A Perfect Match: Deep Learning Towards Enhanced Data Trustworthiness in Crowd-Sensing Systems

Afzal-Houshmand, Sam; Homayoun, Sajad; Giannetsos, Thanassis

Published in:
Proceedings of IEEE International Mediterranean Conference on Communications and Networking

Link to article, DOI:
10.1109/MeditCom49071.2021.9647554

Publication date:
2021

Document Version
Peer reviewed version

A Perfect Match: Deep Learning Towards Enhanced Data Trustworthiness in Crowd-Sensing Systems

Sam Afzal-Houshmand*, Sajad Homayoun*, Thanassis Giannetsos‡
*Technical University of Denmark (DTU), Cyber Security Section, Denmark
‡Ubitech Ltd., Digital Security & Trusted Computing Group, Greece
Email: saaf@dtu.dk, sajho@dtu.dk, agiannetsos@ubitech.eu

***Abstract*—The advent of IoT edge devices has enabled the collection of rich datasets, as part of Mobile Crowd Sensing (MCS), which has emerged as a key enabler for a wide gamut of safety-critical applications ranging from traffic control, environmental monitoring to assistive healthcare. Despite the clear advantages that such unprecedented quantity of data brings forth, it is also subject to inherent data trustworthiness challenges due to factors such as malevolent input and faulty sensors. Compounding this issue, there has been a plethora of proposed solutions, based on the use of traditional machine learning algorithms, towards assessing and sifting faulty data without any assumption on the trustworthiness of their source. However, there are still a number of open issues: how to cope with the presence of strong, colluding adversaries while at the same time efficiently managing this high influx of incoming user data. In this work, we meet these challenges by proposing the hybrid use of Deep Learning schemes (i.e., LSTMs) and conventional Machine Learning classifiers (i.e. One-Class Classifiers) for detecting and filtering out false data points. We provide a prototype implementation coupled with a detailed performance evaluation under various (attack) scenarios, employing both real and synthetic datasets. Our results showcase how the proposed solution outperforms various existing resilient aggregation and outlier detection schemes.**

***Index Terms*—Mobile Crowd Sensing, Adversarial Machine Learning, Data Trustworthiness, LSTM, One-Class Classifier**

I. Introduction

The emergence of resource-rich devices has changed the landscape of mobile sensing with multiple embedded sensors, e.g., accelerometers, cameras, etc. With more than 6 billion mobile subscriptions worldwide, these sensors (collectively) can be used to sense the environment and gather valuable data of unprecedented quality and quantity, practically from everywhere. This new sensing paradigm, *Mobile Crowd Sensing* (MCS) [1], makes individuals and user communities the focal point of the sensing infrastructure.

Despite all benefits of MCS, its applications still face the critical challenges of data quality and integrity of sensed data [2] and could suffer from incorrect contributions due to their inherent open nature; Over the past decade machine learning has been among the top methods to gain hidden insights from IoT data. ML systems are trained using datasets that are assumed to be *representative* and *trustworthy* whilst malicious actors can impact the decision-making algorithms by either targeting the training data (poisoning attacks) or forcing the model to their desired output, e.g., misclassification of abnormal events (evasion attacks).

In this context, security problems have already been addressed in the form of Adversarial Machine Learning (ML). There are many proposed solutions leveraging conventional ML-based techniques towards separating malicious data from benign. Game theory has also been exploited in the design of convolutional neural networks to detect tampering [3]. Concept drift, which is a common phenomenon in IoT data, has also been considered in problems such as feature extraction [4].

However, one main hurdle in all such mechanisms is how well they can operate in a distributed environment, like the one entailed in MCS, comprising many untrustworthy data sources providing falsified data (either as the result of an attack or a malfunctioning sensor). Compounding this issue, Banerjee et. al [5] studied how concept drifts and false data can masquerade the existence of intelligent attackers and their impact on the accuracy of both the clustering and (unsupervised) classification processes; however, they did not investigate the integration of more advanced Deep Learning schemes. What is needed is *a set of advanced deep learning mechanisms for assessing and sifting faulty data without any assumption on the trustworthiness of their sources while coping with intelligent adversaries that try to significantly decrease the overall performance, cause targeted misclassification*.

This paper meets this challenge with FSD - a novel data verification framework employing deep learning techniques to address the issue of possible data poisoning in MCS environments. We define malicious data as falsified data exhibiting different statistical properties (from the real data provided by benign users), and our approach is leveraging sequential relationships of sensory data towards detecting malicious samples generated by adversaries or faulty components. More specifically, our solution named False Sequential data Detection (FSD) offers: (i) Data verification by combining Deep Learning sequential architecture of Long Short Term Memories (LSTMs) with conventional one-class classifiers for distinguishing between "false" and "real" samples, (ii) Proof-of-concept implementation evaluated under various testing scenarios using both real and synthetic datasets in order to have more flexibility on the type of experiments. FSD demonstrates high accuracy even when the distribution of data comes from adversaries that demonstrate very similar behavior with the legitimate user data; i.e, strong colluding adversaries by composing two main attack strategies that represent different aspects of adversarial machine learning.

