Performance comparison with different number of shots on Battle dataset.

# of shot	method	precision	recall	F1-score
1-shot	MAML	0.5792	0.7940	0.6698
	H-LSTM	0.6073	0.5885	0.5825
	FCDGame	**0.6481**	**0.8913**	**0.7475**
2-shot	MAML	0.5464	0.8715	0.6717
	H-LSTM	0.6856	0.7823	0.7223
	FCDGame	**0.7023**	**0.9355**	**0.7799**
3-shot	MAML	0.7038	0.8357	0.7641
	H-LSTM	0.7016	0.9182	0.7930
	FCDGame	**0.7291**	**0.9563**	**0.8187**
4-shot	MAML	0.6952	0.8753	0.775
	H-LSTM	0.7061	0.8608	0.7683
	FCDGame	**0.7307**	**0.9855**	**0.8332**
5-shot	MAML	0.7304	0.9202	0.8144
	H-LSTM	0.7443	0.8950	0.8127
	FCDGame	**0.7471**	**0.9923**	**0.8493**

4.1 Experiment setup

We first introduce the datasets, evaluation metrics, compared baselines and implementation details of our experiments.

Dataset. As described in Section 2.1, we employ the trajectories of two real mobile games to evaluate the performance of FCDGame. We drop the trajectories with a length of less than 20. For each dataset, we select 60% of cheating patterns as known and the remaining as novel patterns. The data labels are obtained from human experts. The statistics of the datasets are shown in Table 1.

Evaluation metrics. We adopt the following metrics: Precision, Recall and F1-score, to evaluate the model performance.

Baseline models. Due to the fact that there is no existing work on few-shot mobile game cheating detection, we compared FCDGame with five methods which are designed for PC game cheating detection and sequence modeling. The details of compared methods are described as follows:

- Rule-based method [29]: The method focuses on the differences in occurrence frequencies of different angles in the motion of robots and humans, using histograms to describe

the distribution of angles. Then the nearest centroid classifier is used to identify cheating samples.

- BiLSTM [23]: BiLSTM is a general model which can efficiently handle sequence data and captures the global and local dependencies of sequences. We train the model with known cheating samples while fine-tuning with few novel cheating samples.
- MAML [8]: We employ BiLSTM as an encoder for the sequence data and update the parameters with the model-agnostic meta-learning algorithm.
- ABLSTM [33]: This method focuses on players' behavior sequences and proposes a BiLSTM with attention for game cheating detection. The model is trained with the same setup as above BiLSTM.
- H-LSTM [35]: A hierarchical model to model sequential information, which can be easily adapted to model the touch trajectories.
- EncDec-AD [18]: An encoder-decoder based model for anomaly detection in sequence data, optimized by minimizing the reconstruction error. We trained the model with a large number of normal samples and then added an extra classifier behind the encoder for cheating detection, which fine-tuned with few novel cheating samples.

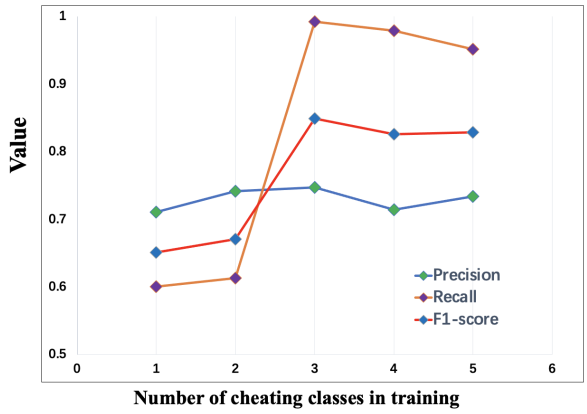


Figure 5: Sensitivity analysis of FCDGame w.r.t. different number of classes in training on Battle dataset.

Implementation Details. We use 2-layers Bilstm as the backbone of the Coordinate-wise trajectory encoding. Then a transformer is introduced for event sequences encoding which consists of 2 layers with 4 self-attention heads. Except for a special note, we set the sample shot $k = 5$ for meta tasks. For the learning rate, we empirically set $\alpha = 0.001$ and $\beta = 0.01$. In addition, we have released the code and data in https://github.com/super1225/cheating-detection.

