2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)

2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT) | 979-8-3503-4649-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/DCOSS-IOT58021.2023.00093

Explainable Artificial Intelligence to Enhance Data Trustworthiness in Crowd-Sensing Systems

Sam Afzal-Houshmand*, Dimitrios Papamartzivanos‡, Sajad Homayoun*, Entso Veliou**, Christian D. Jensen*,
Athanasios (Thanos) Voulodimos†, Thanassis Giannetsos‡
*Technical University of Denmark (DTU), Cyber Security Section, Denmark
‡Ubitech Ltd., Digital Security & Trusted Computing Group, Greece
** Department of Informatics and Computer Engineering, University of West Attica, Athens, Greece
† School of Electrical & Computer Engineering, National Technical Univ. of Athens, Greece
Email: saaf@dtu.dk, sajho@dtu.dk, {dpapamartz,agiannetsos}@ubitech.eu, eveliou@uniwa.gr, thanosv@mail.ntua.gr cdje@dtu.dk

***Abstract*—Around the world there has been an advancement of IoT edge devices, that in turn have enabled the collection of rich datasets as part of the Mobile Crowd Sensing (MCS) paradigm, which in practice is implemented in a variety of safety critical applications. In spite of the advantages of such datasets, there exists an inherent data trustworthiness challenge due to the interference of malevolent actors. In this context, there has been a great body of proposed solutions which capitalize on conventional machine algorithms for sifting through faulty data without any assumptions on the trustworthiness of the source. However, there is still a number of open issues, such as how to cope with strong colluding adversaries, while in parallel managing efficiently the sizable influx of user data. In this work we suggest that the usage of explainable artificial intelligence (XAI) can lead to even more efficient performance as we tackle the limitation of conventional black box models, by enabling the understanding and interpretation of a model's operation. Our approach enables the reasoning of the model's accuracy in the presence of adversaries and has the ability to shun out faulty or malicious data, thus, enhancing the model's adaptation process. To this end, we provide a prototype implementation coupled with a detailed performance evaluation under different scenarios of attacks, employing both real and synthetic datasets. Our results suggest that the use of XAI leads to improved performance compared to other existing schemes.**

***Index Terms*—Explainable AI, Adversarial Machine Learning, Data Trustworthiness, Time-series analysis.**

I. Introduction

Temporal data can be quite unpredictable to humans as it can be hard for someone to understand and explain why changes over time happen by just observing the data as an instance in the form of a picture or text. That is why it can be tricky to represent temporal data as a signal that varies as a function of time due to the ubiquitous nature of temporal data. To enable knowledge extraction out of such data we need to leverage additional methods along with expert knowledge [1].

The abundance of devices worldwide has created a mobile sensing landscape that requires users to feed information about sensory data of the environment, within a sensing paradigm, known as mobile crowd sensing (MCS) [2]. The generated data often take the form of a temporal data set. In terms of security, several issues arise on data quality and integrity due to the openness of these platforms [3], while the same issue affects significantly other domains which capitalize on the collection of data from sensors, such as the smart manufacturing domain [4], [5], [6]. Previous works aim to tackle these issues by making use of Machine Learning. ML systems are trained using MCS datasets that are assumed to be benign, whereas malicious actors attempt to poison them by targeting training data through simple or advanced manipulations or can directly evade models' inference process using "adversarial examples" aiming to deceive completely the model.

There has been a plethora of research works trying to address such security issues using adversarial machine learning leveraging conventional ML-based techniques for distinguishing between malicious and benign data [7]. However, those conventional techniques have proven to be inefficient especially in cases where the discrimination between concept drift and cases where untrustworthy data sources provide falsified data (either as the result of an attack or a malfunctioning sensor) is hard. In fact, the synthesis of concept drift and false data injection can enable an intelligent attacker to hide in the shadows and affect significantly the accuracy of both clustering and classification processes.

In our research path, we have previously worked on applying advanced Deep learning schemes [8] that were capable of distinguishing malicious and benign data, without any assumptions regarding the trustworthiness of data sources, whilst being robust against many colluding intelligent adversaries. In this paper, we take a step forward by addressing the plausible relation between the sensory input from devices over time via the application and integration of explainable artificial intelligence (XAI). XAI is exploited for the purpose of gaining knowledge and insights regarding the temporal data set that typically impose restrictions on human comprehension [1]. Capitalizing on the pipeline of FSD [8], we further expand our method by introducing the aspect of XAI which enables the more exhaustive investigation of the aforementioned security challenges, leading to the proposed XFSD framework.

II. Related Work

A great mass of works applied Machine learning and Deep Learning (DL) in the mobile crowd sensing infrastructures,

2325-2944/23/$31.00 ©2023 IEEE
DOI 10.1109/DCOSS-IoT58021.2023.00093

addressing issues pertaining to concept drift, data integrity, sparse sensing [3] [9] [10]. In this paper, we focus on works that applied XAI to sequential time-series data so we can possibly incorporate it into the existing FSD framework [8] [11]. There is a variety of works that apply XAI to time series but it can be broken down to the distinct categories of Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and non-sequential Neural Network based methodologies [1].

RNNs are a great fit for time series datasets which also encompass the application of LSTM model that retains the whole history of the time data. The XAI methodology used typically within this context is some form of SHAP which is a model-agnostic explanation method. Kim et al. [12] were the first to suggest using SHAP algorithms to explain the output of RNN used on time series data to produce highly accurate outcomes. SHAP has been applied successfully to financial time series prediction and explanation [13] [14] [15] [16].

Regarding CNN-based methods, the prevalent approach of applying XAI to time series is class-activation mapping. This is a post-hoc method that that highlights regions in the input data which affect the CNN output with quite accurate results [17] [18] [19]. Within the category of CNNs, there are other interesting approaches, such as ConvTimeNet which occludes part of the time series and computes the probability of the predicted class [20]. Another approach is Gradient*Input to identify the contribution of the input raw data in time series classification tasks [21]. In terms of non-sequential methods, explanations can be generated using LIME. This approach is less accurate than the sequential explanatory methods but it is more efficient [22], [23] [24].

