SensorGAN: A Novel Data Recovery Approach for Wearable Human Activity Recognition

DINA HUSSEIN and GANAPATI BHAT, Washington State University, USA

Human activity recognition (HAR) and, more broadly, activities of daily life recognition using wearable devices have the potential to transform a number of applications, including mobile healthcare, smart homes, and fitness monitoring. Recent approaches for HAR use multiple sensors on various locations on the body to achieve higher accuracy for complex activities. While multiple sensors increase the accuracy, they are also susceptible to reliability issues when one or more sensors are unable to provide data to the application due to sensor malfunction, user error, or energy limitations. Training multiple activity classifiers that use a subset of sensors is not desirable, since it may lead to reduced accuracy for applications. To handle these limitations, we propose a novel generative approach that recovers the missing data of sensors using data available from other sensors. The recovered data are then used to seamlessly classify activities. Experiments using three publicly available activity datasets show that with data missing from one sensor, the proposed approach achieves accuracy that is within 10% of the accuracy with no missing data. Moreover, implementation on a wearable device prototype shows that the proposed approach takes about 1.5 ms for recovering data in the w-HAR dataset, which results in an energy consumption of 606 μJ. The low-energy consumption ensures that SensorGAN is suitable for effectively recovering data in tinyML applications on energy-constrained devices.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; • **Computing methodologies** → *Machine learning*; • **Human-centered computing** → *Mobile devices;*

Additional Key Words and Phrases: Human activity recognition, wearable electronics, missing data detection, data imputation, clustering, health monitoring

ACM Reference format:
Dina Hussein and Ganapati Bhat. 2024. SensorGAN: A Novel Data Recovery Approach for Wearable Human Activity Recognition. *ACM Trans. Embedd. Comput. Syst.* 23, 3, Article 53 (May 2024), 22 pages.
https://doi.org/10.1145/3609425

1 INTRODUCTION

Human activity recognition (HAR) and, more broadly, activities of daily life recognition using low-power wearable devices are becoming popular. The increasing popularity is due to their applications in healthcare, smart home, and fitness monitoring [1, 12, 20, 21, 29]. For instance, knowing the activities of the user in a free-living environment can guide healthcare professionals in devising personalized treatments for movement disorders [12, 21, 24]. These applications have been enabled by recent advances in sensor technology, low-power processors, and communication technologies.

Authors' address: D. Hussein and G. Bhat, Washington State University, P.O. Box 642752, Pullman, Washington, USA, 99164-2752; emails: {dina.hussein, ganapati.bhat}@wsu.edu.

1539-9087/2024/05-ART53 $15.00
https://doi.org/10.1145/3609425

Table 1. Comparison of HAR Accuracy with Different Set of Sensors for the w-HAR [5] Dataset

Sensors	Accuracy (%)
Accelerometer and Stretch	93
Accelerometer only	85
Stretch sensor only	73

State-of-the-art approaches for HAR use one or more sensors mounted on the body to identify the activities [28, 32, 33]. Multiple sensors are typically used to obtain higher recognition accuracy or to increase the complexity of activities. For instance, the HAR algorithm proposed by Reiss et al. [28] uses a heart rate monitor along with multiple accelerometers mounted on the body to distinguish between walking and ascending stairs. The data from multiple sensors are typically aggregated at a central processing node for activity classification using machine learning algorithms.

Most, if not all, HAR approaches that employ multiple sensors are trained with the assumption that data from all the sensors are available without any errors in the data. This assumption may not hold true in real-world settings for energy-constrained wearable devices. For instance, one or more sensors may turn off periodically due to lack of energy availability in the device [16, 23]. The sensor data may also be missing due to user error, such as forgetting to place a sensor at the appropriate location [17, 19]. The missing sensor data, in turn, may lead to significant degradation of the recognition accuracy. This is because the HAR algorithms are trained with the assumption that data from all sensors are available at runtime. Therefore, there is a strong need to design approaches that account for missing data in one or more sensors while providing accurate HAR.

A canonical approach to handle missing data from one or more sensors is to design individual classification algorithms that do not use data from the respective sensors. For example, in an application with one accelerometer and one gyroscope, we can design three classifiers corresponding to each sensor configuration. Then, at runtime, we can choose the appropriate classifier as a function of the available sensors. However, this approach suffers from a key limitation. Specifically, the accuracy with a subset of sensors may be lower than the accuracy from all the available sensors. Table 1 shows an example of this behavior for the w-HAR dataset [5] that includes accelerometer and stretch sensors. The activity recognition accuracy with both sensors is 93%, while it reduces to 85% and 73% when using only accelerometer or stretch sensors, respectively. This issue can be more severe for complex activities that require the information provided by each sensor. Moreover, loading multiple classifiers for each missing data scenario leads to additional context switching overhead for the wearable devices. Consequently, there is a need for approaches that achieve comparable accuracy to the baseline with full set of sensors while eliminating the need to train individual classifiers for all possible sensor combinations.

This article presents a novel generative approach, referred to as SensorGAN, for missing data recovery. The approach recovers data from missing sensors and uses a single classifier to perform activity classification. Inspired by the success of **generative adversarial networks (GAN)** [9] in the image recognition and natural language processing domains [36, 38], we propose to utilize GANs to seamlessly recover missing data at runtime so that the classifiers can provide high accuracy under all conditions. Conventional GANs are able to generate images or time series data from the latent space to provide additional examples for training or as adversarial data [9]. However, conventional GANs are not suitable for sensor data recovery, since the generated data must accurately represent the activity being performed by the user. Therefore, the proposed SensorGAN

uses conditional GANs and leverages the data of available sensors as conditional inputs. This ensures that the missing sensor data are generated as a function of the available sensors to accurately represent the activity patterns.

The proposed sensor to sensor data generation method, SensorGAN, is an instantiation of conditional GANs [31] that take the desired label (e.g., image of a cat) as one of the conditional inputs. Then, the conditional GAN generates an example of the desired label from the latent space. Conventional conditional GANs use loss functions representing the accuracy of the discriminator in distinguishing between real and generated data to train the network. This is not sufficient for the proposed SensorGAN approach, since our goal is to generate data that follows distribution of the sensor as the user is performing activities. More specifically, the data generated by SensorGAN must satisfy following requirements: (1) the generated time series data must follow the distribution and behavior of real sensor data for the activity of interest, (2) it must have statistical distribution close to the real, observed data, and (3) the generated data should be able to provide classification accuracy that is comparable to the accuracy achieved with real sensor data. To achieve these objectives, we introduce three additional loss functions during the training of the generator. To ensure that the generated data matches the real-world sensor data, we include the **mean-squared error (MSE)** as part of the training process. Minimizing the MSE ensures that generated data closely matches real-world sensor data. Next, to ensure that the statistical features of the distribution of generated data match the actual data features, we introduce feature loss. Specifically, we take the MSE of the statistical features of the real and generated data as the feature loss. This loss is then used as part of the generator optimization. Finally, to ensure that data from SensorGAN provides accurate classification, we first train a baseline HAR classifier with data from all sensors. Then, the classification loss with the generated data is used to optimize the generator. The inclusion of these additional losses enables SensorGAN to effectively recover the missing data.

After training the generator with our optimized loss functions, we deploy it on a wearable device to recover missing data at runtime. If the device detects that any of the sensors has missing data, it uses the generator to first recover missing data. Then, it is used with the data of other sensors to perform activity classification. We validate the proposed SensorGAN approach on three publicly available HAR datasets [5, 28, 29]. Our experiments with all three datasets show that the SensorGAN approach is able to recover data with an average MSE that is significantly smaller than the range of each dataset. Moreover, the classification accuracy with the generated data is within 10% of the accuracy without any missing data. We also implement SensorGAN on a wearable device prototype with the Odroid-XU3 development board [15] to measure the execution time and energy overheads. Measurements with the w-HAR dataset show that the generator incurs 1.5 ms execution time and 606 μJ energy overhead, which makes SensorGAN suitable for use in tinyML applications. In summary, this article makes the following contributions:

- A novel generative approach, SensorGAN, to recover missing sensor data in HAR by leveraging the data available from other sensors as conditional inputs;
- Introduction of MSE, feature loss and classification loss into generative networks for HAR to ensure that the data generated by SensorGAN closely follows real-world sensor data, follows the distribution of statistical features and provides high classification accuracy;
- Experiments with three publicly available datasets [5, 28, 29] to demonstrate that the proposed approach is able to generate accurate sensor data and provide classification accuracy that is within 10% of the baseline with no missing data.

The rest of the article is organized as follows: Section 2 reviews the related work, while Section 3 introduces the preliminaries for HAR. Section 4 sets up the data generation problem, and Section 5 introduces the proposed SensorGAN approach. We provide the experimental results with

three datasets in Section 6 and finally conclude the article with some future research directions in Section 7.

