Detection of Sensors Used for Adversarial Examples against Machine Learning Models

Ade Kurniawan *, Yuichi Ohsita , Shyam Maisuria , Masayuki Murata

Posted Date: 6 November 2023

doi: 10.20944/preprints202311.0328.v1

Keywords: Adversarial examples; compromised sensor; detection; multiple sensors; human activity recognition

Article

Detection of Sensors Used for Adversarial Examples against Machine Learning Models

Ade Kurniawan [1,*], Yuichi Ohsita [2], Shyam Maisuria [1] and Masayuki Murata [1]

[1] Graduate School of Information Science and Technology, Osaka University, Suita 565-0871, Osaka, Japan
[2] Cybermedia Center, Osaka University, Toyonaka 565-0871, Osaka, Japan
* Correspondence: e-mail: k-ade@ist.osaka-u.ac.jp

Abstract: Machine learning (ML) systems that use sensors obtain observations from sensors and use them to recognize and interpret the current situation. These systems are susceptible to sensor-based adversarial example attacks (AEs); if some sensors are vulnerable and can be compromised by an attacker, the attacker can change the output of the system by changing the values of the sensors. The detection of compromised sensors is important to defend the system against sensor-based AEs, because we can check the sensors and replace them by detecting the sensors used by the attacker. In this paper, we propose a method to detect the sensors used in sensor-based AEs by utilizing the features of the attack that cannot be avoided. In this method, we introduced a model called the feature-removable model (FRM), which allows us to select the features used as inputs into the model. Our method detects the sensors used in sensor-based AEs by finding the inconsistencies between the outputs of the FRM obtained by changing the selected features. We evaluated our method using a human activity recognition model with sensors attached to the user's chest, wrist, and ankle. We demonstrate that our method can accurately detect sensors used by the attacker and achieves an average *Recall of Detection* of 0.92, and the average *Precision of Detection* is 0.72.

Keywords: adversarial examples; compromised sensor; detection; multiple sensors; human activity recognition

1. Introduction

The integration of machine learning (ML) with multiple sensors has revolutionized various critical domains, such as healthcare [1], autonomous vehicles [2], and other fields [3]. In these systems, observations from sensor devices are collected via the Internet or a network managed by the service provider. Then, ML models recognize the current situation using the collected observations as their input. The utilization of observations from multiple sensor devices in these systems has shown to significantly enhance recognition accuracy. For instance, Namazi et al. developed a method to detect and track continuous objects surrounding vehicles, locate objects in front of the vehicle, and mitigate occlusion issues to provide more precise readings [4]. Similarly, multiple systems, Zhou et al.[5] and Yuan et al.[6] proposed systems to recognize human activities in low-light indoor conditions by using multiple sensor devices.

However, as systems using ML models have become more widely used, these models have also become the target of attacks. Adversarial examples (AEs) are inputs designed by an adversary that causes a ML system to generate incorrect outputs. Finlayson et al. showed that an adversary can exploit the vulnerability of ML models by creating AEs in critical domains, such as medical [7]. These AEs contain subtle changes that are invisible to humans but can mislead the model's predictions. This poses a significant threat to the systems based on ML models, particularly in sensitive fields.

Considering the systems using multiple sensors, some sensors may be vulnerable and can be used to generate AEs. The data manipulation of sensor data by compromising the software in the sensor device was demonstrated [8]. Monjur et al. have been demonstrated that the data manipulation is also possible by modifying the hardware if the attacker can physically access the sensor device [9]. We also demonstrated that the attacker can change the output of the ML models using multiple sensors if the

 doi:10.20944/preprints202311.0328.v1

attacker can manipulate the values from a part of sensors [10]. That is, manipulation on all sensor values is not necessary for the attacks to change the output values of the ML models. We call this attack *sensor-based AEs*. However, it is difficult to protect all sensor devices, because the risk of the existence of the vulnerable sensor devices increases as the number of sensor devices increases. Therefore, we need a method to protect ML models even if a part of sensors are compromised by the attacker.

Many countermeasures to mitigate the risks posed by AEs have been proposed. One approach is to make the ML models robust against AEs. Adversarial training (AT) is one of the methods used to enhance the robustness of ML models [11]. In this method, ML models are iteratively trained using clean and adversarial examples generated for the current model. By utilizing the AEs, this method enhances the robustness of the model. However, the model trained by AT is only robust against the attack generated during the training and may be vulnerable to new attacks [12].

Another approach to countering AEs is to detect attacks. Especially in the case of sensor-based AEs, the detection of the sensors used in the sensor-based AEs is essential. By detecting the sensors used in the attack, we can check the sensors and replace them. Many methods to detect AEs have also been proposed [13–20]. Hendrycks and Gimpel introduced detection methods for adversarial examples based on identifying implausible gradients [16,17]. Metzen et al. proposed a method based on Lipschitz continuity to detect adversarial perturbations, which involve subtle changes applied to clean examples [19].

However, all existing detection methods only detect attacks and do not detect the features changed by the attacker because they aim to detect the inputs to be blocked. On the other hand, detecting the sensors used by the attacker is essential for systems with multiple sensors, as discussed above. In addition, existing detection methods can be avoided by modifying the AE generation process [21]. This is caused by the features used to detect AEs; the existing methods use features of the AEs, but an attacker with the knowledge of the detector can change the features. That is, the detection method should use the features of the AEs that cannot be avoided by an attacker.

We propose a method to detect the attacks and the sensors used by the attackers. In this method, we introduce a model called the feature-removable model (FRM) that allows us to select the features used as an input into the model. We obtain the outputs of the FRM using all features and features from some of the sensors. If we find inconsistencies between the outputs, our method detects the attacks. Then we also detect the sensors the attacker uses by finding the sensors causing the inconsistency.

After detecting the sensors the attacker uses, we can check and replace them. Nevertheless, our FRM can also be used after the identification; we can obtain the output of the FRM without using the features from the detected sensors to avoid the impact of the attacks, though the accuracy of the model might decrease compared with the case that we can use all features.

To the best of our knowledge, we are the first to propose a method that can detect the sensors used in sensor-based AEs. Our method is based on the features of the sensor-based AEs that the attacker cannot avoid; the output of the ML model is altered when the values from the sensors used by the attacker are incorporated.

The rest of this paper is organized as follows: Section 2 provides an overview of related work on defense techniques against AEs. Section 3 presents a definition of the attack discussed in this paper. Section 4 presents our framework against attacks from a part of sensors in detail. Section 5 evaluates the effectiveness of the proposed countermeasures. Finally, Section 6 concludes the paper and discusses future research directions in this area.

2. Related Work

Many countermeasures to mitigate the risks posed by AEs have been proposed. One of the approaches is the AT. AT is a technique used to make the ML model robust against AEs by incorporating AEs during the training process. AT minimizes the loss for clean inputs and the generated AEs. Ensemble Adversarial Training (EAT) [12] is a method that improves the robustness of AT. In this

 doi:10.20944/preprints202311.0328.v1

method, multiple models are trained at the same time, and AEs generated for a model is used to train the other models.

However, AT has a limitation. It relies on specific attacks generated during training, which hampers its ability to defend against new and unseen attacks [22,23].

Another approach to mitigating the risks posed by AEs is to detect AEs. Hendricks and Gimpel introduced three detection methods, referred to as the H&G detection methods [16,17]. They are based on 1) the coefficients in a PCA-whitened input, 2) the Softmax distributions, and 3) the reconstruction errors of the inputs reconstructed by an auxiliary decoder.