4.2 Performance Comparison

We compare the performance of FCDGame with above-mentioned baseline methods. The results are shown in Table 2. FCDGame consistently achieves the best performance on the two datasets. Among the baselines, H-LSTM shows better performance than ABLSTM, especially on Rookie, with almost 9% improvements on F1-score in 1-shot setting, indicating that the hierarchical structure is suitable for modeling touch trajectories. We also evaluate the performance of FCDGame compared with two typical baselines(*i.e.*, MAML and H-LSTM) in different shot settings. The results on Battle are shown in Table 3. Obviously, with the increase of fine-tuned samples, the performances of baseline methods are improved steadily. However, with the reduction of the labeled samples, their performances drop sharply. For example, in 1-shot setting, the compared methods almost lost the detection ability, *i.e.*, their precision is around 50%. Conversely, despite the performance of FCDGame also decreasing, it still achieves over 60% precision on both Rookie and Battle. We attribute this to the Cross-pattern Meta Learner, which enables the model to adapt to the novel cheating patterns quickly.

4.3 Ablation Study

We conducted an ablation study to analyze the effectiveness of the proposed components in FCDGame. Two variants are proposed: FCDGame-Meta and FCDGame-H. We first verify the role of the Hierarchical Trajectory Encoder with FCDGame-Meta. In detail, FCDGame-Meta replaces the Hierarchical Trajectory Encoder with a BiLSTM-based model and is trained in the manner of Cross-pattern Meta Learner. As shown in Figure 4, FCDGame achieves a better performance than FCDGame-Meta, the F1-score is about 3% improvements on Battle and 16% on Rookie. These observations indicate that the hierarchical architecture is effective in modeling the touch trajectories, and our Hierarchical Trajectory Encoder has a remarkable ability to extract the hierarchical relations between coordinates and events. Also, it validates that the relation information is informative in detecting mobile game cheating patterns.

Then we examine the effectiveness of the Cross-pattern Meta-learner by comparing the performance gap between FCDGame and FCDGame-H. FCDGame-H employs the Hierarchical Trajectory Encoder for embedding and is directly optimized by stochastic gradient descent. The results are shown in Figure 4. Owing to the meta-learner, FCDGame significantly outperforms FCDGame-H on all the metrics, *e.g.*, F1-score, it achieves nearly 5% improvements on the Battle dataset and 15% on the Rookie dataset. These results show that our Cross-pattern Meta-learner can effectively improve the performance of the model to detect novel cheating patterns with few labeled samples from a parameter optimization perspective.

4.4 Sensitivity Analysis

To analyze the influence of choosing different numbers of cheating patterns in meta-training, we provide different numbers of cheating classes in the meta-training stage to analyze the sensitivity of FCDGame. We vary the number of cheating patterns from 1 to 5 and evaluate all combinations of the 5 patterns. Then we report the average performance of the combinations on the Battle dataset in Figure 5. With the increase of cheating patterns, FCDGame achieves a significant performance improvement. The same observation can also be found on the Rookie dataset. There is a significant increase when the number of cheating classes changes to 3, and after that the performance keeps stable. The reason for the results lies in the biased learning of the model, *i.e.*, models trained with several patterns may focus on specific features that are not general for other cheating patterns, which leads to ineffective learning or even

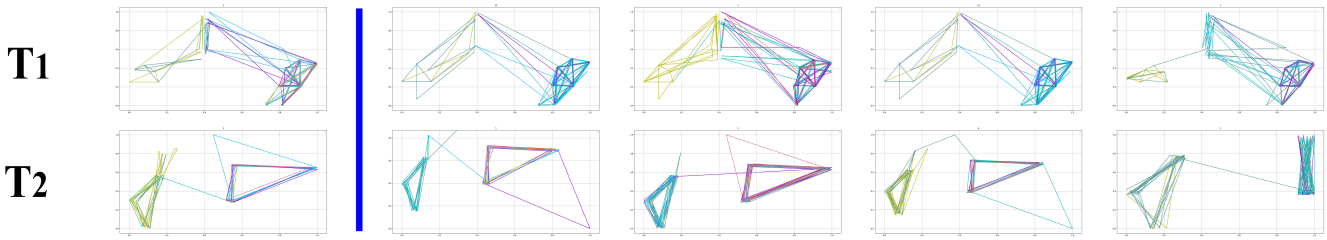


Figure 6: Visualization results of one-shot learning. In each sub-figure, points of different colors indicate different event types.

reverse learning for target cheating patterns. Despite the fact that the biased learning severely damages the generalization and performance of model, FCDGame still achieves good performance with one cheating pattern available in training, which demonstrates the advantages of FCDGame to learn general knowledge and the stability in complex environments.