2 RELATED WORK

HAR is being increasingly used for health monitoring, rehabilitation, and fitness monitoring applications [20, 21, 29]. Indeed, knowing what a user is doing is one of the first steps in prescribing therapy for movement disorders [12, 24]. Early algorithms for HAR typically use a single sensor on the body to monitor sensor data and identify activities [3, 4, 6, 7]. However, increasing number of activities being recognized with HAR algorithms has led to multiple sensors being used for monitoring the activities. For instance, the PAMAP2 dataset [28] uses accelerometers mounted on three locations and a heart rate sensor to monitor user activities. While the addition of more sensors allows recognition of a richer set of activities, it can also lead to performance degradation if one or more sensors have missing data [16, 18, 19, 23].

Sensors in wearable devices can have missing data due to various reasons including human error, battery constraints, or malfunction [16, 18, 19, 23]. To handle the missing data, a number of statistical and machine learning approaches have been proposed [10, 14, 27, 37]. Commonly used statistical methods include using the mean, median, or regression for data imputation [10]. Machine learning algorithms have also been used to handle missing data recently. The machine learning approaches include **k-nearest neighbor (k-NN)**, autoencoders, and GAN [14, 27, 37]. k-NN methods are popular due to their simplicity, however, they are not suitable for low-power wearable devices due to the need to store the entire training data in the memory. GAN-based methods have also been proposed for missing data recovery. In particular, the **generative adversarial imputation network (GAIN)** proposed by Yoon et al. [37] modifies general GANs to impute missing data at runtime. To achieve this, GAIN takes the available data with missing samples and a missing vector to denote the missing samples. Then, it uses the generator to recover missing data. While these data recovery methods are useful, they are generally suitable *only* for recovering isolated missing samples and not long sequences of missing data due to sensor malfunction. Therefore, there is a need to develop approaches that can recover longer sequences of missing data to enable reliable activity classification.

The proposed SensorGAN approach precisely addresses the above challenges and enables recovery of long sequences of missing data. SensorGAN leverages the fact that data from available sensors can be used to generate data of the missing sensor. Then, the recovered sensor data are used along with the trained classifier to obtain reliable activity classification. Using SensorGAN ensures that the HAR algorithm does not need to train a classifier with each sensor. The proposed SensorGAN approach is complementary to MotionTransormer [8], which aims to transfer motion tracking models to new sensor locations where no labeled data are available. MotionTransformer uses knowledge of labeled motion data in one location to generate sequences for another location. However, the MotionTransformer framework is not validated for conditional data generation from one type of sensor to another for activity recognition applications. In contrast, we demonstrate the proposed SensorGAN framework with three publicly available activity datasets. Our experiments show that the data generated using SensorGAN has accuracies that are within 10% of the baseline accuracy with minimal overhead.

3 HAR BACKGROUND

This section first provides a brief overview of the main steps involved in state-of-the-art HAR approaches. Then, we outline the changes to the HAR flow with SensorGAN to enable robust HAR in the presence of missing sensor data.

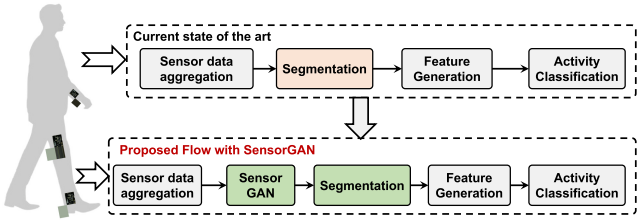

Fig. 1. Overview of the human activity recognition flow.

3.1 State-of-the-art HAR Processing Pipeline

Figure 1 shows the sequence of major steps involved in building a HAR algorithm. One or more sensors are mounted at various locations on the body, as shown in the left side of the figure. Typical sensors used in HAR applications include accelerometer, gyroscopes, stretch sensors, and heart rate sensors [21]. The data obtained from the sensors goes through four main steps described as follows.

Sensor Data Aggregation: The data from multiple sensors are aggregated either on a host device or on one of the sensor nodes so that information from all the sensors can be processed together. This step is required, since the sensors may be discrete units without any physical connections across multiple sensors. Therefore, the data are transferred from each sensor to a central location for further processing.

Segmentation: The aggregated sensor data are then passed through a segmentation algorithm to generate discrete activity windows. Segmentation is an important step to ensure that each segment for classification does not contain multiple activities. Most HAR approaches use fixed-length segments that range from 2 to 10 s in length. The fixed length segments can contain more than one activity in a segment or contain parts of a single activity, which makes it difficult to classify them accurately. More recently, variable length segmentation algorithms have been proposed to ensure that each segment includes a single activity [5, 35]. Variable length segments can aid in reducing the complexity of the HAR classifier. In this article, we utilize the default segmentation algorithms provided by each dataset.

Feature Generation: Supervised learning algorithms commonly used in HAR require features that capture characteristics of sensor data for each activity. Common features used in HAR algorithms include statistical measures such as mean, median, kurtosis, and skewness. Time and frequency domain features, such as discrete wavelet transform and fast Fourier transform, have also been used [5]. Recent work has also proposed one-dimensional **convolutional neural networks (CNN)** [22] for activity recognition. One-dimensional CNNs use the sensor data as inputs instead of handcrafted features.

Classification: The final step in the HAR pipeline is activity classification using the features and supervised learning algorithms, as shown in Figure 1. Early HAR approaches use k-NN, support vector machines, and decision trees [11, 13]. Recent HAR approaches have used neural networks including **multilayer perceptrons (MLP)**, convolutional neural networks, and recurrent neural networks [5, 22, 25, 39]. In this article, we use MLP classifiers while noting that the proposed approach is not dependent on any specific classifier structure.

3.2 Proposed HAR Pipeline with SensorGAN

The typical HAR pipeline shown in Figure 1 is susceptible to inaccurate classification if one of the sensors has missing samples. This can occur due to sensor malfunction, energy limitations, or user error. For instance, the user may forget to place one of the sensors in the appropriate location,

thus leading to missing data. The missing data, in turn, can cause errors in segmentation, feature generation, and classification. To resolve this limitation, we recover any missing data using the proposed SensorGAN approach immediately after the sensor data aggregation, as shown in the lower part of Figure 1. Once the missing data are recovered, they are passed on to segmentation algorithm and other processing blocks to obtain the activity classification. Including the SensorGAN block immediately after data aggregation ensures that all other components in the HAR pipeline can execute seamlessly without requiring any changes.

4 PROBLEM SETUP

We consider a HAR system with M sensors mounted on the user's body. The data collected by the sensors are fed into the segmentation algorithm that creates activity segments. Without loss of generality, we assume that each activity segment contains T samples from each sensor. Consequently, the sensor data in each segment can be represented with the $M \times T$ matrix $\mathcal{X} = \{X_1, X_2, \ldots X_M\} \in \mathbb{R}^{M \times T}$. The sensor data $\mathcal{X}$ are input to a feature generation function g to generate the activity features $\mathcal{X}_g$. The activity label in each window is denoted by a^*. The feature data $\mathcal{X}_g$ and a^* are then used in a supervised learning algorithm to train a classifier F_θ, where θ denotes the classifier parameters. At runtime, features generated from sensor data are fed to the classifier F_θ to obtain the activity prediction $\hat{a}$.

Next, consider that k ($<M$) of the M sensors are unable to monitor the user activity in a given activity window t. Since the activity recognition algorithm expects the data from all sensors, the missing sensor's data in the data matrix $\mathcal{X}$ will get substituted by zeros, another constant value, or noise. We denote the matrix with missing sensor values as $\mathcal{X}_m$. Consequently, the features generated from the $\mathcal{X}_m$ do not match the expected features for an activity, leading to misclassification by classifier F_θ. The misclassifications can have an adverse effect on user health in case of critical activities, such as falls. Given this, our goal is to obtain a general sensor data recovery function f_r^* that uses the data from available sensors and zero-filled missing sensors to recover missing sensor data. Specifically, the recovery function $f_r^* : \mathbb{R}^{M \times T} \rightarrow \mathbb{R}^{M \times T}$ maps the data from $M - k$ available sensors to all the sensors used in HAR. The recovery function generates a matrix of size $M \times T$ that contains both available and missing sensors. From this, we can extract the data for k missing sensors. The mapping function must ensure that the generated data closely matche real-world sensor data for all activities and provides accurate activity classification. Note that in this article, we assume that the wearable system is able to detect presence of missing data and our goal is to obtain the missing data recovery function using data of available sensors.

The general problem of obtaining the recovery function is challenging, since the function has to potentially handle $2^M - 2$ combinations of missing sensors. Two scenarios are excluded in the number of potential combinations, since we do not need sensor data recovery when all sensors are available and the system cannot recover any data if all sensors are missing. We can handle the complexity of obtaining a single recovery function f_r^* by designing individual recovery functions for each of the $2^M - 2$ combinations, however, this may lead to high memory overhead for the wearable device.

