Synthetic Sensor Data for Human Activity Recognition

Fayez Alharbi
Department of Computing
Goldsmiths, University of London
London, United Kingdom
E-mail: falha011@gold.ac.uk

Lahcen Ouarbya
Department of Computing
Goldsmiths, University of London
London, United Kingdom
E-mail: l.ouarbya@gold.ac.uk

Jamie A Ward
Department of Computing
Goldsmiths, University of London
London, United Kingdom
E-mail: j.ward@gold.ac.uk

Abstract— **Human activity recognition (HAR) based on wearable sensors has emerged as an active topic of research in machine learning and human behavior analysis because of its applications in several fields, including health, security and surveillance, and remote monitoring. Machine learning algorithms are frequently applied in HAR systems to learn from labeled sensor data. The effectiveness of these algorithms generally relies on having access to lots of accurately labeled training data. But labeled data for HAR is hard to come by and is often heavily imbalanced in favor of one or other dominant classes, which in turn leads to poor recognition performance.**

In this study we introduce a generative adversarial network (GAN)-based approach for HAR that we use to automatically synthesize balanced and realistic sensor data. GANs are robust generative networks, typically used to create synthetic images that cannot be distinguished from real images. Here we explore and construct a model for generating several types of human activity sensor data using a Wasserstein GAN (WGAN). We assess the synthetic data using two commonly-used classifier models, Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). We evaluate the quality and diversity of the synthetic data by training on synthetic data and testing on real sensor data, and vice versa. We then use synthetic sensor data to oversample the imbalanced training set. We demonstrate the efficacy of the proposed method on two publicly available human activity datasets, the Sussex-Huawei Locomotion (SHL) and Smoking Activity Dataset (SAD). We achieve improvements of using WGAN augmented training data over the imbalanced case, for both SHL (0.85 to 0.95 F1-score), and for SAD (0.70 to 0.77 F1-score) when using a CNN activity classifier.

Keywords—human activity recognition, wasserstein generative adversarial network, imbalance sensor data

I. INTRODUCTION

A. HAR Background

Due to the prevalent use of mobile devices with built-in inertial measurement unit (IMU) sensors, like in smartphones, wearables and other on-body devices [1], human activity recognition (HAR) has gained extensive attention and plays a significant role in several domains, including human–computer interaction (HCI) [2], mobile and ubiquitous computing [3] [4], and human behavior analysis [5]. HAR has empowered researchers to better understand and analyze individual behavior [6]. Thus, HAR has been used in numerous applications in people's daily lives in various areas such as healthcare, elderly monitoring, physical therapy, fitness trackers, and sleep quality monitors [7].

HAR is typically considered a pattern recognition system and uses machine learning methods to acquire useful representations of sensor data that correspond to feature extractions so that the characteristics of pre-defined activities are well recognized by machine learning methods [2].

Human activity data can be described as a multivariate time-series, with further categorical divisions according to the duration and complexity of activities. In this work we consider three categories of activity: hand-to-mouth (HMG), basic, and transportation.

The first category of activity, HMG, is characterized by short durations, less-repetitive and activities frequently confound with each other because of their similar gestures. For example, drinking, eating, smoking while drinking or talking can present similar hand movement.

The second category of activity that we consider are basic activities. Basic activities are characterized by a long interval and come in two forms: static or dynamic. For example, standing still and sitting are static and less-repetitive activities, but running and walking are dynamic and repetitive activities.

The third category that we consider are transportation activities. The period of activity is also long, such as on a bus or on a train where the subject might be sitting or standing.

HAR has been extensively studied in fields like HCI, mobile and ubiquitous computing and human behavior analysis [8]. Traditional machine learning methods are commonly used, such as K-nearest neighbor (KNN), support vector machines (SVM), and decision trees (DT) [8]. In recent years, deep learning methods such as a convolutional neural network (CNN), recurrent neural network (RNN) and Long short-term memory (LSTM) has been used to classify human activities using sensor data and have achieved favorable recognition performance [2] [9]. Since LSTMs in [10] and [11] and 1-D CNN in [12] and [13] have been successful in recognizing activity from raw sensor data and were implemented in both [14] and [15] for sensor data generation, we adopted them for this study.

Traditional machine learning methods depend on handcrafted features. In order to achieve desirable results when using those methods, more feature extraction techniques must be explored to find well-designed and hand-crafted features [16]. Plenty of effort has been devoted to study and design effective features to enhance HAR performance [2] [17]. In contrast, deep-learning methods are capable of automatically learning feature representation and extracting features directly from the sensor data [9].

Consequently, deep learning classification models have been introduced to recognize human activities and replace hand-crafted features, greatly improving the performance of HAR [18].

B. HAR Dataset Challenges

HAR research relies entirely on the amount and the quality of the collected sensor data. Sensor data quality is mostly imperfect and is often with missing data. This occurs due to several factors, such as an individual not wearing a sensor, or a sensor is malfunctioning [19]. Likewise, the sensor data may be extremely imbalanced due to huge individual differences, with limited labels for some activities [7]. Thus, collecting a large enough sample of sensor data of human activity can effectively enhance the performance of HAR models. In this paper we use GANs to produce synthetic data for several types of human activity and we use this data to rebalance the training set.

The overall performance of HAR classifiers can be negatively impacted by a large class imbalance, where classifiers can be skewed towards performing well on the dominant class and less on the minority class.

To overcome the negative impact of imbalanced training data, different approaches have been employed, such as data level resampling. Here data is resampled to make the data more balanced. The Synthetic Minority Over-sampling Technique (SMOTE) was proposed to create synthetic data points from the minority class in a training set [20].

SMOTE defines a neighborhood for each data point of the minority class by identifying its k nearest neighbors. Then SMOTE employs these neighbors to create synthetic data samples using an interpolation of those neighbors [20].

SMOTE was previously used to fix data imbalance in HAR, and was shown to improve classifier performance [5].

The success of extracting appropriate features, or applying deep-learning methods to automatically find features from sensor data, relies on having access to a large quantity of labeled sensor data [21]. However, collecting and labeling such a large amount of sensor data is both difficult and time consuming. As a result of these limitations, some inevitable challenges appear, such as insufficient sample information as well as imbalanced data.