II. Related Work

A problem that is inherent when applying machine learning solutions for false data detection in MCS infrastructures is the concept-drift that occurs in some types of sensors used in emerging applications such as environmental monitoring; e.g., temperature and humidity [5]. Most recent studies, however, mainly focus on challenges in the context of sensing task allocation, sparse sensing, privacy, and data integrity [6]. But only a few works try to address the problem of false data detection [7], [8]. For instance, there are data verification techniques such as Deco [8] which uses spatio-temporal techniques to reconstruct missing values. Deco, however, suffers from high deviations. DETECT-and-CORRECT [9] as a two-phase framework attempts to detect false data by first leveraging a time-series of location data to DETECT suspicious data points while the CORRECT phase marks those points as missing for reconstruction algorithms. This method is not a general algorithm as it relies on additional contextual information of the particular MSC setup, whereas FSD is agnostic to the type of targeted domain. Methods surrounding matrix separation have also been successfully used for separating false data. Light Weight Low Rank and Sparse Matrix Separation (LightLRFMS) [7] is a matrix separation technique but it was only validated on a specific MCS context for environmental monitoring that does not fully consider issues pertaining to concept-drift, which FSD strives to address based on the use of LSTMs. Furthermore, various deep learning techniques have previously been applied in MSC including LSTM models towards predicting traffic flow [10] or user mobility [11]. However, these do not consider the presence of adversaries.

Banerjee et. al [5] showed that most common unsupervised learning algorithms are prone to adversarial infection and, hence, there is a need to leverage a mix of advanced machine learning models for overcoming this challenge. Our paper extends the problem disposition and investigation in [5] by trying to address issues that may be advantageous by the application of deep learning. We will apply some of the tools discussed in [5] to simulate adversaries targeting the datasets to evaluate our proposed model. Overall, there is a consensus that issues of accuracy, as a result of factors such as concept-drift and false data generation, must be addressed by leveraging deep learning based techniques as the ones employed by FSD.

III. Towards Trustworthy MCS Tasks

In MCS platforms, users can participate in the sensing process and upload their contributions to the central server, and collected can be processed by local analytic algorithms towards producing consumable data for requesting applications [12]. In this context, for a specific time window with n time steps and m sensors, we consider a dataset D containing a sequence (S) for each sensor j where $S_j = [v_{1,j}, v_{2,j}, \cdots, v_{i,j}, \cdots, v_{n,j}]$.

Threat Model: The aim of adversarial agents is to mislead the MCS applications towards considering malicious measurement values as legitimate in their services. To this end, an adversary may change the input value $v_{i,j}$ in S_j to $v'_{i,j}$, where $v'_{i,j} \neq v_{i,j}$ to maximize the distortion "$\max\{|v_{i,j} - v^{'}_{i,j}|\}$", where the distortion should be lower than a maximum allowed considered by the adversarial agent.

There are two primary adversarial attack models [13]: 1) pre-training (poisoning) attacks, and 2) post-training (evasion) attacks. In pre-training attacks, adversaries try to inject malicious data in an attempt to poison the training dataset and, thus, decrease the classification accuracy of the classifier. In the post-training attack strategy, adversaries aim at misleading trained classifiers to mis-classify samples towards a malevolent intent. Let us assume $f(x_i) = y_i$ as the mapping function to calculate/map x_i to y_i. In principle, a machine learning technique tries to minimize $|f(x'_i) - y'_i|$ which means minimizing False Positives and False Negatives. On the contrary, an adversarial attacker attempts to maximize the impact of the attack by maximizing $|f(x'_i) - y'_i|$. In the rest of the paper we refer to adversarial data as positive class of data, and legitimate data as negative class.

IV. FSD Conceptual Architecture

The main motivation behind FSD is to benefit from the relationships between samples of different time orders so as to accurately predict the values of the next time step. It uses the distance metrics for comparing the predicted data and real data, in each time step, in order to prepare samples for the one-class classification approach; i.e, draw a boundary around the allowed deviations between predicted and real data values.

Our approach consists of two main phases, namely the *Training* and *Testing* phases. In the *Training Phase*, we build a predictor for each sensory data as well as a final one-class classifier whereas during the *Testing Phase*, we feed a combination of malicious and benign testing samples in order to evaluate how good our trained models behave in separating malicious data from benign values.

A. Training Phase

Figure 1 depicts the underpinnings of the FSD training phase comprising of three steps: 1) training of LSTMs as sequential predictors for each sensor, 2) measuring deviations between predicted and real data vectors, for each time step, and 3) training an one-class classifier on vectors of the observed deviations in order to detect anomalies according to real and expected predicted values (again for each time step).

