Outlier Detection for Streaming Task Assignment in Crowdsourcing

Yan Zhao[1], Xuanhao Chen[2], Liwei Deng[2], Tung Kieu[1], Chenjuan Guo[1], Bin Yang[1], Kai Zheng[2,✉], Christian S. Jensen[1]
[1]Department of Computer Science, Aalborg University, Denmark
[2]University of Electronic Science and Technology of China, China
yanz@cs.aau.dk, xhc@std.uestc.edu.cn, denglw0830@gmail.com,
{tungkvt, cguo, byang}@cs.aau.dk, zhengkai@uestc.edu.cn, csj@cs.aau.dk

ABSTRACT

Crowdsourcing aims to enable the assignment of available resources to the completion of tasks at scale. The continued digitization of societal processes translates into increased opportunities for crowdsourcing. For example, crowdsourcing enables the assignment of computational resources of humans, called workers, to tasks that are notoriously hard for computers. In settings faced with malicious actors, detection of such actors holds the potential to increase the robustness of crowdsourcing platform. We propose a framework called Outlier Detection for Streaming Task Assignment that aims to improve robustness by detecting malicious actors. In particular, we model the arrival of workers and the submission of tasks as evolving time series and provide means of detecting malicious actors by means of outlier detection. We propose a novel socially aware Generative Adversarial Network (GAN) based architecture that is capable of contending with the complex distributions found in time series. The architecture includes two GANs that are designed to adversarially train an autoencoder to learn the patterns of distributions in worker and task time series, thus enabling outlier detection based on reconstruction errors. A GAN structure encompasses a game between a generator and a discriminator, where it is desirable that the two can learn to coordinate towards socially optimal outcomes, while avoiding being exploited by selfish opponents. To this end, we propose a novel training approach that incorporates social awareness into the loss functions of the two GANs. Additionally, to improve task assignment efficiency, we propose an efficient greedy algorithm based on degree reduction that transforms task assignment into a bipartite graph matching. Extensive experiments offer insight into the effectiveness and efficiency of the proposed framework.

CCS CONCEPTS

• **Information systems** → **Location based services**; • **Computing methodologies** → *Machine learning*; • **Human-centered computing** → *Empirical studies in collaborative and social computing*.

KEYWORDS

outlier detection, time series, task assignment, crowdsourcing

ACM Reference Format:
Yan Zhao[1], Xuanhao Chen[2], Liwei Deng[2], Tung Kieu[1], Chenjuan Guo[1], Bin Yang[1], Kai Zheng[2,✉], Christian S. Jensen[1]. 2022. Outlier Detection for Streaming Task Assignment in Crowdsourcing. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3512067

1 INTRODUCTION

Crowdsourcing enables a computing paradigm, where humans, called workers, actively or passively contribute to computing tasks, especially in cases where tasks are intrinsically easier for humans than for computers [27]. Research on crowdsourcing [7, 11, 40] has contributed many techniques for task assignment in different application scenarios, which are based generally on the assumption that workers are trusted and that tasks are meaningful. However, in practice, some workers and task requesters may be malicious. Such actors may complete tasks with low quality or may even sabotage tasks [6, 9, 13], or they may submit "invalid" tasks requests that cannot be finished by their deadlines or that do not bring any rewards to workers. Such malicious actors impact platforms and their workers and requesters negatively. Controlling the quality of workers and tasks is thus a major challenge in crowdsourcing. Recent studies have explored the malicious behavior of workers [8, 28]. Gadiraju et al. [8] analyze prevalent malicious activities in crowdsourcing platforms and study the behavior exhibited by trustworthy and untrustworthy workers. On this basis, they identify different types of malicious behaviors and provide guidelines for task requesters to design crowdsourced surveys efficiently. Wang et al. [28] integrate gold standards into the learning of the quality of workers, where gold standards are questions with correct answers that are known a priori to the crowdsourcing platforms. Typically, the platforms insert a small amount of questions into their tasks, observe the performance of workers, and compute error rates of completed tasks to detect malicious workers. But these studies focus either on a specific task type or use only the error rates to detect malicious workers. They also do not consider malicious task requesters and fail to consider the combination of malicious workers and malicious task requesters in task assignment.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512067

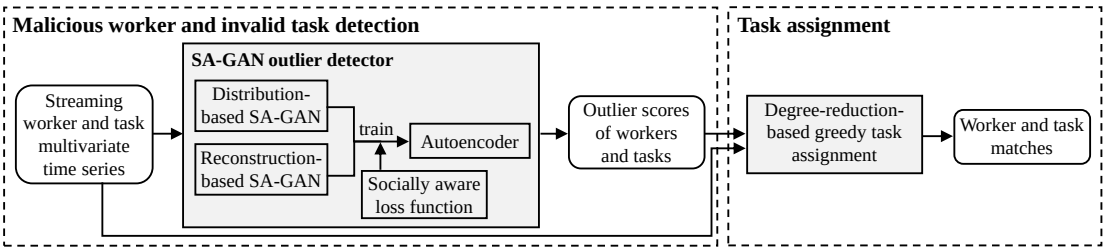


Figure 1: ODSTA Framework Overview

Addressing these unmet challenges, this study goes beyond the state of the art and develops a crowdsourcing framework, called Outlier Detection for Streaming Task Assignment (ODSTA), that aims to enable effective task assignment in settings with malicious workers and task requesters. Specifically, the ODSTA framework consists of a malicious worker and invalid task detection component and a task assignment component, as shown in Figure 1. The arrivals of workers, as well as the arrivals of tasks (or task requests), at a crowdsourcing platform can be modeled as streaming, multivariate time series. Thus, the first component models the detection of malicious workers and invalid tasks as a correlated time series outlier detection problem. The basic task is to determine whether multidimensional data points that represent workers or tasks conform to an expected data distribution. Non-conforming data points then represent malicious workers and invalid tasks. This component is equipped with a novel Socially Aware GAN (SA-GAN) outlier detector that operates in an unsupervised manner. More specifically, we design two SA-GANs, namely a distribution-based SA-GAN and a reconstruction-based SA-GAN, where a socially aware loss function is introduced to achieve better reconstruction. We use the SA-GANs to train an autoencoder adversarially in order to learn the regular patterns of worker and task time series and to calculate outlier scores for workers and tasks. The second component addresses the computational task of matching workers and tasks. Specifically, we propose a greedy algorithm based on degree reduction that transforms task assignment into a bipartite graph matching while considering the outlier scores of workers and tasks. Finally, the worker and task matches, i.e., the task assignments, are obtained.

Our main contributions can be summarized as follows:

1) We propose a general and efficient task assignment framework for crowdsourcing, called Outlier Detection for Streaming Task Assignment (ODSTA), that considers malicious workers and invalid tasks during task assignment.

2) We propose an SA-GAN architecture for worker and task time series outlier detection, where two adversarial training processes are designed to train an autoencoder. To the best of our knowledge, this is the first comprehensive study that aims to identify malicious workers and invalid tasks by modeling them as outliers in time series.

3) An efficient degree-reduction-based greedy algorithm is given that transforms task assignment into a bipartite graph matching.

4) We report on experiments using real data, offering evidence of the effectiveness and efficiency of the proposed framework.

2 PROBLEM STATEMENT

Definition 1 (Worker). *A worker, denoted as $w = (\boldsymbol{w}, a, d)$, is able to perform tasks. A worker is in online mode when the worker is ready to accept tasks; otherwise, the worker is in offline mode. An online worker w is associated with a k-dimensional vector $\boldsymbol{w} \in \mathbb{R}^k$ that describes features of w, e.g., the completion quality of historical tasks, the error rate of completed tasks, and other statistical information derived from historical data, an arrival time $w.a$, and a deadline $w.d$.*

Definition 2 (Task). *A task, denoted by $t = (\boldsymbol{t}, a, d)$, is given by a k'-dimensional vector $\boldsymbol{t} \in \mathbb{R}^{k'}$ that describes the features of the task, where each dimension corresponds to a feature, e.g., task type, task pricing, task popularity, task difficulty, work volume, and skill requirements. A task t also has an arrival time $t.a$, and a task expiration deadline $t.d$.*

Definition 3 (Multivariate Time Series). *A multivariate time series is a time-ordered sequence of vectors or data points, each representing an entity at a specific time instance. In particular, a k-dimensional ($k > 1$) time series $\boldsymbol{S}$ is a sequence of k-dimensional vectors $\boldsymbol{S} = \langle \boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_{|\boldsymbol{S}|} \rangle$, where $|\boldsymbol{S}|$ is the length of $\boldsymbol{S}$, $\boldsymbol{s}_i \in \mathbb{R}^k$ is a k-dimensional vector that describes an entity at time $\mathcal{T}_i$, and each dimension corresponds to a feature.*

We use bold letters, e.g., $\boldsymbol{S}$ and $\boldsymbol{s}$, to denote vectors. In crowdsourcing, a worker multivariate time series is denoted by $\boldsymbol{W} = \langle w_1.\boldsymbol{w}, w_2.\boldsymbol{w}, ..., w_{|\boldsymbol{W}|}.\boldsymbol{w} \rangle$, where $w_i.\boldsymbol{w}$ is a feature vector of worker w_i. Thus, the workers in $\boldsymbol{W}$ arrive at the crowdsourcing platform in the order on their index, i.e., $w_1.a < w_2.a < ... < w_{|\boldsymbol{W}|}.a$. Next, a task multivariate time series is denoted by $\boldsymbol{T} = \langle t_1.\boldsymbol{t}, t_2.\boldsymbol{t}, ..., t_{|\boldsymbol{T}|}.\boldsymbol{t} \rangle$, where each $t_i.\boldsymbol{t}$ is a task feature vector of task t_i.

