Adar: Adversarial Activity Recognition in Wearables

Ramesh Kumar Sah
Washington State University
Pullman, Washington 99164–2752
Email: ramesh.sah@wsu.edu

Hassan Ghasemzadeh
Washington State University
Pullman, Washington 99164–2752
Email: hassan.ghasemzadeh@wsu.edu

***Abstract*—Recent advances in machine learning and deep neural networks have led to the realization of many important applications in the area of personalized medicine. Whether it is detecting activities of daily living or analyzing images for cancerous cells, machine learning algorithms have become the dominant choice for such emerging applications. In particular, the state-of-the-art algorithms used for human activity recognition (HAR) using wearable inertial sensors utilize machine learning algorithms to detect health events and to make predictions from sensor data. Currently, however, there remains a gap in research on whether or not and how activity recognition algorithms may become the subject of adversarial attacks. In this paper, we take the first strides on (1) investigating methods of generating adversarial example in the context of HAR systems; (2) studying the vulnerability of activity recognition models to adversarial examples in feature and signal domain; and (3) investigating the effects of adversarial training on HAR systems. We introduce Adar[1], a novel computational framework for optimization-driven creation of adversarial examples in sensor-based activity recognition systems. Through extensive analysis based on real sensor data collected with human subjects, we found that simple evasion attacks are able to decrease the accuracy of a deep neural network from 95.1% to 3.4% and from 93.1% to 16.8% in the case of a convolutional neural network. With adversarial training, the robustness of the deep neural network increased on the adversarial examples by 49.1% in the worst case while the accuracy on clean samples decreased by 13.2%.**

I. Introduction

Human activity recognition (HAR) is a major research area in the field of mobile and ubiquitous computing. Its applications in health monitoring, patients rehabilitation, assessment of performance in athletes, and gaming industry [1] are the contributing factors for the active research in this area. Moreover, advances in sensor technology and detection algorithms allow for real-time continuous detection of activities with battery-powered devices of small form factor to be used in daily life. In particular, the state-of-the-art algorithms used for HAR using wearable inertial sensors utilize machine learning algorithms to detect various biomarkers and to provide real-time and continuous clinical interventions based on the sensor data.

Recent studies have found that machine learning systems are often vulnerable to adversarial perturbations. Even the addition of a small amount of carefully computed perturbations to the clean samples degrade the performance of machine learning systems significantly [2]–[5]. What distinguishes adversarial perturbations from random noise is that adversarial examples are misclassified far more often than samples that have been perturbed by random noise, even if the magnitude of noise is much larger compared to the adversarial perturbation [3]. Adversarial inputs were first formally described in [6], in which the researchers studied the techniques used by spammers to circumvent spam filters. In recent years, research on adversarial examples has witnessed tremendous growth and in particular have been investigated in great details in computer vision and audio processing researches [4], [7], [8].

Human activity recognition shares many challenges common to other areas such as computer vision and natural language processing but it also has its own unique challenges and requires dedicated set of computational methods [9]. For example, the data used for the classification of activities often comes from a collection of heterogeneous sensors with different characteristics unlike fixed and well-defined sensing modules such as cameras and microphones. Furthermore, unlike other areas where the problem is well defined (e.g., "Is this an image of a dog?" or "Is the word Harry present in this sentence?"), HAR algorithms need to deal with new challenges due to (1) complexity of human movements captured from individual body joints and observed partially with wearable sensors; and (2) lack of concrete definitions, languages or structure of human activities [9]. For starter, human activity is highly complex and diverse and the sensor readings for an activity can be very different even if the activity is performed by the same person under similar conditions compared to, for example, image classification where an image of a dog is always a dog independent of the presentation and context.

All this leads to the conclusion that HAR though sharing many common points with the other research areas also have unique sets of problems and challenges with socially unique outcomes. In fact, the issue of adversarial examples in HAR is mostly an unexplored topic despite the great magnitude of the consequences of adversarial machine learning in behavioral medicine. For example, imagine that someone falls and the detection algorithm fails to detect this event or the system used to estimate clinical measures such as activity level, eating, medication, etc. is underestimating or overestimating due to an adversary. These events can have fatal outcomes and therefore

[1]Software code and experimental data for Adar are available online at *https://github.com/rameshKrSah/Adar*.

requires that we define and assess the effects of adversarial examples in mobile health and behavioral medicine. In this paper, we study the consequences of adversarial examples for activity recognition models and propose an optimization-driven framework called Adar for generating adversarial examples in activity recognition systems. The contributions of this paper are as follows.

- We introduce Adar framework for adversarial examples generation for human activity recognition systems.
- We successfully investigate optimization methods of generating adversarial examples in the context of HAR system.
- We study the vulnerability of activity recognition models to adversarial examples with a comprehensive set of experiments.
- We compare the robustness of a HAR system in feature and signal domain. We also present the result after adversarial training of an activity recognition model.

Note that most of the study focusing on adversarial examples stems from the field of computer vision and speech recognition and it remains to be shown the effect of adversarial examples in other fields such as ubiquitous computing and mobile health. In general, machine learning models that are both highly accurate and robust to adversarial examples remains an open research problem to the research community.

The rest of the paper is structured as follows: In section 2 we explain the Adar framework and we also briefly review the human activity recognition pipeline and explain different adversarial attack methods and defense strategies. Following it, in section 3 we describe our experiments and results.