1) Training LSTMs: In this step, FSD trains separate LSTM models, M, for predicting values of each sensor, for each time step. As LSTM is useful for processing, classifying, and making predictions based on time series, we employ LSTMs on time series of sensor data in this paper. It is worth noting that the LSTMs in Figure 1 can be replaced by any time series based technique to predict next step data. This step results in m trained models based on m sensors, and M_j would be a trained model on S_j for predicting the $p_{i,j}$ value of sensor j at time step i.

2) Measuring deviations between predicted and real values: After training our LSTM models (Ms), FSD feeds all Ss in parallel (S_j to M_j), and store all predicted values ($p_{i,j}$) in

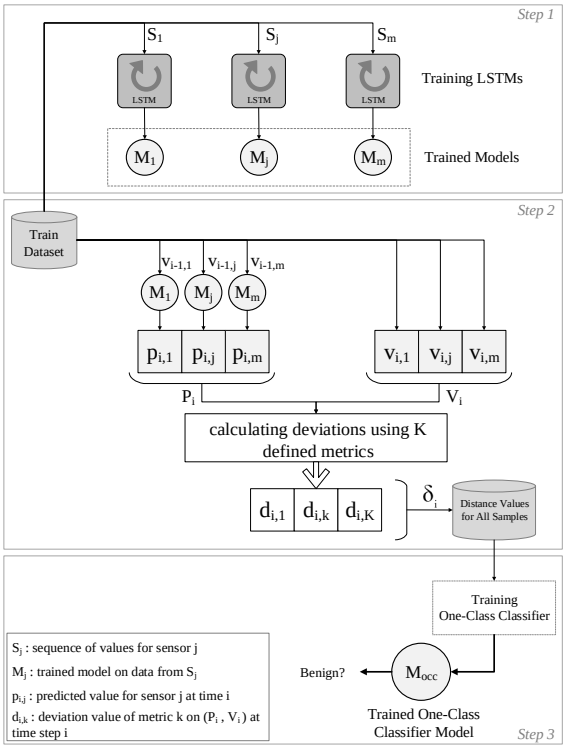


Figure 1: Training phase of FSD

matrix P where each row contains all predicted values by M_j on S_j for each time step. P_j, is column j of P showing the predicted values for sensor j for time step 1 to n, and $P_i = [p_{i,1}, \cdots, p_{i,m}]$ is a vector of all predicted values for m sensors at time step i referring to rows of P.

At time step i, we calculate deviations between predicted vector (P_i) and real vector (V_i) into a new vector $\delta_i = [d_{i,1}, d_{i,2}, \cdots, d_{i,K}]$ using K distance/similarity metrics such as Euclidean distance, cosine similarity, Manhattan distance, or other customized metrics with the ability of calculating one single scalar to show the deviation between the values of the two input vectors. Calculating the differences of predicted and real values we would have a new dataset Δ as a set of δ vectors with size K where $\delta_i = [d_{i,1}, ..., d_{i,k}, ..., d_{i,K}]$, and $d_{i,k}$ is the deviation between vector V_i and P_i calculated by distance metric k (e.g. Euclidean distance).

3) Training one-class classifier: After feeding all legitimate data to our trained models (Ms) and calculating δ_i for each time step i, we have $\Delta = [\delta_1, ..., \delta_i, ..., \delta_n]$ containing all *allowed deviations* between the predictions and the real data for all time step. Dataset Δ is suitable for one-class classifiers trying to find a boundary around the training samples in order to separate them from the data from other distributions, where M_{occ} in Figure 1 would be the trained one-class classifier.

B. Testing phase

Figure 2 shows how FSD works in real time to detect adversarial samples. For each testing time step i, FSD predicts P_i using trained models (Ms), and calculates δ_i as the distance vector between V_i and P_i. Then, M_{occ} classifies δ_i as *Benign* or *Malicious*. As M_{occ} is a trained one-class classifier on legitimate deviations between real and predicted values, the output dictates whether the vector of calculated deviations is legitimate.

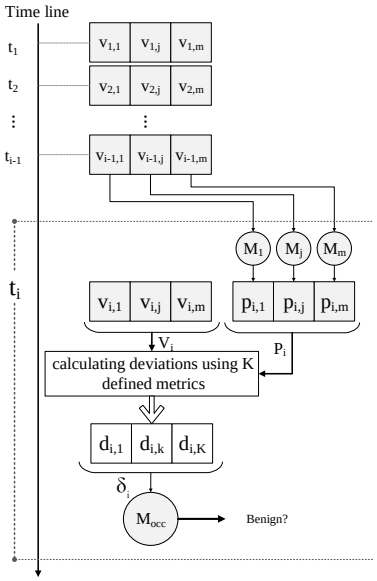


Figure 2: Testing phase of FSD

V. Experimental Setup

In this section, we proceed with presenting our setup for implementing and evaluating FSD under various attack strategies. The dataset considered in this paper originates from *real-world measurements* collected from Data Sensing Lab [14] by sensors deployed at the Strata Clara convention center in 2013 [1]. The dataset contains sensor data from 5 different sensors namely, Temperature (Temp), Humidity (Hum), Pir, Motion and Microphone (Mic). Table I shows the distributions of each sensor type.