Recently, several data generation approaches have emerged based on deep neural networks. The generative adversarial network (GAN) is the most powerful method that has attracted much interest to generate synthetic data. GAN was introduced by Goodfellow [22] to generate images. GAN is learned by competition between two neural network models. The two neural network models are known as the generator and the discriminator. The generator model during the learning process is used to produce new data samples by capturing the distribution of the real data, and the discriminator model is employed to distinguish whether data samples are real or synthetic.

GANs have largely been used to produce synthetic samples in several applications, such as image synthesis [23] and text generation [24]. Yet, few works have been done to develop GANs models for the aim of producing sensor data. SenseGen [14] was the first effort at using GANs to synthesize sensor data. However, the proposed model trained both the generator and the discriminator separately. Subsequently, during the training process, the generator did not learn from the feedback of the discriminator. Recently, researchers have developed a model called SensoryGAN for generating sensor data [15]. SensoryGAN models are capable of capturing the distribution of the original sensor data of human activity, consequently enabling them to generate synthetic sensor data. Yet, SensoryGAN suffers from instability while training.

In this work we use an extended variation of GAN, called the Wasserstein Generative Adversarial Network (WGAN), which has been shown to improve stability when training generator and discriminator networks [25]. We focus on generating synthetic sensor data based on the idea of generative adversarial model and evaluating the quality of the synthetic sensor data using a supervised classifier. This study attempts to shed light on using WGAN for synthesizing sensor data.

An increasing number of studies on HAR attempt to develop and optimize activity recognition models using deep learning [18]. However, there has been little attention paid to investigate the potential of using GANs to both create synthetic sensor data and to rebalance the training data using synthetic data in HAR.

Existing over-sampling methods usually work from training data features (typically extracted by applying a sliding window over the raw data), but in cases where the input is raw sensor data, over-sampling methods might not fully consider the temporal dependencies.

Given that sensor data is multivariate time-series and since the raw data used as input in our study, using GAN approaches based on Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) will preserve the temporal dependencies in the training data [26]. As far we know, no study considered before augmenting the training data for human activities context using GAN approaches, therefore, this study explored on adopting a stable method and demonstrated the effectiveness of GAN approach as an up-sampling method for imbalanced human activity training data.

In particular, we investigate whether it is possible to generate sensor data by applying the WGAN method to several forms of human activity, and to augment real training data with WGAN-generated data. We explore reducing the impact of data imbalance by using WGAN to generate synthetic data of the minority class and examine if the supervised model's performance improves.

The main contributions of this study are the following:

- We explore, and assess, the potential of using WGAN to generate synthetic multimodal sensor data of various activities.
- We built two supervised classification models (1D CNN and LSTM) to validate that the synthetic data preserves the underlying pattern as well as the structure of the original data.
- We resample the imbalanced training data with synthetic data and show how this can be used to improve classifier performance.

The paper is structured as follows. Section II gives a brief overview of GAN. Section III describes our method; we describe the two proposed WGN models to generate synthetic sensor data. Also, we describe the classifiers to evaluate the quality of the produced sensor data as well as the activity

recognition models. Section IV explains the experimental setup in our study. Section V presents the results, and Section VI discussed our findings and presents the conclusions.

II. GENERATIVE ADVERSARIAL NETWORK

GAN is based on the game theory concept of a *minimax game*, where two networks, the generator (G) and the discriminator (D), are trained in an adversarial fashion [27]. The objective of G is to generate synthetic data that D would be unable to differentiate from real data. Contrarily, the aim of D is to distinguish real data from generated synthetic data. Consequently, the objective function of GAN is identified as:

$$\min_G \max_D \; E_{x\sim P_r}[\log(D(x))] + E_{z\sim P_z}[\log(1 - D(G(z)))] \quad (1)$$

Where x is real data, z is sampled data form random noise, such as Gaussian distribution or uniform distribution [28] P_r is the real data distribution and P_z is the generated data distribution.

Kullback–Leibler (KL) divergence and Jensen–Shannon (JS) divergence [28] are important probability measurement metrics that GAN uses when the discriminator is optimized. Those metrics estimate the distribution distance between the real samples and the produced samples. Mode collapse is the problem that constrains the capability of the generator model, which occurs by only allowing the generator models to generate a partial range of samples of the original data distribution. GAN suffers from mode collapse, and this potential limitation leads to learning instability. A possible source of mode collapse is because of the use of KL in GAN training [25].

To overcome mode collapse, the authors in [25] proposed the Wasserstein GAN (WGAN), which uses the Wasserstein distance instead of KL to measures the distance between the original sample and the created sample. WGAN enhances the stability of learning and overcomes the difficulty of mode collapse. The Wasserstein distance is defined as :

$$W(P_r, P_g) = \inf_{\gamma\in\Pi(P_r,P_g)} E_{(x,y)\sim\gamma}\,[\|x - y\|] \quad (2)$$

where $\Pi(P_r, P_g)$ represents the set of all joint distributions $\gamma(x, y)$ and the distance to transform the distribution P_r into the distribution P_g is represented by $\gamma(x, y)$. Because the Wasserstein distance is intractable in practice, the Kantorovich-Rubinstein duality can be used instead as an approximation [25]:

$$W(P_r, P_\theta) = \sup_{\|D\|_L \le 1} \mathrm{E}\,[D(x) - D(g_\theta(z))] \quad (3)$$

where sup is the least upper bound and D is the set of Lipschitz continuous functions that follow this constraint:

$$|D(x_1) - D(x_2)| \le |x_1 - x_2| \quad (4)$$

The WGAN objective is obtained as:

$$\min_G \max_{D\in L} \mathrm{E}_{x\sim P_r}[D(x)] - \mathrm{E}_{\tilde{x}\sim P_g} D(G(\tilde{x}))] \quad (5)$$

In order to apply the Lipschitz constraint on the discriminator, which is also called the critic in WGAN, the authors suggest implementing the parameter to clip the weights of the discriminator. The weights of the discriminator have to be within a specific range [-c, c], where c is controlled hyperparameters [25].

A major variance between WGAN and original GAN is the role of D [28]. The D purpose in GAN is applied as a binary classifier, which differentiates between real and generated samples. However, the function of D in WGAN is to estimate the Wasserstein distance between the generated and the actual data distribution, which is a regression task. Hence, in the last layer of D, in the WGAN, the sigmoid function is eliminated.