III. Towards Trustworthy MCS Tasks

Mobile Crowd Sensing (MSC) information platforms allow users to contribute by uploading their data to a central server, where the collected information can be processed by analytic algorithms for sharing knowledge with other applications of different fields [25]. To put it mathematically, for a specific time window with n time steps and m sensors, we gather a dataset D containing a sequence (S) of values v for each sensor j, having $S_j = [v_{1,j}, v_{2,j}, \cdots, v_{i,j}, \cdots, v_{n,j}]$. $v_{i,j}$ represent the value of sensor j at time step i. D in (1) shows how S sequences build the matrix as the dataset. The sequential structure of the data makes the use of LSTMs a interesting approach for extracting useful insights from the relationships between the various sequence values.

$$D = \begin{bmatrix} v_{1,1} & \dots & v_{1,j} & \dots & v_{1,m} \\ \vdots & & \vdots & & \vdots \\ v_{i,1} & \dots & v_{i,j} & \dots & v_{i,m} \\ \vdots & & \vdots & & \vdots \\ v_{n,1} & \dots & v_{n,j} & \dots & v_{n,m} \end{bmatrix} \quad (1)$$

A. Threat modeling

Colluding adversaries aim to mislead MCS apps by incorporating an adversarial agent for generating malicious measurement values that seem to be legitimate to the application.

There are two main adversarial models which must be taken into consideration for such a platform [7], namely the: 1) pre-training (poisoning) attacks, and 2) post-training (evasion) attacks. With pre-training attacks, the adversaries attempt to inject malicious data to poison the training dataset and, thus, build a model which is formed based on misleading instances. The post-training attacks occur in testing time, i.e. during the inference mode of the model, where the adversaries can craft adversarial data that prevent a machine learning model from correctly identifying the contents of the data, i.e., to mislead the already trained model to mis-classify the injected samples.

In principal, in any ML approach we try to minimize False Positives and False Negatives. On the contrary, the adversarial objective is to maximize the impact of the attack by maximizing these metrics. For the remainder of the paper we refer to the adversarial data as the positive class, and the legitimate data as negative one.

B. Explainable AI

Explainable AI (XAI) is a set of processes and methods that allows human users to comprehend and trust the results and output created by machine learning algorithms in contrast to typical black box approaches of some algorithms, especially neural networks. Machine learning has a long history of being applied to time series data, but most of the solutions make the interpretation difficult, and without a quantitative assessment it may be hard to understand areas of temporal data which may affect the prediction. XAI can tackle this issue by adding additional knowledge that may strengthen the overall model prediction ability [1] [26] [22].

There is a wide spectrum of algorithms to use when applying XAI on time series information. In this paper, we deal with sequential time series data and we exploit two of the most common XAI algorithms, namely the SHAP (SHapley Additive exPlanations) and Class Activation Mapping (CAM). Moreover, we evaluate the impact of a well-known non-sequential based XAI algorithm called LIME (Local Interpretable Model-agnostic Explanations) to study the difference between sequential and non-sequential approaches.

Generally, the pipeline of applying XAI to the numerical values of time series aim to identify the significance that the input values have on the model prediction result. In our model, we take as input the raw data and we obtain a vector of fixed length, determined by a sliding-window, fed into a DL model so that to acquire a prediction. Then, by using an XAI method, we can calculate impact scores which reflect the impact that a input value has on the prediction at a given time. The outcome is a numerical representation which denotes whether a given step in the raw dataset affects the prediction negatively or positively considering the input and the predicted values. In this way, XAI is applied in a holistic manner to the model and explains which are the areas of

importance of the input data. Then, this outcome, i.e., the value obtained by the XAI model, can be used in order to enrich or adjust the NN model accordingly. The rationale behind this approach is that the XAI values are also affected by the adversarial attack strategies and, intrinsically, they bear the information for the presence of malicious data points in a dataset. In this way, XAI can contribute to the identification of attack vectors, which has been applied to the system and the data, by utilizing the generated XAI values. Thus, having these values, we can apply an aggregation model to generate enhanced data vectors that combine the predictions and the XAI importance values, aiming to make the adversarial inputs more distinguishable. In fact, this is the key difference between the FSD approach presented in [8] and the enhanced XAI-assisted method, namely the XFSD, presented in this work.

IV. XFSD Conceptual Architecture

The Fake Sequential Data (FSD) detection framework [8] was created to benefit from the time-based relationships between dataset samples so as to accurately predict the values of the next time step. That system utilized distance metrics for comparing the predicted data and real data, at each time step, in order to enable a one-class classification approach to essentially draw a boundary around the allowed (expected) deviations between predicted and real data values.

The XFSD approach augments the existing data with numerical values from the XAI models which represents the importance of each feature over time. The input to XAI is the raw sequence of values for each feature along with a trained model, while the output is a numerical vector which denotes the importance score of the feature. Then, we apply an aggregation model to concatenate the predictions and the XAI importance values to one dataset. Like most deep learning frameworks, our approach consists of two main phases, namely the *Training* and *Testing* phases which are detailed below.

A. Training Phase

Figure 1 shows the steps followed by XFSD to train the final one-class classifier to distinguish benign samples (not generated by adversaries). Similar to FSD [8], XFSD framework trains a sequence predictor (P_j) for every sensor j that takes an input sequence and predicts the next step value (Step 1 of Figure 1). Step 2, for each timestep, uses the trained predictor (P_j) to estimate the expected value (e_j) at time step i by feeding the sequence (S_j) of all values of sensor j from timestep 0 to timestep $(i-1)$. α_i, then, is the distance vector between the real values and the predictions. α_i is of size K which corresponds to the number of K distance metrics (e.g. Cosine Distance, etc) used. At the same time, feeding the sequence S_j and the predictor P_j to the XAI explainer X_j will generate a vector of values ($\beta_{i,j}$) as the explanation of the data at the current step. Concatenating and flattening α_i and all $\beta_{i,j}$ will result to $\vec{Z}_i$ vector (for row i) of Z (2).