II. Adar Framework Design

Before discussing the details of the Adar framework, we will briefly discuss human activity recognition pipeline and different types of adversarial attacks.

A. Human Activity Recognition Pipeline

The problem of human activity recognition can be defined as: Given a set $W = \{W_0, ..., W_{m-1}\}$ of m equally sized temporal window of sensor readings, such that each window W_i contains a set of sensor reading $S = \{S_{i,0}, ..., S_{i,k-1}\}$, and a set $A = \{a_0, ...a_n - 1\}$ of n activity labels, the goal is to find a mapping function $f : S_i \rightarrow A$ that can be evaluated for all possible values of S_i [10]. As shown in Figure 1 the activity recognition system generally consists of sensing, signal processing, signal segmentation, feature extraction and selection, and classification stages [11]. Raw data from various sensors, such as accelerometer, gyroscope, magnetometer, etc. are collected and passed into the signal processing stage, where filtering, noise removal, etc. are applied to the sensor signals. This is followed by a segmentation stage, where a continuous stream of a signal is divided into temporal windows. For segmentation the sliding window method is used most widely due to its simplicity and real-time performance. After segmentation, statistical and structural features are extracted from each window segment. In feature extraction, the size of the sliding window has significant effects on the accuracy and speed of the system. Larger window size can support the detection of complex activities but result in slower processing whereas smaller window supports faster processing at the expense of accuracy [12]. Usually, a window size of 1-2 second with 50% overlap between the successive windows is considered a good choice [13]. Following the feature extraction, feature selection may be used to select out the best features from the collection of features. Finally, a machine learning classifier is trained on the feature data and the trained classifier is used to make the inference on the future unseen data. Recently, with the success of convolutional neural networks (CNN) several works have been published [14], [15] where a CNN is used for activity detection. What separates the CNN approach from other machine learning approaches is that CNN's completely remove the need to compute features from raw sensor signal for classification. The input to the CNN model is usually the raw sensor signal with or without segmentation, and the CNN model learns the features and the classifier simultaneously during the training process.

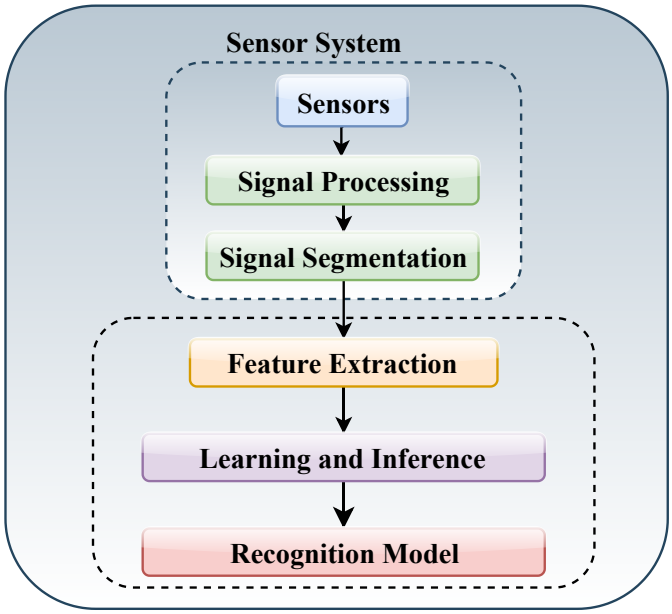

Fig. 1: The general framework of human activity recognition system.

B. Adversarial Attacks

Adversarial examples are inputs formed by applying small but intentional perturbation to the inputs from the dataset such that the perturbed inputs are almost indistinguishable from the true inputs and results in the model outputting an incorrect answer with high confidence [4]. The objective of adversarial learning is to find a perturbation δ which when added to the true inputs X i.e., $X^{adv} = X + \delta$ changes the output of the model. The noise level of the perturbation is constrained by the l_∞ norm denoted by ϵ such that the added perturbation is sufficiently small. In general, an adversary can attack a machine learning system in three ways:

1) Confidence reduction or poisoning attack: In confidence reduction or poisoning the adversary attempts to degrade

the system performance by inducing false connection during training process by corrupting the training data.

2) Evasion attack: There are two types of evasion attack: (1) Misclassification attack, and (2) Source/target misclassification attack. In misclassification attack, the adversary computes adversarial examples which are intended to be classified into any class other than the true class. This is also called untargeted attack. In source/target misclassification attack the adversary defines the target class in which it intends to have the model misclassify an input belonging to any class. This is also called targeted attack.
3) Exploratory attack: In exploratory attack, an adversary tries to gain as much knowledge as possible about the learning algorithm of the target system and the pattern in its training data.

Furthermore, the strength of an adversary is characterized by its ability to operate either in a white-box setting or in a black-box setting [16]. An adversary with access to the target system's architecture, parameters, and data is said to operate in a white-box setting and an adversary with no knowledge of the target system is said to operate in a black-box setting. A large number of evasion attack methods has been proposed that can be used to compute adversarial examples. In our experiments, we have used the simplest attack methods because of their low computational cost. Now, we will briefly explain these attack methods by using the following notations.