4.5 Case study

To illustrate the effectiveness of FCDGame, we visualize some results of 1-shot task in Figure 6. The trajectory T on the left is the sample available in the fine-tuning stage, while the trajectories on the right are the samples classified as cheating by FCDGame. We have the following observations: (1) The detected samples are similar to the given fine-tuned samples. Moreover, the detected samples are diverse and visually different from the fine-tuned samples, which verifies that FCDGame can capture the potential features of the touch trajectory and the encoder is powerful to distinguish different cheating patterns. (2) For the different cheating patterns(T_1 and T_2), FCDGame can achieve good detection performance with only one sample, demonstrating that our model can quickly adapt to novel cheating pattern detection tasks with few samples for fine-tuning.

5 RELATED WORK

5.1 Game cheating detection

There is a continued interest in game cheating detection for many years while most of them are proposed for PC games. The methods can be divided into two categories. The traditional approaches [10, 26] are widely used in industries. For example, the anti-cheating system [28] is built with dynamically maintainable blacklists of cheating processes and identifies the clients with suspicious programs as cheating players. Kesteren[29] introduces a rule-based method that focuses on the distribution of angles in the motion of robots and humans. However, these traditional approaches are mostly limited by the prior knowledge of experts and are poor in the environment with cheats constantly emerging.

Nowadays, the machine learning-based [12, 21, 34] approaches have received considerable critical attention. Some researchers [10, 15, 33] have tried to characterize players' activities with pre-selected handcrafted features. For example, Thawonmas [27] analyses the operation sequences in log data and trains a support vector machine with the specific input features such as action frequency, action type, and time interval. Xu[33] proposes a sequence-level model which focuses on players' behavior sequences and introduces a BiLSTM-based model. However, the performance is poor in the complex sequences. Another trend of machine learning-based approaches is applied with the hypothesis that game cheating would repeat the similar action sequences [5, 7], so the users with high self-similarity would be concerned as illegal users. Lee [16] proposes a detection framework to measure the frequencies of users' repeat activities over time without the dependency on the specific game contents and achieves good performance on the real-world dataset. Since the evaluation of self-similarity requires a large number of action sequences, the self-similarity based methods are limited in practice.

Considering the data privacy and generalization, we focus on the touch trajectories which collected with the sensors under the screen. Compared with the diverse data in PC games, the information in these trajectories is rich but hidden. Moreover, the few-shot problem caused by label scarcity of novel cheating patterns leads to the under-learning of models, which disables most existing detection approaches.

5.2 Trajectory modeling

In recent years, there has been an increasing interest in trajectory modeling with the development of downstream tasks such as trajectory forecasting [25, 39, 42], trajectory retrieval [11, 36, 38, 40], *etc.* Some researchers [23, 37] introduce recurrent neural network(RNN) and its variants(*e.g.*, LSTM) to process the trajectories, which have been proved effective for sequence data in nature language processing [1]. For example, Xue[35] proposes a hierarchical LSTM-based model for pedestrian trajectory prediction, which considers both the influence of social neighborhood and scene layouts. Some researchers [41] also adopt GNN to model the relationship of trajectories. For example, Mo [19] proposes a GNN-RNN based encoder-decoder network for interaction-aware trajectory prediction.

There are also some other literature on trajectory modeling. However, most of them focus on the road networks while little quantitative analysis of touch trajectories.

5.3 Few shot learning

Recently, few-shot learning has attracted extensive attention. The methods can be divided into three categories [17, 31], including data augmentation-based approaches, model-based approaches, and algorithm-based approaches. Date augmentation-based approaches [6, 22] apply prior knowledge to augment the supervised information. For example, Schwartz [22] proposes a modified auto-encoder to extract transferable deformations between pairs of training instances at the same level and apply them to synthesize samples of novel classes. The model-based approaches [17, 24] aim to constrain hypothesis space, and the main strategies include parameter sharing and parameter bundling, which can be applied to share

general information between different tasks. The algorithm-based approaches [2, 9] alter the search strategies for the optimal parameters with the guidance of prior knowledge. Finn [8] proposes a model agnostic algorithm to study the optimal initialization parameters with two-layer optimized strategy.

Few-shot learning is a major area of interest within the field of image classification, target detection, recommendation, *etc.* As far as we know, the study of mobile game detection in a few-shot setting has not been studied.

6 CONCLUSION

we have introduced a new problem of few-shot cheating detection in mobile games and analyzed its challenges in data privacy and label scarcity. To address these challenges, we aim to use the touch trajectories which are passively collected by the game applications as the data source and introduce a hierarchical architecture, namely FCDGame. It can not only capture the nature of hierarchical trajectories, but also extract common features from previously known cheating patterns. FCDGame has three attractive characters, *i.e.*, secure, low demand and general. Extensive experiments on two real games demonstrate the superiority of FCDGame over existing cheating detection methods.