Definition 4 (Multivariate Time Series Outlier Detection). *Assuming that the current time is $\mathcal{T}_i$ ($i > 1$) and given a historical k-dimensional time series $\boldsymbol{S} = \langle \boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_{i-1} \rangle$, we aim to assign an outlier score to each vector $\boldsymbol{s}_i$ such that the higher the outlier score of $\boldsymbol{s}_i$ is, the more likely it is that $\boldsymbol{s}_i$ is an outlier.*

In crowdsourcing setting, we assign outlier scores to each feature vector $w.\boldsymbol{w}$ and $t.\boldsymbol{t}$ in $\boldsymbol{W}$ and $\boldsymbol{T}$, meaning that we assign outlier scores to the corresponding workers and tasks.

When assigning workers to tasks, the crowdsourcing platform will consider all available workers and tasks at a given time instance, and it assigns a task to each worker. Once a task is assigned to a worker, the worker goes offline until the task is completed. A task t can be assigned to a worker w only if their associated time intervals intersect, i.e., $[w.a, w.d) \cap [t.a, t.d) \neq \emptyset$. In single-task assignment mode, the server assigns one task to a worker at a time.

Definition 5 (Task Assignment). *Given a set of workers W and a set of tasks T, a task assignment A is a set of worker-task pairs in the form of (w, t), where each worker or task can be assigned at most once.*

Each pair (w, t) has a utility $U(w, t)$ that may capture the reward given to worker w for completing task t, the spatial distance that

w must travel to complete t, or similar. Since we study general task assignment, we fix neither the features of workers and tasks, nor the basic utility, which are application specific. Different instances of the framework are obtained when using different configurations.

We use $A.U$ to denote the total utility of all the worker-task pairs in A. The problem investigated is stated as follows.

Problem Statement. Given a set of workers and a set of tasks at the current time instance in a crowdsourcing platform, compute the outlier scores for all workers and tasks and use these to obtain an optimal task assignment A_{opt} that maximizes the total utility of non-outlier workers and tasks, i.e., $\forall\, A_i \in \mathbb{A}\ (A_i.U \leq A_{opt}.U)$, where $\mathbb{A}$ denotes all possible assignments that involve only non-outlier workers and tasks.

3 MALICIOUS WORKERS AND INVALID TASKS DETECTION

Since the worker and task data can be deemed as streaming multivariate time series, and the malicious workers and invalid tasks can be regarded as outliers in time series accordingly, we propose an outlier detector, namely SA-GAN outlier detector, to identify the outliers. We first introduce some preliminaries, then give an overview of the outlier detector, and finally provide specifics on each component in the detector.

3.1 Preliminaries

Generative Adversarial Networks. GANs establish min-max adversarial games between a generator (G) and a discriminator (D), which are typically implemented as neural networks. The generator generates fake samples by capturing the distribution of true samples, and the discriminator, usually a binary classifier, aims to classify samples as real or fake (generated by the generator). Given a k-dimensional time series $S = \langle \boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_{|S|} \rangle$, let $Z = \langle \boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_{|Z|} \rangle$ denote noise samples from a random latent space, where $|S| = |Z|$. Following the architecture of a regular GAN [10], we train G and D with the following two-player minimax game:

$$\min_G \max_D \mathbb{E}_{\boldsymbol{s} \sim p_{data}(\boldsymbol{s})}[\log D(\boldsymbol{s})] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))], \tag{1}$$

where $G(\cdot)$ implicitly defines a probability distribution for the generated samples, $D(\boldsymbol{s})$ represents the probability that $\boldsymbol{s}$ comes from the real data, and $p(\boldsymbol{z})$ is a prior on the input noise variables.

Autoencoders. An autoencoder is a feedforward fully-connected neural network, where the number of neurons in the input and output layers are the same and where there are much fewer neurons in the hidden layers [2, 14, 31]. In particular, given a k-dimensional input vector $\boldsymbol{s}_i = (s_i^1, s_i^2, ..., s_i^k)$, where s_i^j denotes the jth $(1 \leq j \leq k)$ feature value of $\boldsymbol{s}_i$, an autoencoder, consisting of an encoder and an decoder, outputs another k-dimensional vector $\boldsymbol{s}'_i = (s_i'^1, s_i'^2, ..., s_i'^k)$ that is a reproduction of the input. Formally, we define the encoder and decoder by two functions f_e and f_d:

$$f_e(\boldsymbol{s}_i) : \mathbb{R}^k \to \mathbb{R}^m, \quad f_d(\boldsymbol{x}) : \mathbb{R}^m \to \mathbb{R}^k, \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^m$ $(m < k)$ denotes an intermediate m-dimensional vector mapped from $\boldsymbol{s}_i \in \mathbb{R}^k$ by the encoder. Next, the decoder maps $\boldsymbol{x}$ to a reconstructed vector $\boldsymbol{s}'_i \in \mathbb{R}^k$. The objective of an autoencoder is to minimize the reconstruction error between $\boldsymbol{s}_i$ and $\boldsymbol{s}'_i$ using the $L1$ distance as follows:

$$\underset{f_e, f_d}{\operatorname{argmin}} \|\boldsymbol{s}_i - \boldsymbol{s}'_i\|_1 = \underset{f_e, f_d}{\operatorname{argmin}} \sum_{j=1}^{k} |s_i^j - s_i'^j| \tag{3}$$

We can use existing algorithms, e.g., gradient descent and back-propagation, to learn the functions, f_e and f_d.

3.2 Solution Overview

The SA-GAN outlier detector is based on two GANs using CNN-based autoencoders as generators and CNNs as discriminators to capture complex patterns and distributions of time series. We use one-dimensional CNNs with filters sliding along the temporal dimension as base models to capture temporal correlations in time series, as recent studies [21, 23] show that CNNs are more robust than LSTMs for time series. The framework has three components: data processing, model training, and outlier detection, as shown in Figure 2.

The data processing component performs data normalization, negative sampling, and segmentation, which are necessary for model training. The model training component encompasses two adversarial training processes, namely a distribution-based SA-GAN (including the encoder as its generator G_1 and a distribution-based discriminator D_1, marked in green) and a reconstruction-based SA-GAN (including the whole autoencoder as its generator G_2 and a reconstruction-based discriminator D_2, marked in blue), which aim to learn the patterns and distributions of time series. We will elaborate the training in Section 3.4. After the model training, the trained model is transferred to the outlier detector in the outlier detection component. When a new time series arrives, it is first normalized and segmented, and then reconstructed by the outlier detector. Finally the outlier score is computed based on the reconstruction error between the original time series and the reconstructed one.

3.3 Data Processing

Data Normalization. We apply min-max normalization to each dimension of the time series $\boldsymbol{S} = \langle \boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_{|S|} \rangle$:

$$\widetilde{s_i^j} = \frac{s_i^j - \min\{S^j\}}{\max\{S^j\} - \min\{S^j\}}, \tag{4}$$

where $\min\{S^j\}$ and $\max\{S^j\}$ denote the minimal and maximal values of the jth dimension.

