A Novel Model to Generate Heterogeneous and Realistic Time-Series Data for Post-Stroke Rehabilitation Assessment

Issam Boukhennoufa, Delaram Jarchi, *Senior Member, IEEE*, Xiaojun Zhai, *Senior Member, IEEE*, Victor Utti, Saeid Sanei, *Senior Member, IEEE*, Tracey K. M. Lee, Jo Jackson, and Klaus D. McDonald-Maier, *Senior Member, IEEE*

***Abstract*—The application of machine learning-based tele-rehabilitation faces the challenge of limited availability of data. To overcome this challenge, data augmentation techniques are commonly employed to generate synthetic data that reflect the configurations of real data. One such promising data augmentation technique is the Generative Adversarial Network (GAN). However, GANs have been found to suffer from mode collapse, a common issue where the generated data fails to capture all the relevant information from the original dataset. In this paper, we aim to address the problem of mode collapse in GAN-based data augmentation techniques for post-stroke assessment. We applied the GAN to generate synthetic data for two post-stroke rehabilitation datasets and observed that the original GAN suffered from mode collapse, as expected. To address this issue, we propose a Time Series Siamese GAN (TS-SGAN) that incorporates a Siamese network and an additional discriminator. Our analysis, using the longest common sub-sequence (LCSS), demonstrates that TS-SGAN generates data uniformly for all elements of two testing datasets, in contrast to the original GAN. To further evaluate the effectiveness of TS-SGAN, we encode the generated dataset into images using Gramian Angular Field and classify them using ResNet-18. Our results show that TS-SGAN achieves a significant accuracy increase of classification accuracy (35.2%-42.07%) for both selected datasets. This represents a substantial improvement over the original GAN.**

***Index Terms*—Generative adversarial networks, mode collapse, stroke rehabilitation, time series.**

Manuscript received 13 October 2022; revised 23 April 2023 and 30 May 2023; accepted 31 May 2023. Date of publication 5 June 2023; date of current version 16 June 2023. This work was supported by the U.K. Engineering and Physical Sciences Research Council under Grant EP/R02572X/1, Grant EP/V034111/1, Grant EP/X015955/1, and Grant EP/V000462/1. *(Corresponding author: Xiaojun Zhai.)*

Issam Boukhennoufa, Delaram Jarchi, Xiaojun Zhai, and Klaus D. McDonald-Maier are with the School of Computer Science and Electronic Engineering, University of Essex, CO4 3SQ Colchester, U.K. (e-mail: ib20472@essex.ac.uk; delaram.jarchi@essex.ac.uk; xzhai@essex.ac.uk; kdm@essex.ac.uk).

Victor Utti is with the Faculty of Health, Education, Medicine and Social Care, School of Allied Health, Anglia Ruskin University, CM1 1SQ Chelmsford, U.K. (e-mail: victor.utti@aru.ac.uk).

Saeid Sanei is with the School of Science and Technology, Nottingham Trent University, Clifton Campus, NG11 8NS Nottingham, U.K. (e-mail: saeid.sanei@ntu.ac.uk).

Tracey K. M. Lee is with the School of Electrical and Electronic Engineering, Singapore Polytechnic, Singapore 139651 (e-mail: tracey_km_lee@ichat.sp.edu.sg).

Jo Jackson is with the School of Sport, Rehabilitation and Exercise Sciences, University of Essex, CO4 3SQ Colchester, U.K. (e-mail: jo.jackson@essex.ac.uk).

Digital Object Identifier 10.1109/TNSRE.2023.3283045

I. Introduction

STROKE is a significant cause of long-term disability globally, affecting millions of people with stroke-related impairments [1]. Post-stroke rehabilitation is critical to aid recovery and enhance functional outcomes [2], [3]. Recently, wearable sensors and machine learning techniques have emerged as promising tools in post-stroke rehabilitation. Wearable sensors can provide continuous and objective monitoring of a patient's daily activities, physical function, and movement patterns [4], [5]. Machine learning algorithms can then analyse and interpret the data collected from wearable sensors to provide personalized feedback, decision-making support, and optimise rehabilitation interventions [6]. The integration of machine learning and wearable sensors has the potential to transform post-stroke rehabilitation by offering personalised, adaptable, and cost-effective solutions to patients, caregivers, and healthcare providers [7].

Machine learning-based post-stroke tele-rehabilitation assessment systems can be broadly categorised into two types based on their objectives: clinical assessment emulation and activity recognition. Clinical assessment emulation systems are designed to quantify the level of correctness in executing the prescribed exercises. On the other hand, activity recognition systems aim to identify specific rehabilitation movements performed by patients and differentiate between them for recording and monitoring purposes [8]. To design an effective and accurate machine learning system, the availability and quality of data are the crucial factors [9]. Data serves as the primary resource used by the system to learn and extract features. Therefore, one major limitation in the development of a performing system is the lack of data or low-quality data [10].