ACKNOWLEDGMENTS

This work has been supported by the National Natural Science Foundation of China under Grant No.: 62077044, 61702470, 62002343.

REFERENCES

[1] S. T. Arasteh. 2020. Generalized LSTM-based End-to-End Text-Independent Speaker Verification. (2020).
[2] S. Azadi, M. Fisher, V. Kim, Z. Wang, E. Shechtman, and T. Darrell. 2017. Multi-Content GAN for Few-Shot Font Style Transfer. (2017).
[3] D. Birant and A. Kut. 2007. ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering* 60, 1 (2007), 208–221.
[4] Kuan-Ta Chen and Li-Wen Hong. 2007. User identification based on game-play activity patterns. In *NETGAMES 2007, Melbourne, Australia, 2007.*
[5] Mark Crovella and Azer Bestavros. 1997. Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Trans. Netw.* 5, 6 (1997), 835–846.
[6] Ekin D. Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. 2019. AutoAugment: Learning Augmentation Strategies From Data. In *CVPR 2019, Long Beach, CA, USA, June 16-20, 2019.* Computer Vision Foundation / IEEE.
[7] Eun-Jo, Lee, Won-Jun, Jo, Hyunchul, Kim, Hyemin, Um, Jina, and Lee. 2016. A Study on Game Bot Detection Using Self-Similarity in MMORPGs. *Journal of the Korea Institute of Information Security & Cryptology* 26, 1 (2016), 93–107.
[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1126–1135.
[9] Chelsea Finn, Kelvin Xu, and Sergey Levine. 2018. Probabilistic Model-Agnostic Meta-Learning. In *NeurIPS 2018, December 3-8, 2018, Montréal, Canada.*
[10] Hiroshi Itsuki, Asuka Takeuchi, Atsushi Fujita, and Hitoshi Matsubara. 2010. Exploiting MMORPG log data toward efficient RMT player detection. In *ACE 2010, Taipei, Taiwan, November 17-19, 2010.* ACM, 118–119.
[11] Quanliang Jing, Di Yao, Chang Gong, Xinxin Fan, Baoli Wang, Haining Tan, and Jingping Bi. 2021. TrajCross: Trajecotry Cross-Modal Retrieval with Contrastive Learning. In *IEEEBigData.* IEEE.
[12] Ah Reum Kang, Seong Hoon Jeong, Aziz Mohaisen, and Huy Kang Kim. 2016. Multimodal Game Bot Detection using User Behavioral Characteristics. (2016).
[13] Ah Reum Kang, Huy Kang Kim, and Jiyoung Woo. 2012. Chatting Pattern Based Game BOT Detection: Do They Talk Like Us? *KSII Trans. Internet Inf. Syst.* (2012).
[14] Frauke Kreuter, Georg Christoph Haas, Florian Keusch, Sebastian Bähr, and Mark Trappmann. 2018. Collecting Survey and Smartphone Sensor Data With an App: Opportunities and Challenges Around Privacy and Informed Consent. *Social Science Computer Review* (2018).
[15] Hyukmin Kwon and Huy Kang Kim. 2011. Self-similarity based Bot Detection System in MMORPG. In *The 3rd International Conference on Internet.*
[16] Eunjo Lee, Jiyoung Woo, Hyoungshick Kim, Aziz Mohaisen, and Huy Kang Kim. 2016. You are a Game Bot!: Uncovering Game Bots in MMORPGs via Self-similarity in the Wild. In *NDSS 2016, San Diego, California, USA, February 21-24, 2016.* The Internet Society.
[17] Jiang Lu, Pinghua Gong, Jieping Ye, and Changshui Zhang. 2020. Learning from Very Few Samples: A Survey. *CoRR* abs/2009.02653 (2020). arXiv:2009.02653
[18] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. *CoRR* abs/1607.00148 (2016). arXiv:1607.00148
[19] X. Mo, Y. Xing, and C. Lv. 2021. Graph and Recurrent Neural Network-based Vehicle Trajectory Prediction For Highway Driving. (2021).
[20] D. Neimark, O. Bar, M. Zohar, and D. Asselmann. 2021. Video Transformer Network. (2021).
[21] Kusno Prasetya and Zheng Da Wu. 2010. Artificial Neural Network for bot detection system in MMOGs. In *NetGames 2010, Taipei, Taiwan, 16-17 November, 2010.* IEEE.
[22] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogério Schmidt Feris, Raja Giryes, and Alexander M. Bronstein. 2018. Delta-encoder: an effective sample synthesis method for few-shot object recognition.