- X, is a true input taken from the dataset. In our case X can be either a feature vector or a window of raw sensor values.
- X^{adv}, is an adversarial sample. It's type will be the same as the type of X.
- y_{true}, is the true class label for the input X.
- $J_\theta(X, y)$, is the cross-entropy cost function of the neural network with parameters θ, given input X and label y. Note that after training the network, the parameters of the network remains fixed.
- $Clip_{x,\epsilon}(A)$, denotes the element-wise clipping of A, with $A_{i,j}$ clipped in the range of $[X_{i,j} - \epsilon,\ X_{i,j} + \epsilon]$.

1) Fast gradient sign method (FGSM): Fast gradient sign method is one of the simplest and computationally efficient technique of generating adversarial examples. Proposed by Goodfellow et al. [4], this method is motivated by the linearizing of the cost function and solving for perturbations bounded by ϵ that maximizes the cost subject to the L_∞ norm.

$$X^{adv} = X + \epsilon sign(\nabla_X J_\theta(X, y_{true})) \tag{1}$$

2) One step target class method: One step target class method tries to maximize the probability $p(y_{target}|X)$ of some specific target class y_{target}, which is unlikely to be the true class for the input X. This is different from the FGSM, in which we try to find adversarial perturbations that increase loss for all classes [7].

$$X^{adv} = X - \epsilon sign(\nabla_X J_\theta(X, y_{target})) \tag{2}$$

3) Basic iterative method: This method is a simple extension of the fast gradient sign method and applies FGSM multiple times with small step size.

$$\begin{aligned} &X_0^{adv} = X, \\ &X_{N+1}^{adv} = Clip_{X,\epsilon}\{X_N^{adv} + \alpha sign(\nabla_X J_\theta(X, y_{true}))\} \end{aligned} \tag{3}$$

α is the step size and is set to ϵ/number of iterations in our experiments. The number of iterations is set to 10 for all iterative methods.

4) Iterative least-likely class method: Similar to the basic iterative method which applies FGSM multiple times the iterative least-likely class method applies the one step target class method multiple times with target class set to the least likely class for the given input.

$$\begin{aligned} &X_0^{adv} = X, \\ &X_{N+1}^{adv} = Clip_{X,\epsilon}\{X_N^{adv} - \alpha sign(\nabla_X J_\theta(X, y_{target}))\} \end{aligned} \tag{4}$$

C. Generating Adversarial Examples

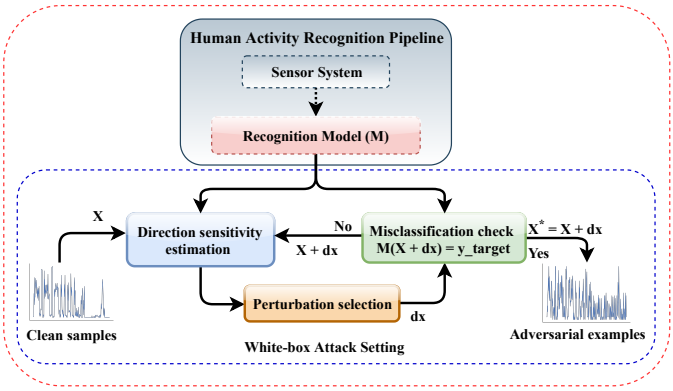

Fig. 2: Adar framework for adversarial examples generation in human activity recognition systems.

Figure 2 shows the proposed Adar framework for computing adversarial examples in human activity recognition systems. This is a general framework and is based on the white-box attack setting, in which an adversary has complete knowledge of the target system. The framework has two components: (1) the human activity recognition pipeline, and (2) the gradient-based attack in a white-box setting. After the HAR pipeline, the trained model is used to compute adversarial examples using gradient-based attacks methods such as fast gradient sign method, basic iterative method or one step target class method. Direction sensitivity estimation finds dimensions of inputs X that will produce the desired adversarial behavior with the smallest perturbation. Perturbation selection uses this knowledge to select perturbations δ affecting sample X's classification [17]. If the resulting sample $X + \delta$ is misclassified by the model M into the target class y_{target}, an adversarial example X^* has been found. If not, the steps can be repeated with updated input $X + \delta$. It has been shown that an adversarial example that was designed to be misclassified by model M_1 is often also misclassified by model M_2 [3]. This is called the transferability property of adversarial examples

and it means that we can generate adversarial examples and perform misclassification attack on a system without having any knowledge of the system [5]. Therefore an adversary operating in a black-box setting can attack a machine learning system by building a substitute model and then computing the adversarial examples using the substitute model. This approach has been successfully demonstrated in [18]. Due to this reason and for the low computational cost of generating adversarial examples the Adar framework is based on the white-box setting.

D. Defense Strategies

There has been a lots of work to build practical defenses against adversarial examples, but a defense technique that is robust against the entire spectrum of adversarial attack is still an open problem. Most of the current defense methods are not adaptive to all types of attacks and one method may block one kind of attack but leaves another vulnerability open for other kinds of attacks [19]. The existing defense mechanism can be categorized into the following types based on their method of implementation.