Another approach is based on ML models to detect AEs. Gong et al. [13] proposed a method based on a binary classifier trained independently from the original model. They trained the binary classifier to output 0 for the clean data and 1 for the AEs. Metzen et al. [19] also proposed a method based on binary classifiers. They constructed binary classifiers as subnetworks connected to each layer of the original main model. Grosse et al. augmented an ML model with an additional output, in which the model is trained to classify adversarial inputs [14]. Hosseini et al. trained an ML model to smoothly decrease its confidence in the original label and instead predicted that the input is "invalid" when it is more perturbed [18]. Miller et al. proposed a detection method for anomaly detection, which utilizes a separate unsupervised learning procedure to detect AEs inputs [20].

In this paper, we discuss a method to detect sensors used in sensor-based AEs. This method should satisfy the following requirements. 1) It should detect AEs accurately. 2) It should detect sensors used in sensor-based AEs. 3) It should utilize the features of sensor-based AEs that cannot be avoided by an attacker. If an attacker can avoid the features used by the detector, AEs that cannot be detected are possible.

Table 1 shows the existing method to detect AEs and includes whether each method meets the requirements. As shown in Table 1, none of the existing methods can detect sensors used in sensor-based AEs, because they did not focus on the sensor-based AEs. Furthermore, none of the existing methods utilizes features that cannot be avoided by the attacker. If an attacker can change the features used by a detector, they can generate AEs that cannot be detected. Carlini et al. evaluated AE detection methods in a white-box setting and demonstrated that AEs capable of evading detection can be generated for all the methods evaluated [21].

In this paper, we propose a method to detect the sensors used in sensor-based AEs that satisfies all of the above requirements. Our method utilizes the features of the attacks; that the results of the classification are change by using the features from the sensors used in the attack. Such features of the attacks cannot be avoided by the attacker.

 doi:10.20944/preprints202311.0328.v1

Table 1. Comparison of different AE detection methods.

Methods	Overview	Detection of attacks	Detection of sensors used in sensor-based AEs	Sure utilization of the unavoidable features of AEs
H & G [16,17]	Detect AEs based on 1) the coefficients in a PCA-whitened input, 2) the Softmax distributions and 3) the reconstruction errors.	Yes	No	No
Metzen et al. [19]	Detect AEs by binary detector sub-networks using the outputs of each layer of the original model as inputs.	Yes	No	No
Gong et al. [13]	Detect AEs by a binary classifier that is trained independently from the original model.	Yes	No	No
Grosse et al. [14]	Augments a ML model with an additional output, in which the model is trained to classify adversarial inputs.	Yes	No	No
Hosseini et al. [18]	Train a ML model so that it outputs lower confidence on the original label and instead predicts that the input is "invalid", as the input is more perturbed	Yes	No	No
Miller et al. [20]	Detect AEs using unsupervised anomaly detectors.	Yes	No"	No
This paper	Detect AEs from a part of sensors and identify the sensors used by the attacker by finding the inconsistency between the results using all features and without using features from a part of sensors	Yes	Yes	Yes

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 November 2023 doi:10.20944/preprints202311.0328.v1

3. Sensor-based Adversarial examples

In this paper, we focus on the system that gathers values from multiple sensors and performs classification tasks based on ML models. We call this system as the target system.

We model the target system as the function $f(x_{0:t})$ where $x_{0:t} = (x_0, x_1, \ldots, x_t)$ is the input of the target system built from the sensor data received from time 0 to time t and x_t is the vector corresponding to the sensor values at time t. We refer to the j-th element of the model's output as $f_j(x_{0:t})$, and $f_j(x_{0:t})$ denotes the probability that the state at time t is classified into the i-th class. $f(x_{0:t})$ represents the classification outcome at time t.

The vector x_t is constructed of the values from multiple sensors. The values of the compromised sensors can be monitored and modified by the attacker. The information of the compromised sensors are represented by the vector $\mathbf{B}$, which is defined as $\mathbf{B} = (b_1, b_2, \ldots, b_m)$; $b_i = 1$ if the i-th value is from the compromised sensor. The sensor values that the attacker can monitor and modify at times t are given by $\dot{x}_t = \mathbf{B} \circ x_t$, where $\circ$ stands for the element-wise product.

Based on the sensor values of the compromised sensors, the attacker creates perturbation. The sensor values including the attacks become $x'_t = x_t + \mathbf{B} \circ G(\dot{x}_{0:t})$ where $G(\dot{x}_{0:t})$ is the attack generator and $\dot{x}_{0:t} = (\dot{x}_0, \dot{x}_1, \ldots, \dot{x}_t)$. The attacker can generate the attacks by training $G(\dot{x}_{0:t})$ so that the output becomes the class the attacker wants. In this paper, we assume that an attacker has sufficient information about the target system.

On the other hand, the victim who manages the target system does not know the compromised sensors but knows the set of sensors with the same risk. For example, sensors connected to the same computer might be compromised at the same time if the computer is vulnerable. The same type of sensors may have the same vulnerability. In this paper, we denote the set of risks by $\mathcal{R}$ and the set of sensors with risk $r \in \mathcal{R}$ by S_r. We assume that the victim has the information of S_r for all $r \in \mathcal{R}$.

4. Framework against sensor-based adversarial examples

The attacker aims to change the classification results by changing the values of the compromised sensors. That is, the classification results obtained from compromised sensors are different from those obtained without using compromised sensors. Our method detects not only sensor-based AEs but also the sensors used in the attack by detecting these inconsistencies.

Figure 1 shows an overview of our method. In our method, we introduce a model called the feature-removed model (FRM), which allows us to select the features used for classification. By using the FRM, we can obtain multiple results by changing the features used for classification. Then, we detect attacks and sensors used in the attack by comparing the results and identifying any inconsistencies

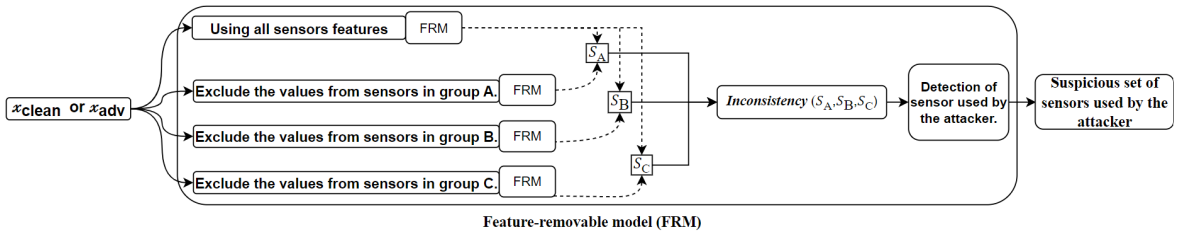


Figure 1. Illustration of countermeasure stage flow.

4.1. Feature removable model

In this subsection, we propose a model called the FRM. The FRM is a model that allows us to exclude specific features. The FRM can be constructed by modifying the first and last layers of the original model. At the first layer, we add a function to exclude the features that are marked to be excluded. The eliminated features are set to 0. Then we apply the scaling based on the dropout [24]. The output of the first layer after scaling is obtained by

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 November 2023 doi:10.20944/preprints202311.0328.v1

$$o_{1,i} = a\left(\frac{N^{\text{all}}}{N^{\text{selected}}}\sum_k (w_{0,k,i}o_{0,k}) + b_{1,i}\right) \tag{1}$$

where $o_{i,j}$ is the value of the j-th node at the i-th layer, $w_{0,k,i}$ and $b_{1,i}$ are the weight and bias, and $a(\cdot)$ is the activation function. The number of all features is N^{all} and the number of selected features is N^{selected}. By this scaling, we have a similar number of activated nodes to the case of using all features even if we exclude some features.