	Temp	Hum	Pir	Motion	Mic
μ	22.5150	36.1389	0.1514	-0.000386	5.2275
σ	1.6171	6.0058	0.3583	0.3544	5.1334

Table I: Mean (μ) and standard deviation (σ) of base dataset

LSTM and One-class Classifier Architectures

As we are dealing with sequential sensor data with specific time steps, Recurrent Neural Networks based techniques - like the leveraged LSTMs - provide an attractive root of trust. In this context, in order to better justify the reasoning behind moving from Multi Layer Perceptron (MLPs) towards more complex time oriented classification techniques, we compare the best Root Mean Square Errors (RMSE) achieved by MLP and LSTMs. Table II shows the RMSE values between predicted and real corresponding values for each sensor. We name LSTMs with 20 units as $LSTM_{best}$ as it showed the best performance in our experiments with different number of units. LSTM with one LSTM unit ($LSTM_1$) and conventional Neural Networks with 10 hidden layers (MLP_{best}) are reported to compare with $LSTM_{best}$.

	Temp	Hum	Pir	Motion	Mic
MLP_{best}	0.208	0.350	0.797	1.152	2.322
$LSTM_1$	0.037	0.081	0.347	0.521	1.832
$LSTM_{best}$	0.029	0.080	0.301	0.216	1.639

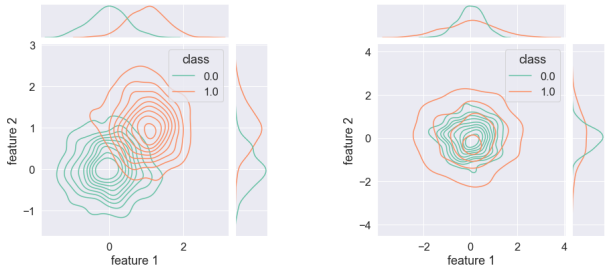
Table II: Comparing RMSEs achieved by the best trained MLP model, one-layer LSTM, and best trained LSTMs.

Adversarial Behaviour: As FSD is a framework for characterizing between inlying and outlying sensory data reports in the presence of adversarial malicious users, we assume that colluding adversaries are attacking the data collection process (poisoning attack) or the classification process (evasion attack). Therefore, we have effectively categorized attack strategies into pre-training and post-training attack strategies respectively. We measure the performance of the FSD framework based on different levels of distortion that colluding adversaries try to impose on the data trustworthiness. We consider two main attack approaches for each attack strategy: *1) distribution attacks (Attack Cases I & II)*, which manipulate mean or standard deviation to generate adversarial samples which are different from the benign ones; *2) position attacks (Attack Cases III, IV & V)*, which manipulate the position of injecting adversarial samples in the sequence of values which in turn targets the order of samples. Position attacks can only be applied after a distribution attack for changing the order of samples to be injected in the sequence.

Attack Case I: Adversaries may affect the system uncertainty about the true value of the sensory data by setting $\sigma' = \sigma$ and $\mu' \neq \mu$ which represents a malicious standard deviation equal to the standard deviation of the legitimate data points with smaller/larger μ as Equation (1) where λ is a scalar value as the deviation factor. Adversarial samples generated by $\lambda > 2$ are not attractive in our data as they are easier to detect due to their lower overlap with benign samples.

$$\begin{cases} \mu' = \mu + (\lambda \times \sigma) & 0 < \lambda \leq 2 \\ \sigma' = \sigma \end{cases} \tag{1}$$

Figure 3a compares the distributions of legitimate samples and adversarial samples generated by Attack Case I for a simple dataset with only two features, where $\lambda = 1.0$.



(a) (b)

Figure 3: Distributions of legitimate samples (class 0) vs. adversarial samples (class 1) for two features with $\mu = 0$ and $\sigma = 1.0$; (a) Attack Case I, and (b) Attack Case II.

Attack Case II: Adversaries may affect the system uncertainty by selecting an adversarial distribution based on equation 2 where the adversary's goal is to keep the same mean but changing the standard deviation.

$$\begin{cases} \mu' = \mu \\ \sigma' = \sigma + (\lambda \times \sigma) & 0 < \lambda \leq 2 \end{cases} \tag{2}$$

Figure 3b shows the distributions of a dataset with legitimate and adversarial data from Attack Case II with $\lambda = 1.0$. This attack case is designed to better reflect the real-world case scenarios where adversaries attempt to gradually change the classification behaviour by performing concept drifting by modifying the standard deviation over time.