1) Adversarial training: The general idea of adversarial training is to augment the training data with adversarial examples during the training process. Adversarial training increases the robustness of the model against white-box attacks when the attack method used to augment the training set is the same as the method used by the attacker for the attack [20]. In a black-box setting and for an attack method different from that used for adversarial training this approach performs poorly.
2) Gradient hiding: Gradient hiding is a natural defense against gradient-based attacks, and is based on hiding information about the model's gradient from an adversary. However, this defense is easily fooled by learning a substitute model and using the gradients of the substitute model to craft adversarial examples [19].
3) Defensive distillation: Distillation is a way to transfer knowledge from a large neural network to a smaller one. In [17] the authors used distillation to formulate a defense against adversarial attacks. In defensive distillation, the output probability of a neural network M is used as the label for training a newer network M^* of same architecture on the same training data.

III. Experiments and Results

A. Dataset

In our experiments, we have used the UCI dataset [21] compiled from a group of 30 participants each wearing a smartphone on their waist and performing 6 different activities: standing, sitting, laying, walking, walking upstairs, and walking downstairs. The sensor data consists of a 3-axial accelerometer and a 3-axial gyroscope sampled at a frequency of 50 Hz. The sensor readings were filtered to remove noise and then segmented into windows of size 2.56 seconds (i.e. 128 readings per window) with 50% overlap. The acceleration values were separated into gravity and body component using a Butterworth low pass filter with a cutoff frequency of 0.3 Hz. From each window, 561 features was computed from time and frequency domain. There are 10299 samples in the dataset with 7352 training samples and 2947 test samples. The availability of raw sensor reading in the dataset allows us to explore adversarial examples in both feature and signal domains.

B. Adversarial examples in feature domain

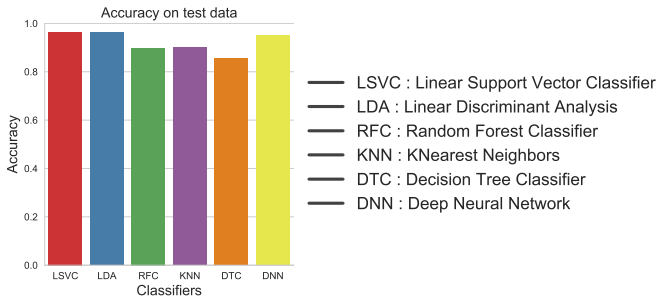

Fig. 3: The accuracy of different classifiers computed on 2947 test samples in the dataset.

In the first experiment, we trained multiple classifiers on the full (561) features data. Figure 3 shows the accuracy of these classifiers on the test data. A deep neural network (DNN) of architecture (64, 32, 6) with l2-regularization (0.001) on the first and second layer gave the best results in terms of accuracy and complexity. We settled on this configuration after testing multiple DNNs with different configurations. *TensorFlow* [22] was used to build the DNN with training parameters: 50 epoch, mini-batch size of 32, Adam optimizer, and sparse categorical cross-entropy loss. All other classifiers were trained with their default settings using the *sklearn* [23] library. We used the *CleverHans* [24] library to compute adversarial examples in all cases.

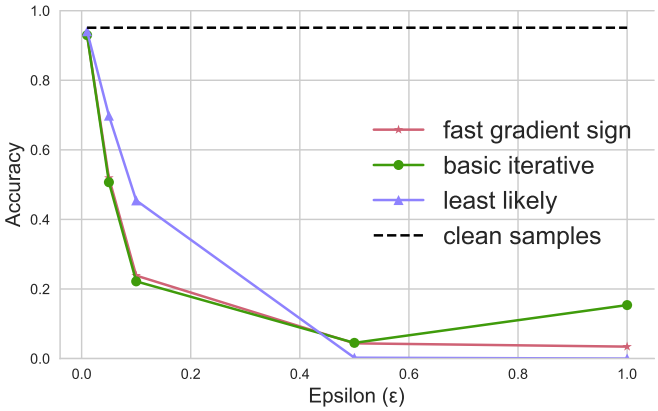

Fig. 4: The accuracy of the DNN model for the untargeted attack. The value of ϵ varies from 0.01 to 1.0 and for iterative methods the number of iteration was set to 10.

Figure 4 shows the accuracy of the DNN model at different values of $\epsilon \in [0.01, 0.05, 0.1, 0.5, 1]$, for the untargeted attack. With an increase in the value of ϵ the success rate of the adversarial examples also increased (as shown by the decrease in accuracy) for all methods. Initially, the least likely method

was the least effective but with the increase in ϵ it was able to exceed other methods. The fast gradient sign method and the basic iterative method performed almost the same at all values of ϵ except at $\epsilon = 1$. At $\epsilon = 1$ the success rate of the basic iterative method decreased compared to the fast gradient sign method. We experimented with different values for the number of iterations and found that with a lower number of iterations, the basic iterative method was able to achieve a similar success rate as that of fast gradient sign method. This is due to the nature of data, which is normalized between [-1.0, 1.0], such that at higher values of ϵ, the gradient of the cost function starts overshooting and the iterative method fails to find the optimum perturbations.

For the transferability of attack, we tested the adversarial examples computed for the untargeted attack on the DNN model on all other classifiers. Figure 6, shows the accuracy of these classifiers. We can see the transferability of attack exists and the accuracy of all classifiers decreased with increase in the value of ϵ. The k-nearest neighbors (kNN) classifier was found to be the most robust compared to other classifiers in all cases. This is consistent with the observation in [25], where the authors showed that a kNN classifier is usually more robust to adversarial examples and its robustness depends on the properties of the data distribution and on the value of k. However, it has been shown that with complex attack methods adversarial examples can be computed that can fool a kNN model with high success rate [26].