III. METHOD

A. Data Processing

In our study we consider two different types of input data to evaluate our proposed models: raw input (e.g. direct accelerometer or gyroscope derived readings), and feature data (extracted handcrafted features from the raw data, such as the mean over a sliding window).

Fig.1 shows the pre-processing pipeline for each of the two types of data. For both, the first step is to low-pass filter the data using a 3rd-order Butterworth filter. We then calculate the root-sum-squared magnitude ($\sqrt{x^2 + y^2 + z^2}$) for each 3-axis sensor to ensure the data is invariant to shifting orientation of the smartphones. The data is then segmented into non-overlapping windows, or frames. For the raw data, each frame is a matrix of size: length of the window × the number of sensor channels.

Five features are calculated over each frame: *mean*, *standard deviation*, *minimum*, *maximum*, and *zero crossing rate*. These features are computationally cheap and proven to be effective for HAR [29]. Each frame is then a vector of size: number of extracted features × number of sensor channels.

Both raw and feature data is then scaled using min-max normalization [30].

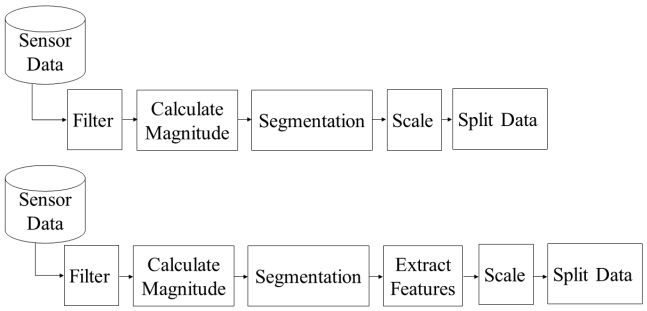

Figure 1. PIPELINES FOR RAW (TOP) & FEATURES (BOTTOM)

Finally, the dataset is split into training, validation and testing sets (70 % for training, 15 % for validation and 15 % for testing), using the stratified split data method from scikit-learn [31]. This method balances the number of data samples of the classes in each split. We used Python [31] and Keras [32] to implement our models.

B. WGAN

Human activities are heterogeneous [2], therefore a unified WGAN model might not be enough to learn several distributions of different human activities. To counter this, we build two different types of activity-specific WGAN model, one designed for relatively HMG (e.g. smoking while in a group conversation), plus static activities lasting a relatively long time (e.g. sitting), and another for more dynamic, short-term activities (e.g. running).

We fine-tuned a WGAN model on each activity class to find suitable layers of the generator and the discriminator, the dimension of the noise vector, learning rate, and epochs. The hyperparameters for each model were obtained over a number

of trials, validated using the validation set.

Figure 2 shows the two models we use. Model-1 has a generator based on one LSTM layer with 25 memory cells and uses a Tanh activation on its output. The generator's responsibility is to generate data from the noise data that has a similar structure to the real sensor data.

The discriminator has a single 1D CNN layer using 10 filters with ReLU activation function and a dense layer with Tanh activation function. The output layer has a single neuron without an activation function. The discriminator's responsibility is to predict if its input is real or not based on its Wasserstein distance.

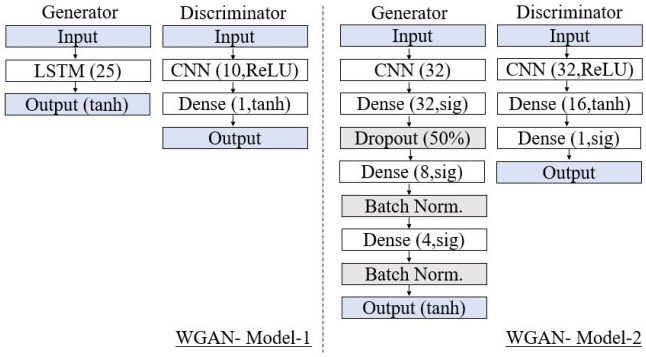

Figure 2. WGAN MODEL-1 (left) AND MODEL-2 (right)

Model-2, by contrast, has a generator built based on a 1D CNN with 32 filters. The model utilized dense layer with 32 units and sigmoid activation function. We applied dropout with a rate of 50% and a dense layer with 8 units that used sigmoid activation function. We then added a batch normalization layer and a dense layer with 4 neurons, which applied sigmoid activation function. We again applied batch normalization layer. The output layer of the generator was dense with the Tanh as activation function.

The discriminator used 1D CNN of 32 kernels with ReLU activation and a dense layer of 16 units with Tanh activation function. We also added a dense layer of one unit with sigmoid activation function. The output layer is a further dense layer of one neuron, but without an activation function.

C. Assessing Synthetic Sensor Data

To evaluate the synthetic sensor data, we used the GAN-train and GAN-test methods [33]. GAN-train involves training on synthetic sensor data but testing on real sensor data. A high performance with this reveals that the GAN is capable of producing a realistic and diverse output, and is consequently not suffering from mode collapse. Conversely, GAN-test is trained on real data and tested on synthesized data. This gives a complementary measure of synthesized data quality [33].

We evaluate two commonly used classifiers: 1D CNN and LSTM. CNNs are formed by stacking several processing units, including convolutional layers, pooling layers, and fully connected (dense) layers [34]. These stacked layers enable CNNs to extract features automatically from raw sensor data. As a comparison we also evaluate synthetic sensor data in a dynamic RNN-based model. We used LSTM which can learn long-term dependencies by using a memory cell that is comprised of an input gate, output gate, and forget gate. LSTM is specifically designed to model temporal dynamics in sequences such as sensor data [35].

We used categorical cross-entropy as the loss function for training both 1D CNN and LSTM classifiers. The training hyperparameters, including the number of epochs, learning rates, and optimizer functions, differed between datasets and classification tasks. When evaluating classes with a limited number of samples, for example, the number of epochs had to be limited to avoid overfitting. Some hand-tuning of these hyper parameters was therefore required.

1) 1D CNN Supervised Model

Figure 3 shows the CNN layout for *n* sensor streams. Each individual input sensor, such as accelerometer magnitude or extracted features from accelerator magnitude, is first passed to a single 1D CNN layer. The first layer uses 9 filters with ReLU activation function. A dropout layer is then added with a rate of 50%. We also implemented a max-pooling layer (with a kernel of 2). The output of each subnet is then flattened, concatenated, and passed to a dense layer with 15 units with ReLU activation functions. The output SoftMax activation layer is finally used for classification [36].