1) Step1: Train time-series predictors: This step trains a predictor (P) for each sensor that is able to predict sensor value of timestep i based on the sequence of values at previous

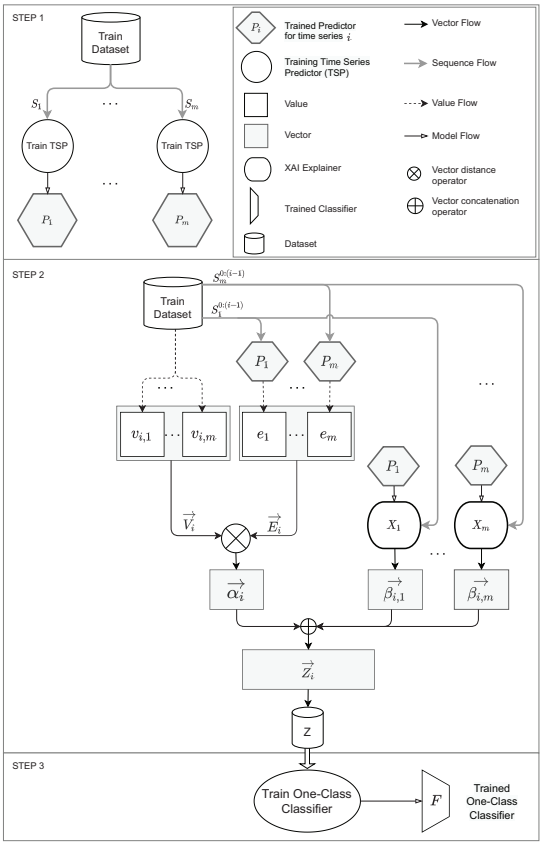


Figure 1: Training phase of XFSD

timesteps. Although this step is not limited to sequential based algorithms, we recommend using Recurrent neural networks (RNNs) and Convolutions neural networks (CNNs) as they are widely used in predicting time series. The output of Step 1 (*Predictors*) will be used later in Step 2.

2) Step2: Explainable feature extraction: We apply an XAI algorithm (any of the SHAP, CAN, LIME) which effectively calculates a numerical representation of the important areas of the data input based on the sequential model predictor. Step 2 of Figure 1 shows the procedure of computing $\vec{\alpha}$ and $\vec{\beta}$ into $\vec{Z}_i$. The output of this step is matrix Z that will be used in Step 3 to train the final one class classifier.

$$Z = \begin{bmatrix} z_{1,1} & \dots & z_{1,k} & \dots & z_{1,q} \\ \vdots & & \vdots & & \vdots \\ z_{2,1} & \dots & z_{i,k} & \dots & z_{i,q} \\ \vdots & & \vdots & & \vdots \\ z_{n,1} & \dots & z_{n,k} & \dots & z_{n,q} \end{bmatrix} \quad (2)$$

where $q = K + (m \times a)$ in (2).

3) Step3: Training one-class classifier: The completion of step 2 means that all legitimate data have been fed to the *Predictors* and the system calculated Z for each time step i containing all *allowed deviations* between the predictions and the real data plus the XAI values for all time steps. Z matrix

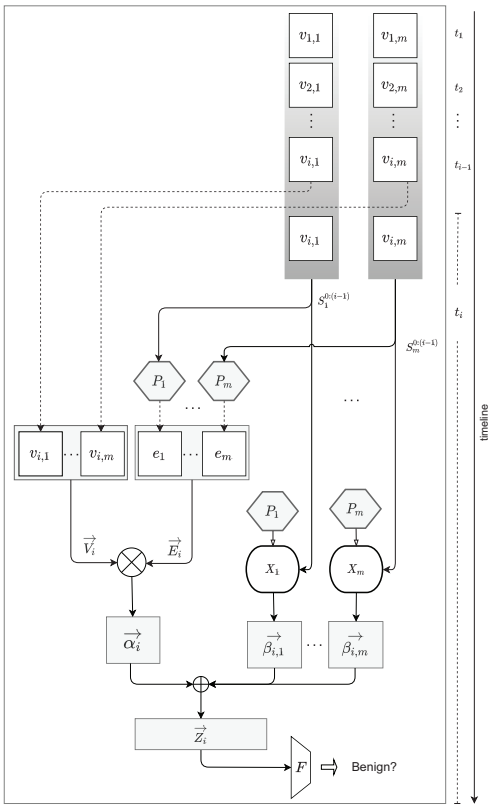


Figure 2: Testing phase of XFSD

is suitable for a one-class classifiers as it is synthesized by feeding only benign data to the predictors and the explainers. The one-class classifier attempts to find a boundary around the training samples (based on benign samples) in order to distinguish them from data that follow other distributions. F in Figure 1 represents the trained one-class classifier.

B. Testing phase

Figure 2 shows how XFSD works in real time to detect adversarial samples. For each testing time step i, XFSD uses predictor P_j to produce e_j for sensor j. Then it concatenates and flatten $\vec{\alpha}$ and $\vec{\beta}$ into $\vec{Z}_i$ so that to feed the one-class classifier F in order to specify if the data received in the current time step follows the distribution of benign data. As F is a trained one-class classifier on legitimate deviations between real and predicted values as well as the XAI values, the output dictates whether $\vec{Z}_i$ falls within the boundaries of the legitimate behavior or not. If F suggests that $\vec{Z}_i$ is not in the distribution of benign samples, then $\vec{V}_i$ will be flagged as malicious, as one or more sensor values at time step i in $\vec{V}_i$ were generated by adversaries.

V. Experimental Setup

XFSD is an extension of FSD which introduces the use of XAI on numerical values as a way of improving the detection rate of malicious data. In this work, we offer an experimental testbed with more advanced attack strategies in order to witness the performance improvements of XFSD.

A. Adversarial Behavior

Following the threat model presented in section III-A, the adversary may attempt to feed malicious data points which are then included in the training of the deep learning model used for the classification or sends malicious data points *after* the model and the classifier has been trained on the real benign data points. Therefore, we categorize the attack strategies into pre-training and post-training attack strategies, respectively.

The performance of the framework was evaluated based on different levels of distortion. In XFSD we consider two main attack approaches for each attack strategy: *1) distribution attacks (Attack Cases I & II)* and (Attack Cases VI & VII), which manipulate the mean or standard deviation to generate adversarial samples which are different from the benign ones; and *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 towards changing the order of samples to be injected.

Attack Case I: The adversaries may affect the system uncertainty by setting $\sigma' = \sigma$ and $\mu' \neq \mu$ which represents a malicious standard deviation equal to the standard deviation of the legitimate data points, but with smaller/larger μ. In Equation (3), λ is a scalar value as the deviation factor. Adversarial samples generated with $\lambda > 2$ are not attractive as they are easier to detect due to their lower overlap with benign samples. That is, we limit λ between 0 and 2 to study more realistic data distributions.