Attack Case III: Adversaries may affect the system uncertainty about the true values by selecting a different distribution by changing the *order* of legitimate and malicious data points in the sequence of data. In this case all legitimate data comes before malicious data points in the sequence.

Attack Case IV: In this case all malicious data come before benign data points in the sequence as opposed to Case III.

Attack Case V: Adversaries may affect the system uncertainty by putting malicious batches of data in between legitimate data batches.

VI. Results and Analysis

In this section, we evaluate the performance of FSD in the presence of strong adversaries under the two aforementioned attack strategies (pre-training and post-training). We use F-Measure as our evaluation metric since it directly considers True/False Positive Rates as well as Recall and Precision metrics. The dataset is partitioned into two sub-sets of training and testing sets with 60% and 40% of data respectively.

A. Strategy 1: Impact of Adversaries in Pre-training

In the pre-training strategy, detecting malicious samples would be more difficult than in the post-training case since the classifier has been trained to recognize the adversarial samples as legitimate. Therefore, increasing the rate of adversaries has a higher impact on the classifier as it may consider all or parts of the injected malicious samples as "noise", thus, changing the overall perception of the classifier for the sensed phenomenon. Usually, the attackers tend to choose higher λ values during the pre-training attack strategy as their end-goal would be to mislead the classifier to cover a broader area and misclassify the attacking samples as legitimate in testing time. Due to limited space, we only reported the results based on changing deviations in μ' for Attack Cases III, IV and V.

1) Distribution Attacks (Cases I & II): Tables III & IV report the results for Attack Cases I & II, respectively. Each row of the tables show F-Measures for a specific λ value where the attacker generates malicious data using the previously described equations. In the pre-training strategy, increasing λ should result in a decrease in the adversarial data rate, thus, leading to a less accurate classification due to the adversarial samples affecting the classifier in order to cover malicious data as legitimate by expanding the area under the legitimate cluster (see Figure 3a). Tables III & IV clearly show this performance degradation. Similarly, increasing the rate of adversarial samples in pre-training prevents the classifier

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
0.5	0.813	0.768	0.727	0.718	0.704	0.668	0.652	0.634	0.613
1	0.768	0.737	0.696	0.676	0.679	0.643	0.608	0.591	0.558
2	0.721	0.701	0.662	0.635	0.613	0.601	0.579	0.556	0.506

Table III: F-measures for Attack Case I in pre-training strategy

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
0.5	0.755	0.741	0.719	0.700	0.681	0.661	0.641	0.597	0.564
1	0.716	0.690	0.681	0.668	0.648	0.632	0.598	0.548	0.514
2	0.678	0.668	0.665	0.658	0.621	0.607	0.565	0.516	0.468

Table IV: F-measures for Attack Case II in pre-training.

from ignoring the impact of malicious data in the training procedure. Thus more adversarial samples mean more power to mislead the trained classifier. Therefore, F-Measure of ML techniques should be decreasing when increasing the rate of adversarial data in pre-training due to the increase in quantity of False Positives and False Negatives. As expected, FSD faced performance degradation by increasing the rate of malicious samples (Adversarial Rate in Tables III & IV). However, the results are significant in comparison to conventional machine learning techniques where FSD achieved F-measures of 0.556 for Attack Case I and 0.516 for Attack Case II with a lot of malicious samples (40%) and a significant deviation ($\lambda = 2$) which is higher than the random guessing.

The results also show that FSD is more robust against the changes in mean (Attack Case II) in comparison to changes in standard deviation (Attack Case I). We expected to see higher F-Measures for $\lambda = 0.5$ as we had the same mean value with a small change in standard deviation for adversarial samples, however, we realized that the spikes in values of *pir* and *mic* over time made it difficult for FSD to assess their values in the context of a normal distribution.

2) Position Attacks (Cases III, IV & V): In Attack Case III all legitimate data appear before the malicious data in the pre-training strategy. Depending on the rate of malicious samples, the model may ignore parts of adversarial samples. As an example with 5% of adversarial data, the LSTMs model learns from 95% of legitimate data before learning from the rest of 5% malicious data, which have less impact on the final performance in comparison with 55% legitimate and 45% malicious samples. Tables V, VI and VII support the fact that the higher rates of adversarial data cause the higher negative impact on the knowledge gained by the model from the legitimate data. The results also show that for adversarial rates $\geq 45\%$, random guessing works better because FSD also learns from malicious data in training time which entails higher false negatives rate at testing time. Comparing Table V and Table VI reveals that the samples at the beginning of the sequences has more impact on the learning process. For example, 5% of malicious samples had more negative impact on F-Measure value when FSD learned from malicious samples before benign samples in comparison with Attack Case III. Attack Case V let the adversary have malicious samples in each malicious data batch, therefore more concept drift in comparison to the other cases. Increasing the rate of adversarial samples entails that the attacker has higher chance of preventing LSTMs to learn with sufficient legitimate data due to too many concept drifts on the input data.