Due to the complexity of a single recovery function f_r^* and the overhead of individual recovery functions for $2^M - 2$ combinations, we consider a subset of the problem where one sensor is missing at a time. More specifically, our goal is to design a recovery function $f_r : \mathbb{R}^{M \times T} \rightarrow \mathbb{R}^{M \times T}$ that maps the data from $M - 1$ available sensors to all sensors, including the missing sensor. The data for missing sensor are then extracted from the generated data matrix of size $M \times T$. The input matrix to the recovery functions consists of both available sensors and zero-filled missing sensor. The simplified problem is acceptable for HAR applications, since they typically employ less than five

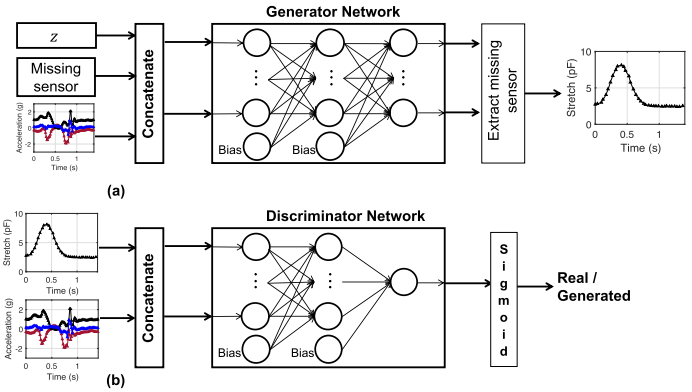

Fig. 2. Overview of the SensorGAN architecture.

sensors [21, 30], which does not result in significant overhead. We leave the generalized function f_r^* as future work while demonstrating the value of sensor data recovery on the smaller problem with one missing sensor.

5 PROPOSED SENSORGAN APPROACH

5.1 Overview

The goal of the SensorGAN is to train a function f_r that recovers missing sensor data to enable reliable activity recognition. To this end, SensorGAN employs **conditional generative adversarial networks (cGAN)** [31] to recover the missing data. GANs have been popular recently for data augmentation in image processing, natural language processing, and medical imaging [2, 9]. In these applications, GANs are used when there are insufficient data for training a machine learning algorithm. However, conventional GANs are unable to generate data for a required label or condition. For instance, in medical imaging, we may have fewer examples of a rare condition and it is essential to generate additional examples of the rare condition. cGANs provide the ability to generate data for specific classes or labels. Specifically, cGANs take the desired label as an input and generate data conditioned on the label. For instance, in an image classification problem, the cGAN may take a desired label "cat" as the input to generate images of cats. Inspired by the success of cGANs, we propose to use them as the basis for SensorGAN. However, unlike traditional cGANs, the conditional input in SensorGAN is time series data available from the sensors. The output generated by SensorGAN is a time series to be used in classification instead of being used for augmenting training data. Consequently, we introduce additional loss functions that ensure that our recovery function f_r accurately generates missing data that closely resembles the real-world sensor behavior. The rest of this section provides details on the SensorGAN architecture, loss functions, and the training procedure to accurately recover missing sensor data.

5.2 SensorGAN Architecture

Figure 2 shows the architecture of SensorGAN. Following the general structure of GANs, we include a discriminator and generator in SensorGAN. The generator G is used at runtime to recover missing data while the discriminator is used in the training process to ensure that the generator is able to accurately follow the missing sensor data. The generator in SensorGAN takes two inputs: a latent vector z and the data from available sensors $X_m \in \mathbb{R}^{M \times T}$. The input sensor data matrix contains both available and missing sensors. The data for missing sensors are substituted with zeros. These input data are first concatenated and reshaped to form an input vector to the generator. The

input vector is then passed through a fully connected neural network to obtain the sensor values, denoted by $\hat{\mathcal{X}}_m$. We choose a fully connected neural network for the generator, since our goal is to deploy the generator on a low-power wearable device. As such, the lower number of computations in a fully connected neural network when compared to convolutional or recurrent networks will ensure that the generator is energy efficient. The output of the generator is a $\mathbb{R}^{M \times T}$ matrix representing both available and missing sensors. Since our goal is to recover data for missing sensors, we extract data for missing sensor using a mask $\mathcal{M}$. The mask specifies a zero wherever a sensor is available and contains a value of one for missing sensors. Recovered data for the missing sensor are then concatenated with available sensors to form the full sensor data for a window. The sensor data are then fed to the feature generation and classification blocks to obtain activity classification.

The discriminator architecture, shown in Figure 2(b), takes either observed values or generated data for a given sensor. That is, the discriminator takes data of a single sensor as input at a time. The main goal of the discriminator is to determine if the input sensor data are real or generated. Similar to the generator, we use a fully connected neural network to implement the discriminator. The output of the discriminator is a number between zero (generated) to one (real) that specifies the probability of the input being real or generated.

5.3 SensorGAN Objective

The general objective in cGANs consists of a minimax game where the discriminator is trained to maximize the accuracy of predicting real versus generated data and the generator is trained to minimize the probability of the discriminator mispredicting generated data as real data. Using this, the objective function is written as

$$\min_{G} \max_{D} \mathbb{E}[\log D(\mathcal{X}^*|\mathcal{X}_m) + \log(1 - D(G(z|\mathcal{X}_m)|\mathcal{X}_m)], \tag{1}$$

where $D(\cdot)$ denotes the probability outputs produced by the discriminator, $\mathcal{X}^*$ is the ground truth for the missing sensor data, and G is the generator. It is computationally difficult to directly optimize for the objective function in Equation (1). Therefore, most GAN approaches use an iterative process where the discriminator is optimized with a fixed generator and the generator is optimized with a given discriminator. As part of this, a separate objective function is defined for the discriminator and the generator, as follows.

5.4 Discriminator Loss Function

The primary goal of the discriminator is to accurately distinguish between real and generated data. To this end, the discriminator optimization solves the following problem:

$$\max_{D} \mathbb{E}[\log D(\mathcal{X}^*|\mathcal{X}_m) + \log(1 - D(G(z|\mathcal{X}_m)|\mathcal{X}_m)]. \tag{2}$$

The objective problem ensures that the discriminator correctly identifies observed data as real and generated data as fake. In practice, the optimization in Equation (2) is converted to a minimization using binary cross entropy so that standard stochastic optimization algorithms can be employed to train the discriminator.

5.5 Generator Loss Function

SensorGAN Generator Goals: The generator in SensorGAN has two primary goals. First, we would like to maximize the classification accuracy in the presence of missing sensor values. However, just recovering classification accuracy may not be sufficient to accurately reconstruct the time-series sensor data. Accurate reconstruction of time-series data can offer additional insights in many applications. For instance, in health monitoring applications, the physician may look at

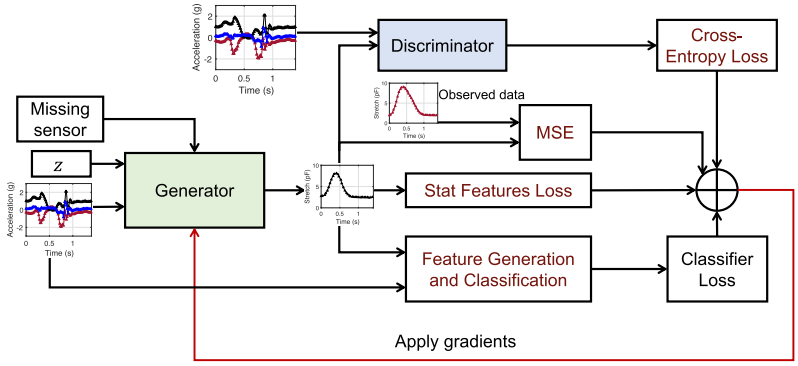

Fig. 3. Overview of SensorGAN training.

classification results and sensor data patterns to arrive at a conclusion. In activity monitoring sensor data patterns offer insight into limb movement or acceleration data for individual users. Therefore, our second goal is to ensure that sensor data are reconstructed accurately for inspection by physicians or other users of the application. The generator must ensure that the following conditions are satisfied to meet the above objectives: (1) The discriminator must be unable to distinguish between real and generated data, (2) the sensor values given by the generator must closely match the observed data, (3) have statistical distribution close to the real, observed data, and (4) provide high classification accuracy with an activity classifier trained using real sensor data. To achieve these objectives, we use four loss functions to train the generator, as shown in Figure 3.

5.5.1 Discriminator Loss. The most commonly used loss function in generator training quantifies the accuracy of the discriminator on the generated data. That is, the generator is trained such that the discriminator is unable to classify the data as fake, i.e., not real-world data. The discriminator loss function is expressed as

$$l_D = \log(1 - D(G(z|\mathcal{X}_m)|\mathcal{X}_m)]. \tag{3}$$

The loss is calculated with a fixed discriminator. Minimizing l_D leads to the discriminator predicting data from the generator as real-world data, which means that the discriminator is unable to distinguish between real and generated data.

5.5.2 Mean-squared Error Loss. The second objective of the generator in SensorGAN is to ensure that the values of the recovered sensor data are as close to the real-world sensor data as possible. To achieve this, we introduce the MSE between generated and observed sensor data as an additional loss. The MSE can be expressed as

$$l_{MSE} = \mathbb{E}(\mathcal{X}^* - \hat{\mathcal{X}}_m)^2, \tag{4}$$

where $\hat{\mathcal{X}}_m$ is the generated data and $\mathbb{E}$ is the expectation operator. The MSE loss ensures that each data point generated is close in magnitude to the ground-truth data. That is, the generator data closely matches actual sensor data when we minimize the MSE loss. As such, generated data can be examined by users or health experts to obtain additional insights into user activities.