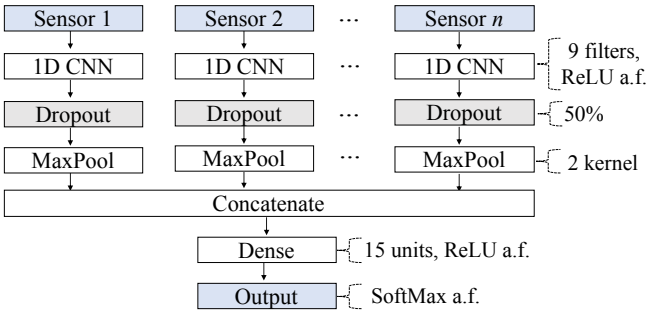

Figure 3. CNN MODEL ARCHITECTURE

2) LSTM Supervised Model

Figure 4 shows the LSTM stacked layers in the second classification model. Each sensor stream is also independently processed to extract features and to capture longer temporary patterns. The LSTM layer uses 15 units and a Tanh activation function. We apply a dropout layer with a rate of 10%. Another layer of LSTM has 10 units and a Tanh activation function. The patterns from the individual pipelines are then concatenated together. We use a final dropout layer with a rate of 50%, and a dense layer with 8 neurons and a ReLU activation function. Finally, the output layer uses a SoftMax activation function.

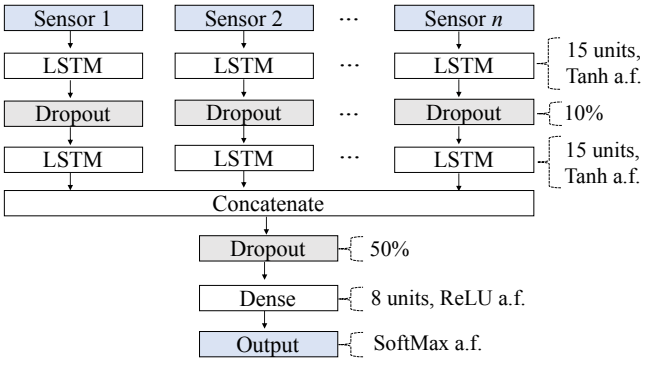

Figure 4. LSTM MODEL ARCHITECTURE

D. Oversampling Training Set with Synthetic Sensor Data

After we generate and evaluate the quality of the synthetic sensor data produced by WGAN, we use it to oversample the minority activity in the training set. We then evaluate the entire oversampled dataset using the two classifiers, CNN and LSTM. As a baseline comparison, we also run these classifiers using the original, imbalanced, data.

E. Extracting Features from Synthetic Sensor Data

We extract features (mean, standard deviation, minimum, maximum, and zero crossing rate) from the synthetic sensor data of the minority class and use these to oversample the activity that is least represented in the training set. We refer to this as the WGAN-Features method.

To investigate the efficiency of WGAN-Features, we compare it to a commonly used oversampling method, SMOTE. SMOTE is used to mitigate the problems caused by imbalanced training data by oversampling classes that are less well represented. First, SMOTE selects a random data sample from a minority activity and determines k nearest neighbors for that data sample, typically k = 5. Then, an arbitrarily chosen neighbor is selected, and a synthetic data sample is generated at a randomly selected point on the line connecting the two data samples in feature space [20].

The study makes three evaluations: an evaluation of WGAN-Features, an evaluation using SMOTE, and a baseline evaluation using features calculated from the original data. As before, we use CNN and LSTM classifiers.

F. Performance Measures

Typically, accuracy is used as a measure of classifier performance, however, in the case where a dataset is imbalanced, accuracy is unsuitable as it is skewed towards more common classes [37]. To counter this, we use precision and recall for each class, and then the weighted mean of these over all classes (the F1 score) [38]. *Precision* records the proportion of class predictions that are correct, and *Recall* records the proportion of actual class instances that are correct. The balanced F1 score used here treats classes equally, irrespective of how frequently a class appears:

$$\text{F1 score} = \frac{1}{m}\sum_{i=1}^{classes}\frac{2 * Precison_i * Recall_i}{Precison_i + Recall_i}$$

IV. Experiment Setup

A. Datasets

Two public datasets are used in this study: the Sussex-Huawei Locomotion (SHL) [39], and the Smoking Activities Dataset (SAD) [40].

SHL records real-life activities of three subjects over three days, and includes eight different locomotion and transportation activities, including *walk*, *run*, *still*, *bike*, *car*, *bus*, *subway*, and *train*. The subjects carried four smartphones at four locations (hand, hip, torso, and bag). Of the 16 recorded sensor modalities, we use 6: accelerometer, gyroscope, magnetometer, linear acceleration, orientation, and gravity, from both hand and hip. We use a subset of the dataset from the days where the same activities are performed by subjects one and three. These activities are *walk*, *run*, *still*, *bike*, and *bus*. Figure 5 (left) shows the proportion of each class in the dataset (with, e.g., *run* making up 3% of samples).

SAD was collected from 11 participants over 3 months. Each participant wore a smartwatch on the right wrist as well as a smartphone in the right pocket to capture data. These were embedded with accelerometers and gyroscopes, which were the sensors adopted in this study.

The SAD dataset is divided into three subsets according to the activities performed. Here we use only one of these (Subset 2) because it includes more activities and participants Figure 5 (right) shows the imbalanced activity distributions for SAD. Participants performed 8 activities, divided into complex and simple. Complex activities include: smoking while standing (*SmokeST*), smoking while sitting (*SmokeSD*), smoking while in a group conversation (*SmokeGP*), drinking while standing (*DrinkST*), drinking while sitting (*DrinkSD*). The simple activities are *stand*, *sit*, and *eat*.

Both datasets were preprocessed using the pipeline described above. The SHL data is low-pass filtered, using a 3rd-order Butterworth filter with a corner frequency 20Hz. This was then segmented using a 3s window [41]. The SHL dataset is sampled at 100Hz, the length of the window size was 3 seconds, and there are 12 sensor channels, so the raw matrix size for each frame is (300,12). When we extract feature from each frame the shape of the vector becomes (5,12). The total dataset size is 15280 frames (~13 hours).