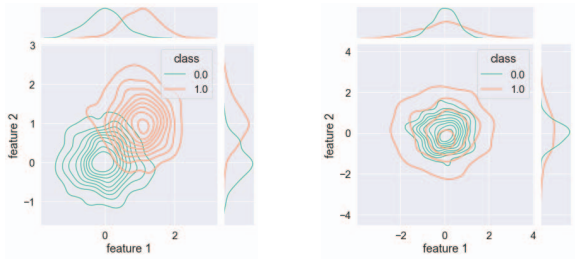
$$\begin{cases} \mu' = \mu + (\lambda \times \sigma) & 0 < \lambda \leq 2 \\ \sigma' = \sigma \end{cases} \tag{3}$$

To further illustrate what the actual structure of this attack we refer to Figure 3a where there is a comparison of the distributions of legitimate samples and adversarial samples generated by Attack Case I for a simple dataset with only two features, where $\lambda = 1.0$. The adversaries may choose the λ value depending on the attack strategy (pre-training or post-training) and the number of attacking samples they want to inject/test notated as a percentage of the total number of samples. As seen in the figure, changing the mean value would have a greater impact on changing the distribution as it will not have a huge overlapping region with the actual distribution of the legitimate data. The intuition is that trained classifiers should actually work better in detecting malicious data if they were trained solely on a legitimate distribution.

Attack Case II: Adversaries may affect the system by selecting an adversarial distribution based on equation 4 where the adversary's goal is to keep the same mean but changing the standard deviation (SD).

$$\begin{cases} \mu' = \mu \\ \sigma' = \sigma + (\lambda \times \sigma) & 0 < \lambda \leq 2 \end{cases} \tag{4}$$

Similar to case I, we refer to Figure 3b where one can see the



(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.

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 behavior by performing concept drifting and by modifying the SD.

Attack Case III: In this case of a positional attack strategy, the adversaries may affect the system 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: The opposite of case III, in this case all malicious data come before benign data points in the sequence.

Attack Case V: The adversaries attempt to affect the system uncertainty by putting malicious batches of data in between legitimate data batches. Variations of this approach inspired us for the expansion of positional attacks including more specific patters including randomization, normal and different sized windows of malicious data batches inserted.

Attack Case VI: In this case we perform an availability attack which aims to maximize the error at the point where the adversaries are not detected by the model. Practically, this means that we feed the system a dataset during the pre-training which will broaden the boundaries that the classifier can draw for benign samples, and thus, making adversarial inputs indistinguishable during the testing phase. The focus is on increasing the variance in the data poisoning process. To do so, we use bilevel optimization approach so that to define the optimal attack strategy, i.e., to define the optimal shift, distribution, and the number of malicious samples which will shift the outcome as much as possible from the truth. We apply this method on cases I and II i.e. shifting the μ and σ.

Attack Case VII: In this case we perform a targeted attack focusing on steering the model estimation towards a certain prediction. In this concept, the attacker poisons the model so that to target a pre-determined outcome. In contrast to availability attack of case VI, where the attacker's objective is to deviate the response of the model in an untargeted manner, in the targeted attack, the attacker aims to have control over the model's prediction and this requires to develop an optimized adversarial strategy. Practically, we generate a dataset for a specific pre-determined target, we feed it during pre-training and we observe the effects through the testing phase. To achieve this, we make use of bilevel optimization to obtain the optimal variable to be manipulated in order to steer the model to the pre-determined target. We apply the targeted attack considering the cases I and II i.e., shifting the μ and σ.

VI. Results

In this section, we evaluate the performance of XFSD. For that purpose we utilize F-measure metric as it considers True/False positive rates, recall and precision. We divided the dataset into two subsets with a 60%-40% split for the training and testing respectively. The dataset considered in this paper originates from real world measurements collected from Data Sensing Lab [27] by sensors deployed at the Strata Clara convention center in 2013.

In addition, before converging to a specific configuration setup for our testbed, we performed a number of experiments in order to define the "optimal" size of the sliding-window. A sliding window of 1000 instances was the best fit for our model based on the acquired F-measure performance results. Thus, with the optimal setup and by incorporating the XAI methods, we report the results and we perform a comparative analysis against FSD [8] in order to advocate the merit of XFSD.

Note that for attack cases VI and VII we have considered only pre-training attacks so that to evaluate the performance of XFSD framework against more advanced poisoning attacks compared to the more simple poisoning approaches of cases I and II. The following sections are structured as follows: Firstly, we focus on distribution attack cases (i.e. I and II). An analysis on the positional attacks follows (i.e. attack cases III, IV and V). The analysis concludes with a section focusing on the advanced attack cases (i.e. VI and VII) where both cases include manipulation of both μ and σ.

A. Results of XFSD Distribution attacks

1) XFSD case I and II in pre-training attack strategy: With the pre-training attack strategy the adversaries try to poison the training datasets. It is generally harder to detect malicious samples in the pre-training than the post-training since classifiers are trained to recognize the adversarial samples as legitimate. Increasing the rate of adversarial inputs in pre-training has a higher impact on classifiers. For the same reason an attacker would chose a higher deviation factor as the end-goal is to mislead the classifier's perception in that scenario.

Tables I and II summarize the results for attack cases I and II, respectively for the pre-training attack strategy. Each row of the tables show F-measure for a specific λ value used by the attacker to generate malicious data. It is expected that by increasing λ or the adversarial data rate, this should lead to less accurate classification in the pre-training strategy, as this would skew the perception of the classifier. This F-measure degradation tendency is in fact reflected in the results of Tables I and II. All these reflections are as expected from

what we know from FSD without the XAI values. The interesting aspect is the positive impact of applying different XAI methods. As showcased in the tables the tendencies are more or less the same across the different algorithms, implying that the impact of different attack cases in pre-training strategies is the same regardless of the algorithm applied. The only difference is the scale of accuracy among their different XAI methods. Generally, the introduction XAI to FSD provides a proportional improvement as degradation is the same with respect to the different configurations of adversarial attacks between FSD and XFSD. However, the F-measure is clearly higher for XFSD showcasing a significant improvement when adding XAI values, advocating the motivation of this work for introducing XAI values as the additional information to improve the overall performance of classifiers.