The data obtained from sensors are called time series (TS) [11]. TS data is organised in a sequential time-dependent manner. If it varies just on one axis, it is called a univariate

TS; if it varies on multiple axes, it is called a multivariate TS [12]. The more available and good-quality the TS is, the better the assessment algorithms perform. Unfortunately, in healthcare applications in general, or stroke rehabilitation application in particular, it is not always possible to have sufficient data due to the condition of the patients that does not always allow them to attend a testing session [13]. A potential solution to address this issue is to generate synthetic data, (artificial or dummy) that possess adequate information to mimic the real-world data gathered under naturalistic circumstances. This procedure, known as data augmentation or the creation of surrogate data, involves producing synthetic data from real data. It is essential that the synthetic data accurately reflect the statistical distribution of the original dataset and be realistic. In essence, the generated data should bear sufficient resemblance to the authentic data to enable their use in training the machine learning models without introducing biases or errors.

In computer vision, data augmentation is already an established processing step [14]. In addition to the fact that it helps understand the configuration of the collected data it also assists with the model generalisation capability by decreasing over fitting and increasing the characterisation boundary of the trained models [15]. In the TS domain, most data augmentation techniques involve random transformations such as introducing random noise on the data, scaling and slicing both in the time and frequency domains [16]. The issue with this random transformation is that, there is a plethora of TS having different properties and sequential patterns, and each dataset has its own characteristics. e.g. what may be applied to the data collected from accelerometer can not be applied to electroencephalogram data. But with the advances made in machine learning domain, new models have been introduced that allow a personalised spawning of data that takes into account the input dataset characteristics. This is known as Generative Adversarial Networks (GANs).

GAN is a type of deep learning model which captures the inner probabilistic distribution of actual data and generates new data that are comparable to the original [17]. Many GAN architectures have been proposed over the years for manipulating the data. It can be either distance based [18], latent space based [19], or through changing the configuration [20]. One limitation of GAN is seen as mode collapse, when the generated data fail to take into account all the elements of the real dataset [21]. In other words, the generator spawns the data for few modes while omitting other elements. This results in an a synthetic dataset without learning all the information from the real one.

The present study aims to address the aforementioned issue by utilising two distinct real-world datasets representative of post-stroke tele-rehabilitation. These datasets include a clinical assessment emulation of exercises belonging to the Action Research Arm Test (ARAT), which exhibits a highly unbalanced distribution and limited data availability, thus highlighting the necessity of data augmentation. The second dataset pertains to activity recognition and encompasses 18 diverse daily activities. In order to augment the datasets, a GAN architecture is initially employed. However, it is observed that the generated data is afflicted by mode collapse. Therefore, a novel model is proposed which employs objective metrics to alleviate the issue of mode collapse and enhance the quality of the generated data.

The contributions of this work are summarised as follows:

- We propose a new GAN model by coupling it with a Siamese network (SN), to add another layer that allows to generate more heterogeneous data.
- The resulting model generates more diverse TS than the original GAN, as proved using the longest common subsequence (LCSS). Classifying the original data using the generated TS increased from 63% in the original GAN to 98.2% in the proposed model, for the first dataset and from 48% to 90.8% in the second.
- Encoding TS into images permitted to increase the classification performance, thus improving the post-stroke tele-rehabilitation assessment.

The remainder of the paper is organised as follows: firstly in Section II we introduce the dataset, then in Section III we show the results from the GAN model, followed by those from our proposed model in Section IV and demonstrate the effectiveness of our approach. Thereafter giving an overall results summary in Section IV-D and conclude the paper in Section V.

II. Datasets

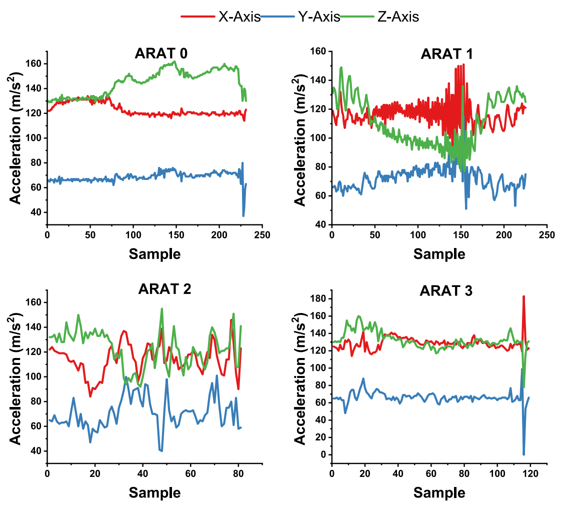
This study utilises two datasets that belong to the post-stroke rehabilitation categories. Section II-A examines an ARAT-based dataset introduced by Lee et al. in [22]. Section II-B explores the WISDM Smartphone and Smart-watch Activity and Biometrics Dataset proposed by Weiss in [23].