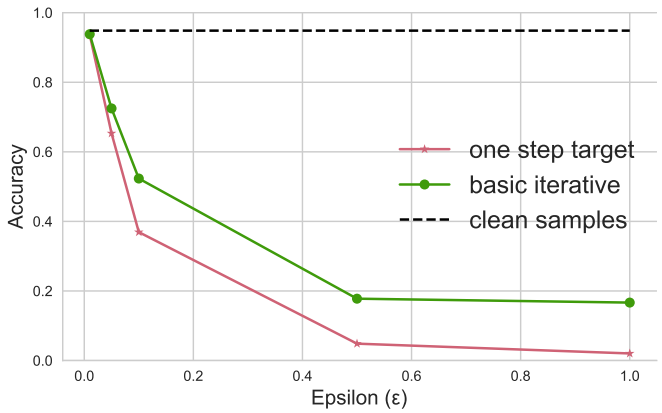

Fig. 5: The accuracy of the DNN model for targeted attack. The value of ϵ varies from 0.01 to 1.0 and for iterative methods the number of iteration was set to 10.

Figure 5 shows the accuracy of the DNN model at different values of ϵ for the targeted attack. One step target class method and the basic iterative method was used to compute the adversarial examples with **Sit** activity class as the target class. Figure 7 shows the confusion matrix of the DNN model on the clean test samples and adversarial examples. In the figure, WalkU and WalkD represent walking upstairs and walking downstairs respectively. It is clear that the one step target class method is more competent at untargeted misclassification attack than the basic iterative method, and the basic iterative method is more successful at targeted misclassification attack. This is due to the fact that one step target method adds ϵ-scaled noise to each sample whereas the iterative methods exploit much finer perturbations and therefore is able to find perturbations that cause the classifier to classify an input into a particular class.

C. Adversarial examples in signal domain

One another approach of building human activity recognition system is by using data-driven methods such as hidden Markov model (HMM), hidden conditional random fields (HCRF), and convolutional neural network (CNN). These methods learn the features and classifier simultaneously from the raw sensor signal without any prior knowledge of sensor signals and eliminate the need to calculate hand-crafted features for classification. In recent years, methods involving CNN's have shown very good results compared to other methods. This is because the signal representation learned by CNN models capture the local dependency and scale invariance in the signal exceptionally well.

In this section, we will discuss adversarial examples in the signal domain. The UCI dataset has sensor readings from an accelerometer and a gyroscope, making it a multi-modal dataset. For multi-modal data 2D convolutional neural networks are considered to work best because of their ability to capture local dependency among the temporal dimension as well as spatial dimension [27]. For the purpose of simplicity, we have used a 1D kernel in the following experiments and we are confident that similar results will be obtained with a 2D kernel. We have used a ConvNet architecture composed of two 1D convolutional layers with 64, 32 filters and kernel size of 5 and 3 respectively. This is followed by $3 * 1$ max-pooling layer and then three dense layers of sizes 64, 32, and 6.

TABLE I: The accuracy of the CNN model for the untargeted attack with different attack methods and at different values of ϵ. The accuracy was computed on all 2947 test samples in the dataset.

Epsilon (ϵ)	Attack methods		
	Fast gradient sign	Basic iterative	Least likely
$\epsilon = 0.01$	90.77%	90.63%	92.09%
$\epsilon = 0.05$	70.44%	64.43%	75.63%
$\epsilon = 0.1$	44.18 %	34.06%	39.02%
$\epsilon = 0.5$	26.63%	3.49%	1.69%
$\epsilon = 1$	16.83%	1.42%	1.01%

Table I shows the accuracy of the CNN model for untargeted misclassification attack. With the increase in the value of ϵ, the accuracy of the model decreases for all attack methods. The decrease in accuracy for the same value of ϵ is largest for the basic iterative method. This is because the basic iterative method is better suited to search the model's input space and find perturbations that can fool the target system with higher confidence. The similar pattern follows for the targeted attack as seen in Table II. It shows the success rate of the targeted attack which is defined as the ratio of the number of adversarial examples that are classified into the target class (**Sit** activity) to the total number of samples. One interesting thing to note

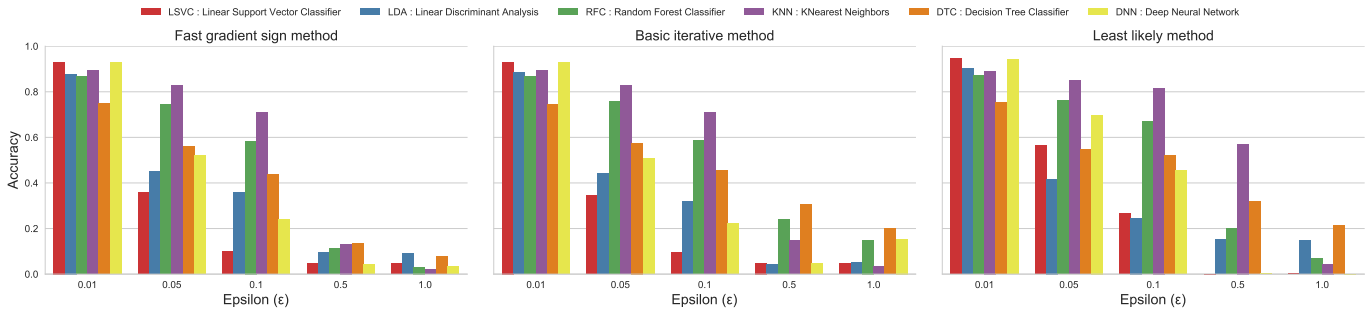

Fig. 6: The accuracy of different classifiers on the adversarial examples computed using fast gradient sign, basic iterative, and least likely methods for the untargeted attack on DNN model. The default settings were used for training all classifiers, except the deep neural network which we described ahead.