5.5.3 Classifier Loss. The data recovered by the generator are eventually used in the classifier F_θ to obtain the activity classification. Therefore, it is critical that the generated data leads to accurate classifications. To this end, we include the classifier loss as part of generator training.

Specifically, we use the categorical cross-entropy loss for the MLP classifier, as follows:

$$l_{CE} = -\sum_{i=1}^{A} a_i \log(p_i), \tag{5}$$

where A is the number of activities, a_i is the activity label, and p_i are activity probabilities at the output layer. During each iteration of the generator training, we use the recovered data along with the data of available sensors to obtain the feature vector. The feature vector is then fed to the classifier F_θ to classify the activities. The classification output is then used in Equation (5) to calculate the classifier loss. Using classifier loss in the generator training ensures that the activity classifications using generated data are accurate.

5.5.4 Statistical Feature Loss. The final consideration in the generator training is ensuring that the statistical properties of the generated data are same as the observed sensor data. To this end, we include the mean-squared difference in the statistical properties of the two sensor signals as an additional loss. Specifically, we include six statistical features in the loss function: {mean, variance, kurtosis, skewness, maximum, minimum}. Using these, the statistical loss function can be written as

$$l_{stat} = \mathbb{E}(S_f(\mathcal{X}^*) - S_f(\hat{\mathcal{X}}_m))^2, \tag{6}$$

where S_f is the function that calculates the statistical features as a function of the sensor data. Minimizing the statistical feature loss ensures that data distribution of generated data matches actual data, thus aiding in accurately recovering data patterns. This, in turn, leads to accurate feature generation and classification as well.

Combining the four losses, we can write the final loss for the generator training as

$$\mathcal{L}_G = l_D + \lambda_1 l_{MSE} + \lambda_2 l_{CE} + \lambda_3 l_{stat}, \tag{7}$$

where λ_1, λ_2, and λ_3 are the weights of respective loss functions. We set these parameters based on a design space exploration to achieve the lowest overall loss and highest accuracy.

5.6 SensorGAN Training

Algorithm 1 describes the overall process of training the generator and discriminator in SensorGAN. The algorithm takes the missing sensor data, ground truth for the missing sensor $\mathcal{X}^*$, and the activity classifier F_θ as primary inputs. It also takes the number of training iterations N as an additional input. As the first step, the training algorithm initializes parameters for the generator and discriminator, respectively. In each iteration of the training, we first randomly select a missing sensor. The missing sensor values are set to zero and generate sensor data matrix $\mathcal{X}_m$. Using this, the algorithm first obtains a new discriminator by optimizing the loss function in Equation (2). Then, we use the current generator to obtain the missing sensor data using $\mathcal{X}_m$. The generated data are then used in conjunction with the ground-truth data and activity classifier to obtain the four losses described in the previous section. The combined loss is optimized to train a new generator G_n. At the end of N iterations, the algorithm returns the optimized generator and discriminator. Finally, the generator is deployed on the wearable device to recover data at runtime to provide reliable activity classification.

6 EXPERIMENTAL EVALUATION

6.1 Experimental Setup

6.1.1 Wearable Device. We use a wearable device based on the Odroid-XU3 development board [15] processor as the main processing unit. Since each dataset used to evaluate SensorGAN has different set of sensors, we consider that the processor present on the Odroid-XU3 development board is representative of the common processor used to process and recover sensor

ALGORITHM 1: SensorGAN Training

Input: Activity Classifier F_θ, Ground-truth sensor data $\mathcal{X}^*$, Number of iterations N
Initialize weights for discriminator D
Initialize weights for generator G
for $n = 1, 2, \ldots, N$ **do**
    Randomly choose a missing sensor
    Generate missing sensor data matrix $\mathcal{X}_m$
    Train D_n by optimizing loss in Equation (2)
    $\hat{\mathcal{X}}_m$ Generate sensor data using G_n and $\mathcal{X}_m$
    Calculate discriminator loss l_D
    Calculate MSE loss l_{MSE} using $\hat{\mathcal{X}}_m$ and $\mathcal{X}^*$
    Generate activity features using $\hat{\mathcal{X}}_m$ and $\mathcal{X}_m$
    Calculate classifier loss l_{CE}
    Calculate statistical feature loss l_{stat} using Equation (6)
    Train G_n by optimizing the combined loss
end
return Last discriminator D_N and generator G_N

data. The Exynos 5422 processor on the Odroid-XU3 board integrates four high-performance ARM Cortex-A15 cores and four low-power ARM Cortex-A7 cores that are used to perform segmentation, feature generation, and classification. The Odroid-XU3 board also provides power sensors to profile power consumption of programs running on the board. The processor is also used to recover missing data using the SensorGAN generator. We also note that any low-power processor can be used to perform the activity recognition and recover data using SensorGAN.

6.1.2 Evaluation Metrics. The primary goal of SensorGAN is to recover missing data such that it resembles real-world sensor data and is able to provide accurate classification. To this end, we use the MSE and classification accuracy to evaluate the quality of the data recovery by SensorGAN. Specifically, we calculate the MSE of the recovered data with respect to the ground truth for each dataset. Next, to obtain the classification accuracy, we first train an activity classifier with real-world observed sensor data. We use the trained classifier to recognize activities with the recovered data. That is, instead of using the real-world sensor data as input to the classifier, we use recovered data. The activities classified with the recovered data are then used to obtain the classification accuracy with SensorGAN. Another goal of SensorGAN is to ensure that the generated data has statistical features that are similar to the real-world sensor data. To analyze this, we compare the mean, variance, and skewness of the observed and generated data. We also use the **t-distributed stochastic neighbor embedding (t-SNE)** [34] method to visualize the data generated by SensorGAN and real-world observed sensor data. t-SNE is a dimensionality reduction technique that aids in the visualization of high dimensional data by reducing the data to two or three dimensions. To achieve this, t-SNE constructs a probability distribution such that similar points are nearby and dissimilar points are far with high probability. If two distributions are close to each other, then their samples will be close in the t-SNE map, while dissimilar distributions will have larger distances between samples. Overall, these metrics allow us to evaluate the quality of data generated by the generator in SensorGAN.

6.1.3 Baseline Methods. We employ three baselines for evaluation of classification accuracy with SensorGAN: zero-filling, average-filling, and sensor-suite specific classifiers. Brief descriptions of each baseline are as follows:

Zero-filling: Most devices for wearable HAR use memory buffers on the processor to store incoming sensor data before performing activity classification. The memory buffers are typically initialized to zero for each activity window. As such, we consider zero-filling of missing data as one of the baselines.
Average-filling: One can also substitute missing sensor data with an average case of sensor data observations. Specifically, we can store the average value of a sensor in an activity window by taking average of training data. The average-filling case is used as a second baseline due to its simplicity and ease of implementation.
Sensor-suite Specific classifiers: A commonly used approach to handle missing data is to train individual classifiers for each possible set of missing data scenario so that we can perform activity classification with available set of sensors. Therefore, we use it as a baseline for comparisons. At the same time, training multiple classifiers suffers from two key limitations: (1) the number of required classifiers increases with sensors. For instance, for a system with M potential sensors, we require $2^M - 2$ classifiers. (2) The system may need to load individual classifier for each window if the set of missing sensors changes across windows. This leads to additional runtime overhead for context switching for classification.

6.2 Datasets

We use three publicly available activity datasets to evaluate the performance of SensorGAN. This section describes the characteristics, features used for classification, and missing data model for each dataset.

6.2.1 w-HAR [5]. The w-HAR dataset includes accelerometer and stretch sensor data for 22 users and eight activities. The eight activities monitored by the w-HAR dataset are: {*Jump, lie down, sit, stand, walk, stairs up, stairs down, and transition*}. The accelerometer is mounted on the user's ankle while the stretch sensor is mounted on the knee. The accelerometer is sampled at 250 Hz and the stretch sensor is sampled at 25 Hz. w-HAR employs a variable-length segmentation technique that creates a new activity segment whenever the stretch sensor has a local minimum. The variable length segments ensure that each segment contains a single activity, except for segment where there is an activity transition. Overall, using variable-length segmentation, the w-HAR dataset provides a total of 4,740 activity windows.

We employ the feature set provided by w-HAR to train the classifiers. Specifically, the features from the accelerometer include the **discrete wavelet transform (DWT)**, mean, and variance. Since the stretch sensor is repetitive in nature, the features for the stretch sensor include the **fast Fourier transform (FFT)** of the sensor data. The minimum and maximum of the stretch sensor also provide useful information, therefore they are included as additional features. We use this feature set provided in w-HAR to train the MLP classifier.

To model the missing data behavior in w-HAR, we consider that the data from the stretch sensor is missing. This is a reasonable assumption in w-HAR, because it is a discrete sensor unit that communicates to the accelerometer and processor for activity recognition. The communication channels may experience packet misses, leading to missing data. Therefore, we use the accelerometer data to recover the missing data from the stretch sensor.