The results show that FSD achieved higher F-Measures in detecting samples generated by Attack Case III in comparison to the two other Attack Cases. The results also prove that the superiority of FSD in detecting samples from Attack Case III diminishes by increasing λ or the rate of malicious samples. For example, based on Tables V & VI, the difference between F-Measures for $\lambda = 0.5$ and adversarial rate 5% is $(0.059 = 0.842 - 0.783)$, while for $\lambda = 2.0$ and adversarial rate 45% is only $(0.011 = 0.482 - 0.471)$. Due to page limits, we reported all position attacks using Attack Case I, however, FSD performance on cases using Attack Case II having similar trend. Moreover, FSD was more robust against attacks with one concept drift (Attack Cases III & IV) in comparison to attacks that try to trigger more drifts on the data preventing the model to learn enough knowledge from legitimate data. FSD also achieved high performance in cases with high rates of malicious samples implying superiority over conventional machine learning studied at [5].

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
0.5	0.842	0.736	0.701	0.687	0.672	0.651	0.646	0.626	0.561
1	0.753	0.717	0.675	0.659	0.635	0.626	0.625	0.601	0.516
2	0.718	0.688	0.651	0.631	0.616	0.608	0.596	0.565	0.482

Table V: F-measures for Attack Case III (benign first) with attack samples generated by Attack Case I in pre-training.

To give a comprehensive overview of FSD under various position Attack Cases, Table VIII compares F-Measures of Attack Cases III, IV and V. To calculate values of Table VIII, we subtracted and averaged all values of two comparing Attack Cases. For example, to calculate first row of Table VIII which compares F-Measures of Attack Case III to Attack Case IV, we first subtracted all values of Table VI from the paired values of Table V, and then calculated the average of all differences for each column shown in Table VIII. Negative values in the table shows that FSD achieved higher F-Measures for the second attack case.

B. Strategy 2: Impact of Adversaries in Post-training

This strategy generates samples for bypassing the classifier during testing time without accessing the training dataset. Therefore, we first trained the FSD on legitimate data to evaluate this strategy. In post-training strategy, it is easier for a

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
0.5	0.783	0.719	0.683	0.676	0.663	0.647	0.638	0.612	0.554
1	0.729	0.683	0.649	0.639	0.622	0.602	0.611	0.581	0.497
2	0.702	0.647	0.612	0.591	0.585	0.585	0.571	0.559	0.471

Table VI: F-measures for Attack Case IV (malicious first) with attack samples generated by Attack Case I in pre-training.

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
0.5	0.782	0.735	0.699	0.648	0.624	0.610	0.601	0.571	0.492
1	0.758	0.683	0.646	0.623	0.593	0.574	0.567	0.534	0.443
2	0.717	0.655	0.611	0.589	0.563	0.541	0.511	0.486	0.421

Table VII: F-measures for Attack Case V with attack samples generated by Attack Case I in pre-training.

	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
CaseIII-CaseIV	0.033	0.030	0.027	0.023	0.017	0.017	0.015	0.013	0.012
Case V-Case III	-0.018	-0.022	-0.023	-0.039	-0.047	-0.053	-0.062	-0.067	-0.067
Case V- Case IV	0.014	0.008	0.004	-0.015	-0.030	-0.036	-0.047	-0.053	-0.055

Table VIII: An overview of the F-Measures of FSD under different Attack Cases III, IV and V in pre-training.

trained classifier to detect non-overlapping malicious samples as the classifier has the knowledge to cover the legitimate area. Therefore, an adversary with this strategy tends for lower λ since they have to follow the distributions of legitimate data with minor changes for a successful evasion attack.

1) Distribution Attacks (Cases I & II): Table IX shows the results of classifying poisoned testing dataset by Attack Case I. FSD achieved higher F-Measures when we increased the rate of malicious examples (Adversarial Rate) because it was trained on legitimate data, so detecting malicious samples would be easier if the adversary inject more samples. This is because malicious data may create a new cluster with new distribution, so it is easier to detect them in cases with enough adversarial samples (higher rates of adversarial samples).

Comparing Table III in pre-training to the results from post-training (Table IX) shows opposite trends for both increasing the rate of adversaries and increasing the deviation factor (λ). Opposite to pre-training attacks, increasing the rate of adversarial samples may make it easier for the classifier to separate malicious data in post-training strategy. Higher deviation factors make an adversarial cluster with higher distances to the legitimate cluster resulting to higher F-Measures for post-training and lower F-Measures in pre-training.

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
0.5	0.551	0.582	0.598	0.606	0.629	0.648	0.677	0.698	0.743
1	0.573	0.595	0.621	0.643	0.678	0.688	0.714	0.735	0.763
2	0.592	0.600	0.642	0.666	0.697	0.701	0.745	0.765	0.815

Table IX: F-measures for Attack Case I in post-training.