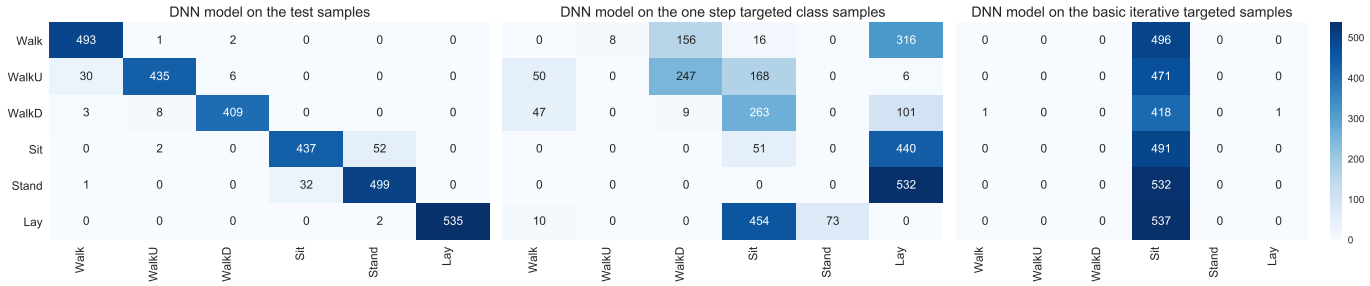

Fig. 7: Confusion matrix of the DNN model on true test samples and targeted adversarial examples computed using one step target class method and basic iterative method at $\epsilon = 1.0$. The y-axis represents the true label and the x-axis represents the predicted label.

TABLE II: The success rate of the targeted adversarial attack on the CNN model at different values of ϵ. The values represent the number of samples that were successfully classified into the target class over the total number of samples. The target activity class was **Sit**.

Epsilon (ϵ)	Attack methods	
	One step target class	Basic iterative
$\epsilon = 0.01$	17.71%	17.78%
$\epsilon = 0.05$	22.12%	23.31%
$\epsilon = 0.1$	30.50 %	36.81%
$\epsilon = 0.5$	4.64%	95.01%
$\epsilon = 1$	0%	90.19%

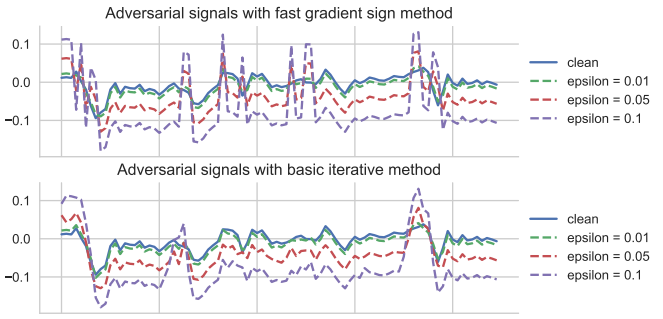

Fig. 8: Adversarial signal for the x-channel of body acceleration at different values of ϵ.

here is that at higher values of ϵ one step target class method performs poorly compared to at lower values of ϵ. This is because at higher values of ϵ the input samples are completely destroyed by the addition of ϵ-scaled noise to them. Since the adversarial examples computed for the CNN model is in the signal domain, and we call these **adversarial signals**.

Figure 8 shows the x-channel of the body acceleration signal at different values of ϵ for fast gradient sign method and basic iterative method. As we can see, the fast gradient sign method is just adding ϵ-scaled noise to the clean signal whereas the basic iterative method is following the signal pattern more closely and the added noise preserves the temporal characteristics of the signal.

D. Adversarial training as a defense

Adversarial training, which augments the training data with adversarial examples, is a well-known defense used to improve the robustness of a machine learning system against adversarial attacks. In this section, we will discuss the effect of adversarial examples on a deep neural network (DNN) trained using adversarial training. We also compare the performance of this model with a baseline mode i.e., a DNN model without adversarial training. We used the algorithm defined in [28] for adversarial training of the DNN model. In all cases we trained the model with a mini-batch size of 32, in which half the samples were clean and the other half were adversarial. It has been shown that if a fixed value of ϵ is used to compute adversarial examples during adversarial training then the trained model become robust only to that specific value of ϵ [28]. Hence, in our experiments at each epoch a random value of ϵ, taken from a truncated normal distribution defined in interval [0, 1] with underlying normal distribution $N(\mu = 0, \sigma = 0.5)$ was used to compute adversarial examples. In particular, we tested three cases of adversarial training for the DNN model. In the first case, we used the fast gradient sign method to

compute adversarial examples during the adversarial training. After training, the robustness of the trained model was tested on the adversarial examples computed using fast gradient sign method at different values of ϵ. Table III shows the accuracy of the baseline model and an adversarially trained model. As we can see adversarial training increased the robustness of the baseline model at all values of ϵ but also resulted in decrease in accuracy by 13.88% on clean test samples.