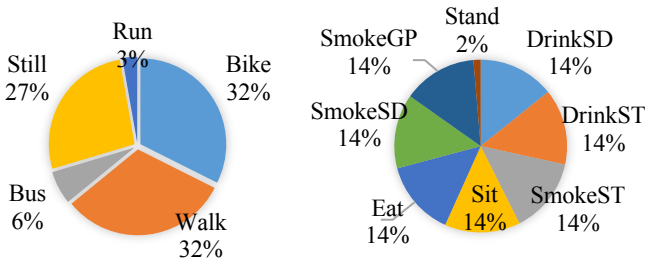

Figure 5. DISTRIBUTION FOR SHL (LEFT) AND SAD (RIGHT)

The SAD dataset is sampled at 50 Hz. We adjust the low-pass filter with a corner frequency of 10Hz, with segment windows of 9s (to capture the longer-term complexities of activities). Given the 4 sensor channels, each raw data frame is sized (450, 4). The feature-extracted vector is then (5,4). The total dataset size for SAD is 11776 frames (~30 hours).

B. Evaluation Setup

We evaluate our approach in three stages: first we generate and evaluate the quality and diversity of the synthetic data. Then we evaluate the raw synthetic data when used to oversample imbalanced datasets. Finally, we evaluate the synthetic data when it is converted into features.

1) Evaluation of Synthetic Data

We assess the quality and diversity of our synthetic data in two ways: by using generated data to train our classifiers, which we then evaluate using real test data (GAN-train), and using real data to train, which we then evaluate on generated data (GAN-test). Because we are interested in producing the best quality data for each class, separate WGAN models were fine-tuned to match each minority class of interest. For relatively long-term static data in both datasets, like *still* and *bus* (in SHL), and s*mokeGP* and s*tand* (in SAD), the WGAN Model-1 worked best. The faster-changing data of *run* (SHL) was better characterized using Model-2. TABLE 1 and TABLE 2 show the parameters used for each class model.

The SHL activity generator produced 100 frames of synthetic sensor data for *bus*, *still*, and *run*. The SAD activity generator produced 50 frames of synthetic data for two activities: *smokeGP* and *stand*.

TABLE 1. **SHL** HYPERPARAMETERS FOR WGAN MODELS

Activity	Noise Vector	Learning Rate	Epochs	WGAN Model
Bus	10	0.0005	1000	1
Run	5	0.03	1000	2
Still	10	0.0005	1000	1

TABLE 2. **SAD** HYPERPARAMETERS FOR WGAN MODELS

Activity	Noise Vector	Learning Rate	Epochs	WGAN Model
SmokeGP	10	0.0005	1000	1
Stand	10	0.0005	1000	1

TABLE 3. HYPERPARAMETERS FOR MODELS ASSESSING THE QUALITY OF SYNTHETIC DATA FOR BOTH DATATSETS

Classifier	1D – CNN	LSTM
Optimizer	SGD	ADAM
Learning Rate	0.00001	0.0001
Epochs	15	15

Once generated the data is evaluated using the two classifiers, 1D-CNN and LSTM, with the respective hyperparameters shown in TABLE 3.

2) Raw Data Oversampling Evaluation

We evaluate how our method might be used in a real-world situation. We use the WGAN models to oversample each minority activity in the training set (*run* in SHL, *stand* in SAD), and we use the new, oversampled, datasets to compare classifier performance. As a baseline, we also calculate the performances without oversampling.

3) Feature Data Oversampling Evaluation

We evaluate our oversampling method when applied to features extracted from the raw data. We compare three approaches. First, we extract features from the oversampled synthetic sensor data produced by our WGAN method (we call these WGAN-Features). Second, we compare these against those obtained using the, state-of-the-art, SMOTE method. Third, we evaluate a baseline using features from the imbalanced dataset.

C. Classifier Setup

Both the raw data and feature data oversampling evaluations are carried out using 1D-CNN and LSTM classifiers. TABLE 4 and TABLE 5 show the hyperparameters for the classification models on SHL data and SAD, respectively.

TABLE 4. **SHL** HYPERPARAMETERS FOR CLASSIFICATION

Classifier	1D - CNN		LSTM	
Input data	Raw Input	Feature Input	Raw Input	Feature Input
Optimizer	SGD	SGD	ADAM	ADAM
Learning Rate	0.001	0.001	0.001	0.0001
Epochs	20	50	35	50

TABLE 5. **SAD** HYPERPARAMETERS FOR CLASSIFICATION

Classifier	1D - CNN		LSTM	
Input data	Raw Input	Feature Input	Raw Input	Feature Input
Optimizer	SGD	SGD	ADAM	ADAM
Learning Rate	0. 0001	0.01	0.001	0.001
Epochs	20	100	50	100

V. RESULTS

A. Evaluating the Synthetic Data

TABLE 6 and TABLE 7 show the GAN-train classifier results used to assess the diversity and quality of the new sensor samples (trained on synthetic, tested on real). On the SHL data, the F1 score using 1D-CNN was 0.76, but for LSTM it was 0.59. On the SAD data, the F1 score for 1D-CNN was 0.75, and for LSTM it was 0.99. The equivalent GAN-test results, measuring how well the synthesized samples match the real distributions, all returned perfect F1 scores (1.00) across all datasets and classes. These results reveal that the characteristics of the synthesized data strongly match real data, and that the samples are relatively diverse – with the exception of the *run* and *still* classes when using LSTM.

TABLE 6. CLASSIFIER PERFORMANCE FOR GAN-TRAIN (**SHL**)

Classifier	1D - CNN		LSTM	
Activity	Recall	Precision	Recall	Precision
Bus	0.98	0.87	0.99	0.63
Run	0.95	0.36	0.75	0.43
Still	0.69	1.00	0.29	1.00
F1 Score	0.76		0.59	

TABLE 7. CLASSIFIER PERFORMANCE FOR GAN-TRAIN (**SAD**)

Classifier	1D - CNN		LSTM	
Activity	Recall	Precision	Recall	Precision
SmokeGP	0.91	0.98	1.00	1.00
Stand	0.77	0.44	0.97	1.00
F1 Score	0.75		0.99	

B. Rebalancing the Training Set with Raw Data

TABLE 8 shows the SHL dataset results for CNN using raw sensor data. The baseline F1 for this is 0.85, which increases to 0.95 after oversampling the minority class, *run*. Note that, by adding 100 synthetic samples of *run* to the training set, the recall of this class rises dramatically from 0.20 to 0.71. Similarly, on the SAD results, shown in

Table 9, oversampling the minority class, *stand,* creates a jump in *stand's* recall from 0.25 to 0.83, and a 7% increase in overall F1 score. This is achieved by adding only 50 new samples.