[23] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. 2019. The Performance of LSTM and BiLSTM in Forecasting Time Series. In *2019 IEEE International Conference on Big Data (Big Data).* 3285–3292.
[24] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *CVPR'18.* 1199–1208.
[25] Haining Tan, Di Yao, Tao Huang, Baoli Wang, Quanliang Jing, and Jingping Bi. 2021. Meta-Learning Enhanced Neural ODE for Citywide Next POI Recommendation. In *MDM.* IEEE.
[26] Ruck Thawonmas and Yoshitaka Kashifuji. 2010. Detection of MMORPG Misconducts Based on Action Frequencies, Types and Time-Intervals. In *DMIN 2010, July 12-15, 2010, Las Vegas, Nevada, USA.*
[27] R. Thawonmas and Y. Kashifuji. 2010. Detection of MMORPG Misconducts Based on Action Frequencies, Types and Time-Intervals. In *DMIN 2010, July 12-15, 2010, Las Vegas, Nevada, USA.*
[28] Yuan Tian, Eric Chen, Xiaojun Ma, Shuo Chen, and Patrick Tague. 2016. Swords and Shields - A Study of Mobile Game Hacks and Existing Defenses. In *Conference on Computer Security Applications.*
[29] Marlieke van Kesteren, Jurriaan Langevoort, and Franc Grootjen. 2009. A step in the right direction: Botdetection in MMORPGs using movement analysis. In *BNAIC 2009.* 129–136.
[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv* (2017).
[31] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. 2020. Generalizing from a Few Examples: A Survey on Few-shot Learning. *ACM Comput. Surv.* (2020).
[32] C Xca, C Jxa, Z. B. Rui, C. A. Wei, A Jf, and B Cl. 2021. TrajVAE: A Variational AutoEncoder model for trajectory generation. (2021).
[33] Jiarong Xu, Yifan Luo, Jianrong Tao, Changjie Fan, Zhou Zhao, and Jiangang Lu. 2020. NGUARD+: An Attention-based Game Bot Detection Framework via Player Behavior Sequences. *ACM Trans. Knowl. Discov. Data* 14, 6 (2020), 65:1–65:24.
[34] Yongjun Xu, Xin Liu, and Xin Cao.etc. 2021. Artificial intelligence: A powerful paradigm for scientific research. *The Innovation* 2, 4 (2021), 100179. https://www.sciencedirect.com/science/article/pii/S2666675821001041
[35] Hao Xue, Du Q Huynh, and Mark Reynolds. 2018. SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction. In *2018 WACV.* IEEE, 1186–1194.
[36] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. 2019. Computing Trajectory Similarity in Linear Time: A Generic Seed-Guided Neural Metric Learning Approach. In *ICDE.* IEEE.
[37] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. 2019. Computing Trajectory Similarity in Linear Time: A Generic Seed-Guided Neural Metric Learning Approach. In *2019 IEEE 35th International Conference on Data Engineering (ICDE).*
[38] Di Yao, Gao Cong, Chao Zhang, Xuying Meng, Rongchang Duan, and Jingping Bi. 2020. A Linear Time Approach to Computing Time Series Similarity based on Deep Metric Learning. *IEEE Transactions on Knowledge and Data Engineering* (2020).
[39] Di Yao, Chao Zhang, Jian-Hui Huang, and Jingping Bi. 2017. SERM: A Recurrent Model for Next Location Prediction in Semantic Trajectories. In *CIKM.*
[40] Di Yao, Chao Zhang, Zhihua Zhu, Qin Hu, Zheng Wang, Jian-Hui Huang, and Jingping Bi. 2018. Learning deep representation for trajectory clustering. *Expert Syst. J. Knowl. Eng.* (2018).
[41] H. Yao, X. Tang, W. Hua, G. Zheng, and Z. Li. 2018. Modeling Spatial-Temporal Dynamics for Traffic Prediction. (2018).
[42] Chongjian Yue, Lun Du, Qiang Fu, Wendong Bi, Hengyu Liu, Yu Gu, and Di Yao. 2022. HTGN-BTW: Heterogeneous Temporal Graph Network with Bi-Time-Window Training Strategy for Temporal Link Prediction. *arXiv* (2022).
[43] J. Zhang, H. Luan, M. Sun, F. F. Zhai, J. Xu, M. Zhang, and Y. Liu. 2018. Improving the Transformer Translation Model with Document-Level Context. (2018).