The reasoning behind this improved performance is that the adversarial strategies also impact the explanation values obtained by XAI. That is, the explanation values are a valuable addition for better distinguishing between legitimate and malicious instances. Furthermore, in its foundation, XAI provides values that explain the impact of input to a NN model. With that knowledge it is possible to adjust weights that may improve a model mitigating issues like concept-drift.

Another part of the experimentation of this project was to find the best XAI algorithm to employ. We had some insights why certain algorithms may perform better given the dataset we are working with. In fact, we expected SHAP to outperform the other methods as it is designed for scenarios using RNN or LSTM, which is the basis of FSD, which is widely recognized as the conventional way of handling time-series information. To illustrate this performance lets take an example of $\lambda = 2$ and 40% adversarial rate for all algorithms in case I in pre-training strategy. SHAP is clearly the most accurate method in terms of F-measure. However, it is interesting to note that LIME achieves worse performance than FSD. The reasoning is that LIME is a non-sequential algorithm which indicate that there is a significant amount of information that can be extracted from the sequential order of our data, and LIME fails to capture this behavior.

2) XFSD case I and II in post-training attack strategy: In post-training scenario the attacker attempt to generate samples for bypassing the classifiers during testing time without any access to the training data. Thus, for this attack strategy the models are trained on legitimate data and then the overall classification is evaluated using manipulated data. In the post-training strategy it should be easier for a classifier to detect non-overlapping malicious samples as the classifier has the knowledge to cover the legitimate area. Therefore, adversaries will tend to not deviate the incoming poisoned dataset from the legitimate dataset i.e. not change λ much.

As shown in III and IV we can see that we achieve higher F-measure scores when the malicious samples are increased and the deviation λ is larger. This makes sense since the malicious data points in those cases are far from the boundaries of the legitimate behavior, making it easier to be detected by the classifier. Again these are expected results for XFSD from what we have previously learned from FSD and, as in the scenario of pre-training strategy, the post-training strategy shows similar tendencies across models for all configurations in both attack cases in post-training.

When focusing on the behavior of the different XAI algorithms, unsurprisingly the tendencies are the same as they were for the pre-training cases. XFSD-SHAP is the strongest and performs with a higher F-measure than base FSD proving that adding XAI-values can have a positive effect as discussed throughout this paper. The dominance of SHAP is evident across the results of Tables III and IV regardless the λ and the adversarial rate configurations. The same highlights can be extracted for the post-training case as for the pre-taining one, suggesting that SHAP is more accurate, as it is destined to work well with RNN and time series data, while the contribution of XAI is evident based on the performance results of XFSD against FSD.

B. Results of XFSD positional attacks

With the architecture settled in terms of scenario and algorithms, we are using similar visualization of aggregates to review the difference of different positional attacks.

1) Attack case III, IV, V in pre-training: Our approach takes advantage of the sequential relationship between the data samples from different time steps. That is, an adversary could launch positional attacks to bypass the classifier. To emulate this behavior we consider that in attack case III all legitimate data appear before the malicious data, in case IV all legitimate samples appear after the malicious data, while in case V malicious data distributed in between legitimate samples.

Depending on the adversarial rate, the model may ignore parts of adversarial samples, e.g., with 5% of adversarial data, a model for attack case III learns 95% of legitimate data before learning from the 5% of malicious data, which has less impact on the final performance in comparison to 55% legitimate and 45% malicious samples. This impact is shown in Tables V, VI and VII. It is evident that the higher adversarial rates cause higher negative impact on the model.

Among the different attack cases, the lowest impact is for attack case IV, in which the model learns first on malicious data. The highest negative impact appears for attack case III, while attack case V, as expected is in between the other two attack cases. Overall, based on the experimentation results across all cases, XFSD is proved to perform better, advocating the beneficial impact of the XAI over the FSD approach.

2) Attack case III, IV, V in Post-training: The results are reflected in Tables VIII, IX and X and as before the comparative tendencies reveal that XAI algorithms do not differ much depending on the order or the distribution of malicious samples. SHAP is the most accurate variant showcasing the best performance especially for the attack case V.

C. Advanced data poisoning

Taking inspiration from Miao et al. [28] we added more advanced adversarial techniques based on their optimal attack framework which uses a bilevel optimization. The two types

of data poisoning attacks i.e., the availability attack (case VI) and the target attack (case VII) were applied to evaluate XFSD effectiveness. The outcomes which are mainly on the pre-training strategy (data poisoning) showcase that the optimization and these types of attacks are highly efficient, even with the addition of XAI values, rendering the system beneath random guessing during some adversarial configurations. The order in which the results are presented in Tables XI, XII, XIII and XIV are 1) SHAP, 2) CAM, 3) LIME, and 4) FSD.

The outcome for this experimentation using optimal standard deviation and different adversarial rates based on the newly introduced optimized strategy is a great opportunity to gain more insight into the robustness of our model. Our model is still performing well within the area, but for these challenging attack cases it dips beneath random guessing in certain configurations. It is evident again that XAI gives an advantage to the XFSD approach, while SHAP remains the most prominent XAI method. It is noted that the advanced attack strategies with the optimization is more effective in skewing the perception of the classifiers.

D. Aggregation Method

In an effort to reduce the dimmentionality of the problem and to explore whether a different aggregation approach of the XAI values may have a positive impact to the model, we extended attack case I, as reported in Table XV. We proceeded to an aggregation before the concatenation of the fixed length vectors from the time window. However, as can be seen in Table XV, no improvement is reported, implying that a different aggregation approach cannot be considered a prominent method for improving the model.

VII. Conclusion

The purpose of XFSD was to improve the existing FSD framework. The landscape of MCS for IoT present a plethora of research challenges with one of the more prominent ones being to address data trustworthiness. With FSD we took a first step, while with XFSD we attempt to find a way for improving the robustness of the models that can be derived from the framework by leveraging advanced deep learning capabilities with explainability qualities. The XAI extension provided an notable improvement, making classifiers able to distinguish between legit and malicious samples. In our work we showcased that, for time series data, the use of LSTM/RNN in combination with SHAP was more accurate than applying it with CAM. This can be explained by the way CAM works in combination with CNN for time-series i.e., CAM considers areas of importance, whereas SHAP considers the whole ordinal information sequence. This is observed to be advantageous in dataset such as ours in which one has to consider the issues pertaining to concept drift, i.e., unforeseen changes over time. The information that can be extracted from sequential order of data is proven important considering the improvement in the accuracy of the classifiers when introducing XAI values and the better performance when compared to non-sequential algorithms, such as LIME which failed to contribute in the XFSD setup. As future work, we aim to explore more robust approaches in XFSD that could tackle the challenging availability and targeted adversarial attacks.