Negative Sampling. To address the class-imbalanced problem, where normal samples by far exceed abnormal ones are universal in outlier detection which presents additional challenges to outlier detection in time series, we introduce negative sampling [24] to create negative samples (i.e., abnormal samples) from the observed positive samples (i.e., normal samples) globally and locally when processing the training data. Specifically, we design a simple negative sampling method to obtain synthetic global and local negative samples. Specifically, given a set of positive time series samples $\boldsymbol{S} = \langle \boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_{|S|} \rangle$ from training data, there are sufficient positive samples, but only few negative samples are expected because time series contain only few outliers. The probability that any point drawn from the time series is normal is nearly one. Therefore, we regard range $[\min\{S^j\}, \max\{S^j\}]$ as the normal range for the jth dimension $(1 \leq j \leq k)$, where $\min\{S^j\}$ and $\max\{S^j\}$ denote the

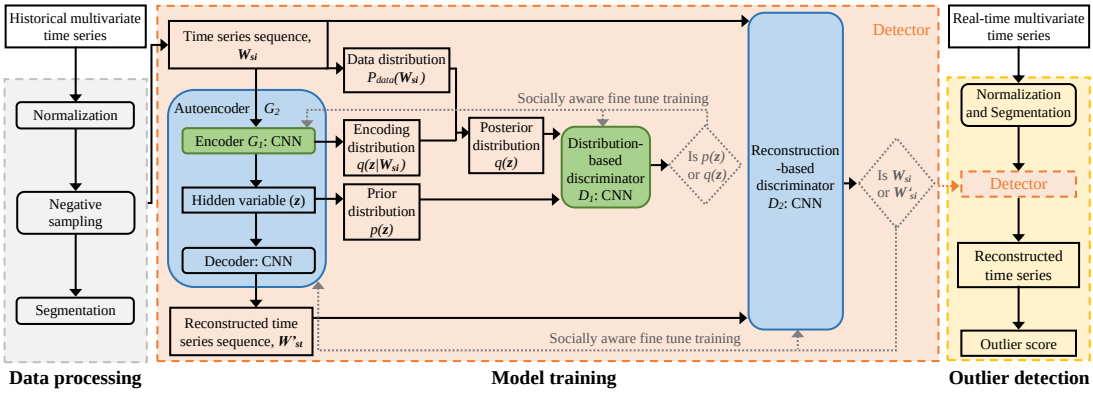


Figure 2: SA-GAN Outlier Detector

minimal and maximal values of the jth dimension. We generate negatives in a range that is slightly larger than the normal range and then use GANs for reconstruction. Specifically, the value in the jth dimension of negative samples $\boldsymbol{S'} = \langle \boldsymbol{s'}_1, \boldsymbol{s'}_2, ..., \boldsymbol{s'}_{|S'|} \rangle$ is chosen independently and uniformly from range $[\min\{S^j\} - \delta, \max\{S^j\} + \delta]$, where δ is a small positive number denoting the deviation of negative samples from the normal range. The sampling ratio, denoted as $R = \frac{|S'|}{|S|}$, is the ratio between the synthetic and the original sample size. We generate negative samples based on the sampling ratio on the whole time series (i.e., in a global manner) or on each time series subsequence segmented by a sliding window (i.e., in a local manner). The choice of global versus local generation needs to be fitted to time series data.

Data Segmentation. Instead of treating each feature independently, the SA-GAN outlier detector considers the entire feature set concurrently to capture latent interactions among features. We divide the time series into subsequences with a sliding window. Given an entity $\boldsymbol{s}_i$ at time $\mathcal{T}_i$, we use $\boldsymbol{W}_{\boldsymbol{s}_i}$ to denote the corresponding subsequence of $\boldsymbol{s}_i$ segmented by a sliding window, where $\boldsymbol{W}_{\boldsymbol{s}_i} = \langle \boldsymbol{s}_{i-|\boldsymbol{W}|+1}, \boldsymbol{s}_{i-|\boldsymbol{W}|+2}, ..., \boldsymbol{s}_i \rangle$ and $|\boldsymbol{W}|$ is the length of $\boldsymbol{W}_{\boldsymbol{s}_i}$.

3.4 Model Training

Distribution-based GAN Training. In the Distribution-based SA-GAN, the encoder is regarded as the generator and takes a time series subsequence $\boldsymbol{W}_{\boldsymbol{s}_i}$ as input and outputs a hidden variable $\boldsymbol{z}$. The Distribution-based SA-GAN aims to guide the matching between the prior distribution $p(\boldsymbol{z})$ and the posterior distribution $q(\boldsymbol{z})$ of the hidden variable $\boldsymbol{z}$, in which $q(\boldsymbol{z})$ is calculated according to Equation 5.

$$q(\boldsymbol{z}) = \int_{\boldsymbol{W}_{\boldsymbol{s}_i}} q\left(\boldsymbol{z} \mid \boldsymbol{W}_{\boldsymbol{s}_i}\right) p_{data}\left(\boldsymbol{W}_{\boldsymbol{s}_i}\right) d\boldsymbol{W}_{\boldsymbol{s}_i}, \tag{5}$$

where $q\left(\boldsymbol{z} \mid \boldsymbol{W}_{\boldsymbol{s}_i}\right)$ is the encoding distribution and $p_{data}\left(\boldsymbol{W}_{\boldsymbol{s}_i}\right)$ is the data distribution. We assume that $q\left(\boldsymbol{z} \mid \boldsymbol{W}_{\boldsymbol{s}_i}\right)$ is a Gaussian distribution and use the parametrization of Kingma and Welling [15] for back-propagation through the encoder.

The distribution-based discriminator D_1 is used to guide the posterior $q(\boldsymbol{z})$ to match the prior $p(\boldsymbol{z})$, which tries to maximize the following loss function:

$$\begin{aligned}\mathcal{L}_{D_1} = & \mathop{\mathbb{E}}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log\left(D_1(\boldsymbol{z})\right)\right] \\ & + \mathop{\mathbb{E}}_{\boldsymbol{W}_{\boldsymbol{s}_i}\sim p_{data}\left(\boldsymbol{W}_{\boldsymbol{s}_i}\right)}\left[\log\left(1 - D_1\left(G_1\left(\boldsymbol{W}_{\boldsymbol{s}_i}\right)\right)\right)\right],\end{aligned} \tag{6}$$

where G_1 is the corresponding generator, i.e., the encoder, which tries to minimize the following loss function:

$$\mathcal{L}_{G_1} = \mathop{\mathbb{E}}_{\boldsymbol{W}_{\boldsymbol{s}_i}\sim p_{data}\left(\boldsymbol{W}_{\boldsymbol{s}_i}\right)}\left[\log\left(1 - D_1\left(G_1\left(\boldsymbol{W}_{\boldsymbol{s}_i}\right)\right)\right)\right] \tag{7}$$

Existing GAN learning strategies pursue convergence to Nash equilibria [1, 10, 41]. In contrast, we incorporate the concept of social awareness into the training process and aim at improving the social welfare of the generator and discriminator under self-play rather than pursuing Nash equilibrium solutions. Considering that social welfare is the common interests or goals between the generator and discriminator, we sum up their objectives, $\max\{\mathcal{L}_{D_1}\}$ and $\min\{\mathcal{L}_{G_1}\}$.

$$\begin{aligned}&\max\left\{\mathcal{L}_{D_1}\right\} + \min\left\{\mathcal{L}_{G_1}\right\} \\ &= \max\Big\{\mathop{\mathbb{E}}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log\left(D_1(\boldsymbol{z})\right)\right] \\ &+ \mathop{\mathbb{E}}_{\boldsymbol{W}_{\boldsymbol{s}_i}\sim p_{data}\left(\boldsymbol{W}_{\boldsymbol{s}_i}\right)}\left[\log\left(1 - D_1\left(G_1\left(\boldsymbol{W}_{\boldsymbol{s}_i}\right)\right)\right)\right]\Big\} \\ &+ \min\Big\{\mathop{\mathbb{E}}_{\boldsymbol{W}_{\boldsymbol{s}_i}\sim p_{data}\left(\boldsymbol{W}_{\boldsymbol{s}_i}\right)}\left[\log\left(1 - D_1\left(G_1\left(\boldsymbol{W}_{\boldsymbol{s}_i}\right)\right)\right)\right]\Big\}\end{aligned} \tag{8}$$

Given that $\min\{\mathcal{L}_{G_1}\} = \max\{-\mathcal{L}_{G_1}\}$, we get

$$\max\left\{\mathcal{L}_{D_1}\right\} + \min\left\{\mathcal{L}_{G_1}\right\} = \max\Big\{\mathop{\mathbb{E}}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log\left(D_1(\boldsymbol{z})\right)\right]\Big\} \tag{9}$$

Equation 9 aims to maximize the probability of recognizing a real sample, which is regarded as the common goal of the generator and discriminator, called a socially aware goal. For the discriminator, we can see from Equation 6 that the socially aware goal is also its individual goal. The generator can generate high-quality fake samples with the social goal since it aims to generate fake samples to fool the discriminator.