TABLE 10 and TABLE 11 show the equivalent results for each dataset when the LSTM classifier is used. Unlike the CNN case, no improvement is had here from oversampling the minority classes. However, here LSTM performs much better on the baseline cases than CNN.

C. Rebalancing the Training Set with Features

TABLE 12 includes the results obtained using CNN on features extracted from SHL data. The baseline (no oversampling) F1 score is 0.93. By oversampling the minority *run* activity using SMOTE, the F1 falls to 0.92. But using WGAN-Features, the F1 rises to 0.94. The equivalent results, shown in TABLE 13, for SAD, where *stand* is oversampled, reveal an F1 increase from 0.88 (baseline), through 0.93 (SMOTE), to 0.94 for WGAN-Features.

The LSTM results, however, are not improved by oversampling. On the SHL dataset, shown in TABLE 14, WGAN-Features is equivalent to the baseline (0.89 F1), with SMOTE showing a marginal improvement (to 0.90 F1). With SAD, displayed in TABLE 15, again WGAN-Features is

equivalent to baseline (0.95 F1), while SMOTE performs slightly worse (0.93 F1).

TABLE 8. **CNN** PERFORMANCE ON **SHL** DATA, COMPARING BASELINE VS. WGAN OVERSAMPLING.

Activity	Baseline		Oversampled with WGAN	
	Recall	Precision	Recall	Precision
Bike	0.99	0.93	0.98	0.97
Bus	0.96	0.99	0.96	0.99
Run	0.20	1.00	**0.71**	0.97
Still	1.00	1.00	1.00	1.00
Walking	0.99	0.95	0.99	0.94
F1 Score	0.85		**0.95**	

TABLE 9. **CNN** PERFORMANCE ON **SAD**, COMPARING BASELINE VS. WGAN OVERSAMPLING.

Activity	Baseline		Oversampled with WGAN	
	Recall	Precision	Recall	Precision
DrinkSD	0.88	0.67	0.88	0.73
DrinkST	0.77	0.73	0.83	0.74
Eat	*0.29*	*0.59*	*0.29*	*0.62*
Sit	0.96	1.00	0.96	1.00
SmokeGP	0.78	0.67	0.79	0.66
SmokeSD	0.85	0.89	0.85	0.90
SmokeST	0.77	0.69	0.76	0.69
Stand	0.25	1.00	**0.83**	1.00
F1 Score	0.70		**0.77**	

TABLE 10. **LSTM** PERFORMANCE ON **SHL** DATA, COMPARING CLASSIFER FOR BASELINE VS. WGAN OVERSAMPLING

Activity	Baseline		Oversampled with WGAN	
	Recall	Precision	Recall	Precision
Bike	0.99	0.99	0.98	0.99
Bus	0.99	0.96	0.97	0.99
Run	0.82	1.00	0.84	0.81
Still	0.96	0.99	1.00	1.00
Walking	0.98	0.97	0.99	0.95
F1 Score	0.96		0.95	

TABLE 11. **LSTM** PERFORMANCE ON **SAD**, COMPARING CLASSIFER FOR BASELINE VS. WGAN OVERSAMPLING

Activity	Baseline		Oversampled with WGAN	
	Recall	Precision	Recall	Precision
DrinkSD	0.94	0.73	0.90	0.78
DrinkST	0.73	0.67	0.82	0.68
Eat	0.18	0.60	0.08	0.54
Sit	0.97	1.00	0.96	1.00
SmokeGP	0.78	0.57	0.68	0.52
SmokeSD	0.78	0.99	0.81	0.82
SmokeST	0.78	0.65	0.78	0.62
Stand	0.88	1.00	0.96	1.00
F1 Score	0.74		0.72	

VI. DISCUSSION

Using raw synthetic sensor data, produced by WGAN, to oversample minority activities in imbalanced training data can boost classifier performance. Extracting features from this synthetic data also has the potential to boost performance, however the choice of classifier plays a role in how well this may work.

The CNN-based evaluation reveals just how well our synthetic data oversampling method can work, both when working on raw data and on feature data. When trained on the baseline case of imbalanced raw data, the CNN classifier tends to miss under-represented classes (see the low baseline recall rates for *run* in TABLE 8 and *stand* in Table 9). However, performance improves considerably when these classes are oversampled using WGAN – with the recall for *run* rising from 0.20 to 0.71, and *stand* from 0.25 to 0.83.

TABLE 12. **CNN** PERFORMANCE ON **SHL** DATA FEATURES, COMPARING BASELINE, SMOTE, AND WGAN OVERSAMPLING.

Activity	Baseline		Oversampled with SMOTE		Oversampled with WGAN-Features	
	Recall	Precision	Recall	Precision	Recall	Precision
Bike	0.98	0.96	0.99	0.95	0.97	0.95
Bus	0.98	0.98	0.96	0.97	0.96	0.98
Run	0.59	1.00	0.59	1.00	**0.67**	1.00
Still	0.99	0.98	0.97	0.96	0.99	0.97
Walking	0.99	0.97	0.98	0.98	0.99	0.98
F1 Score	0.93		0.92		**0.94**	

TABLE 13. **CNN** PERFORMANCE ON **SAD** FEATURES, COMPARING BASELINE, SMOTE, AND WGAN OVERSAMPLING

Activity	Baseline		Oversampled with SMOTE		Oversampled with WGAN-Features	
	Recall	Precision	Recall	Precision	Recall	Precision
DrinkSD	0.86	0.96	0.93	0.97	0.92	0.95
DrinkST	0.95	0.88	0.98	0.88	0.98	0.87
Eat	*0.92*	*0.90*	*0.96*	*0.93*	*0.94*	*0.93*
Sit	0.94	0.98	0.94	1.00	0.95	1.00
SmokeGP	0.93	0.71	0.97	0.83	0.96	0.91
SmokeSD	0.70	0.93	0.83	0.99	0.93	0.95
SmokeST	0.87	0.86	0.89	0.95	0.89	0.97
Stand	0.83	1.00	0.92	0.96	**0.92**	1.00
F1 Score	0.88		0.93		**0.94**	