6.2.2 Shoaib et al. [29]. The Shoaib dataset provides accelerometer data for 10 users performing seven activities: {*Biking, walking downstairs, jogging, sitting, stand, walking upstairs, and walk*}. The dataset includes accelerometer sensors at five locations on the body: left pocket, right pocket, wrist, belt, and upper arm. Each accelerometer is sampled at 50 Hz. The authors of the dataset use fixed length segments where each segment contains 200 samples. We follow the same segmentation in evaluating SensorGAN and training the classifier.

Table 2. SensorGAN Generator Architecture for Each Dataset H* Represents the Hidden Layer Number

Datasets/SensorGAN Generator	I/P Layer	H1	H2	H3	O/P Layer	λ_1	λ_2	λ_3
w-HAR	1,240	32	16	8	620	100	100	1
Shoaib	6,000	128	64	32	3,000	200	200	1
PAMAP2	9,216	128	64	32	4,608	200	200	—

The features generated for the Shoaib et al. dataset include the minimum, maximum, DWT, FFT and variance of each accelerometer. Using the features, we train an activity classifier to recognize activities in the Shoaib et al. dataset.

To model the missing data in the Shoaib et al. dataset, we consider one of the five sensors is unavailable at any given time. Then, we use available data from other sensors to recover missing data using SensorGAN. Making one sensor missing at a time results in five missing data scenarios for the Shoaib dataset.

6.2.3 PAMAP2 [28]. The PAMAP2 dataset is one of the most commonly used datasets for designing HAR classifiers. The dataset provides data for nine users performing six activities: {*lying, sitting, standing, walking, running and cycling*}. The data are collected with three accelerometers placed on the wrist of the dominant arm, chest, and the dominant side's ankle. The sampling rate of all three accelerometers is 100 Hz.

The authors of the PAMAP2 dataset recommend fixed length segments where each segment contains 512 samples. This is equivalent to a window length of three to five seconds. For each segment, we generate a subset of the features proposed by the PAMAP2 authors. Specifically, we obtain the mean, maximum, FFT, standard deviation, and absolute integral of each sensor in each window. These features are used to train the baseline activity classifier for the PAMAP2 dataset.

To evaluate the effectiveness of the data generated by SensorGAN, we assume that the data from one of the sensors are unavailable. Then, we use the available data from other sensors to recover the data for the missing sensor. This results in three missing data scenarios for the PAMAP2 dataset.

6.3 SensorGAN Architecture and Training

This section describes the architecture of the generator and discriminator in SensorGAN. The generator for all three datasets consists of a fully connected neural network with three hidden layers. Each fully connected hidden layer uses the tanh activation function while the output layer uses linear activation. The input size and number of neurons for each dataset are different because of the variations in sampling rates and segment length. Table 2 summarizes the number of inputs and number of neurons for each dataset. The input to the generator consists of the available and missing sensor data in a segment and the latent vector. The two are concatenated to form a single input to the generator. The input vector size of each dataset is shown in the second column of the table. The input data from each sensor are normalized linearly to a range of −1 to 1. Next, we determine the number of neurons in each hidden layer through a design space exploration. The goal of the design space exploration is to obtain a generator with the highest performance while minimizing the size and computational cost. At the end of the design space exploration, we obtain the network structures shown in the third to fifth columns of Table 2. Finally, the output layer provides the data recovered for each dataset. The number of outputs are governed by the segment length and sampling rate of each dataset. The discriminator in SensorGAN follows similar fully connected structure as the generator with some changes in the number of inputs. Specifically, the input for the discriminator consists of either observed values or generated data for a given sensor. That is, the discriminator takes data of a single sensor as input at a time. The output in the

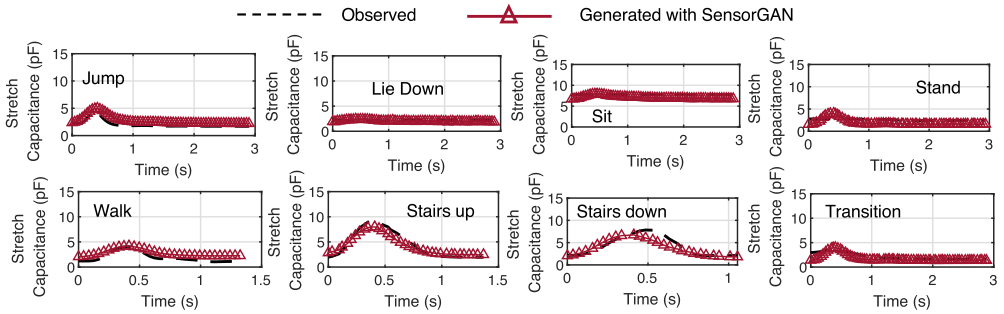

Fig. 4. Comparison of the observed and generated data for each activity in the w-HAR dataset.

Table 3. MSE of Data Generated by SensorGAN

Dataset	MSE	Data Range
w-HAR	9.1	0 to 34
Shoaib	0.1	−2 to 2
PAMAP2	20.2	−145 to 156

discriminator consists of a single neuron with the sigmoid activation function. The output of the discriminator specifies probability of sensor data being either real or generated.

As described in Section 5.5, the generator training includes parameters λ_1, λ_2, and λ_3 as the weights given to each loss function. We determine the values of these parameters via a design space exploration. The final values of λ_1, λ_2, and λ_3 are shown in the right side of Table 2. We note that the feature loss (λ_3) is not used for the PAMAP2 dataset, since the accuracy does not improve significantly with the feature loss.

SensorGAN Training: We implement the SensorGAN training flow in Python using the PyTorch library [26]. We use 600, 150, and 350 epochs of training for w-HAR, Shoaib, and PAMAP2 datasets, respectively. All the SensorGAN training is performed using a server that contains 32 Intel Xeon Gold 6226R cores with 192 GB of memory.

6.4 Accuracy of Generated Sensor Data

We start the experimental validation with an analysis of the accuracy of the data generated by SensorGAN. To this end, Figure 4 shows the observed and generated data for each activity in the w-HAR dataset. The observed data are shown using a black dashed line while the generated data are shown using red lines with triangle markers. We see that the generated data closely matches real-world ground-truth data. In fact, for most of the activities the generated data overlaps with the observed data. This shows that the generator in SensorGAN is able to accurately recover data for all activities. Table 3 summarizes MSE of generated data for all three datasets. Specifically, the table provides average MSE loss for all missing data scenarios in each dataset. Observed data for the missing sensor are used as the ground-truth values in the MSE calculation. The MSE analysis allows us to analyze if the generated data is able to follow observed sensor data.

The table shows that the generated data has an MSE of only 0.1 for the Shoaib et al. dataset. The MSE is slightly higher for the w-HAR and PAMAP2 datasets. This is expected because of the higher range of data values for the two datasets, as shown in the third column of the table. Overall, the data generated using SensorGAN closely follows the real-world sensor data.

Next, we compare the statistical features of the generated data with the observed data. Specifically, we analyze the distribution of the difference ($\|S_f(\text{Original data}) - S_f(\text{Generated data})\|$) for

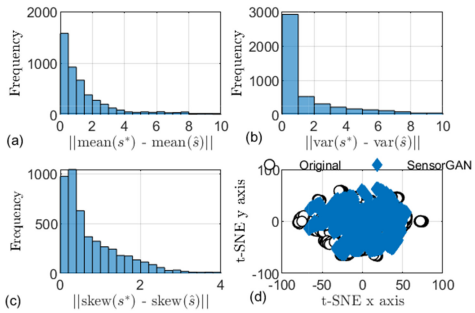

Fig. 5. Distribution of the difference between the statistical features of the generated and the original data using (a) mean, (b) variance, (c) skewness, and (d) t-SNE for the w-HAR dataset when the stretch sensor is missing.

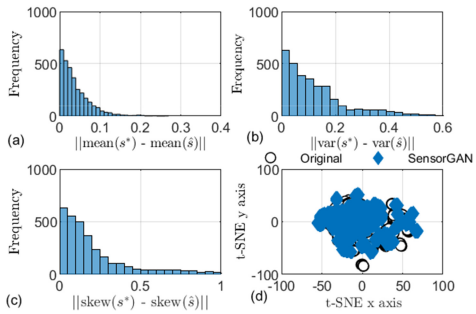

Fig. 6. Distribution of the difference between the statistical features of the generated and the original data using (a) mean, (b) variance, (c) skewness, and (d) t-SNE for the Shoaib et al. when the right pocket sensor is missing.

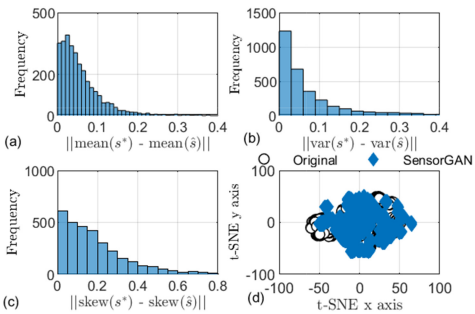

Fig. 7. Distribution of the difference between the statistical features of the generated and the original data using (a) mean, (b) variance, (c) skewness, and (d) t-SNE for the Shoaib et al. dataset when wrist sensor is missing.