In the final layer, we employed the sigmoid activation function to to allow the outputs for multiple classes to be large. If essential features are excluded, the FRM may be unable to identify the class. By allowing large outputs for multiple classes, the FRM can handle such cases by outputting large probabilities for all possible classes.

We train the FRM so that the outputted probabilities for all possible classes become large even if we exclude some features. In this paper, we train the FRM by selecting the features to be excluded randomly. During the training, we use the following loss function.

$$L^{\text{"removed-feature"}}(Y,T) = -\sum_i w(t_i)(t_i \log y_i + (1-t_i)\log(1-y_i)) \tag{2}$$

where Y is the model's output, t_i is the i-th element of T, y_i is the i-th element of Y, and T is the training label. If the training label is i, t_i is set to 1. If not, 0. $w(t_i)$ is defined as the weight for t_i. We set $w(0) \ll w(1)$ to include the training label in the output. By using this loss function, we set a large penalty for the case that the actual class is not included in the output classes.

4.2. Detection of Attacks and Identification of Compromised Sensors

Our method detects the sensor-based AE attacks and the sensors used in the attack by comparing the results of the FRM. If the outputted probability for a class obtained by using all features is high but the corresponding probability obtained by excluding values from a set of sensors, the sensors are suspicious. Based on such inconsistencies, our method detects sensors used in the sensor-based AEs.

Algorithm 1 shows the steps for detecting sensor-based AEs and the suspicious sensors used in the AEs. In these steps, we focus on the classes whose probabilities in the prediction results using all features exceed the threshold (α). Such classes are extracted as indices at Line 2 in **Algorithm 1**. Then, we obtain the results by excluding the values from the sensors with the risk ($r \in R$) and compare the results of the classes in the indices. If the difference exceeds a threshold (β) (Lines 7-11 in **Algorithm 1**), we detect the attacks and mark the sensors with the risk (r) as suspicious.