VIII. Acknowledgment

This work was supported by European Commission under the STAR project GA No. 956573.

References

[1] T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, and N. Diaz Rodriguez, "Explainable artificial intelligence (xai) on timeseries data: A survey," 04 2021.

[2] S. Gisdakis, T. Giannetsos, and P. Papadimitratos, "Shield: a data verification framework for participatory sensing systems," 07 2015.

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

[4] E. Veliou, D. Papamartzivanos, S. A. Menesidou, P. Gouvas, T. Giannetsos *et al.*, "Artificial intelligence and secure manufacturing: Filling gaps in making industrial environments safer," *Trusted Artificial Intelligence in Manufacturing*, p. 30, 2021.

[5] J. M. Rožanec, D. Papamartzivanos, E. Veliou, T. Anastasiou, J. Keizer, B. Fortuna, and D. Mladenić, "Machine beats machine: Machine learning models to defend against adversarial attacks," 2022.

[6] T. Anastasiou, S. Karagiorgou, P. Petrou, D. Papamartzivanos, T. Giannetsos, G. Tsirigotaki, and J. Keizer, "Towards robustifying image classifiers against the perils of adversarial attacks on artificial intelligence systems," *Sensors*, vol. 22, no. 18, 2022.

[7] 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.

[8] S. Afzal-Houshmand, S. Homayoun, and T. Giannetsos, "A perfect match: Deep learning towards enhanced data trustworthiness in crowdsensing systems," in *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, 2021, pp. 258–264.

[9] B. Guo, Z. Yu, X. Zhou, and D. Zhang, "From participatory sensing to mobile crowd sensing," in *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, 2014, pp. 593–598.

[10] 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.

[11] 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.

[12] J.-Y. Kim and S.-B. Cho, "Electric energy consumption prediction by deep learning with state explainable autoencoder," *Energies*, vol. 12, p. 739, 02 2019.

[13] K. E. Mokhtari, B. P. Higdon, and A. Başar, "Interpreting financial time series with shap values," in *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '19. USA: IBM Corp., 2019, p. 166–172.

[14] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *Entropy*, vol. 23, p. 18, 12 2020.

[15] D. Dataman. Explain your model with the shap values. [Online]. Available: https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d

[16] B. Khaleghi. The how of explainable ai: Post-modelling explainability. [Online]. Available: https://towardsdatascience.com/the-how-of-explainable-ai-post-modelling-explainability-8b4cbc7adf5f

[17] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 1578–1585.

[18] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Accurate and interpretable evaluation of surgical skills from kinematic data using fully convolutional neural networks," 08 2019.

[19] F. Oviedo, Z. Ren, S. Sun, C. M. Settens, Z. Liu, N. T. P. Hartono, R. Savitha, B. L. DeCost, S. I. P. Tian, G. Romano, A. G. Kusne, and T. Buonassisi, "Fast classification of small x-ray diffraction datasets using data augmentation and deep neural networks," *ArXiv*, vol. abs/1811.08425, 2018.

[20] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, and G. Shroff, "Convtimenet: A pre-trained deep convolutional neural network for time series classification," 07 2019, pp. 1–8.

[21] L. Zhou, C. Ma, X. Shi, D. Zhang, W. Li, and L. Wu, "Salience-cam: Visual explanations from convolutional neural networks via salience score," 07 2021, pp. 1–8.

[22] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1135–1144. [Online]. Available: https://doi.org/10.1145/2939672.2939778

[23] L. Hulstaert. Understanding model predictions with lime. [Online]. Available: https://towardsdatascience.com/understanding-model-predictions-with-lime-a582fdff3a3b

[24] A. Sharma. Decrypting your machine learning model using lime. [Online]. Available: https://towardsdatascience.com/decrypting-your-machine-learning-model-using-lime-5adc035109b5

[25] 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.

[26] D. Mercier, A. Dengel, and S. Ahmed, "Patchx: Explaining deep models by intelligible pattern patches for time-series classification," 02 2021.

[27] D. sensing Lab, ""strata santa clara dataset"." [Online]. Available: http://datasensinglab.com/

[28] 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.

APPENDIX A
EXPERIMENTAL RESULTS

	λ	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.963	0.929	0.941	0.933	0.915	0.881	0.865	0.848	0.826
	1	0.952	0.882	0.909	0.889	0.895	0.856	0.823	0.804	0.771
	2	0.914	0.837	0.876	0.849	0.828	0.819	0.792	0.769	0.719
CAM	0.5	0.886	0.841	0.8	0.791	0.777	0.741	0.725	0.707	0.686
	1	0.841	0.81	0.769	0.749	0.752	0.716	0.681	0.664	0.631
	2	0.794	0.774	0.735	0.708	0.686	0.674	0.652	0.629	0.579
LIME	0.5	0.85	0.805	0.764	0.755	0.741	0.705	0.689	0.671	0.65
	1	0.805	0.774	0.733	0.713	0.716	0.68	0.645	0.628	0.595
	2	0.758	0.738	0.699	0.672	0.65	0.638	0.616	0.593	0.543
FSD	0.5	0.843	0.797	0.756	0.747	0.733	0.695	0.682	0.663	0.642
	1	0.797	0.765	0.725	0.703	0.707	0.672	0.637	0.619	0.587
	2	0.749	0.729	0.691	0.664	0.641	0.629	0.608	0.583	0.533

Table I: Results of various XAI algorithms + FSD for Attack Case I in pre-training attack strategy