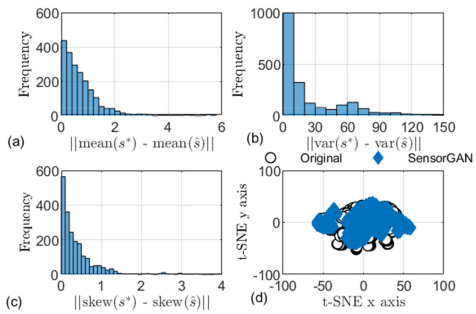

Fig. 8. Distribution of the difference between the statistical features of the generated and the original data using (a) mean, (b) variance, (c) skewness, and (d) t-SNE for the PAMAP2 dataset when the ankle sensor is missing.

the three statistical features, namely, mean, variance, and skewness. The data distribution difference also uses real, observed sensor data as ground-truth values for evaluations. We evaluate the statistical features and their error, since accurate generation of data will lead to accurate feature generation for the classifier. This, in turn, leads to accurate activity classification, as we show later. We also analyze the t-SNE map for the original and generated data. Figures 5–8 show the distribution of the statistical features for the three datasets. For each dataset, we see that the distribution peaks at zero and reduces quickly. This means that the statistical features of the generated data closely match the features of real-world data. Moreover, the t-SNE distribution of the generated data almost overlaps with the t-SNE distribution of observed data, further highlighting the fact that the generated data closely matches real-world data.

6.5 Classification Accuracy with SensorGAN

One of the primary goals of SensorGAN is to ensure that the classification accuracy with the generated data is close to the accuracy with the real-world observed data. This section analyzes effectiveness of data generated by SensorGAN with respect to classification accuracy. We start

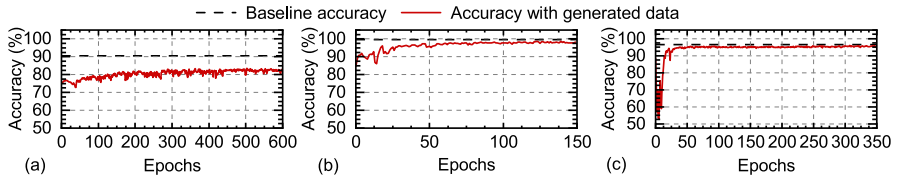

Fig. 9. Classification accuracy during generator training for (a) w-HAR, (b) Shoaib, and (c) PAMAP2 datasets.

Table 4. Confusion Matrix for w-HAR with Actual Data

	Jump	Lie Down	Sit	Stand	Walk	Stairs Up	Stairs Down	Transition
J	**424**	0	0	0	13	2	6	13
L	0	**474**	0	0	0	0	0	0
S	0	0	**660**	28	0	0	0	8
St	0	1	6	**569**	6	0	0	38
W	21	0	1	14	**1904**	1	28	38
SU	6	0	0	0	0	**101**	2	0
SD	7	0	0	0	14	16	**62**	0
T	5	4	12	12	14	1	3	**226**

Table 5. Confusion Matrix for w-HAR with Generated Data

	Jump	Lie Down	Sit	Stand	Walk	Stairs Up	Stairs Down	Transition
J	**420**	0	0	1	16	4	5	12
L	0	**474**	0	0	0	0	0	0
S	0	0	**522**	34	0	0	0	140
St	0	0	138	**398**	4	0	0	80
W	24	0	2	11	**1,904**	2	18	46
SU	5	0	0	0	0	**98**	6	0
SD	7	0	0	0	15	17	**60**	0
T	2	4	52	36	16	2	3	**162**

with a description of the activity classifier used for each dataset. Then, we analyze the accuracy with generated data.

Activity Classifier: We employ a fully connected neural network with two hidden layers as the activity classifier for each dataset. The two hidden layers contain four and eight neurons, respectively. The number of neurons in the output layer is equal to the number of activities in each dataset. The hidden layers use the "ReLU" activation, while the output layers use softmax activation. We train the classifiers with features generated with the original sensor data, i.e., without any missing samples. We use the same classifier to identify activities when the device encounters missing data.

Accuracy with Generator Training: We start with an analysis of the activity accuracy during generator training, as shown in Figure 9. The figure shows the overall activity recognition accuracy while the generator is being trained. The classification accuracy after each epoch of generator training is shown with a red line. We also show the baseline accuracy with the original sensor data using a dashed black line. We observe that the classification accuracy with the generated data is low at the initial stages of generator training. This is expected, since the generator has not learned the distribution of data. The classification loss is then provided as feedback to the generator training algorithm. With this feedback, the accuracy quickly improves. Specifically, for the w-HAR dataset the accuracy improves to 83%, which is within 8% of the baseline accuracy, as shown in Figure 9(a). We see similar improvements in the classification accuracy for the Shoaib et al. and PAMAP2 datasets as well. This shows that the classification loss feedback provided to the generator training algorithm is able to improve the classification accuracy.

Next, we analyze the confusion matrices of activity classification with real-world observed data and generated data. Tables 4 and 5 show the confusion matrices for the w-HAR dataset with the original stretch sensor data and generated data, respectively. The stretch sensor is considered missing in this scenario for accuracy evaluations. We also obtain the confusion matrix when the missing data are filled with zeros and average filling as points of comparison in Tables 6 and 7. The accuracy with zero-filled sensor values offers a useful baseline, because the wearable device will typically observe zero data when sensor data are missing. Similarly, average-filling case provides an additional baseline that may be employed to impute data. We see that using the original data

Table 6. Confusion Matrix for w-HAR with Zeros Data for Missing Sensor

	Jump	Lie Down	Sit	Stand	Walk	Stairs Up	Stairs Down	Transition
J	**411**	1	0	0	33	1	5	7
L	0	**473**	0	1	0	0	0	0
S	0	0	**1**	49	1	0	0	645
St	0	2	1	**524**	11	0	0	82
W	18	0	1	6	**1,947**	0	15	20
SU	3	0	0	0	1	**82**	22	1
SD	8	0	0	0	28	9	**54**	0
T	7	4	0	61	39	1	4	**161**

Table 7. Confusion Matrix for w-HAR with Average Filling for Missing Sensor

	Jump	Lie Down	Sit	Stand	Walk	Stairs Up	Stairs Down	Transition
J	**420**	0	0	1	13	6	5	13
L	0	**474**	0	0	0	0	0	0
S	0	0	**102**	42	1	0	0	551
St	0	1	131	**401**	0	0	0	87
W	31	1	2	16	**1,871**	1	21	64
SU	5	0	0	0	0	**100**	4	0
SD	6	0	0	0	15	17	**61**	0
T	3	4	18	45	14	3	3	**187**

Table 8. Confusion Matrix for the Shoaib et al. with Actual Data

	Biking	Down Stairs	Jogging	Sitting	Stand	Up Stairs	Walk
B	**449**	0	0	0	0	1	0
DS	0	**412**	1	0	2	23	12
J	0	0	**440**	0	0	0	10
S	0	0	0	**450**	0	0	0
St	0	0	0	0	**450**	0	0
US	0	19	0	0	0	**431**	0
W	0	11	0	0	0	0	**439**

Table 9. Confusion Matrix for the Shoaib et al. Wrist Missing Scenario with Generated Data

	Biking	Down Stairs	Jogging	Sitting	Stand	Up Stairs	Walk
B	**425**	0	0	25	0	0	0
DS	0	**383**	0	0	4	25	38
J	0	0	**420**	0	0	0	30
S	0	0	0	**450**	0	0	0
St	0	0	0	0	**450**	0	0
US	0	19	0	0	0	**430**	1
W	0	6	1	0	0	1	**442**

Table 10. Confusion Matrix for the Shoaib et al. Wrist Missing Scenario with Zero-filling

	Biking	Down Stairs	Jogging	Sitting	Stand	Up Stairs	Walk
B	**3**	0	0	444	0	3	0
DS	0	**87**	2	0	0	4	357
J	0	0	**418**	0	0	0	32
S	0	0	0	**450**	0	0	0
St	0	0	0	0	**450**	0	0
US	0	174	0	0	0	**262**	14
W	0	1	428	0	0	0	**21**

Table 11. Confusion Matrix for the Shoaib et al. Wrist Missing Scenario with Average-filling

	Biking	Down Stairs	Jogging	Sitting	Stand	Up Stairs	Walk
B	**144**	0	0	302	0	4	0
DS	0	**202**	3	0	11	32	202
J	0	0	**400**	0	0	0	50
S	0	0	0	**450**	0	0	0
St	0	0	0	0	**450**	0	0
US	0	39	0	0	17	**392**	2
W	0	1	167	0	1	1	**280**

leads to high classification accuracy for all activities. However, when we use the generated data for classification, the accuracy reduces for a few activities. Specifically, the number of correct classifications for sit, stand, and transition are reduced. This is because sit and stand have similar patterns for the sensor data with minor differences in the magnitude. As a result, the generator is unable to accurately recover the data in all cases, especially when the sit pattern of a user is close to the stand pattern of another user. In contrast, classification with zero-filled sensor data experiences significant drop in accuracy. For instance, the classification has an accuracy of zero for sit and majority of sitting activities are classified as transition. This shows that the data generated by SensorGAN enables significant accuracy gains when compared to the default zero-filled data. We observe similar results for average-filling case where majority of sit activities are classified as transition and several stand activities are classified as sit. Overall, the classification accuracy for the w-HAR dataset is within 8% of the accuracy with no missing data while the accuracy with zero-filled data is less than 77%, as shown in Figure 10.