Let SG_1 denote $\mathop{\mathbb{E}}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log\left(D_1(\boldsymbol{z})\right)\right]$ and IG_1 denote $\mathop{\mathbb{E}}_{\boldsymbol{W}_{\boldsymbol{s}_i}\sim p_{data}\left(\boldsymbol{W}_{\boldsymbol{s}_i}\right)}\left[\log\left(1 - D_1\left(G_1\left(\boldsymbol{W}_{\boldsymbol{s}_i}\right)\right)\right)\right]$. Based on a regular GAN framework [10], we train the SA-GAN with a two-player min-max game by introducing the socially aware degree $sa_1 \in (0, 1)$, which reflects the socially aware degree of the generator and discriminator, as shown in Equation 10.

$$\min_{G_1}\max_{D_1}\left\{sa_1 \cdot SG_1 + (1 - sa_1) \cdot IG_1\right\}, \tag{10}$$

where sa_1 can be adjusted adaptively based on the relative performance between the generator and discriminator. During each training round, if SG_1 exceeds IG_1, sa_1 increases; otherwise, sa_1 decreases. We adopt two strategies for updating sa_1, namely a *random strategy* (Equation 11) and a *learning rate based strategy*

(Equation 12).

$$sa_1 = \begin{cases} random(sa_1, 1) & \text{if } SG_1 > IG_1 \\ random(0, sa_1) & \text{if } SG_1 < IG_1 \\ sa_1 & \text{otherwise} \end{cases} \tag{11}$$

$$sa_1 = sa_1 + \alpha \cdot \frac{SG_1 - IG_1}{SG_1 + IG_1}, \tag{12}$$

where $random(a, b)$ is a random number from range (a, b), and α is the learning rate of sa_1, which is usually set to 0.001.

Reconstruction-based GAN Training. In this SA-GAN, the autoencoder is regarded as the generator (denoted as G_2), which generates the reconstructed time series, $\boldsymbol{W'}_{\boldsymbol{s}_i} = G_2(G_1(\boldsymbol{W}_{\boldsymbol{s}_i}))$. A reconstruction-based discriminator D_2 that compares the difference between the original and reconstructed time series is proposed to avoid overfitting. Similarly, the reconstruction-based SA-GAN can be trained with the following socially aware two-player minimax game:

$$\min_{G_2} \max_{D_2} \left\{ sa_2 \cdot SG_2 + (1 - sa_2) \cdot IG_2 \right\} \tag{13}$$

$$SG_2 = \mathop{\mathbb{E}}_{\boldsymbol{W}_{\boldsymbol{s}_i} \sim p_{data}(\boldsymbol{W}_{\boldsymbol{s}_i})} \left[\log \left(D_2(\boldsymbol{W}_{\boldsymbol{s}_i}) \right) \right] \tag{14}$$

$$IG_2 = \mathop{\mathbb{E}}_{\boldsymbol{W}_{\boldsymbol{s}_i} \sim p_{data}(\boldsymbol{W}_{\boldsymbol{s}_i})} \left[\log \left(1 - D_2 \left(G_2 \left(G_1(\boldsymbol{W}_{\boldsymbol{s}_i}) \right) \right) \right) \right], \tag{15}$$

where sa_2 is the socially aware degree of the Reconstruction-based SA-GAN.

Autoencoder Training. For the autoencoder training, we use the loss functions of the encoder and decoder, i.e., $\mathcal{L}_{G_1}$ and $\mathcal{L}_{G_2}$, as the adversarial regularization to achieve robustness of the autoencoder, where $\mathcal{L}_{G_2} = \mathop{\mathbb{E}}_{\boldsymbol{W}_{\boldsymbol{s}_i} \sim p_{data}(\boldsymbol{W}_{\boldsymbol{s}_i})} \left[\log \left(1 - D_2 \left(G_2 \left(G_1(\boldsymbol{W}_{\boldsymbol{s}_i}) \right) \right) \right) \right]$. The loss function of the autoencoder is shown in Equation 16.

$$\mathcal{L} = \mathcal{L}_0 + \lambda_1 \mathcal{L}_{G_1} + \lambda_2 \mathcal{L}_{G_2} \tag{16}$$

$$\mathcal{L}_0 = \mathbb{E}_{\boldsymbol{W}_{\boldsymbol{s}_i} \sim p_{data}(\boldsymbol{W}_{\boldsymbol{s}_i})} \left\| \boldsymbol{W}_{\boldsymbol{s}_i} - G_2 \left(G_1(\boldsymbol{W}_{\boldsymbol{s}_i}) \right) \right\|_1, \tag{17}$$

where $\mathcal{L}_0$ is the reconstruction loss, and λ_1 and λ_2 are parameters that control the contributions of $\mathcal{L}_{G_1}$ and $\mathcal{L}_{G_2}$.

To achieve stability during the training, we employ feature matching loss [22] to replace the adversarial regulations, $\mathcal{L}_{G_1}$ and $\mathcal{L}_{G_2}$. Let $F(\cdot)$ be a function of an intermediate layer in the discriminator, which can be regarded as a feature function. The feature matching measures the $L2$ distance between the feature representation of the original and generated data. Thereby, the adversarial regulations, $\mathcal{L}_{G_1}$ and $\mathcal{L}_{G_2}$, can be recomputed as follows:

$$\begin{aligned} \mathcal{L}_{G_1} &= \| \mathbb{E}_{\boldsymbol{W}_{\boldsymbol{s}_i} \sim p_{data}(\boldsymbol{W}_{\boldsymbol{s}_i})} F_1(G_1(\boldsymbol{W}_{\boldsymbol{s}_i})) - \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})} F_1(\boldsymbol{z}) \|_2 \\ \mathcal{L}_{G_2} &= \| \mathbb{E}_{\boldsymbol{W}_{\boldsymbol{s}_i} \sim p_{data}(\boldsymbol{W}_{\boldsymbol{s}_i})} \left(F_2(\boldsymbol{W}_{\boldsymbol{s}_i}) \right. \\ &\quad \left. - F_2 \left(G_2 \left(G_1(\boldsymbol{W}_{\boldsymbol{s}_i}) \right) \right) \right) \|_2, \end{aligned} \tag{18}$$

where $F_1(\cdot)$ and $F_2(\cdot)$ are the functions of the intermediate layer of D_1 and D_2, respectively. The autoencoder and the two discriminators are trained simultaneously.

3.5 Outlier Detection

Upon training we can use the SA-GAN model to detect outliers. Taking the time series subsequence $\boldsymbol{W}_{\boldsymbol{s}_i} = \langle \boldsymbol{s}_{i-|\boldsymbol{W}|+1}, \boldsymbol{s}_{i-|\boldsymbol{W}|+2}, ..., \boldsymbol{s}_i \rangle$ as input, the detector aims to reconstruct $\boldsymbol{s}_i$ and uses the reconstruction error to compute its outlier score $Score(\boldsymbol{s}_i)$, which can be calculated as follows:

$$Score(\boldsymbol{s}_i) = \left\| \boldsymbol{s}_i - \boldsymbol{s'}_i \right\|_1, \tag{19}$$

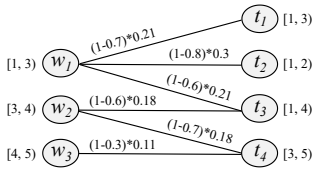


Figure 3: Worker-Task Bipartite Graph

where $\boldsymbol{s'}_i$ is the reconstruction of $\boldsymbol{s}_i$. A higher outlier score means that $\boldsymbol{s}_i$ is more likely to be an outlier.

4 TASK ASSIGNMENT

We propose a greedy algorithm based on degree reduction by transforming task assignment into a bipartite graph matching.

4.1 Bipartite Graph Generation

We first transform the task assignment problem in time instance $\mathcal{T}$ into a bipartite graph. The graph is represented by $G = (V, E)$, where V includes a left node set W (that is the worker set in time instance $\mathcal{T}$) and a right node set T (that is the task set in time instance $\mathcal{T}$), and E denotes the set of edges. Each node $w \in W$ ($t \in T$) has an arrival time denoted by $w.a$ ($t.a$), and a deadline denoted by $w.d$ ($t.d$). Each edge $e(w, t) \in E$ has a basic utility denoted by $U(w, t)$, which is application-specific and can be set by task reward, spatial distance, etc. Inevitably, in crowdsourcing applications, malicious workers and invalid tasks may exist. In such situations, workers cannot finish the assigned tasks on time or cannot complete the tasks correctly. Therefore, we consider the outlier factor in task assignment. More specifically, each (w, t) pair has an outlier-based utility, denoted by $\mathcal{U}(w, t) = (1 - \max\{Score(w.\boldsymbol{w}), Score(t.\boldsymbol{t})\}) * U(w, t)$, where $max\{Score(w.\boldsymbol{w}), Score(t.\boldsymbol{t})\}$ denotes the maximum outlier score among worker w and task t that can be calculated by the SA-GAN outlier detector in Section 3. A worker-task pair with a lower outlier score is more likely to have a higher outlier-based utility. Note that there is no need to consider the edges $e(w, t)$ with $[w.a, w.d) \cap [t.a, t.d) = \emptyset$ for matching since in this case one node expires before the other arrives. A task assignment on a bipartite graph G is denoted by $A = \{(w, t) | w \in W, t \in T\}$. It is a set of node pairs where each node appears at most once.