	λ	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.875	0.861	0.839	0.821	0.802	0.782	0.762	0.718	0.684
	1	0.836	0.811	0.802	0.789	0.769	0.753	0.719	0.669	0.635
	2	0.799	0.789	0.784	0.779	0.742	0.727	0.684	0.638	0.589
CAM	0.5	0.866	0.852	0.83	0.811	0.792	0.772	0.752	0.708	0.675
	1	0.827	0.801	0.792	0.779	0.759	0.743	0.709	0.659	0.625
	2	0.789	0.779	0.776	0.769	0.732	0.718	0.676	0.627	0.579
LIME	0.5	0.858	0.844	0.822	0.803	0.784	0.764	0.744	0.7	0.667
	1	0.819	0.793	0.784	0.771	0.751	0.735	0.701	0.651	0.617
	2	0.781	0.771	0.768	0.761	0.724	0.71	0.668	0.619	0.571
FSD	0.5	0.791	0.777	0.755	0.736	0.717	0.697	0.677	0.633	0.6
	1	0.752	0.726	0.717	0.704	0.684	0.668	0.634	0.584	0.55
	2	0.714	0.705	0.701	0.694	0.657	0.644	0.601	0.552	0.504

Table II: Results of various XAI algorithms + FSD for Attack Case II in pre-training attack strategy

	λ	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.698	0.709	0.738	0.759	0.782	0.804	0.832	0.862	0.901
	1	0.72	0.738	0.755	0.774	0.796	0.839	0.853	0.905	0.945
	2	0.729	0.747	0.763	0.795	0.815	0.857	0.871	0.939	0.986
CAM	0.5	0.647	0.66	0.687	0.71	0.734	0.755	0.782	0.813	0.853
	1	0.671	0.69	0.708	0.725	0.747	0.79	0.803	0.855	0.896
	2	0.678	0.699	0.713	0.746	0.766	0.807	0.823	0.89	0.947
LIME	0.5	0.579	0.592	0.619	0.642	0.666	0.687	0.714	0.745	0.785
	1	0.603	0.622	0.64	0.657	0.679	0.722	0.735	0.787	0.828
	2	0.61	0.631	0.645	0.678	0.698	0.739	0.755	0.822	0.879
FSD	0.5	0.578	0.609	0.623	0.633	0.654	0.675	0.703	0.723	0.769
	1	0.599	0.622	0.646	0.669	0.705	0.714	0.74	0.761	0.789
	2	0.617	0.627	0.669	0.693	0.724	0.727	0.771	0.791	0.842

Table III: Results of various XAI algorithms + FSD for Attack Case I in post-training attack strategy

	λ	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.668	0.688	0.712	0.739	0.791	0.843	0.868	0.897	0.942
	1	0.738	0.756	0.769	0.787	0.807	0.856	0.877	0.921	0.967
	2	0.78	0.787	0.801	0.817	0.841	0.872	0.888	0.932	0.982
CAM	0.5	0.665	0.685	0.709	0.737	0.788	0.842	0.865	0.895	0.94
	1	0.735	0.755	0.766	0.783	0.805	0.855	0.875	0.918	0.965
	2	0.779	0.784	0.798	0.815	0.839	0.869	0.884	0.928	0.978
LIME	0.5	0.651	0.671	0.695	0.723	0.774	0.828	0.851	0.881	0.926
	1	0.721	0.741	0.752	0.769	0.791	0.841	0.861	0.904	0.951
	2	0.765	0.77	0.784	0.801	0.825	0.855	0.87	0.914	0.991
FSD	0.5	0.568	0.588	0.612	0.64	0.691	0.745	0.768	0.798	0.843
	1	0.638	0.658	0.669	0.686	0.708	0.758	0.778	0.821	0.868
	2	0.682	0.687	0.701	0.718	0.742	0.772	0.787	0.831	0.908

Table IV: Results of various XAI algorithms + FSD for Attack Case II in post-training attack strategy

	λ	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.971	0.865	0.83	0.816	0.801	0.78	0.775	0.755	0.69
	1	0.882	0.846	0.804	0.788	0.764	0.755	0.754	0.73	0.645
	2	0.847	0.817	0.78	0.76	0.745	0.737	0.725	0.694	0.611
CAM	0.5	0.901	0.856	0.815	0.806	0.792	0.756	0.74	0.722	0.701
	1	0.856	0.825	0.784	0.764	0.767	0.731	0.696	0.679	0.646
	2	0.809	0.789	0.75	0.723	0.701	0.689	0.667	0.644	0.594
LIME	0.5	0.83	0.785	0.744	0.735	0.721	0.685	0.669	0.651	0.63
	1	0.785	0.754	0.713	0.693	0.696	0.66	0.625	0.608	0.575
	2	0.738	0.718	0.679	0.652	0.63	0.618	0.596	0.573	0.523
FSD	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 for different XAI algorithms.

	λ	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.958	0.894	0.858	0.851	0.838	0.822	0.813	0.787	0.729
	1	0.904	0.858	0.824	0.814	0.797	0.777	0.786	0.756	0.672
	2	0.877	0.822	0.787	0.766	0.76	0.76	0.746	0.734	0.646
CAM	0.5	0.841	0.796	0.755	0.746	0.732	0.696	0.68	0.662	0.641
	1	0.796	0.765	0.724	0.704	0.707	0.671	0.636	0.619	0.586
	2	0.749	0.729	0.69	0.663	0.641	0.629	0.607	0.584	0.534
LIME	0.5	0.77	0.725	0.684	0.675	0.661	0.625	0.609	0.591	0.57
	1	0.725	0.694	0.653	0.633	0.636	0.6	0.565	0.548	0.515
	2	0.678	0.658	0.619	0.592	0.57	0.558	0.536	0.513	0.463
FSD	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%
SHAP	0.5	0.954	0.907	0.871	0.82	0.796	0.782	0.773	0.743	0.664
	1	0.93	0.855	0.818	0.795	0.765	0.746	0.739	0.706	0.615
	2	0.889	0.827	0.783	0.761	0.735	0.713	0.683	0.658	0.593
CAM	0.5	0.814	0.769	0.728	0.719	0.705	0.669	0.653	0.635	0.614
	1	0.769	0.738	0.697	0.677	0.68	0.644	0.609	0.592	0.559
	2	0.722	0.702	0.663	0.636	0.614	0.602	0.58	0.557	0.507
LIME	0.5	0.77	0.725	0.684	0.675	0.661	0.625	0.609	0.591	0.57
	1	0.725	0.694	0.653	0.633	0.636	0.6	0.565	0.548	0.515
	2	0.678	0.658	0.619	0.592	0.57	0.558	0.536	0.513	0.463
FSD	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.