We perform the confusion matrix analysis for the Shoaib et al. as well. Tables 8 and 9 show the confusion matrices for the Shoaib dataset for actual data and for scenario when the wrist data is missing. Data from other four sensors are used to recover the wrist data. Moreover, Tables 10 and 11 show the confusion matrices when the missing data are filled with zeros and average values,

Table 12. PAMAP2 Confusion Matrix with Actual Data

	Lie Down	Sit	Walk	Run	Cycle
L	**347**	25	0	0	2
S	4	**710**	1	0	1
W	3	2	**454**	0	0
R	0	4	1	**179**	1
C	1	2	0	0	**311**

Table 13. PAMAP2 Confusion Matrix with Generated Data

	Lie Down	Sit	Walk	Run	Cycle
L	**351**	23	0	0	0
S	7	**708**	0	0	1
W	11	447	**1**	0	0
R	4	11	5	**150**	15
C	1	8	0	0	**305**

Table 14. PAMAP2 Confusion Matrix with Zeros Data for Missing Sensor

	Lie Down	Sit	Walk	Run	Cycle
L	**12**	17	0	0	345
S	64	**420**	0	0	232
W	295	3	**0**	0	161
R	115	41	1	**2**	26
C	0	0	0	0	**314**

Table 15. PAMAP2 Confusion Matrix with Average Filling for Missing Sensor

	Lie Down	Sit	Walk	Run	Cycle
L	**215**	159	0	0	0
S	0	**715**	0	0	1
W	1	457	**0**	0	1
R	0	165	0	**0**	20
C	1	9	0	0	**304**

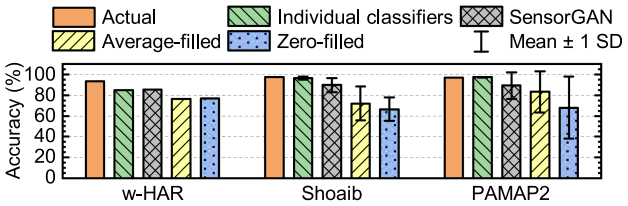

Fig. 10. Comparison of classification accuracies all datasets.

respectively. Similar to the w-HAR dataset, we see that the baseline classifier achieves high accuracy for all the activities. The high accuracy is maintained with the generated data as well, except for stairs up and down activities. This is expected, because stairs up and down have similar patterns of sensor data and features, which can lead to misclassifications. In contrast, the zero-filled sensor data experiences zero accuracy for multiple activities, showing the benefits of the proposed SensorGAN approach. The average-filled case also suffers from low accuracy for biking, walking, and downstairs activities. The overall accuracy of classification with the generated data is 95.23%, which is within 2% of the baseline 97.48% accuracy, while the accuracy with zero-filled and average-filled data drops to 53% and 73%, respectively.

Finally, we compare the confusion matrices with the PAMAP2 dataset in Tables 12, 13, 14, and 15, respectively. The confusion matrices show the case when ankle sensor is missing in the PAMAP2 dataset. Following the recommendation of the PAMAP2 authors, we combine sitting and standing into one activity, because distinguishing between the two requires a sensor on the hip. Similar to the other two datasets, the baseline classifier achieves high accuracy for all activities while the zero-filled and average-filled case have low accuracy. With the generated data, the accuracy reduces for the walking activity, while maintaining similar accuracies for other activities. The accuracy for walking reduces due to high variability of sensor data during walking and the generator is unable to accurately recover data. At the same time, we note that the generator is able to accurately recover data when the other two sensors are missing.

6.5.1 Summary of Classification Accuracy. Figure 10 shows a comparison of accuracy for the three datasets when considering all missing data scenarios. Specifically, the bars for w-HAR show the accuracy when stretch sensor is missing. Bars for Shoaib and PAMAP2 show the average

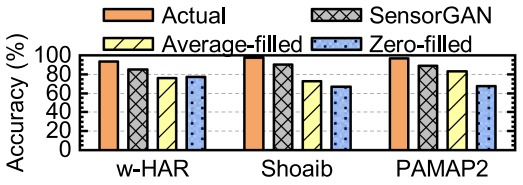

Fig. 11. Comparison of the accuracies all datasets when a missing sensor is chosen randomly in each window.

classification accuracy across all one sensor missing cases. This amounts to five cases for the Shoaib dataset and three cases for the PAMAP2 dataset. The errors bars in the figure show one standard deviation from the mean for Shoaib and PAMAP2 datasets. We see that SensorGAN achieves average accuracy that is close to the actual accuracy for all three datasets. In contrast, accuracy of zero-filled and average-filled cases are lower than SensorGAN. They also have higher standard deviation when compared to SensorGAN, which means that the accuracy has higher variations across missing data scenarios. Sensor-suite specific classifiers cases achieve better accuracy, however, they incur additional context switching overhead and are unable to recover raw data patterns. Overall, SensorGAN is able to achieve average accuracy that is within 10% of the actual accuracy for all three datasets.

6.6 Accuracy with Randomly Missing Sensors

Accuracy evaluations in the previous section assume that a given sensor is missing for all activity windows in a dataset. However, in real-world scenarios, the missing sensor may change for each activity window as energy availability changes across sensors. Therefore, we perform additional accuracy evaluations by randomly choosing a missing sensor in each activity window for all three datasets. After choosing the missing sensor, we either use SensorGAN, zero-filling, or average-filling to recover missing data and perform activity recognition. Figure 11 shows the classification accuracy with random missing sensors for all three datasets. We see that SensorGAN achieves higher accuracy when compared to zero-filling and average-filling scenarios. SensorGAN is also able to achieve accuracy that is within 10% of the accuracy with actual data, thus showing its effectiveness. In summary, SensorGAN is able to handle random missing sensor scenarios and recover classification accuracy.

6.7 Implementation Overhead

The proposed SensorGAN generator is implemented on the Odroid-XU3 board to measure the overhead of data recovery at runtime. We start by implementing a dedicated function for the generator architecture in C. The parameters for the generator are stored on the device in a comma separated value format. The generator function takes available sensor data and missing sensor values as inputs. Then, using stored parameters, the function generates data for the missing sensor. Since our goal is to impute missing data before activity classification, we integrate the function with the HAR processing pipeline, as shown in Figure 1. The integrated HAR processing pipeline is stored as a user-space binary on the Odroid-XU3 board.

At runtime, the HAR processing pipeline is executed as a user-space application on the Odroid-XU3 board. The application starts by loading HAR classifier parameters and generator parameters into memory. Then, the classifier is executed for each window to obtain the user activity. If missing data are detected, the generator function is executed to recover missing data, so that we can perform accurate activity classification.

To understand the implementation overhead of SensorGAN, we first characterize the memory requirements of storing the generator weights on the wearable device. As shown in Table 16, w-HAR

Table 16. Memory Storage of SensorGAN Generator

Datasets	Storage in kB
w-HAR	183
Shoaib	3,509
PAMAP2	5,368

dataset requires 183 kB of memory to implement SensorGAN. As the inputs to the model in Shoaib and PAMAP2 are larger, the memory requirements increase for Shoaib et al. and PAMAP2 datasets. Specifically, Shoaib et al. requires 3,509 kB and PAMAP2 requires 5,368 kB of memory. We also measure the execution time and energy overhead of running the generator function for w-HAR on the Odroid-XU3 board. GNU C timing libraries are used to profile the execution time of the generator in SensorGAN. Specifically, we insert timing functions before and after each invocation of the generator function to measure the execution time. We run the generator function using A7 cores on the Odroid board, because they are closer to processors used in commercial wearable devices. We observe that each invocation of the SensorGAN generator takes about 1.5 ms for w-HAR (average of 1,000 iterations), which results in an energy consumption of 606 μJ. In summary, the SensorGAN architecture is suitable for effectively recovering data in tinyML applications on energy-constrained devices.

7 CONCLUSIONS AND FUTURE WORK

Human activity recognition using wearable devices has the potential to enable a number of interesting applications, such as health monitoring, rehabilitation, and fitness improvement. However, HAR approaches with multiple sensors may suffer from reliability issues if one or more sensors have missing data. To address this limitation, we proposed a novel generative approach, referred to as SensorGAN, to recover the missing data at runtime. Our approach is guided by the fact that we can use the data from available sensors to generate the missing data. The SensorGAN approach also includes additional losses in the generator training to ensure that the recovered data are able to provide accurate activity classification. Experiments with three publicly available datasets show that SensorGAN effectively recovers missing data and achieves accuracy that is within 10% of the classifier accuracy with no missing data. Implementation of SensorGAN on a low-power processor for the w-HAR dataset shows that each invocation of SensorGAN takes about 1.5 ms, which results in an energy consumption of 606 μJ. Our immediate future work includes developing low-overhead algorithms for detecting missing data and the generalized recovery function that can handle all combinations of missing sensor data.