Algorithm 1 Detection of sensors used to generate AEs

Require: α (threshold for prediction), β (threshold for inconsistency), F (features), *model* (trained feature removable model), S (set of sensors), S_r (set of sensors with the risk r), R (set of risks)
Ensure: *DetectedSensors*

```
1:  P_all ← model.predict_using_selected_sensors(F, S)
2:  indices ← extract_indices_exceeding_threshold(P_all, α)
3:  P'_all ← extract_result_for_indices(P_all, indices)
4:  for each r in R do
5:      P_part ← model.predict_using_selected_sensors(F, S\R_s)
6:      P'_part ← extract_result_for_indices(P_part, indices)
7:      if |P'_all − P'_part| > β then
8:          for each s in S_r do
9:              DetectedSensor.Add(s)
10:         end for
11:     end if
12: end for
13: return DetectedSensor
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 November 2023 doi:10.20944/preprints202311.0328.v1

Algorithm 1 returns the set of suspicious sensors. Thus, if no suspicious sensors are found, we regard the current input as clean. But if at least one sensor is included in the set of suspicious sensors, we detect sensor-based AEs.

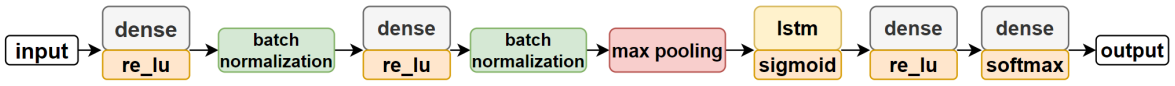
5. Experiment

5.1. Original target model, Dataset, and attacks

In our experiment, we used a system that recognized human activity as the target system. In this system, three devices were mounted at three places on the chest, left ankle, and right wrist. All three devices had 3D accelerometers; the device on the chest had an ECG sensor, and the other devices had 3D gyroscopes and 3D magnetometers. Each device sends its monitored value to the server. The server recognizes the user's current activity by using the deep neural network (DNN) to handle time-series data sent from the devices.

Figure 2a depicts the DNN structure utilized in our experiment, which was inspired by the HAR model proposed by [25]. In this experiment, we use this structure as the original model. In this structure, we construct a vector by concatenating the sensor values obtained at each time slot and use it as an input. This model handles the continuous inputs by using an LSTM layer, and outputs the probabilities of the activities for the inputs.

From this original model, we constructed an FRM, modifying only the first and last layers, as detailed in Section IV. To train the FRM, we utilized the Adam optimizer with a learning rate of 0.001 and a batch size of 32 for 100 epochs. We used a custom binary cross-entropy loss function for training FRM with weight parameters. We set w (0) to 1.0 and w (1) to 20.0.



(a)

(b)

Figure 2. The target model architecture (a) and the architecture of the generator AE model (b)

In this experiment, we use the MHealth dataset [26]. This dataset includes 12 physical activities (standing, sitting, lying down, walking, climbing stairs, bending forward, lifting arms forward, knees, cycling, jogging, running, and jumping back and forth) for ten subjects. In this paper, we regard subjects 9 and 10 as the target subjects for whom the attacks are generated, and the data from the other subjects are used to train the FRM. The Mhealth dataset includes time series of sensor values. We extract the data with a length of 500 from this dataset for training and validation by using a sliding window. Table 2 shows the number of data extracted for each subject and each class.

In this paper, we assume that the attacker has compromised one of the three devices and can change the values of all sensors on the compromised device. We assumed that the victim does not know the compromised sensor but is aware that the sensors in the same device carry the same risk.

We generate the attack by using the attack generator trained for the target model (FRM)[10]. Figure 2b shows the structure of the generator. The generator is trained so that the outputs of the FRM using all features become the class the attacker wants. When training the generator, we assume that the attacker has enough information about the target model (FRM), and we use the data from subjects 1 to 8. For training the generator, we used the Adam optimizer with a learning rate of 0.002 and a batch size of 256 for 25 epochs.

 doi:10.20944/preprints202311.0328.v1

Table 2. Data that we used for each class and individual.

Subject	Standing	Lying Down	Walking	Climbing Stairs	Waist Bends Forward	Frontal Elevation of Arms	Knees Bending	Cycling	Jogging	Running	Jump Front and Back
1	3072	3072	3072	3072	3072	3132	3278	3179	3072	3072	1024
2	3072	3072	3072	3072	3132	3132	3380	3430	3072	3072	1024
3	3072	3072	3072	3072	3132	3132	3266	3379	3175	3072	1024
4	3072	3072	3072	3072	3132	3132	3266	3379	3175	3072	1024
5	3072	3072	3072	3072	3132	3132	7265	7141	2784	3072	1024
6	3072	3072	3072	3072	2766	2894	2816	3072	3072	3072	1024
7	3072	3072	3072	3072	2766	2894	2816	3072	3072	3072	1024
8	3072	3072	3072	3072	2458	2765	2867	3072	3072	3072	1025
9	3072	3072	3072	3072	2458	2765	2867	3072	3072	3072	1025
10	3072	3072	3072	3072	2458	2765	2867	3072	3072	3072	1025

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 November 2023 doi:10.20944/preprints202311.0328.v1

On top of TensorFlow and Keras, we constructed the FRM, original model, and generator models and trained them with four NVIDIA Quadro RTX 6000 GPU cards.

5.2. Property of the feature removable model without attack

Before demonstrating our method to detect the attacks, we present the properties of the FRM, comparing them with the original model. We define the identified class as the class whose corresponding output exceeds the threshold, where the threshold $= 0.60$. We use PrecisionPrecision and RecallRecall as metrics for this comparison. The definitions of PrecisionPrecision and RecallRecall are:

$$Precision = \frac{tp}{tp + fp} \tag{3}$$

$$Recall = \frac{tp}{tp + fn} \tag{4}$$

where tp is the number of activities that are correctly identified as the target class, fp is the number of activities that are identified as the target class but whose actual class is different, and fn is the number of activities whose actual class is the target class but that are not identified as the target class. The FRM can output probabilities larger than the threshold for multiple classes. If probabilities for multiple classes are lager than the threshold, we independently count all classes to calculate tp and fp.

Table 3 shows the comparison between the original model and FRM using all features. Table 3 indicates that the FRM is similar to the original model, while the *Precision* of the FRM is slightly lower than that of the original model. The *Precision* of the FRM is caused by that the FRM allows large probabilities for multiple classes. However, the FRM achieves a sufficiently high PrecisionPrecision, which is greater than 0.95.

Table 3. The full results of the original model compared with the features-removed model.