The reported F-Measures in Table X indicate that FSD achieved higher performance in cases where adversaries change σ' without updating μ' (Attack Case II). Similar to Attack Case I, there is an increase in F-Measure along with the increase in λ and number of malicious data points in the post-training attack strategy. Comparing the results of Tables IX and X, we can see that FSD achieved better performance in cases where adversarial data were generated by updating σ' with the same mean. FSD achieved values of 0.551 and 0.566 for Attack Case I and Attack Case II, respectively, which are higher than the random guessing in our worst-case experiments with $\lambda = 0.5$ and 45% of adversarial data.

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
0.5	0.566	0.586	0.610	0.638	0.689	0.743	0.766	0.796	0.841
1	0.636	0.656	0.667	0.684	0.706	0.756	0.776	0.819	0.866
2	0.680	0.685	0.699	0.716	0.740	0.770	0.785	0.829	0.906

Table X: F-measures for Attack Case II in post-training

2) Position Attacks (Case III, IV & V): Table XI describes F-Measures of detecting adversarial samples when all benign samples appeared before malicious samples in post-training. More adversarial samples with higher deviation means more distance between benign and malicious clusters. Therefore, FSD achieved high F-measure of 0.903 in *post-training* strategy when the adversaries injected a lot of malicious samples (45%) with high deviation of 2.

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
0.5	0.599	0.615	0.651	0.681	0.715	0.734	0.781	0.800	0.832
1	0.632	0.655	0.681	0.713	0.759	0.777	0.810	0.833	0.866
2	0.678	0.681	0.702	0.734	0.774	0.795	0.832	0.851	0.903

Table XI: F-measures for Attack Case III (benign first) with attack samples generated by Attack Case I in post-training.

Table XII shows F-Measures of FSD in testing dataset under Attack Case IV in post-training. As LSTMs work based on the history of data elements in time series data, the attacker in Attack Case IV attempts to affect the initial weights of LSTMs. Therefore, we see lower F-measures in comparison to Attack Case III. Comparing Tables XI & XII shows that FSD is more efficient if the adversary positions all benign samples at the beginning of testing data.

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
0.5	0.487	0.536	0.588	0.609	0.637	0.666	0.696	0.743	0.785
1	0.534	0.597	0.601	0.652	0.705	0.718	0.767	0.769	0.800
2	0.597	0.607	0.648	0.679	0.722	0.741	0.783	0.799	0.851

Table XII: F-measures for Attack Case IV (malicious first) with samples generated by Attack Case I in post-training.

Table XIII reflects F-Measures of FSD where the adversary positions the malicious data in between benign samples. FSD showed better performance under this attack in comparison to Attack Case IV, however the results of Attack Case V are more or less similar to Attack Case III. F-Measures of higher than 0.785 with 45% of adversaries and $\lambda = 0.5$ is significant for FSD under Attack Cases III-V in comparison to conventional machine learning at [5].

Table XIV gives an overview of F-Measures for Attack Cases III, IV and V. The first tow of the table shows that FSD achieved higher F-measures when the adversaries put benign samples before the attacking samples. The second row of Table XIV shows that increasing the rate of adversarial samples to more than 25% in Attack Case V makes the detection easier for FSD to detect malicious samples in comparison to Attack Case III with adversarial rate of $\lambda > 25\%$.

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
0.5	0.579	0.611	0.658	0.683	0.729	0.749	0.780	0.803	0.860
1	0.592	0.631	0.676	0.704	0.748	0.771	0.814	0.838	0.878
2	0.606	0.661	0.680	0.732	0.771	0.790	0.839	0.859	0.901

Table XIII: F-measures for Attack Case V post-training attack strategy with attack samples generated by Attack Case I.

	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Case III-Case IV	0.097	0.070	0.065	0.062	0.061	0.060	0.059	0.057	0.055
Case V-Case III	-0.044	-0.016	-0.006	-0.003	0.000	0.001	0.003	0.005	0.012
Case V-Case IV	0.053	0.054	0.059	0.059	0.061	0.061	0.062	0.063	0.067

Table XIV: An overview of the F-Measures of FSD under different Attack Cases III, IV and V in post-training.

VII. Discussion and Critique

As it is commonly the case for any relatively young research area, the landscape of MCS for IoT is fragmented into various families based on the emerging research challenges. Undoubtedly, data trustworthiness is a prominent challenge with unprecedented number of consequences. We consider this paper as the first step towards the development of a holistic framework, which will improve data trustworthiness by leveraging advanced deep learning capabilities.

For the evaluation, the system uses different distance measures comprising data vectors of all 5 features originating from data sources of various trustworthiness levels; containing both benign and malicious data samples. This segregation is denoted by a scalar applied to the standard deviation of each feature ensures that the data distribution considered for each feature is balanced. However, there are a number of challenges to be considered: First of all, features do not behave the same way over time as reflected in their distribution presented in our evaluation due to changing statistical properties over time. Typically, a feature can be modelled by a normal distribution, however, some features possess a lot of spikes and shifts frequently which makes them harder to predict. Furthermore, this affects the overall performance as all features are merged to one representative vector, meaning that features with varying behavior over time can affect the overall performance negatively, as was also reflected in our results.