REFERENCES

[1] Sizhe An, Ganapati Bhat, Suat Gumussoy, and Umit Ogras. 2023. Transfer learning for human activity recognition using representational analysis of neural networks. *ACM Trans. Comput. Healthcare* 4, 1, Article 5 (Mar. 2023), 21 pages. https://doi.org/10.1145/3563948

[2] Sizhe An, Yigit Tuncel, Toygun Basaklar, Gokul K. Krishnakumar, Ganapati Bhat, and Umit Y. Ogras. 2021. MGait: Model-based gait analysis using wearable bend and inertial sensors. *ACM Trans. Internet Things* 3, 1 (2021), 1–24.

[3] Muhammad Arif, Mohsin Bilal, Ahmed Kattan, and S. Iqbal Ahamed. 2014. Better physical activity classification using smartphone acceleration sensor. *J. Med. Syst.* 38, 9 (2014), 95.

[4] Ling Bao and Stephen S. Intille. 2004. Activity recognition from user-annotated acceleration data. In *Proceedings of the International Conference on Pervasive Computing*. 1–17.

[5] Ganapati Bhat, Nicholas Tran, Holly Shill, and Umit Y. Ogras. 2020. w-HAR: An activity recognition dataset and framework using low-power wearable devices. *Sensors* 20, 18 (2020), 5356.

[6] Alberto G. Bonomi, Annelies H. C. Goris, Bin Yin, and Klaas R. Westerterp. 2009. Detection of type, duration, and intensity of physical activity using an accelerometer. *Med. Sci. Sports Exerc.* 41, 9 (2009), 1770–1777.

[7] Judit Bort-Roig, Nicholas D. Gilson, Anna Puig-Ribera, Ruth S. Contreras, and Stewart G. Trost. 2014. Measuring and influencing physical activity with smartphone technology: A systematic review. *Sports Med.* 44, 5 (2014), 671–686.
[8] Changhao Chen, Yishu Miao, Chris Xiaoxuan Lu, Linhai Xie, Phil Blunsom, Andrew Markham, and Niki Trigoni. 2019. MotionTransformer: Transferring neural inertial tracking between domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 8009–8016.
[9] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. 2018. Generative adversarial networks: An overview. *IEEE Signal Process. Mag.* 35, 1 (2018), 53–65.
[10] Ton De Waal, Jeroen Pannekoek, and Sander Scholtus. 2011. *Handbook of Statistical Data Editing and Imputation.* Vol. 563. John Wiley & Sons.
[11] Jakob Doppler, Gerald Holl, Alois Ferscha, Marquart Franz, Cornel Klein, Marcos dos Santos Rocha, and Andreas Zeidler. 2009. Variability in foot-worn sensor placement for activity recognition. In *Proceedings of the International Symposium on Wearable Computers*. IEEE, 143–144.
[12] Alberto J. Espay et al. 2016. Technology in Parkinson's disease: Challenges and opportunities. *Movt. Disord.* 31, 9 (2016), 1272–1282.
[13] Dawud Gordon, Hedda Rahel Schmidtke, Michael Beigl, and Georg Von Zengen. 2010. A novel micro-vibration sensor for activity recognition: Potential and limitations. In *Proceedings of the International Symposium on Wearable Computers (ISWC'10)*. IEEE, 1–8.
[14] Zijian Guo, Yiming Wan, and Hao Ye. 2019. A data imputation method for multivariate time series based on generative adversarial network. *Neurocomputing* 360 (2019), 185–197.
[15] Hardkernel. 2014. ODROID-XU3. Retrieved from https://www.hardkernel.com/shop/odroid-xu3/
[16] Tahera Hossain, Md Ahad, Atiqur Rahman, and Sozo Inoue. 2020. A method for sensor-based activity recognition in missing data scenario. *Sensors* 20, 14 (2020), 3811.
[17] Dina Hussein, Taha Belkhouja, Ganapati Bhat, and Janardhan Rao Doppa. 2022. Reliable machine learning for wearable activity monitoring: Novel algorithms and theoretical guarantees. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design (ICCAD'22)*. Association for Computing Machinery.
[18] Dina Hussein, Aaryan Jain, and Ganapati Bhat. 2022. Robust human activity recognition using generative adversarial imputation networks. In *Proceedings of the Design, Automation ,and Test in Europe Conference & Exhibition (DATE'22)*. 84–87.
[19] Kai Kunze and Paul Lukowicz. 2014. Sensor placement variations in wearable activity recognition. *IEEE Pervas. Comput.* 13, 4 (2014), 32–41.
[20] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. 2011. Activity recognition using cell phone accelerometers. *ACM SigKDD Explor. Newslett.* 12, 2 (2011), 74–82.
[21] Oscar D. Lara et al. 2012. A survey on human activity recognition using wearable sensors. *IEEE Commun. Surveys Tut.* 15, 3 (2012), 1192–1209.
[22] Song-Mi Lee, Sang Min Yoon, and Heeryon Cho. 2017. Human activity recognition from accelerometer data using Convolutional neural network. In *Proceedings of the IEEE International Conference on Big Data and Smart Computing (BigComp'17)*. IEEE, 131–134.
[23] Shengzhong Liu, Shuochao Yao, Yifei Huang, Dongxin Liu, Huajie Shao, Yiran Zhao, Jinyang Li, Tianshi Wang, Ruijie Wang, Chaoqi Yang, and Tarek Abdelzaher. 2020. Handling missing sensors in topology-aware IoT applications with gated graph neural network. *Proc. ACM Interact., Mobile, Wear. Ubiq. Technol.* 4, 3 (2020), 1–31.
[24] Walter Maetzler, Josefa Domingos, Karin Srulijes, Joaquim J. Ferreira, and Bastiaan R. Bloem. 2013. Quantitative wearable sensors for objective assessment of parkinson's disease. *Move. Disord.* 28, 12 (2013), 1628–1637.
[25] Abdulmajid Murad and Jae-Young Pyun. 2017. Deep recurrent neural networks for human activity recognition. *Sensors* 17, 11 (2017), 2556.
[26] Adam Paszke et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Info. Process. Syst.* 32 (2019).
[27] Ivan Miguel Pires, Faisal Hussain, Nuno M. Garcia, and Eftim Zdravevski. 2020. Improving human activity monitoring by imputation of missing sensory data: Experimental study. *Future Internet* 12, 9 (2020), 155.
[28] Attila Reiss and Didier Stricker. 2012. Introducing a new benchmarked dataset for activity monitoring. In *Proceedings of the 16th International Symposium on Wearable Computers*. IEEE, 108–109.
[29] Muhammad Shoaib et al. 2014. Fusion of smartphone motion sensors for physical activity recognition. *Sensors* 14, 6 (2014), 10146–10176.
[30] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul J. M. Havinga. 2015. A survey of online activity recognition using mobile phones. *Sensors* 15, 1 (2015), 2059–2085.
[31] Kaleb E. Smith and Anthony O. Smith. 2020. Conditional GAN for timeseries generation. Retrieved from https://arXiv:2006.16477

[32] Christina Strohrmann, Holger Harms, and Gerhard Troster. 2011. What do sensors know about your running performance? In *Proceedings of the 15th Annual International Symposium on Wearable Computers*. IEEE, 101–104.
[33] Amarnag Subramanya, Alvin Raj, Jeff A. Bilmes, and Dieter Fox. 2012. Recognizing activities and spatial context using wearable sensors. Retrieved from https://arXiv:1206.6869
[34] Laurens Van Der Maaten. 2014. Accelerating t-SNE using tree-based algorithms. *J. Mach. Learn. Res.* 15, 1 (2014), 3221–3245.
[35] Aiguo Wang, Guilin Chen, Jing Yang, Shenghui Zhao, and Chih-Yung Chang. 2016. A comparative study on human activity recognition using inertial sensors in a smartphone. *IEEE Sensors J.* 16, 11 (2016), 4566–4578.
[36] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. 2017. DualGAN: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE International Conference on Computer Vision*. 2849–2857.
[37] Jinsung Yoon, James Jordon, and Mihaela Schaar. 2018. GAIN: Missing data imputation using generative adversarial nets. In *Proceedings of the International Conference on Machine Learning*. PMLR, 5689–5698.
[38] Yinhe Zheng, Guanyi Chen, and Minlie Huang. 2020. Out-of-domain detection for natural language understanding in dialog systems. *IEEE/ACM Trans. Audio, Speech, Lang. Process.* 28 (2020), 1198–1209.
[39] Muhammad Zubair, Kibong Song, and Changwoo Yoon. 2016. Human activity recognition using wearable accelerometer sensors. In *Proceedings of the IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. IEEE, 1–5.

Received 2 September 2022; revised 30 April 2023; accepted 4 June 2023