Figure 3 shows a worker-task bipartite graph, where $\{w_1, w_2, w_3\}$ is the left node set, and $\{t_1, t_2, t_3, t_4\}$ is the right node set. An edge exists between w_i ($i = 1, 2, 3$) and t_j ($j = 1, 2, 3, 4$) when $[w_i.a, w_i.d) \cap [t_j.a, t_j.d) \neq \emptyset$. For example, w_1 and t_2 has an edge (with the outlier-based utility $(1 - 0.8) * 0.3 = 0.06$) between them as $[1, 3) \cap [1, 2) \neq \emptyset$.

To optimize the total utility of normal workers and tasks, we aim to maximize the total outlier-based utility, which gives priority to normal workers and tasks during task assignment.

4.2 Degree-Reduction-based Greedy Algorithm

The greedy algorithm is to utilize a degree reduction strategy, which consists of the following steps.

Step 1. Calculate the degree of each node.

Step 2. Find the node $v \in W \cup T$ with the minimal degree, and denote the nodes that are connected with v as $C(v)$, i.e., $\forall u \in C(v)$, $e(v, u)$ exists. Then perform the following actions.

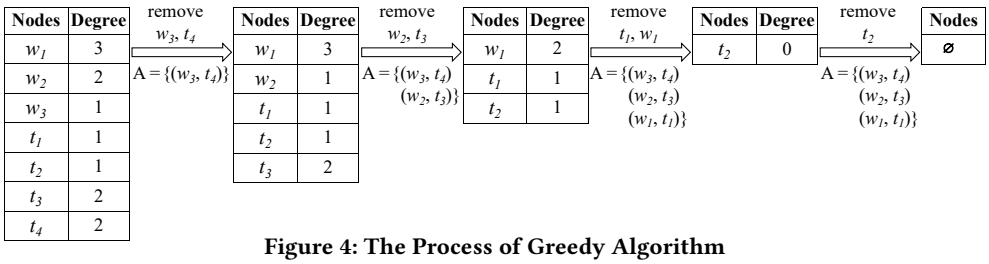


Figure 4: The Process of Greedy Algorithm

1) If node $u \in C(v)$ with the maximal weight can be found, (v, u) is a worker-task matching and added into task assignment A. Nodes v and u are then removed from $W \cup T$. Update the degree of nodes in $C(v)$ and $C(u)$, i.e., the degree of all the nodes in $C(v)$ and $C(u)$ should be reduced by 1.

2) Otherwise (i.e., $C(v) = \emptyset$), node v is removed from $W \cup T$. Update the degree of nodes in $C(v)$.

Step 3. Repeat *Step 2* until $W \cup T = \emptyset$.

The intuition of the above algorithm is that the nodes with less edges are more likely to be assigned unsuccessfully when they are assigned later, so we give priority to them. The time complexity for this algorithm is $O((|W| + |T|) * |maxC|)$, where $|maxC| = \max_{v \in W \cup T} |C(v)|$.

Figure 4 illustrates the degree reduction process for the worker-task bipartite graph in Figure 3, which first removes nodes w_3 and t_4 since w_3 is the node with the minimal degree and $t_4 \in C(w_3)$ ($C(w_3) = \{t_4\}$) has the maximal outlier-based utility. Then we add the worker-task pair (w_3, t_4) into the task assignment A, i.e., $A = \{(w_3, t_4)\}$. Accordingly, w_3 and t_4 are removed from the graph, and the degree of nodes in $C(w_3)$ and $C(t_4)$ are reduced by 1. Subsequently, nodes w_2, t_3, t_1, w_1, t_2 are removed, respectively, following the principle of *Step 2*. Finally, we obtain the task assignment $A = \{(w_3, t_4), (w_2, t_3), (w_1, t_1)\}$.

5 EXPERIMENTAL EVALUATION

We evaluate the performance of the malicious workers and invalid tasks detection, and the task assignment. Due to the lack of benchmark for streaming crowsourcing task assignment algorithms, we use two real multivariate time series datasets, MSL and SMAP [12], to simulate the streaming crowdsourcing scenario. The consecutive points with multiple dimensions in the real time series data are good candidates for workers and tasks (with multiple features) which appear continuously and in time sequences on crowdsourcing platforms. The datasets are described in Appendix B.1.

5.1 Performance of Malicious Workers and Invalid Tasks Detection

Evaluation Methods. We evaluate the following methods: VAE [15], EncDec-AD [19], GANomaly [1], BeatGAN [41], SA-GAN-Random (our SA-GAN model using the random strategy to update the socially aware degree), and SA-GAN-Learn (our SA-GAN model using the learning rate based strategy to update the socially aware degree), where the details are given in Appendix B.2.

Metrics. We adopt the commonly-used metric, *F1-score (F1)*, to evaluate the accuracy of the six methods, where the outlier threshold selection is shown in Appendix B.3. The larger the *F1* is, the more accurate the method is. We also evaluate the efficiency, including the training and testing time. The parameter settings are shown in Appendix B.4.

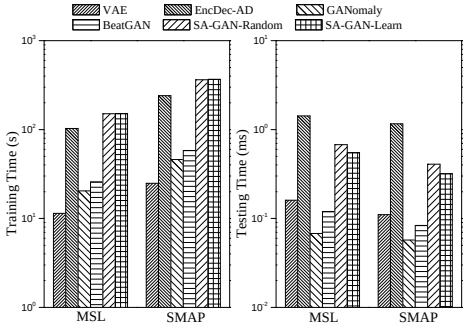


Figure 5: Training Time and Testing Time

Table 1: F1-score on Two Datasets

Models	F1		
	MSL	SMAP	Total
VAE	0.361	0.332	0.336
EncDec-AD	0.243	0.193	0.199
GANomaly	0.317	0.327	0.325
BeatGAN	0.308	0.304	0.323
SA-GAN-Random	0.335	0.349	**0.354**
SA-GAN-Learn	**0.363**	**0.357**	0.331

Accuracy. We report the *F1* values in Table 1. The best performance by an existing method (VAE, EncDec-AD, GANomaly, and BeatGAN) is underlined, and the overall best performance is marked in bold. We also calculate the total *F1* values, which is computed based on the total number of correctly detected outliers, the total number of falsely detected outliers, and the total number of falsely assigned normal entities on the two datasets. For both datasets, SA-GAN-Learn achieves the highest F1-score, which outperforms the best among the baseline methods by 0.55% and 7.53% in MSL and SMAP, respectively. SA-GAN-Random achieves competitive accuracy in MSL and has a higher *F1*-score than other start-of-the-art methods in SMAP. When considering the total accuracy, SA-GAN-Random shows the best performance. Overall, the SA-GANs can achieve high accuracy.

Efficiency. As efficiency is important for online outlier detection, we study the training time (of each epoch) and testing time (of each entity) for all the methods on two datasets. From Figure 5, we can see that although VAE, GANomaly, and BeatGAN take less time for training and testing, they perform worse than the SA-GANs in terms of accuracy, shown in Table 1. Figure 5 also shows that SA-GAN methods run in less than a millisecond when detecting outliers, which indicates their feasibility in real outlier detection scenarios.

5.2 Performance of Task Assignment

Evaluation Methods. We study the following algorithms

1) KM: The Kuhn-Munkras algorithm [16] that achieves the optimal assignment without considering outliers, i.e., malicious workers and invalid tasks.

2) DR: The Degree-Reduction-based greedy algorithm without considering outliers.

3) KM-Outlier: The KM algorithm considering outliers.

4) DR-Outlier: The DR algorithm considering outliers.

Metrics. Four main metrics are compared for the above algorithms, i.e., total utility of the correct task assignments (marked as *total utility*), assignment accuracy of workers that is the ratio between the number of correct assigned workers and the total number of assigned workers (marked as *worker assignment accuracy*), assignment accuracy of tasks that is the ratio between the number of correct assigned tasks and the total number of assigned tasks (marked as *task assignment accuracy*), and CPU time for finding task assignments. A correct assigned worker or task means that the assigned worker or task is normal. A larger *worker* or *task assignment accuracy* implies more accurate assignments.