TABLE 14. **LSTM** PERFORMANCE ON **SHL** DATA FEATURES, COMPARING BASELINE, SMOTE, AND WGAN OVERSAMPLING

Activity	Baseline		Oversampled with SMOTE		Oversampled with WGAN-Features	
	Recall	Precision	Recall	Precision	Recall	Precision
Bike	0.92	0.95	0.93	0.94	0.92	0.96
Bus	0.95	0.91	0.94	0.91	0.94	0.92
Run	0.53	1.00	**0.61**	1.00	0.53	1.00
Still	0.93	0.94	0.93	0.95	0.94	0.93
Walking	0.98	0.94	0.97	0.94	0.98	0.93
F1 Score	0.89		**0.90**		0.89	

TABLE 15. **LSTM** PERFORMANCE ON **SAD** FEATURES, COMPARING BASELINE, SMOTE, AND WGAN OVERSAMPLING

Activity	Baseline		Oversampled with SMOTE		Oversampled with WGAN-Features	
	Recall	Precision	Recall	Precision	Recall	Precision
DrinkSD	0.91	0.97	0.88	0.99	0.91	0.97
DrinkST	0.95	0.93	0.90	0.93	0.96	0.91
Eat	0.95	0.97	0.92	0.98	0.96	0.95
Sit	0.98	1.00	0.98	0.99	0.99	0.99
SmokeGP	0.98	0.87	0.98	0.83	0.96	0.89
SmokeSD	0.92	0.97	0.88	0.96	0.93	0.97
SmokeST	0.94	0.95	0.96	0.85	0.92	0.94
Stand	0.96	1.00	0.92	1.00	0.92	1.00
F1 Score	0.95		0.93		0.95	

When trained on feature data, CNN performs slightly better than when trained on raw data. This is likely due to the inclusion of features that capture signal dynamics, like zero-crossing rate – which is particularly useful at capturing short-term periodicity (or lack of) in classes like *run*.

Oversampling the minority classes and then generating features improves the results even further, as demonstrated by the higher recall rates using CNN for both *run* and *stand* in TABLE 12 and TABLE 13. In both datasets, the proposed WGAN-Features method produces superior results to SMOTE.

As an aside, the usefulness of hand-crafted features can also be seen in the vastly improved performance of recognising the eat activity when comparing

TABLE 9 (raw) with TABLE 13 (features). This shows a recall/precision rise from 0.29/0.59 to 0.92/0.90 for raw data

versus features on the baseline imbalanced case. This improvement is largely due to the dynamic information provided by the features.

The LSTM-based evaluation results are not as clear-cut as they are for CNN. As a starting point, LSTM is better able to capture the dynamics of activities like *run* and *stand* directly from raw data, without the need for features as was the case with CNN. Thus, the LSTM baseline results are already quite high (e.g. the recall of *run* in TABLE 10 starts at 0.82). (One caveat to this is that LSTM fails to adequately classify the *eat* activity without the help of features, as shown in TABLE 11. Because the *eat* activity is not as repetitive over short timescales as dynamic activities (like *walk* and *run*), it may be the case that using a longer window size might help LSTM on raw *eat* activity data.)

The addition of extra samples using WGAN on the raw data with LSTM has only a modest effect on some classes (e.g. *run*) while reducing the performance on others (e.g. *stand*), but overall there is a 1% to 2% drop in F1 score. And on feature data, WGAN-Features has almost identical performance as the baseline. SMOTE has a mixed result, with a 1% improvement in F1 score on the SHL data, but a 2% drop compared to baseline on SAD.

Further analysis needs to be done to ascertain why the performance of our WGAN-based oversampling is poorer when LSTM is used as the classifier versus CNN.

When evaluated using CNN, oversampling minority classes using WGAN can reap great improvements to classification performance. It can even outperform the state-of-the-art SMOTE method. Also, unlike SMOTE, it can perform just as well generating raw sensor data as well as derived features. And unlike other GAN-based oversampling methods, WGAN has the benefit of improved stability during training [25].

In future work, we plan to investigate the computational complexity, and potential overheads of using WGAN-based oversampling. WGAN uses potentially more processing power than alternatives like SMOTE. However, we would argue that for such a small number of data channels, as is typical in HAR, as opposed to the thousands typically used with, say, video, then this overhead is negligible.

The two datasets used here are relatively diverse and cover a fairy wide range of human activities. However here we only consider 5 minority classes. In a further study, we plan to explore the use of this method on a wider variety of classes and datasets.

VII. CONCLUSION

In this study, we introduce the use of a Wasserstein Generative Adversarial Network (WGAN) to generate sensor data for human activity recognition. We investigated WGAN on 5 different classes of human activity that were under-represented across 2 publicly available datasets. We evaluated the diversity and quality of the generated synthetic sensor data, and found F1-scores of over 75% when a CNN classifier is trained on synthetic and tested on real data, and 100% when it is trained on real data and tested on synthetic. We also oversampled imbalanced training sets using synthetic data and found overall F1 performance improvements of between 7% and 10% (again using CNN classifiers on raw data). More modest improvements (1% to 2%) were found when comparing CNN-classified WGAN-features against features produced using SMOTE. However, similar evaluations using LSTM found no immediate advantage from our method. As there are currently no widely recognized approaches or frameworks to evaluate synthetic sensor data, the work in this paper makes some promising steps, upon which we will explore further in future work.

REFERENCES

[1] O. Steven Eyobu and D. Han, "Feature Representation and Data Augmentation for Human Activity Classification Based on Wearable IMU Sensor Data Using a Deep LSTM Neural Network," *Sensors*, vol. 18, no. 9, p. 2892, 2018.

[2] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Comput. Surv.*, vol. 1, no. June, pp. 1–33, 2014.

[3] M. Shoaib, S. Bosch, O. Incel, H. Scholten, and P. Havinga, "A Survey of Online Activity Recognition Using Mobile Phones," *Sensors*, vol. 15, no. 1, pp. 2059–2085, 2015.

[4] T. Plötz, N. Y. Hammerla, and P. Olivier, "Feature Learning for Activity Recognition in Ubiquitous Computing," *Proceeding IJCAI'11 Proc. Twenty-Second Int. Jt. Conf. Artif. Intell.*, vol. Volume 2, pp. 1729–1734, 2011.