TABLE III: Accuracy of the baseline and an adversarially trained models on clean test samples and adversarial examples. Fast gradient sign method was used to compute adversarial examples for both training and testing.

Epsilon (ϵ)	Classifiers	
	Baseline	Adversarial training
clean	95.11%	81.23%
$\epsilon = 0.01$	92.97%	90.12%
$\epsilon = 0.05$	51.95%	57.31%
$\epsilon = 0.1$	23.85%	46.31%
$\epsilon = 0.5$	4.37%	50.69%
$\epsilon = 1$	3.42%	52.49%

TABLE IV: Accuracy of the baseline and an adversarially trained models on clean test samples and adversarial examples. One step target class method with the least likely class as the target class was used to compute adversarial examples for both training and testing.

Epsilon (ϵ)	Classifiers	
	Baseline	Adversarial training
clean	95.11%	81.23%
$\epsilon = 0.01$	92.53%	94.70%
$\epsilon = 0.05$	48.18%	69.05%
$\epsilon = 0.1$	22.97%	55.24%
$\epsilon = 0.5$	1.9%	23.04%
$\epsilon = 1$	1.15%	16.38%

TABLE V: Accuracy of the baseline and an adversarially trained models on clean test samples and adversarial examples computed using the basic iterative method at different ϵ values. The target classes for the adversarial examples were set to the least likely class. Here the first adversarially trained model $M1$ uses fast gradient sign method to compute adversarial examples and model $M2$ uses one step target class method with the least likely class as the target label to compute adversarial examples during adversarial training.

Epsilon (ϵ)	Classifiers		
	Baseline	Adversarial training fgsm	Adversarial training one step target
clean	95.11%	81.23%	81.23%
$\epsilon = 0.01$	94.09%	94.97%	95.38%
$\epsilon = 0.05$	69.83%	85.91%	80.59%
$\epsilon = 0.1$	45.47%	82.21%	67.05%
$\epsilon = 0.5$	0.23%	44.14%	4.3%
$\epsilon = 1$	0%	3.35%	0.61%

In the second case, we used the one step target class method with the least likely class as the target class to compute adversarial examples for both training and testing. Table IV shows the accuracy of the baseline model and an adversarially trained model. Compared to the adversarial training using the FGSM method, the one step target class method has improved the model performance at smaller values of ϵ but at larger values, the performance has decreased significantly.

Finally, in the third case, we tested the robustness of adversarially trained models $M1$(trained using fast gradient sign method) and $M2$ (trained using one step target class method) on the adversarial examples computed using the basic iterative method with least likely class as the target class. Table V shows the result. As expected the iterative method was able to degrade the performance of both $M1$ and $M2$ severely at higher values of ϵ. This shows that iterative methods are more competent at finding adversarial examples that can fool a model with high confidence.

IV. Conclusions and Future Work

In this paper, we introduced the Adar computational framework for computing adversarial examples for inertial sensor-based activity recognition systems. We investigated different gradient-based evasion attack methods and showed that even computationally simple attack methods can significantly degrade the performance of the activity recognition models. In particular, we found that the accuracy of the deep neural networks trained on the features data decreased from 95.2% to 3.4% and the accuracy of the convolutional neural network trained on the window segments of sensor signal decreased from 93.1% to 16.8%. We also showed the transferability of untargeted attacks in the feature domain and found that the k-nearest neighbor and decision tree classifiers were more robust compared to the other classifiers at all values of ϵ. We observed that adversarial training provides robustness to adversarial examples generated using one-step attack methods, but fails against iterative attack methods. In the worst case with adversarial training the robustness of the deep neural networks increased by 49.1% but at the same time suffered a loss in accuracy by 13.2% on clean test samples.

Our results in this paper have extremely important implications in behavioral medicine and mobile health, where detecting human behavior is at the center of clinical interventions. The vulnerability of activity recognition models to adversarial examples not only jeopardize the validity of behavioral interventions that rely on accurate detection of physical and behavioral context such as dietary intake, medication adherence, opioid dependence, smoking behavior, hydration status, and physical activity but also opens up risk points that can have serious impacts on people lives. In this work, we only scratched the surface studying the consequences of adversarial examples in human activity recognition systems, and there remains a need to fully understand and define adversarial attacks for HAR systems in a much greater extent. From the lack of standardization to the nature of the HAR systems, the topic of adversarial example has many facets in activity recognition models. The results of our work also ask many important questions. Should we consider the CNN model more suitable for activity recognition acknowledging the fact that

the CNN models were more robust compared to DNN models and also has an advantage of learning features and classifier directly from the sensor data? We also showed the existence of adversarial signals using CNN models. This opens up newer dimensions in the study of adversarial attacks because the ability to generate adversarial examples at the raw signal level implies that many other data processing blocks such as segmentation and pre-processing are also vulnerable to adversarial attacks. This suggests that the study of adversarial attacks in the context of sensor-based systems need to extend beyond conventional feature-level attacks. What are ramifications of adversarial signals and how can we use the information from the adversarial signals to impact the sensor system remains an open question. Some of the limitations of our work are as follows:

1) Since there are no standard datasets and activity recognition models we have carried out our experiments with simple models built from scratch and trained on a single dataset. A follow up on our work can show the effects of adversarial examples on activity recognition models with multiple datasets.
2) In our experiments, we have only considered the white-box attack setting. A follow up on our work will be to design adversarial attacks on activity recognition model in the black-box setting.