Effect of $|W|$. We first study the effect of $|W|$. In Figures 6(a) and 7(a), the utilities for all methods naturally increase as $|W|$ gets larger. Although DR and DR-Outlier generate less utility than KM and KM-Outlier, they run much faster than KM and KM-Outlier (cf. Figures 6(d) and 7(d)). The CPU time of DR is only 6.79%–34.09% of those of KM and KM-Outlier, and the CPU time of DR-Outlier is only 7.90%–38.03% of those of KM and KM-Outlier, which demonstrate the superiority of DR and DR-Outlier for solving the streaming task assignment problem. For worker and task assignment accuracy—cf. Figures 6(b), 6(c), 7(b), and 7(c), the outlier-based methods (i.e., KM-Outlier and DR-Outlier) outperform their counterparts (i.e., KM and DR) in most situations. In some cases when $|W|$ is small, the outlier-based methods have similar or same assignment accuracy with their counterparts since there are fewer outliers with fewer workers, where the outlier-based methods fail to show their superiority. To save space, in the following experiments, we do not report results for SMAP, as these are similar to those obtained for MSL. The results for SMAP are given Appendix C.

Effect of $|S|$. In Figure 8(a), the utilities of all methods increase when $|S|$ grows. The reason behind it is self-evident, that is, with more tasks to be assigned, each worker receives more available tasks such that a worker has more chance to be assigned a task with higher utility. Apparently, KM achieves the highest total utility, followed by KM-Outlier, DR, and DR-Outlier. In Figures 8(b) and 8(c), the outlier-based methods perform better than their counterparts or perform similar with them in terms of worker and task assignment accuracy. This is mainly because outliers are much fewer than the normal data points (i.e., normal workers and tasks), which is less likely to be assigned. Besides, it is difficult to detect all the outliers correctly by outlier detection methods including our SA-GAN outlier detector, which also leads to the above results. For efficiency, the DR-related methods still run much faster than the KM-related methods in Figure 8(d). Besides, the CPU time of DR and DR-Outlier remain almost unchanged regardless of $|S|$, demonstrating the adaptability of DR-related methods in different data-intensive scenarios.

Effect of x. We proceed to consider the effect of x, the valid time of workers and tasks. As expected, the total utilities of all methods increase gradually as x grows, shown in Figure 9(a). This is because when x increases, there will be more available tasks for each worker. Workers then have more choices, which may lead to the increasing utilities. For worker and task assignment accuracy, KM-Outlier (DR-Outlier) clearly outperforms KM (DR) in Figures 9(b) and 9(c). Figure 9(d) shows that the CPU time of KM and KM-Outlier exhibit an increasing trend when increasing x, and become more and more time-consuming compared to the others.

6 RELATED WORK

Recent studies in crowdsourcing provide solutions to a variety of task assignment problems [3–5, 17, 18, 25, 26, 29, 30, 32–39]. However, none of these studies target robustness towards malicious actors, including malicious workers and task requesters, thus making them vulnerable to malicious behavior. Quality assurance is a major challenge in crowdsourcing. Specifically, tasks may be completed with poor results if workers are malicious or if tasks cannot reasonably be completed by their deadlines or if completions carry little benefit to workers. A few studies consider malicious user behavior [8, 13, 20, 28]. However, these studies either focus on a specific task type or rely solely on the error rate of task results to detect malicious workers and do not consider other factors. Moreover, these studies do not consider malicious task requesters and thus do not account for the combination of malicious workers and tasks submitted by such requesters in task assignment.

To better ensure the task assignment quality, it is attractive to integrate mechanisms that can estimate worker and task quality accurately, so that low-quality workers and tasks can be disregarded. Complementing existing studies, our study aims to detect both malicious workers and task requesters to enable more robust task assignment in crowdsourcing.

7 CONCLUSION

We propose a general efficient outlier detection framework for streaming task assignment in crowdsourcing, that aims to achieve high utility among normal workers and tasks while considering outliers including malicious workers and invalid tasks. Specifically, we propose an unsupervised outlier detection architecture that enables the capture of complex patterns and distributions in the worker and task multivariate time series to detect malicious workers and invalid tasks. Besides, we propose a greedy algorithm based on degree reduction to assign tasks to suitable workers to adapt to the real streaming corwdsourcing applications. An extensive empirical study offers evidence that the framework is capable of advancing the state of the art in terms of outlier detection accuracy, assignment accuracy, and computational efficiency.

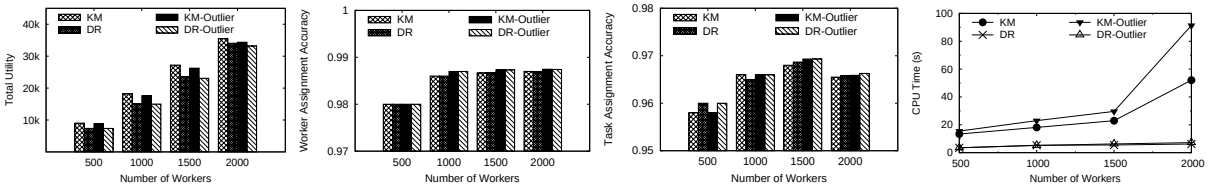
ACKNOWLEDGMENTS

This work is partially supported by NSFC (No. 61972069, 61836007 and 61832017), and Shenzhen Municipal Science and Technology R&D Funding Basic Research Program (JCYJ20210324133607021).

REFERENCES

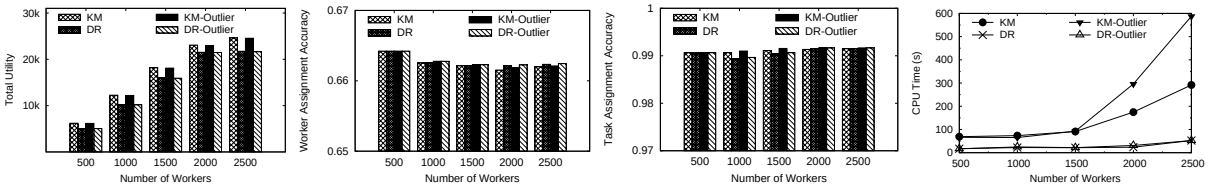
[1] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. 2018. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *ACCV*. 622–637.

[2] Xuanhao Chen, Liwei Deng, Feiteng Huang, Chengwei Zhang, Zongquan Zhang, Yan Zhao, and Kai Zheng. 2021. DAEMON: Unsupervised Anomaly Detection and Interpretation for Multivariate Time Series. In *ICDE*. 2225–2230.



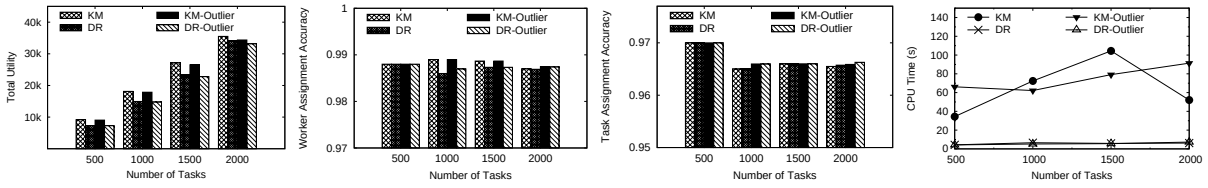
(a) Total Utility (b) Worker Assignment Accuracy (c) Task Assignment Accuracy (d) CPU Time (s)

Figure 6: Effect of $|W|$ on MSL



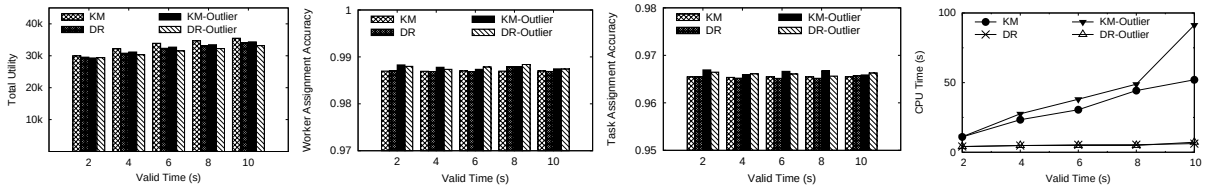
(a) Total Utility (b) Worker Assignment Accuracy (c) Task Assignment Accuracy (d) CPU Time (s)

Figure 7: Effect of $|W|$ on SMAP



(a) Total Utility (b) Worker Assignment Accuracy (c) Task Assignment Accuracy (d) CPU Time (s)

Figure 8: Effect of $|S|$ on MSL



(a) Total Utility (b) Worker Assignment Accuracy (c) Task Assignment Accuracy (d) CPU Time (s)

Figure 9: Effect of x on MSL

[3] Peng Cheng, Xiang Lian, Lei Chen, and Cyrus Shahabi. 2017. Prediction-Based Task Assignment in Spatial Crowdsourcing. In *ICDE*. 997–1008.