Secondly, with colluding adversaries attacking the data collection process by feeding the classifier with falsified data, it is evident that not only does the order of how the data is provided matter significantly in the system's ability to classify malicious users, but so does the time window size. This is why the use of LSTM agents is beneficial as it considers the sequential order of data. In this work, we have investigated the impact that this ordering of data has along the observed distribution. However, in the future it may be fruitful to also explore the open challenges regarding the impact of selecting various time windows and consider the influence this might have on the performance of the classifier. Recall that if the time window is increased, we effectively increase the overall knowledge of the classification process. However, it should be noted that larger time windows might introduce sever scalability issues for real-world applications.

While the current status and functionality of FSD has been evaluated against advanced data poisoning attacks, it is also our intention to perform a detailed analysis considering also even more intelligent adversaries targeting different facets of the MCS paradigm such as evasion attacks and more advanced data poisoning techniques like the one proposed by Miao et. al [15]. As another research direction, we may study how extracted features from the sequential information between data elements from other sensors can contribute to detect malicious samples in a sensor.

VIII. Conclusion

Data trustworthiness has always been one of the main concerns that current MCS platforms are facing. FSD works based on measuring the deviation between the predicted and the real received sensor values. We used LSTMs and One-Class SVM to build FSD. However, it is simply possible to replace them with alternative techniques working on time series and one-class classification. As an intelligent framework, FSD improves data trustworthiness by providing the capability of detecting falsified data generated by adversaries under both pre-training and post-training attack strategies. In our experiments we designed five attack cases consisting of two distribution attack cases and three position attack cases to evaluate both attack strategies. FSD achieved higher performance when it visits more benign samples before malicious samples in both studied strategies.

IX. Acknowledgment

This work was supported by Innovation Fund Denmark (IFD) under SecDNS project and the European Commission under the STAR project, Grant Agreements no. 956573.

References

[1] T. Giannetsos, S. Gisdakis, and P. Papadimitratos, "Trustworthy people-centric sensing : Privacy, security and user incentives road-map," in *2014 13th Annual Mediterranean Ad Hoc Networking Workshop, MED-HOC-NET 2014 :*, 2014, pp. 39–46, qC 20150108.

[2] N. P. Owoh and M. M. Singh, "Security analysis of mobile crowd sensing applications," *Applied Computing and Informatics*, vol. ahead-of-print, no. ahead-of-print, Jul. 2020.

[3] A. S. Chivukula and W. Liu, "Adversarial learning games with deep learning models," in *Int. Joint Conference on Neural Networks*, 2017.

[4] R. C. Cavalcante, L. L. Minku, and A. L. Oliveira, "Fedd: Feature extraction for explicit concept drift detection in time series," in *Int. Joint Conference on Neural Networks*, 2016.

[5] N. Banerjee, T. Giannetsos, E. Panaousis, and C. Cheong Took, "Unsupervised learning for trustworthy iot," 07 2018, pp. 1–8.

[6] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, R. Huang, and X. Zhou, "Mobile crowd sensing and computing," *ACM Computing Surveys*, vol. 48, no. 1, pp. 1–31, Sep. 2015.

[7] X. Li, K. Xie, X. Wang, G. Xie, D. Xie, Z. Li, J. Wen, Z. Diao, and T. Wang, "Quick and accurate false data detection in mobile crowd sensing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1339–1352, 2020.

[8] L. Cheng, L. Kong, C. Luo, J. Niu, Y. Gu, W. He, and S. Das, "Deco: False data detection and correction framework for participatory sensing," in *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*. IEEE, Jun. 2015.

[9] B. Wang, L. Kong, L. He, F. Wu, J. Yu, and G. Chen, "I(TS, CS): Detecting faulty location data in mobile crowdsensing," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, Jul. 2018.

[10] Y. Tian, K. Zhang, J. Li, X. Lin, and B. Yang, "Lstm-based traffic flow prediction with missing data," *Neurocomputing*, vol. 318, 2018.

[11] W. Yang, G. Sun, X. Ding, and X. Zhang, "Budget-feasible user recruitment in mobile crowdsensing with user mobility prediction," in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, 2018, pp. 1–10.

[12] R. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, Nov. 2011.

[13] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, "A taxonomy and survey of attacks against machine learning," *Computer Science Review*, vol. 34, p. 100199, Nov. 2019.

[14] D. S. Lab, ""strata santa clara dataset"." [Online]. Available: http://datasensinglab.com/data/

[15] C. Miao, Q. Li, H. Xiao, W. Jiang, M. Huai, and L. Su, "Towards data poisoning attacks in crowd sensing systems," 06 2018, pp. 111–120.