	Original Model		**Features-removed Model**	
Activities	*Precision*	*Recall*	*Precision*	*Recall*
Standing	1.00	1.00	0.98	1.00
Sitting	1.00	1.00	0.99	1.00
Lying down	1.00	1.00	0.99	1.00
Walking	1.00	1.00	0.97	1.00
Climbing stairs	0.98	1.00	0.96	0.99
Waist bends forward	1.00	1.00	0.99	1.00
Frontal elevation of arms	0.98	1.00	0.99	1.00
Knees bending	1.00	0.96	0.99	0.97
Cycling	0.96	1.00	0.98	1.00
Jogging	0.98	0.93	0.95	1.00
Running	0.94	0.97	0.98	1.00
Jump front and back	0.95	0.97	0.98	1.00
Average	**0.98**	**0.98**	**0.98**	**0.99**

*Red color represents 1.00, while yellow indicates a score less than 1.00

5.3. Property of the attack

In this section, we evaluate the attacks on FRM using all the features. Table 4 show the attack success rates based on sensors located on the ankle, wrist, and chest, respectively.

In this table, we consider attacks that result in the outputted probability of the FRM using all features for the target class exceeding 0.60 as successful attacks. We define the attack success ratio as $\frac{N_{\text{success}}}{N_{\text{attack}}}$, where N_{success} is the number of successful attacks and N_{attack} is the number of generated attacks.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 November 2023 doi:10.20944/preprints202311.0328.v1

Table 4. The success ratio of generated sensor-based AEs

Sensors attach	**Ground Truth Class**	**Target Attack Class**											
		St	*Si*	*Ly*	*Wa*	*Cs*	*Wb*	*Fe*	*Kb*	*Cy*	*Jo*	*Ru*	*Ju*
Ankle	Standing (*St*)		1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.90	0.80	0.00	1.00
	Sitting (*Si*)	0.20		1.00	0.30	0.90	1.00	1.00	1.00	0.83	0.80	1.00	1.00
	Lying down (*Ly*)	1.00	1.00		0.00	1.00	1.00	1.00	1.00	1.00	0.40	0.00	1.00
	Walking (*Wa*)	0.00	1.00	0.00		0.80	1.00	1.00	1.00	0.00	0.00	1.00	0.00
	Climbing stairs (*Cs*)	0.50	0.90	1.00	1.00		1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Waist bends forward (*Wb*)	0.70	1.00	1.00	1.00	1.00		1.00	1.00	1.00	0.00	1.00	1.00
	Frontal elevation of arms (*Fe*)	0.00	1.00	1.00	1.00	0.80	1.00		0.00	1.00	1.00	1.00	1.00
	Knees bending (*Kb*)	1.00	1.00	1.00	1.00	1.00	0.90	1.00		0.90	0.90	1.00	1.00
	Cycling (*Cy*)	0.00	1.00	1.00	1.00	1.00	0.00	1.00	0.80		1.00	1.00	1.00
	Jogging (*Jo*)	1.00	1.00	0.70	0.00	1.00	1.00	1.00	1.00	1.00		1.00	1.00
	Running (*Ru*)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00		1.00
	Jump front and back (*Ju*)	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	
Wrist	Standing (*St*)		1.00	0.00	0.00	1.00	0.00	1.00	1.00	0.00	0.20	0.00	0.00
	Sitting (*Si*)	1.00		0.00	0.00	1.00	0.60	0.50	1.00	0.80	0.00	0.00	0.10
	Lying down (*Ly*)	0.00	1.00		1.00	0.50	0.00	0.00	0.70	1.00	0.00	0.00	1.00
	Walking (*Wa*)	0.60	1.00	0.60		1.00	0.00	1.00	0.30	0.00	0.00	1.00	0.00
	Climbing stairs (*Cs*)	1.00	0.30	0.00	0.00		0.90	0.00	0.00	1.00	1.00	1.00	0.00
	Waist bends forward (*Wb*)	1.00	0.70	0.40	0.30	0.40		0.60	1.00	1.00	0.70	1.00	1.00
	Frontal elevation of arms (*Fe*)	1.00	0.70	1.00	1.00	0.90	0.00		0.20	0.00	0.90	0.90	1.00
	Knees bending (*Kb*)	0.00	1.00	1.00	0.00	0.70	1.00	1.00		1.00	1.00	1.00	1.00
	Cycling (*Cy*)	0.00	1.00	1.00	0.00	1.00	1.00	0.00	1.00		1.00	1.00	1.00
	Jogging (*Jo*)	0.10	0.00	0.00	0.70	0.00	0.20	0.00	0.00	1.00		1.00	1.00
	Running (*Ru*)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.80		1.00
	Jump front and back (*Ju*)	1.00	1.00	0.30	0.00	0.60	0.00	0.00	0.00	1.00	1.00	1.00	

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 November 2023 doi:10.20944/preprints202311.0328.v1

Table 4. *Cont.*

Sensors attach	**Ground Truth Class**	**Target Attack Class**										
		St	*Si*	*Ly*	*Wa*	*Cs*	*Wb*	*Fe*	*Kb*	*Cy*	*Jo*	*Ru*	*Ju*
Chest	Standing (StSt)		0.80	0.00	1.00	0.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00
	Sitting (*Si*)	1.00		1.00	0.40	0.80	0.00	0.80	0.00	0.50	0.00	0.00	0.00
	Lying down (LyLy)	0.80	0.00		1.00	0.70	1.00	1.00	0.10	0.40	1.00	0.00	0.10
	Walking (*Wa*)	0.00	1.00	0.00		0.00	1.00	0.00	0.00	0.00	0.00	1.00	0.00
	Climbing stairs (*Cs*)	0.50	0.90	1.00	1.00		0.20	0.70	0.00	1.00	1.00	0.00	1.00
	Waist bends forward (*Wb*)	0.00	0.00	1.00	1.00	1.00		1.00	0.00	0.00	0.00	0.00	0.00
	Frontal elevation of arms (*Fe*)	0.00	0.00	0.00	0.00	0.00	1.00		0.00	1.00	0.00	1.00	0.00
	Knees bending (*Kb*)	0.60	0.00	1.00	1.00	1.00	0.00	1.00		1.00	0.00	0.00	0.70
	Cycling (*Cy*)	1.00	0.80	1.00	0.90	0.70	1.00	1.00	0.60		0.00	0.70	0.00
	Jogging (JoJo)	0.00	0.00	0.00	0.70	0.40	1.00	0.00	0.00	0.10		1.00	1.00
	Running (*Ru*)	1.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.00	0.00		0.50
	Jump front and back (*Ju*)	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.60	0.00	1.00	

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 November 2023 doi:10.20944/preprints202311.0328.v1

Table 4 shows that the ankle, wrist, and chest sensors group is susceptible to a wide range of adversarial attacks. Many of the ground-truth classes were highly vulnerable in this table. However, Table 4 also shows that some attacks were difficult to succeed. The rest of this section focuses on the successful attacks.

5.4. Accuracy of Detection

In this subsection, we evaluate our attack detection by using two metrics, false negative rate (FNR) and false positive rate (FPR) defined by

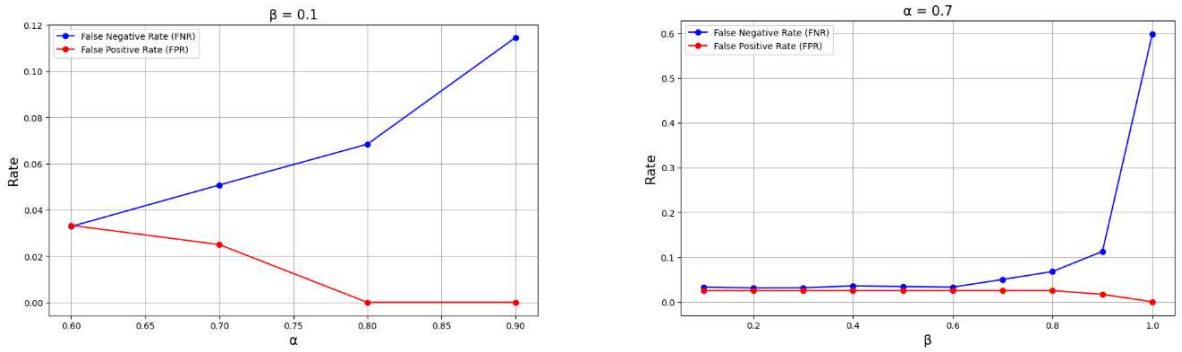
$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}} \tag{5}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{6}$$

where TP is the number of sensor-based AEs that are correctly detected by our method, FN is the number of sensor-based AEs that cannot be detected, FP is the number of clean data that are mistakenly detected as sensor-based AEs, and TN is the number of clean data that are not detected by our method.

Our detection method has two parameters: α and β. These parameters have a large impact on the detection of sensor-based AEs. In this evaluation, we changed these parameters to investigate their impact on both FNR and FPR. Figure 3 shows the results. This figure shows a trade-off between the FPR and FNR. A higher value of these parameters tends to decrease FPR at the expense of an increased FNR.

This figure also shows that we can achieve the FNR less than 0.03 without letting the FPR exceed 0.06.



(a) **(b)**

Figure 3. The results of impact when FNR and FPR were α value is changing and $\beta = 0.1$ (a) and result of impact when the β value is changing and $\alpha = 0.7$ (b)

5.5. Accuracy of Detection of sensors used in sensor-based AEs.

Our method also detects the sensors used in sensor-based AEs. In this subsection, we evaluate the accuracy of the detection by using two metrics, *Precision Of Detected Sensors* and *Recall Of Detected Sensors* as defined by

$$\textit{Precision Of Detected Sensors} = \frac{\textit{Truly Detected Sensors}}{\textit{Truly Detected Sensors} + \textit{Falsely Detected Sensors}} \tag{7}$$