[4] Peng Cheng, Xiang Lian, Zhao Chen, Rui Fu, Lei Chen, Jinsong Han, and Jizhong Zhao. 2015. Reliable Diversity-based Spatial Crowdsourcing by Moving Workers. *VLDBJ* 8, 10 (2015), 1022–1033.

[5] Yue Cui, Liwei Deng, Yan Zhao, Bin Yao, Vincent W Zheng, and Kai Zheng. [n. d.]. Hidden poi ranking with spatial crowdsourcing. In *KDD*.

[6] C. Eickhoff and A. de Vries. 2011. How crowdsourcable is your task. In *WSDM*. 11–14.

[7] Lee Erickson, Irene Petrick, and Eileen Trauth. 2012. Hanging with the right crowd: Matching crowdsourcing need to crowd characteristics.. In *AMCIS*. 1–9.

[8] Ujwal Gadiraju, Ricardo Kawase, Stefan Dietze, and Gianluca Demartini. 2015. Understanding malicious behavior in crowdsourcing platforms: The case of online surveys. In *CHI*. 1631–1640.

[9] R. Gennaro, C. Gentry, and B. Parno. 2010. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*. 465–482.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *NIPS* 27 (2014), 2672–2680.

[11] Chien-Ju Ho and Jennifer Vaughan. 2012. Online task assignment in crowdsourcing markets. In *AAAI*. 45–51.

[12] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *SIGKDD*. 387–395.

[13] P. G. Ipeirotis, F. Provost, and J. Wang. 2010. Quality management on amazon mechanical turk. In *SIGMOD Workshops*. 64–67.

[14] Tung Kieu, Bin Yang, Chenjuan Guo, Razvan-Gabriel Cirstea, Yan Zhao, Yale Song, and Christian S. Jensen. 2022. Anomaly Detection in Time Series with Robust Variational Quasi-Recurrent Autoencoders.. In *ICDE*.

[15] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[16] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[17] Xiang Li, Yan Zhao, Jiannan Guo, and Kai Zheng. 2020. Group task assignment with social impact-based preference in spatial crowdsourcing. In *DASFAA*. 677–693.

[18] Xiang Li, Yan Zhao, Xiaofang Zhou, and Kai Zheng. 2020. Consensus-Based Group Task Assignment with Social Impact in Spatial Crowdsourcing. *Data Science and Engineering* 5, 4 (2020), 375–390.

[19] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148* (2016).

[20] David Oleson, Alexander Sorokin, Greg Laughlin, Vaughn Hester, John Le, and Lukas Biewald. 2011. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. In *AAAI Workshops*. 43–48.

[21] Pranav Rajpurkar, Awni Y Hannun, Masoumeh Haghpanahi, Codie Bourn, and Andrew Y Ng. 2017. Cardiologist-level arrhythmia detection with convolutional neural networks. *arXiv preprint arXiv:1707.01836* (2017).

[22] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498* (2016).

[23] Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman. 2017. Stock price prediction using LSTM, RNN and CNN-sliding window model. In *ICACCI*. 1643–1647.

[24] John Sipple. 2020. Interpretable, multidimensional, multimodal anomaly detection with negative sampling for detection of device failure. In *ICML*. 9016–9025.

[25] Yongxin Tong, Jieying She, Bolin Ding, and Libin Wang. 2016. Online Mobile Micro-Task Allocation in Spatial Crowdsourcing. In *ICDE*. 49–60.

[26] Yongxin Tong, Libin Wang, Zimu Zhou, Bolin Ding, Lei Chen, Jieping Ye, and Ke Xu. 2017. Flexible Online Task Assignment in Real-time Spatial Data. *PVLDB* 10, 11 (2017), 1334–1345.

[27] Yongxin Tong, Zimu Zhou, Yuxiang Zeng, Lei Chen, and Cyrus Shahabi. 2019. Spatial Crowdsourcing: A Survey. *VLDBJ* 29, 1 (2019), 217–250.
[28] Jing Wang, Panagiotis G Ipeirotis, and Foster Provost. 2011. Managing crowdsourcing workers. In *Conference on Business Intelligence*. 10–12.
[29] Ziwei Wang, Yan Zhao, Xuanhao Chen, and Kai Zheng. 2021. Task Assignment with Worker Churn Prediction in Spatial Crowdsourcing. In *CIKM*.
[30] Jinfu Xia, Yan Zhao, Guanfeng Liu, Jiajie Xu, Min Zhang, and Kai Zheng. 2019. Profit-driven Task Assignment in Spatial Crowdsourcing.. In *IJCAI*. 1914–1920.
[31] Yan Xia, Xudong Cao, Fang Wen, Gang Hua, and Jian Sun. 2015. Learning discriminative reconstructions for unsupervised outlier removal. In *ICCV*. 1511–1519.
[32] Guanyu Ye, Yan Zhao, Xuanhao Chen, and Kai Zheng. 2021. Task Allocation with Geographic Partition in Spatial Crowdsourcing. In *CIKM*.
[33] Yan Zhao, Jiannan Guo, Xuanhao Chen, Jianye Hao, Xiaofang Zhou, and Kai Zheng. 2021. Coalition-based task assignment in spatial crowdsourcing. In *ICDE*. 241–252.
[34] Yan Zhao, Yang Li, Yu Wang, Han Su, and Kai Zheng. 2017. Destination-aware Task Assignment in Spatial Crowdsourcing. In *CIKM*. 297–306.
[35] Yan Zhao, Jinfu Xia, Guanfeng Liu, Han Su, Defu Lian, Shuo Shang, and Kai Zheng. 2019. Preference-aware task assignment in spatial crowdsourcing. In *AAAI*. 2629–2636.
[36] Yan Zhao, Kai Zheng, Yue Cui, Han Su, Feida Zhu, and Xiaofang Zhou. 2020. Predictive task assignment in spatial crowdsourcing: a data-driven approach. In *ICDE*. 13–24.
[37] Yan Zhao, Kai Zheng, Jiannan Guo, Bin Yang, Torben Bach Pedersen, and Christian S Jensen. 2021. Fairness-aware Task Assignment in Spatial Crowdsourcing: Game-Theoretic Approaches. In *ICDE*. 265–276.
[38] Yan Zhao, Kai Zheng, Yang Li, Han Su, Jiajun Liu, and Xiaofang Zhou. 2019. Destination-aware Task Assignment in Spatial Crowdsourcing: A Worker Decomposition Approach. *TKDE* (2019), 2336–2350.
[39] Yan Zhao, Kai Zheng, Hongzhi Yin, Guanfeng Liu, Junhua Fang, and Xiaofang Zhou. 2020. Preference-aware task assignment in spatial crowdsourcing: from individuals to groups. *TKDE* (2020).
[40] Yudian Zheng, Jiannan Wang, Guoliang Li, Reynold Cheng, and Jianhua Feng. 2015. QASCA: A quality-aware task assignment system for crowdsourcing applications. In *SIGMOD*. 1031–1046.
[41] Bin Zhou, Shenghua Liu, Bryan Hooi, Xueqi Cheng, and Jing Ye. 2019. BeatGAN: Anomalous Rhythm Detection using Adversarially Generated Time Series.. In *IJCAI*. 4433–4439.

A NOTATION

Table 2 lists notation used throughout the paper.

Table 2: Summary of Notation

Symbol	Definition
w	A worker
$w.\boldsymbol{w}$	k-dimensional vector describing the features of worker w
$w.a$	Arrival time of worker w
$w.d$	Deadline of worker w
W	A worker set
t	A task
$t.\boldsymbol{t}$	k'-dimensional vector describing the features of task t
$t.a$	Arrival time of task t
$t.d$	Deadline of task t
T	A task set
S	A k-dimensional ($k > 1$) time series
s	A k-dimensional ($k > 1$) vector
$\mathcal{T}$	A time instant
$\boldsymbol{W}$	A worker multivariate time series
$\boldsymbol{T}$	A task multivariate time series
(w, t)	A pair of a worker w and a task t
A	A task assignment
$U(w, t)$	Utility of worker-task pair (w, t)
$A.U$	The total utility in task assignment A
A_{opt}	The optimal task assignment
$\mathbb{A}$	A task assignment set

B EXPERIMENTAL SETUP

We conduct the experiments using PyTorch (v1.0.0 with Python 3.6.4) on a Linux (EulerOS 4.8.5) machine with NVIDIA Tesla P100 16GB HBM2 GPU and 64G memory.