ACKNOWLEDGMENT

This work was supported in part by the United States National Science Foundation under grant CNS-1750679. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.

REFERENCES

[1] J. Bort-Roig, N. D. Gilson, A. Puig-Ribera, R. S. Contreras, and S. G. Trost, "Measuring and influencing physical activity with smartphone technology: a systematic review," *Sports Medicine (Auckland, N.Z.)*, vol. 44, May 2014.

[2] B. Biggio *et al.*, "Evasion attacks against machine learning at test time," *arXiv:1708.06131 [cs]*, vol. 7908, pp. 387–402, 2013. [Online]. Available: http://arxiv.org/abs/1708.06131

[3] C. Szegedy, W. Zaremba *et al.*, "Intriguing properties of neural networks," *ICLR*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *CoRR*, 2014. [Online]. Available: http://arxiv.org/abs/1412.6572

[5] N. Papernot, P. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *ArXiv e-prints*, 2016. [Online]. Available: http://arxiv.org/abs/1605.07277

[6] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial classification," in *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*, 2004, p. 99. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1014052.1014066

[7] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, Jul. 2016. [Online]. Available: http://arxiv.org/abs/1607.02533

[8] N. Carlini, P. Mishra, and other, "Hidden voice commands," in *In 25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 513–530. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/carlini

[9] A. Bulling, U. Blanke, and B. Schiele, "A Tutorial on Human Activity Recognition Using Body-worn Inertial Sensors," *ACM Comput. Surv.*, vol. 46, pp. 1–33, Jan. 2014. [Online]. Available: http://doi.acm.org/10.1145/2499621

[10] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys & Tutorial*, vol. 15, pp. 1192–1209, 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6365160/

[11] L. T. Nguyen, "Enhanced human activity recognition on smartphone by using linear discrimination analysis recursive feature elimination algorithm," in *Context-Aware Systems and Applications*. Springer International Publishing, 2017, pp. 72–81.

[12] G. Ding, J. Tian, J. Wu, Q. Zhao, and L. Xie, "Energy efficient human activity recognition using wearable sensors," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Apr. 2018, pp. 379–383.

[13] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas, "Window Size Impact in Human Activity Recognition," *Sensors (Basel, Switzerland)*, vol. 14, pp. 6474–6499, Apr. 2014. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4029702/

[14] A. Bevilacqua, K. MacDonald, A. Rangarej *et al.*, "Human Activity Recognition with Convolutional Neural Networks," in *Machine Learning and Knowledge Discovery in Databases*. Springer International Publishing, 2019, vol. 11053, pp. 541–552. [Online]. Available: http://link.springer.com/10.1007/978-3-030-10997-4_33

[15] S. Ha and S. Choi, "Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2016, pp. 381–388.

[16] I. Goodfellow, P. McDaniel, and N. Papernot, "Making machine learning robust against adversarial inputs," *Commun. ACM*, vol. 61, no. 7, pp. 56–66, Jun. 2018. [Online]. Available: http://doi.acm.org/10.1145/3134599

[17] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks," *arXiv:1511.04508 [cs, stat]*, Nov. 2015. [Online]. Available: http://arxiv.org/abs/1511.04508

[18] N. Papernot, P. McDaniel, I. Goodfellow *et al.*, "Practical Black-Box Attacks Against Machine Learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 2017, pp. 506–519. [Online]. Available: http://doi.acm.org/10.1145/3052973.3053009

[19] A. Chakraborty, M. Alam, V. Dey *et al.*, "Adversarial Attacks and Defences: A Survey," *arXiv:1810.00069 [cs, stat]*, Sep. 2018. [Online]. Available: http://arxiv.org/abs/1810.00069

[20] T.-J. Chang, Y. He, and P. Li, "Efficient Two-Step Adversarial Defense for Deep Neural Networks," *arXiv:1810.03739 [cs, stat]*, Oct. 2018. [Online]. Available: http://arxiv.org/abs/1810.03739

[21] D. Anguita, A. Ghio, L. Oneto *et al.*, "A public domain dataset for human activity recognition using smartphones," in *21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013*, 2013, pp. 437–442.

[22] M. Abadi, A. Agarwal *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: http://tensorflow.org/

[23] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] N. Papernot, F. Faghri *et al.*, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.

[25] Y. Wang, S. Jha, and K. Chaudhuri, "Analyzing the Robustness of Nearest Neighbors to Adversarial Examples," *arXiv:1706.03922 [cs, stat]*, Jun. 2017. [Online]. Available: http://arxiv.org/abs/1706.03922

[26] C. Sitawarin and D. Wagner, "On the Robustness of Deep K-Nearest Neighbors," *arXiv:1903.08333 [cs, stat]*, Mar. 2019. [Online]. Available: http://arxiv.org/abs/1903.08333

[27] S. Ha, J. Yun, and S. Choi, "Multi-modal Convolutional Neural Networks for Activity Recognition," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, Oct. 2015, pp. 3017–3022.

[28] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial Machine Learning at Scale," *arXiv:1611.01236 [cs, stat]*, Nov. 2016. [Online]. Available: http://arxiv.org/abs/1611.01236