$$\textit{Recall Of Detected Sensors} = \frac{\textit{Truly Detected Sensors}}{\textit{Truly Detected Sensors} + \textit{Misses Detected Sensors}} \tag{8}$$

where *Truly Detected Sensors* is the number of sensors that are correctly detected as the sensors used in the sensor-based AEs, *Falsely Detected Sensors* is the number of sensors detected as the sensors used in the sensor-based AEs but are not used in the AEs, and *MissedDetected* is the number of sensors that are used in the sensor-based AEs but are not detected as the sensors used in the AEs. If our method

 doi:10.20944/preprints202311.0328.v1

detects all sensors used in the AEs, the Recall of Detected Sensors becomes 1.00. It is important to detect sensors used by attackers as suspicious, even if some legitimate sensors are mistakenly detected. Therefore, the *Recall Of Detected Sensors* is more important than *Precision Of Detected Sensors*.

Table 5 shows the *Recall Of Detected Sensors*. In this table, the green cells are the cells whose values are larger than 0.50, the yellow cells are the cells whose values are larger than 0.50 and less than 0.80, and the red cells are the cells whose values are larger than 0.80.

Table 5 shows that our method achieves high*Recall Of Detected Sensors*. *Recall Of Detected Sensors* calculated for all cases is 0.92. That is, most of the sensors used in the AEs can be detected by our method. This is because our method uses the features of the attack, which causes the prediction results to change when the attacker uses the features from sensors. This feature cannot be avoided by the attacker because they are unable to alter the results of the FRM, which extracts the features from the sensors used by the attacker.

However, the *Recall Of Detected Sensors* for some cases are not 1.00; for example, it is 0.79 in the case of the AEs from standing class to sitting class by compromising the wrist sensor. Such misdetection occurs when the target class cannot be distinguished from the actual class. In this case, the FRM, excluding the values from the sensors used in the AEs, outputs a high probability even for the target class. One approach to solving this problem is to add more sensors to make the system more redundant so that any class can be distinguished even if we exclude some sensors, which is one of our future research topics.

Table 6 shows *Precision Of Detected Sensors*. This table indicates that our method achieves high *Precision Of Detected Sensors* in most cases. The *Precision Of Detected Sensors* calculated for all cases is 0.72. But *Precision Of Detected Sensors* becomes low in some cases. For example, the *Precision Of Detected Sensors* for the case that values from the chest sensor device are changed so that the state of lying down is miss-classified into the state of knee bending is 0.22. In such cases, inconsistencies are found by extracting the features from the other sensors. However, even in such cases, we can successfully detect sensors used in the AEs as suspicious sensors. Therefore, our method can be used as a trigger to check suspicious sensors.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 November 2023 doi:10.20944/preprints202311.0328.v1

Table 5. Recall Of Detected Sensors

Sensors attach	Ground Truth Class	Target Attack Class											
		St	*Si*	*Ly*	*Wa*	*Cs*	*Wb*	*Fe*	*Kb*	*Cy*	*Jo*	*Ru*	*Ju*
Ankle	Standing (*St*)		1.00	N/A	1.00	1.00	N/A	1.00	1.00	1.00	1.00	N/A	1.00
	Sitting (*Si*)	1.00		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Lying down (*Ly*)	1.00	1.00		N/A	1.00	1.00	1.00	1.00	1.00	1.00	N/A	1.00
	Walking (*Wa*)	N/A	1.00	N/A		1.00	1.00	1.00	1.00	N/A	N/A	1.00	N/A
	Climbing stairs (*Cs*)	1.00	1.00	1.00	1.00		1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Waist bends forward (*Wb*)	1.00	1.00	1.00	1.00	1.00		1.00	1.00	1.00	N/A	1.00	1.00
	Frontal elevation of arms (*Fe*)	N/A	1.00	1.00	1.00	1.00	1.00		N/A	1.00	1.00	1.00	1.00
	Knees bending (*Kb*)	1.00	1.00	1.00	1.00	1.00	0.91	1.00		0.91	1.00	1.00	1.00
	Cycling (*Cy*)	N/A	1.00	1.00	1.00	1.00	N/A	1.00	1.00		1.00	1.00	1.00
	Jogging (*Jo*)	1.00	1.00	1.00	1.00	0.96	1.00	1.00	1.00	1.00		1.00	1.00
	Running (*Ru*)	1.00	1.00	N/A	N/A	1.00	1.00	1.00	1.00	1.00	1.00		1.00
	Jump front and back (*Ju*)	N/A	1.00	N/A	N/A	1.00	1.00	1.00	1.00	1.00	1.00	1.00	
Wrist	Standing (*St*)		0.79	N/A	N/A	1.00	N/A	1.00	1.00	N/A	1.00	N/A	N/A
	Sitting (*Si*)	1.00		N/A	N/A	1.00	1.00	1.00	1.00	1.00	N/A	N/A	1.00
	Lying down (*Ly*)	N/A	1.00		1.00	1.00	N/A	N/A	1.00	1.00	N/A	N/A	1.00
	Walking (*Wa*)	1.00	1.00	1.00		1.00	N/A	1.00	1.00	N/A	N/A	1.00	N/A
	Climbing stairs (*Cs*)	1.00	1.00	N/A	N/A		1.00	N/A	N/A	1.00	1.00	1.00	N/A
	Waist bends forward (*Wb*)	1.00	1.00	1.00	1.00	1.00		1.00	1.00	1.00	1.00	1.00	1.00
	Frontal elevation of arms (*Fe*)	1.00	0.95	1.00	1.00	1.00	N/A		1.00	N/A	1.00	1.00	1.00
	Knees bending (*Kb*)	N/A	1.00	1.00	N/A	1.00	1.00	1.00		1.00	1.00	1.00	1.00
	Cycling (*Cy*)	N/A	1.00	1.00	N/A	1.00	1.00	N/A	1.00		1.00	1.00	1.00
	Jogging (*Jo*)	1.00	N/A	N/A	1.00	N/A	1.00	N/A	N/A	1.00		0.82	1.00
	Running (*Ru*)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1.00	0.98		1.00
	Jump front and back (*Ju*)	1.00	1.00	1.00	N/A	1.00	N/A	N/A	N/A	1.00	1.00	1.00	

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 November 2023 doi:10.20944/preprints202311.0328.v1

Table 5. *Cont.*

Sensors attach	Ground Truth Class	Target Attack Class										
		St	*Si*	*Ly*	*Wa*	*Cs*	*Wb*	*Fe*	*Kb*	*Cy*	*Jo*	*Ru*	*Ju*
Chest	Standing (*St*)		1.00	N/A	1.00	N/A	1.00	1.00	N/A	N/A	N/A	N/A	N/A
	Sitting (*Si*)	1.00		1.00	1.00	1.00	N/A	1.00	N/A	1.00	N/A	N/A	N/A
	Lying down (*Ly*)	1.00	1.00		1.00	1.00	1.00	1.00	1.00	1.00	1.00	N/A	1.00
	Walking (*Wa*)	N/A	1.00	N/A		N/A	1.00	N/A	N/A	N/A	N/A	1.00	N/A
	Climbing stairs (*Cs*)	1.00	0.86	1.00	1.00		1.00	1.00	N/A	1.00	1.00	N/A	1.00
	Waist bends forward (*Wb*)	N/A	N/A	1.00	1.00	1.00		1.00	N/A	N/A	N/A	N/A	N/A
	Frontal elevation of arms (*Fe*)	N/A	N/A	N/A	N/A	N/A	1.00		N/A	1.00	N/A	1.00	N/A
	Knees bending (*Kb*)	1.00	N/A	1.00	1.00	1.00	N/A	1.00		1.00	N/A	N/A	1.00
	Cycling (*Cy*)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00		N/A	1.00	N/A
	Jogging (*Jo*)	N/A	N/A	N/A	1.00	1.00	1.00	N/A	N/A	1.00		1.00	1.00
	Running (*Ru*)	1.00	N/A	N/A	N/A	N/A	1.00	1.00	N/A	N/A	N/A		1.00
	Jump front and back (*Ju*)	N/A	N/A	N/A	N/A	N/A	1.00	1.00	N/A	1.00	N/A	1.00	

Table 6. Precision Of Detected Sensors

Sensors attach	Ground Truth Class	Target Attack Class										
		St	*Si*	*Ly*	*Wa*	*Cs*	*Wb*	*Fe*	*Kb*	*Cy*	*Jo*	*Ru*	*Ju*
Ankle	Standing (*St*)		1.00	N/A	0.41	0.39	N/A	0.64	1.00	1.00	0.67	N/A	1.00
	Sitting (*Si*)	0.64		1.00	0.45	0.60	0.90	1.00	1.00	0.66	0.49	1.00	1.00
	Lying down (*Ly*)	1.00	1.00		N/A	0.82	1.00	1.00	1.00	1.00	0.39	N/A	1.00
	Walking (*Wa*)	N/A	1.00	N/A		1.00	1.00	1.00	1.00	N/A	N/A	1.00	N/A
	Climbing stairs (*Cs*)	1.00	1.00	1.00	0.21		1.00	1.00	1.00	1.00	0.54	1.00	1.00