[5] Y. Chen and C. Shen, "Performance Analysis of Smartphone-Sensor Behavior for Human Activity Recognition," *IEEE Access*, vol. 5, pp. 3095–3110, 2017.

[6] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. M. Havinga, "Complex human activity recognition using smartphone and wrist-worn motion sensors," *Sensors (Switzerland)*, vol. 16, no. 4, pp. 1–24, 2016.

[7] S. Ramasamy Ramamurthy and N. Roy, "Recent trends in machine learning for human activity recognition—A survey," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 8, no. 4, pp. 1–11, 2018.

[8] Ó. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Commun. Surv. Tutorials*, 2013.

[9] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A Survey," *Pattern Recognition Letters*, 2018.

[10] N. Y. Hammerla, S. Halloran, and T. Ploetz, "Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables," *Ijcai*, pp. 1533–1540, 2016.

[11] A. Murad and J. Y. Pyun, "Deep recurrent neural networks for human activity recognition," *Sensors (Switzerland)*, vol. 17, no. 11, 2017.

[12] M. Zeng *et al.*, "Convolutional Neural Networks for Human Activity Recognition using Mobile Sensors," 2014.

[13] F. Moya Rueda, R. Grzeszick, G. Fink, S. Feldhorst, and M. ten Hompel, "Convolutional Neural Networks for Human Activity Recognition Using Body-Worn Sensors," *Informatics*, vol. 5, no. 2, p. 26, 2018.

[14] M. Alzantot, S. Chakraborty, and M. B. Srivastava, "SenseGen: A Deep Learning Architecture for Synthetic Sensor Data Generation," *Proc. First Int. Conf. on. IEEE*, pp. 66–73, 2017.

[15] J. Wang, Y. Chen, Y. Gu, Y. Xiao, and H. Pan, "SensoryGANs: An Effective Generative Adversarial Framework for Sensor-based Human Activity Recognition," in *Proceedings of the International Joint Conference on Neural Networks*, 2018.

[16] S. J. Preece, J. Y. Goulermas, L. P. J. Kenney, and D. Howard, "A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data," *IEEE Trans. Biomed. Eng.*, 2009.

[17] V. Elvira, A. Nazábal-Rentería, and A. Artés-Rodríguez, "A novel feature extraction technique for human activity

recognition," in *IEEE Workshop on Statistical Signal Processing Proceedings*, 2014.

[18] H. F. Nweke, Y. W. Teh, M. A. Al-garadi, and U. R. Alo, "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges," *Expert Systems with Applications*, vol. 105, no. August. pp. 233–261, 2018.

[19] D. Wu, Z. Wang, Y. Chen, and H. Zhao, "Mixed-kernel based weighted extreme learning machine for inertial sensor based human activity recognition with imbalanced dataset," *Neurocomputing*, vol. 190, pp. 35–49, 2016.

[20] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, 2002.

[21] N. Jaques, S. Taylor, A. Sano, and R. Picard, "Multimodal autoencoder: A deep learning approach to filling in missing sensor data and enabling better mood prediction," in *2017 7th International Conference on Affective Computing and Intelligent Interaction, ACII 2017*, 2018.

[22] I. J. Goodfellow *et al.*, "Generative Adversarial Networks," *arXiv e-prints*, p. arXiv:1406.2661, Jun. 2014.

[23] A. Odena, C. Olah, and J. Shlens, "Conditional Image Synthesis with Auxiliary Classifier GANs," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 2017, pp. 2642–2651.

[24] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient," Sep. 2016.

[25] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *eprint arXiv:1701.07875*. p. arXiv:1701.07875, 01-Jan-2017.

[26] F. Li, K. Shirahama, M. A. Nisar, L. Köping, and M. Grzegorzek, "Comparison of Feature Learning Methods for Human Activity Recognition Using Wearable Sensors," *Sensors*, vol. 18, no. 3, p. 679, 2018.

[27] Y. B. Ian J. Goodfellow, Jean Pouget-Abadie∗, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair†, Aaron Courville, "Generative Adversarial Nets Ian," *Vet. Immunol. Immunopathol.*, 2013.

[28] Z. Wang, Q. She, and T. E. Ward, "Generative Adversarial Networks: A Survey and Taxonomy," *arXiv e-prints*, p. arXiv:1906.01529, Jun. 2019.

[29] M. Arif, M. Bilal, A. Kattan, and S. I. Ahamed, "Better physical activity classification using smartphone acceleration sensor," *J. Med. Syst.*, 2014.

[30] F. Li, K. Shirahama, M. A. Nisar, L. Köping, and M. Grzegorzek, "Comparison of feature learning methods for human activity recognition using wearable sensors," *Sensors (Switzerland)*, 2018.

[31] F. Pedregosa and G. Varoquaux, *Scikit-learn: Machine learning in Python*, vol. 12. 2011.

[32] F. Chollet, "Keras," *GitHub Repos.*, 2015.

[33] K. Shmelkov, C. Schmid, and K. Alahari, "How Good Is My GAN?," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

[34] J. B. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *IJCAI International Joint Conference on Artificial Intelligence*, 2015.

[35] Y. Chen, K. Zhong, J. Zhang, Q. Sun, and X. Zhao, "LSTM Networks for Mobile Human Activity Recognition," *Int. Conf. Artif. Intell. Technol. Appl.*, no. Icaita, pp. 50–53, 2016.

[36] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "DeepSense: A unified deep learning framework for time-series mobile sensing data processing," in *26th International World Wide Web Conference, WWW 2017*, 2017.

[37] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, 2009.

[38] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manag.*, 2009.

[39] H. Gjoreski *et al.*, "The University of Sussex-Huawei Locomotion and Transportation Dataset for Multimodal Analytics with Mobile Devices," *IEEE Access*, 2018.

[40] M. Shoaib, H. Scholten, P. J. M. Havinga, and O. D. Incel, "A hierarchical lazy smoking detection algorithm using smartwatch sensors," in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services, Healthcom 2016*, 2016.

[41] A. Das Antar, M. Ahmed, M. S. Ishrak, and M. Atiqur Rahman Ahad, "A comparative approach to classification of locomotion and transportation modes using smartphone sensor data," in *UbiComp/ISWC 2018 - Adjunct Proceedings of the 2018 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2018 ACM International Symposium on Wearable Computers*, 2018.