	λ	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.698	0.714	0.75	0.78	0.814	0.833	0.88	0.899	0.931
	1	0.731	0.754	0.78	0.812	0.858	0.876	0.909	0.932	0.965
	2	0.777	0.78	0.801	0.833	0.873	0.894	0.931	0.95	0.981
CAM	0.5	0.656	0.669	0.696	0.719	0.743	0.764	0.791	0.822	0.862
	1	0.68	0.699	0.717	0.734	0.756	0.799	0.812	0.864	0.905
	2	0.687	0.708	0.722	0.755	0.775	0.816	0.832	0.899	0.956
LIME	0.5	0.585	0.598	0.625	0.648	0.672	0.693	0.72	0.751	0.791
	1	0.609	0.628	0.646	0.663	0.685	0.728	0.741	0.793	0.834
	2	0.616	0.637	0.651	0.684	0.704	0.745	0.761	0.828	0.885
FSD	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 VIII: F-measures for Attack Case III (benign first) with attack samples generated by Attack Case I in post-training.

	λ	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.617	0.666	0.718	0.739	0.767	0.796	0.826	0.873	0.915
	1	0.664	0.727	0.731	0.782	0.835	0.848	0.897	0.899	0.93
	2	0.727	0.737	0.778	0.809	0.852	0.871	0.913	0.929	0.981
CAM	0.5	0.596	0.609	0.636	0.659	0.683	0.704	0.731	0.762	0.802
	1	0.62	0.639	0.657	0.674	0.696	0.739	0.752	0.804	0.845
	2	0.627	0.648	0.662	0.695	0.715	0.756	0.772	0.839	0.896
LIME	0.5	0.525	0.538	0.565	0.588	0.612	0.633	0.66	0.691	0.731
	1	0.549	0.568	0.586	0.603	0.625	0.668	0.681	0.733	0.774
	2	0.556	0.577	0.591	0.624	0.644	0.685	0.701	0.768	0.825
FSD	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 IX: F-measures for Attack Case IV (malicious first) with samples generated by Attack Case I in post-training.

	λ	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.699	0.731	0.778	0.803	0.849	0.869	0.9	0.923	0.98
	1	0.712	0.751	0.796	0.824	0.868	0.891	0.934	0.958	0.981
	2	0.726	0.781	0.8	0.852	0.891	0.91	0.959	0.979	0.989
CAM	0.5	0.616	0.629	0.656	0.679	0.703	0.724	0.751	0.782	0.822
	1	0.64	0.659	0.677	0.694	0.716	0.759	0.772	0.824	0.865
	2	0.647	0.668	0.682	0.715	0.735	0.776	0.792	0.859	0.916
LIME	0.5	0.535	0.548	0.575	0.598	0.622	0.643	0.67	0.701	0.741
	1	0.559	0.578	0.596	0.613	0.635	0.678	0.691	0.743	0.784
	2	0.566	0.587	0.601	0.634	0.654	0.695	0.711	0.778	0.835
FSD	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 X: F-measures for Attack Case V post-training attack strategy with attack samples generated by Attack Case I.

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Optimal	0.701	0.681	0.642	0.615	0.593	0.581	0.559	0.536	0.486
Optimal	0.686	0.666	0.627	0.6	0.578	0.566	0.544	0.521	0.471
Optimal	0.674	0.654	0.615	0.588	0.566	0.554	0.532	0.509	0.459
Optimal	0.661	0.641	0.602	0.575	0.553	0.541	0.519	0.496	0.446

Table XI: Optimized Data poisoning availability attack case VI for the scenario of μ

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Optimal	0.687	0.677	0.674	0.667	0.63	0.616	0.574	0.525	0.477
Optimal	0.674	0.664	0.661	0.654	0.617	0.603	0.561	0.512	0.464
Optimal	0.662	0.652	0.649	0.642	0.605	0.591	0.549	0.5	0.452
Optimal	0.647	0.637	0.634	0.627	0.59	0.576	0.534	0.485	0.437

Table XII: Optimized Data poisoning availability attack case VI for the scenario with σ

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Optimal	0.788	0.757	0.716	0.696	0.699	0.663	0.628	0.611	0.578
Optimal	0.638	0.607	0.566	0.546	0.549	0.513	0.478	0.461	0.428
Optimal	0.625	0.594	0.553	0.533	0.536	0.5	0.465	0.448	0.415
Optimal	0.613	0.582	0.541	0.521	0.524	0.488	0.453	0.436	0.403

Table XIII: Optimized Data poisoning target attack case VII for the scenario with μ

λ	Adversarial Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Optimal	0.717	0.707	0.704	0.697	0.66	0.646	0.604	0.555	0.507
Optimal	0.702	0.692	0.689	0.682	0.645	0.631	0.589	0.54	0.492
Optimal	0.677	0.667	0.664	0.657	0.62	0.606	0.564	0.515	0.467
Optimal	0.662	0.652	0.649	0.642	0.605	0.591	0.549	0.5	0.452

Table XIV: Optimized Data poisoning target attack case VII for the scenario of σ

	λ	Adversarial Rate								
		5%	10%	15%	20%	25%	30%	35%	40%	45%
SHAP	0.5	0.865	0.813	0.825	0.817	0.799	0.765	0.749	0.732	0.71
	1	0.836	0.766	0.793	0.773	0.779	0.74	0.707	0.688	0.655
	2	0.798	0.721	0.76	0.733	0.712	0.703	0.676	0.653	0.603
CAM	0.5	0.77	0.725	0.684	0.675	0.661	0.625	0.609	0.591	0.57
	1	0.725	0.694	0.653	0.633	0.636	0.6	0.565	0.548	0.515
	2	0.678	0.658	0.619	0.592	0.57	0.558	0.536	0.513	0.463
LIME	0.5	0.734	0.689	0.648	0.639	0.625	0.589	0.573	0.555	0.534
	1	0.689	0.658	0.617	0.597	0.6	0.564	0.529	0.512	0.479
	2	0.642	0.622	0.583	0.556	0.534	0.522	0.5	0.477	0.427
FSD	0.5	0.724	0.681	0.64	0.631	0.617	0.579	0.566	0.547	0.526
	1	0.681	0.649	0.609	0.587	0.591	0.556	0.521	0.503	0.471
	2	0.633	0.613	0.575	0.548	0.525	0.513	0.492	0.467	0.417

Table XV: F-measures for Attack Case I pre-training attack strategy where the default aggregation is applied before concatenation of fixed length vectors