	Waist bends forward (*Wb*)	1.00	1.00	1.00	0.50	0.50		1.00	1.00	1.00	N/A	1.00	1.00
	Frontal elevation of arms (*Fe*)	N/A	1.00	1.00	1.00	0.53	1.00		N/A	1.00	0.61	1.00	1.00
	Knees bending (*Kb*)	0.91	1.00	1.00	0.91	0.83	1.00	1.00		1.00	1.00	1.00	1.00
	Cycling (*Cy*)	N/A	1.00	1.00	0.57	0.83	N/A	1.00	1.00		0.42	1.00	1.00
	Jogging (*Jo*)	1.00	1.00	1.00	1.00	0.66	1.00	1.00	1.00	1.00		1.00	1.00
	Running ()	1.00	1.00	N/A	N/A	0.78	1.00	1.00	1.00	1.00	0.57		1.00
	Jump front and back (*Ju*)	N/A	1.00	N/A	N/A	0.66	1.00	1.00	1.00	1.00	1.00	1.00	

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 November 2023 doi:10.20944/preprints202311.0328.v1

Table 6. *Cont.*

Sensors attach	Ground Truth Class	Target Attack Class											
		St	*Si*	*Ly*	*Wa*	*Cs*	*Wb*	*Fe*	*Kb*	*Cy*	*Jo*	*Ru*	*Ju*
Wrist	Standing (*St*)		1.00	N/A	N/A	0.60	N/A	1.00	0.68	N/A	0.64	N/A	N/A
	Sitting (*Si*)	1.00		N/A	N/A	0.75	0.57	0.50	0.90	0.84	N/A	N/A	1.00
	Lying down (*Ly*)	N/A	1.00		0.64	0.90	N/A	N/A	1.00	1.00	N/A	N/A	0.60
	Walking (*Wa*)	1.00	1.00	1.00		1.00	N/A	1.00	0.39	N/A	N/A	1.00	N/A
	Climbing stairs (*Cs*)	1.00	1.00	N/A	N/A		1.00	N/A	N/A	1.00	0.87	1.00	N/A
	Waist bends forward (*Wb*)	1.00	1.00	1.00	0.59	0.57		0.67	0.72	1.00	0.48	1.00	1.00
	Frontal elevation of arms (*Fe*)	1.00	1.00	0.80	0.68	0.55	N/A		0.44	N/A	0.70	1.00	1.00
	Knees bending (*Kb*)	N/A	1.00	1.00	N/A	1.00	1.00	1.00		1.00	1.00	1.00	1.00
	Cycling (*Cy*)	N/A	1.00	1.00	N/A	1.00	1.00	N/A	1.00		0.57	1.00	1.00
	Jogging (JoJo)	1.00	N/A	N/A	0.58	N/A	1.00	N/A	N/A	1.00		1.00	1.00
	Running (*Ru*)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1.00	0.54		1.00
	Jump front and back (*Ju*)	1.00	1.00	1.00	N/A	0.56	N/A	N/A	N/A	1.00	1.00	1.00	
Chets	Standing (StSt)		0.69	N/A	0.36	N/A	1.00	0.56	N/A	N/A	N/A	N/A	N/A
	Sitting (*Si*)	0.36		1.00	0.36	0.69	N/A	0.33	N/A	0.53	N/A	N/A	N/A
	Lying down (*Ly*)	0.33	0.22		0.48	0.44	1.00	1.00	0.22	1.00	0.64	N/A	1.00
	Walking (*Wa*)	N/A	1.00	N/A		N/A	1.00	N/A	N/A	N/A	N/A	1.00	N/A
	Climbing stairs (*Cs*)	0.55	1.00	1.00	0.60		1.00	0.63	N/A	1.00	0.82	N/A	1.00
	Waist bends forward (*Wb*)	N/A	N/A	0.83	0.50	1.00		0.48	N/A	N/A	N/A	N/A	N/A
	Frontal elevation of arms (*Fe*)	N/A	N/A	N/A	N/A	N/A	1.00		N/A	0.56	N/A	1.00	N/A
	Knees bending (*Kb*)	0.55	N/A	1.00	1.00	0.85	N/A	1.00		1.00	N/A	N/A	1.00
	Cycling (*Cy*)	1.00	1.00	1.00	0.71	0.47	1.00	1.00	0.48		N/A	1.00	N/A
	Jogging (*Jo*)	N/A	N/A	N/A	0.44	0.24	1.00	N/A	N/A	0.36		1.00	0.85
	Running (*Ru*)	1.00	N/A	N/A	N/A	N/A	1.00	1.00	N/A	N/A	N/A		0.58
	Jump front and back (*Ju*)	N/A	N/A	N/A	N/A	N/A	1.00	1.00	N/A	0.60	N/A	1.00	

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 November 2023 doi:10.20944/preprints202311.0328.v1

5.6. Mitigation of sensor-based AEs by excluding the detected sensors

The FRM can be used after detection of sensor-based AEs and compromised sensors, because the FRM can output the classification results even if some features are excluded. Therefore, we demonstrate the performance of the FRM by excluding the values from the detected sensors.

Table 7 presents the *Precision* (P) and *Recall* (R) of the FRM excluding the values from the sensors used in the AEs. This table also includes the results of the original model in the cases without attacks.

Table 7 indicates that the FRM excluding sensors used in the AEs achieves high *Precision* and *Recall* in most cases. In some cases, the *Precision* becomes low. For example, *Precision* for the class of climbing stairs is 0.71 when excluding the values from the sensors of the ankle device. This is because the values from the ankle device are essential to distinguish the classes. Even in this case, the FRM can output the actual classes by outputting large probabilities for multiple possible classes.

Table 7. Performance comparison of the original target model without AEs and FRM exluding the values from the sensors used in the AEs

Ground Truth Class	Original Target Model		Ankle		Wrist		Chest	
	P	R	P	R	P	R	P	R
Standing	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99
Sitting	1.00	1.00	0.99	0.99	0.76	0.99	1.00	1.00
Lying down	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99
Walking	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.99
Climbing stairs	0.99	1.00	0.71	0.99	1.00	1.00	0.99	0.99
Waist bends forward	1.00	1.00	1.00	1.00	0.99	0.99	1.00	1.00
Frontal elevation of arms	0.98	1.00	0.99	0.99	0.99	0.99	0.99	0.99
Knees bending	1.00	0.96	1.00	0.99	0.99	0.99	0.75	0.99
Cycling	0.96	1.00	1.00	0.99	0.99	0.99	0.99	0.99
Jogging	0.98	0.93	1.00	0.99	0.99	0.99	0.99	0.99
Running	0.94	0.97	0.99	0.99	1.00	1.00	0.99	0.99
Jump front and back	0.95	0.97	0.99	0.99	0.99	0.99	0.99	0.99
Average	**0.98**	**0.98**	**0.97**	**0.99**	**0.97**	**0.99**	**0.99**	**0.99**

*Red color represents 1.00, while yellow indicates a score less than 1.00

6. Conclusions

In this study, we propose a method to detect the sensors used in the sensor-based AEs. Our method is based on utilizing the features of the sensor-based AEs that the attacker cannot avoid. The output of the ML model changes when the values obtained from the sensors used in the AE are incorporated. To evaluate the impact of sensor values, we have introduced the feature-removable model (FRM), which allows us to select the features to be used. The FRM outputs the possible classes classified using the selected features. By comparing the results of the FRM when selecting different features, we can find any inconsistencies and detect the sensors causing such inconsistencies. After detecting the sensors used in the AEs, we can proceed to check and replace them. Nevertheless, our FRM can also be used after detection. We can obtain the output of the FRM without using the features from the detected sensors in order to avoid the impact of the attacks. However, it is important to note that the accuracy of the model might decrease compared to the case where we can use all features.

Through experimental evaluations, a model for human activity recognition was tested using three devices attached to the user's chest, wrist, and ankle. We have demonstrated that our method detect the sensors used in the AEs, even when one-third of the sensors are used by the attacker.

For future research, it is crucial to identify the key features that have a significant impact on the model's output. This exploration can offer improved protection against sensor-based AEs. By identifying the crucial features and understanding their impact, we can strengthen the model's resilience. This proactive approach ensures the system's robustness in the face of potential adversarial challenges.

Author Contributions: Conceptualization, AK and YO; methodology, AK and YO; software, AK and YO; validation, YO and MM; formal analysis, AK and SM; investigation, AK; resources, MM; data curation, AK and YO; writ-ing—original draft preparation, AK; writing—review and editing, YO; visualization, AK and SM; supervi-sion, MM and YO; project administration, YO; funding acquisition, YO and MM. All authors have read and agreed to the published version of the manuscript