B.1 Datasets

We perform experiments to evaluate the malicious workers and invalid tasks detection on two datasets: MSL (Mars Science Laboratory rover), and SMAP (Soil Moisture Active Passive satellite) [12]. MSL and SMAP include training and testing sets, where outliers in each testing sets are labeled. We train models on each dataset, the statistics of which are shown in Table 3 including the subset number, dimension number, training set size, testing set size, and outlier ratio.

For evaluating the performance of task assignment, we choose two time series (i.e., two subsets) with the similar length from MSL and SMAP to represent the worker and task time series, respectively. Specifically, we use the subset "T-5" ("E-3") in MSL (SMAP) to represent a worker time series and the subset "T-4" ("E-6") in MSL (SMAP) to represent a task time series, each point denoting a worker or a task and the vector denoting the feature vector of the worker or the task accordingly. Since the original datasets do not contain the arrival time and deadline of workers and tasks, we synchronize the worker and task time series, i.e., the first point is set to 0s, and the time gap between two consecutive points is set to 1s, so that each point has a time instance that is regarded as the arrival time of the corresponding worker or task. Then we generate the deadline of each worker or task from $(a, a + x]$ randomly, where a is the arrival time of the worker or task, and x ($x = 2s, 4s, 6s, 8s, 10s$ with

Table 3: Statistics of Datasets

Dataset name	Subset number	Dimension number	Training set size	Testing set size	Outlier ratio
MSL	27	55	58,317	73,729	10.72%
SMAP	55	25	135,183	427,617	13.13%

Table 4: Negative Sampling Settings

Datasets	R	δ	Sampling manners
MSL	0.03	0.05	local
SMAP	0.04	0.01	global

Table 5: Parameters Settings in Task Assignment

Parameter	Value
Number of workers $\|W\|$ (MSL)	500, 1000, 1500, 2000
Number of workers $\|W\|$ (SMAP)	500, 1000, 1500, 2000, 2500
Number of tasks $\|S\|$ (MSL)	500, 1000, 1500, 2000
Number of tasks $\|S\|$ (SMAP)	500, 1000, 1500, 2000, 2500
Valid time x of workers and tasks(MSL/SMAP)	2s, 4s, 6s, 8s, 10s

a default value of 10s) denotes the valid time of the worker or task. The basic utility of each worker or task is generated randomly from [10, 20]. Moreover, we set the granularity of a time instance as 2000s (2500s) in MSL (SMAP), during which the available workers and tasks will be packed and input to our framework. We run the task assignment algorithms over 2000s (7500s) in MSL (SMAP), and report the average results.

B.2 Evaluation Methods of Malicious Workers and Invalid Tasks Detection

We evaluate the following methods when evaluating the performance of malicious workers and invalid tasks detection.

1) VAE: a Variational Autoencoder [15], where the network structure of VAE is same with that of SA-GANs.

2) EncDec-AD: an encoder-decoder-based seq2seq model [19], where the hidden layer is composed of LSTM units.

3) GANomaly: a conditional GAN model [1], which jointly learns the generation of data and the latent space.

4) BeatGAN: a GAN model [41], which employs an adversarial generation approach to reconstructed data.

5) SA-GAN-Random: our SA-GAN model using the random strategy to update the socially aware degree.

6) SA-GAN-Learn: our SA-GAN model using the learning rate based strategy to update the socially aware degree.

B.3 Outlier Threshold Selection

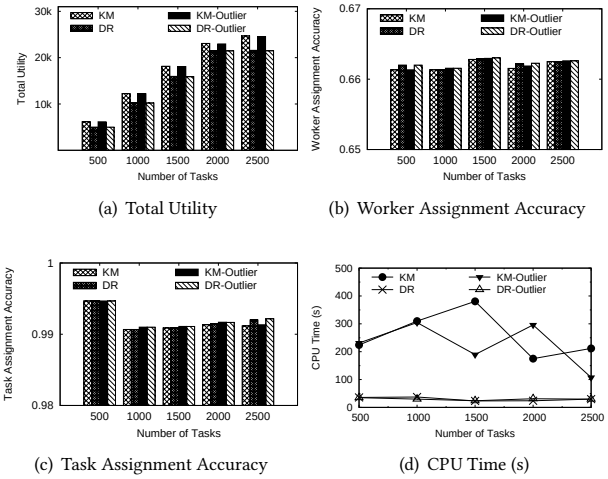
The outlier threshold is generated from the range [0, 1] with the initial value 0 and step 0.01, which means the threshold is 0, 0.01, 0.02, ..., 1. The outlier score of each method is normalized to the unit interval using min-max scaling. We compute the *F1* value with each threshold and report the best *F1* value.

B.4 Parameter Settings

Parameter settings of Malicious Workers and Invalid Tasks Detection. For all models, we use *Adam* as the optimizer, of which the momentums β_1 is set to 0.9 and β_2 is set to 0.999. The batch size, sequence length, and the number of training iterations are set to 50, 128, and 20, respectively. The settings (including sampling ratio R, deviation of negative samples δ, and sampling manners) of negative sampling is shown in Table 4.

For SA-GANs, the learning rate of two discriminators are set to 0.0003, and that of the autoencoder is set to 0.001. We set $\lambda_1 = 1$ and $\lambda_2 = 1$ in MSL, and $\lambda_1 = 1$ and $\lambda_2 = 0.1$ in SMAP. Next, we elaborate the detailed settings of the autoencoder and two discriminators in SA-GANs.

In the autoencoder, G_1 consists of three parts: the basic encoder part, the mean part, and the standard deviation part. The kernel's size and number of each layer of the basic encoder part are $32(4/2/1) - 64(4/2/1) - 128(4/2/1) - 256(4/2/1) - 512(4/2/1)$, along with the mean part $128(4/1/0)$ and the standard deviation part $128(4/1/0)$, where $32(4/2/1)$ means that the number of filters is 32, the size of each filter is 4, the stride is 2, and the padding is 1. G_2 consists of six one-dimensional transposed convolutional layers, i.e., $512(4/1/0) - 256(4/2/1) - 128(4/2/1) - 64(4/2/1) - 32(4/2/1) - M(4/2/1)$, where M is the number of dimensions of the input data. For two discriminators, D_1 and D_2 have the same structure with the basic encoder part, and have a layer with the structure of $1(4/1/0)$ and Sigmoid activation.



(a) Total Utility (b) Worker Assignment Accuracy

(c) Task Assignment Accuracy (d) CPU Time (s)

Figure 10: Effect of $|S|$ on SMAP

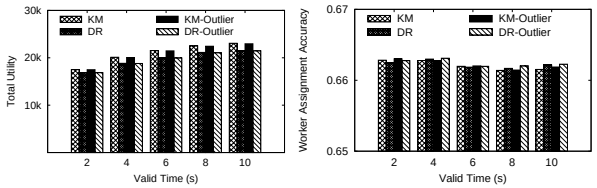
Parameter settings of Task Assignment. Table 5 shows the parameter settings of the task assignment phase, where the default values are underlined on both MSL and SMAP.

C EXPERIMENTAL RESULTS FOR SMAP

We give the results of effect of $|S|$ and x for SMAP in Figures 10 and 11.

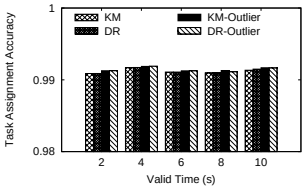
Effect of $|S|$. As expected, the total utilities of all methods increase as $|S|$ grows (see Figure 10(a)). This is because that with more tasks to be assigned, workers tend to have more available tasks to choose to increase the total utilities of of the correct task assignments. Figures 10(b) and 10(c) show that the methods considering outliers, i.e., KM-Outlier and DR-Outlier, perform better than their counterparts in terms of worker and task assignment accuracy. For efficiency, Figure 10(d) shows that our DR-related methods have great superiority compared with the others.

Effect of x. We also study the effect of x for SMAP. As can be seen in Figures 11(a), 11(b), and 11(c), KM-Outlier and DR-Outlier can achieve higher worker/task assignment accuracy than KM and DR, respectively, while sacrificing some utilities. The efficiency of DR-related methods is stable regardless of x, while KM-related methods deteriorate at a significantly faster pace in respect of efficiency, which are shown in Figure 11(d). Overall, our proposed DR-Outlier method can provide an excellent trade-off between effectiveness and efficiency.

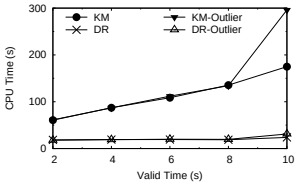


(a) Total Utility

(b) Worker Assignment Accuracy



(c) Task Assignment Accuracy



(d) CPU Time (s)

Figure 11: Effect of x on SMAP