Funding: This work was supported by the Cabinet Office (CAO), Cross-ministerial Strategic Innovation Promotion Program (SIP), and "Cyber Physical Security for IoT Society" (funding agency: NEDO).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable

Data Availability Statement: Not applicable

Conflicts of Interest: The authors declare no conflict of interest

Abbreviations

The following abbreviations are used in this manuscript:

α	Alpha (Parameter)
β	Beta (Parameter)
AE	Adversarial Examples
AT	Adversarial Training
Cs	Climbing stairs
Cy	Cycling
EAT	Ensemble Adversarial Training
ECG	Electrocardiogram
Fe	Frontal elevation of arms
FN	False Negatives in subsection Accuracy of Detection
fp	False positives in subsection Property of the feature removable model without attack
FP	False Positives in subsection Accuracy of Detection
FPR	False Positive Rate
FNR	False Negative Rate
FRM	Feature-Removable Model
HAR	Human Activity Recognition
H & G	Hendricks and Gimpel
Jo	Jogging
Ju	Jump front and back
Kb	Knees bending
Ly	Lying down
ML	Machine Learning
Ru	Running
Si	Sitting
St	Standing
TN	True Negatives in subsection Accuracy of Detection
TP	True Positives in subsection Accuracy of Detection
tp	True Positive in subsection Property of the feature removable model without attack
Wa	Walking
Wb	Waist bends forward
fn	False Negative in subsection Property of the feature removable model without attack

References

1. Miao, S.; Dang, Y.; Zhu, Q.; Li, S.; Shorfuzzaman, M.; Lv, H. A Novel Approach for Upper Limb Functionality Assessment Based on Deep Learning and Multimodal Sensing Data. *IEEE Access* **2021**, *9*, 77138–77148. https://doi.org/10.1109/ACCESS.2021.3080592.

2. Karle, P.; Fent, F.; Huch, S.; Sauerbeck, F.; Lienkamp, M. Multi-Modal Sensor Fusion and Object Tracking for Autonomous Racing. *IEEE Transactions on Intelligent Vehicles* **2023**, pp. 1–13. https://doi.org/10.1109/TIV.2023.3271624.
3. Zhang, X.; Zheng, P.; Peng, T.; Li, D.; Zhang, X.; Tang, R. Privacy-preserving activity recognition using multimodal sensors in smart office. *Future Generation Computer Systems* **2023**, *148*, 27–38. https://doi.org/10.1016/j.future.2023.05.023.
4. Namazi, E.; Mester, R.; Li, J.; Lu, C.; Tang, M.; Xiong, Y. Traffic Awareness Through Multiple Mobile Sensor Fusion. *IEEE Sensors Journal* **2022**, *22*, 11903–11914. https://doi.org/10.1109/JSEN.2022.3171070.
5. Zhou, H.; Zhao, Y.; Liu, Y.; Lu, S.; An, X.; Liu, Q. Multi-Sensor Data Fusion and CNN-LSTM Model for Human Activity Recognition System. *Sensors* **2023**, *23*, 4750. https://doi.org/10.3390/s23104750.
6. Yuan, L.; Andrews, J.; Mu, H.; Vakil, A.; Ewing, R.; Blasch, E.; Li, J. Interpretable Passive Multi-Modal Sensor Fusion for Human Identification and Activity Recognition. *Sensors* **2022**, *22*. https://doi.org/10.3390/s22155787.
7. Finlayson, S.G.; Bowers, J.D.; Ito, J.; Zittrain, J.L.; Beam, A.L.; Kohane, I.S. Adversarial attacks on medical machine learning. *Science* **2019**, *363*, 1287–1289. https://doi.org/10.1126/science.aaw4399.
8. Classen, J.; Wegemer, D.; Patras, P.; Spink, T.; Hollick, M. Anatomy of a Vulnerable Fitness Tracking System. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* **2018**, *2*, 1–24. https://doi.org/10.1145/3191737.
9. Monjur, M.M.R.; Heacock, J.; Calzadillas, J.; Mahmud, M.S.; Roth, J.; Mankodiya, K.; Sazonov, E.; Yu, Q. Hardware Security in Sensor and its Networks. *Frontiers in Sensors* **2022**, *3*. https://doi.org/10.3389/fsens.2022.850056.
10. Kurniawan, A.; Ohsita, Y.; Murata, M. Experiments on Adversarial Examples for Deep Learning Model Using Multimodal Sensors. *Sensors* **2022**, *22*, 8642. https://doi.org/10.3390/s22228642.
11. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* **2015**, pp. 1–11, [1412.6572].
12. Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; McDaniel, P. Ensemble Adversarial Training: Attacks and Defenses. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings* **2017**, pp. 1–20, [1705.07204].
13. Gong, Z.; Wang, W. Adversarial and Clean Data Are Not Twins. In Proceedings of the Proceedings of the Sixth International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, New York, NY, USA, jun 2023; pp. 1–5, [1704.04960]. https://doi.org/10.1145/3593078.3593935.
14. Grosse, K.; Papernot, N.; Manoharan, P.; Backes, M.; McDaniel, P. Adversarial Examples for Malware Detection. In Proceedings of the Computer Security – ESORICS 2017; Foley, S.N.; Gollmann, D.; Snekkenes, E., Eds., Cham, 2017; pp. 62–79.
15. Grosse, K.; Manoharan, P.; Papernot, N.; Backes, M.; McDaniel, P. On the (Statistical) Detection of Adversarial Examples. *ArXiv* **2017**, [1702.06280]. https://doi.org/https://doi.org/10.48550/arXiv.1702.06280.
16. Hendrycks, D.; Gimpel, K. Early Methods for Detecting Adversarial Images. *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings* **2016**, pp. 1–9, [1608.00530].
17. Hendrycks, D.; Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* **2019**, pp. 1–12, [1610.02136].
18. Hosseini, H.; Chen, Y.; Kannan, S.; Zhang, B.; Poovendran, R. Blocking Transferability of Adversarial Examples in Black-Box Learning Systems. *ArXiv* **2017**, [1703.04318].
19. Metzen, J.H.; Genewein, T.; Fischer, V.; Bischoff, B. On Detecting Adversarial Perturbations. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* **2017**, pp. 1–12, [1702.04267].
20. Miller, D.; Wang, Y.; Kesidis, G. When not to classify: Anomaly detection of attacks (ADA) on DNN classifiers at test time, 2019, [1712.06646]. https://doi.org/10.1162/neco_a_01209.
21. Carlini, N.; Wagner, D.A. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security* **2017**.
22. Tramèr, F.; Boneh, D. Adversarial training and robustness for multiple perturbations. In Proceedings of the Advances in Neural Information Processing Systems, 2019, Vol. 32, pp. 1–11, [1904.13000].
23. Kang, D.; Sun, Y.; Brown, T.; Hendrycks, D.; Steinhardt, J. Transfer of Adversarial Robustness Between Perturbation Types. *ArXiv* **2019**, [1905.01034].

24. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfittin. *Journal of Machine Learning Research 15* **2014**, *15*, 1929–1958.
25. Mutegeki, R.; Han, D.S. A CNN-LSTM Approach to Human Activity Recognition. In Proceedings of the 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC). IEEE, feb 2020, pp. 362–366. https://doi.org/10.1109/ICAIIC48513.2020.9065078.
26. Banos, O.; Garcia, R.; Holgado-Terriza, J.A.; Damas, M.; Pomares, H.; Rojas, I.; Saez, A.; Villalonga, C. mHealthDroid: A Novel Framework for Agile Development of Mobile Health Applications. In Proceedings of the Ambient Assisted Living and Daily Activities; Pecchia, L.; Chen, L.L.; Nugent, C.; Bravo, J., Eds., Cham, 2014; pp. 91–98.