A. ARAT Dataset

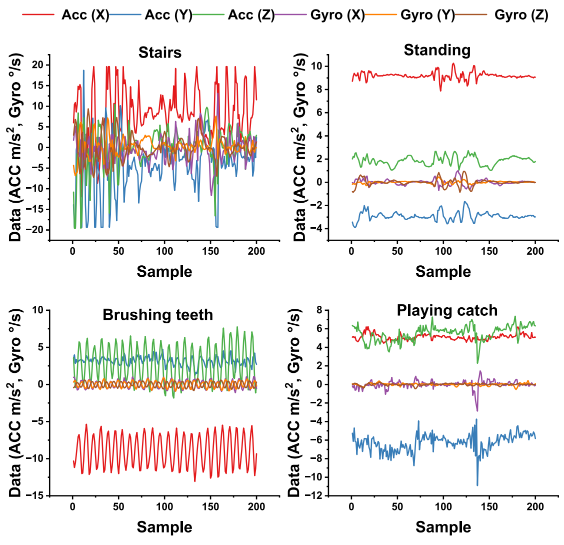
The dataset includes ARAT motions [24], which are rated on a four-point scale. A score of 3 indicates satisfactory completion within 5 seconds, while a score of 0 denotes non-completion due to factors such as inability to grasp the cube or use fingers to manipulate it. The score also considers the time taken to complete the task, where a score of 2 indicates completion with difficulty or taking an abnormally long time, and a score of 1 represents partial completion. The trial involved 34 stroke patients undergoing rehabilitation over a 60-day period in a hospital setting. Each patient performed a set of ARAT motions up to three times in a single session, with data continuously recorded and manually segmented into individual trials. Notably, the scores were awarded on a session basis, suggesting averaging over trials, and multiple therapists scored the sessions, introducing score variability despite briefings. Each class used in this study comprises a distinct number of segments, and the length of each segment varies, as presented in Table I. The data acquisition is sampled at 30 Hz.

TABLE I
Number of TS Segments per Class in the ARAT Dataset

ARAT score	0	1	2	3
Number of segments	3	6	38	31



(a) ARAT dataset.



(b) WISDM dataset.

Fig. 1. Some of the original TS segments for the datasets.

TABLE II
Activity Recognition Dataset Labels

Activity orientation	Activities
Non-hand-oriented activities	Walking (A), Jogging (B), Stairs (C), Sitting (D), Standing (E), Kicking (M)
Hand-oriented activities (Eating)	Eating soup (H), Eating chips (I), Eating pasta (J), Drinking (K), Eating sandwich (L)
Hand-oriented activities (General)	Typing (F) , Playing catch (O) , Dribbling (P), Writing (Q), Clapping (R), Brushing teeth (G), Folding clothes (S)

As depicted in Table I, the dataset is unbalanced and contains an insignificant number of segments for contemporary algorithm analysis such as deep learning. This feature renders the dataset an excellent candidate for the exploration of generating synthetic data. Furthermore, the dataset is comprised of naturalistic data recorded in a real-life scenario, thereby offering added value to its application in real-world scenarios. The segments are multivariate TS chunks of varying lengths, derived from triaxial acceleration as previously mentioned. Each TS in a particular class has dimensions of $n \times 3 \times t$, where n represents the number of segments in that class, 3 indicates the number of axes (X, Y, Z), and t denotes the sequence length.

B. Activity Recognition Dataset

This dataset was created in late 2019 and comprises various complex daily activities that includes 18 activities, as shown in Table II, which were performed by 51 distinct participants for a duration of three minutes. The data was collected using two IMU sensors, namely triaxial accelerometer and triaxial gyroscope, which were connected to a smartwatch and smartphone, respectively. The smartwatch was worn on the participant's dominant hand, while the smartphone was placed on the waist. Both sensors recorded data at a frequency of 20 Hz. Each line of the dataset contains three axes acceleration data, three axes gyroscope data, and the activity abbreviation that was performed. As per findings in our previous works [25], [26], we only utilised data from the smartwatch, as the smartphone data did not contribute to improving classification accuracy. The dataset was segmented into 10-second chunks corresponding to 200 readings using non-overlapping sliding windows, and every segment of data was labeled with the most recurring activity label. The resulting dataset structure was $16854 \times 6 \times 200$ where 16854 represents the total number of TS segments, 6 represents the number of axes (tri-axial gyro + triaxial acceleration), and 200 represents the length of the segments.

Figure 1 displays samples from each dataset, Figure 1a display some segments from each ARAT class and Figure 1b shows segments from four of the activity recognition datasets. Time axis "sample" refers to the acquisition index knowing that the first dataset has 30 acquisitions per second while the 2^{nd} has 20.

III. 2D Systems

In this section, we present the process of generating synthetic data from the original dataset using a GAN. The aim is to produce data that satisfies two key conditions: first, the generated data must be realistic and accurately represent the statistical distribution of the original dataset. Second, it should not suffer from mode collapse. We describe the proposed GAN structure in detail and demonstrate the generated synthetic data in the following sections.

A. GAN

The proposed GAN is composed of two parts: the first part encompasses a generator that takes as its input random noise vectors z and generates dummy data while the second part "the discriminator" takes the real TS data and the dummy TS data generated by the generator as input, and outputs a number that is corresponding to the probability of the input being real. The GAN employs the Nash equilibrium game principle

TABLE III
ARCHITECTURE OF THE PROPOSED GANS FOR THE TWO DATASETS

Dataset	Network architecture	Layer type	Nodes, activation, normalisation
ARAT	Generator	Input vector	32
		Fully Connected Layer	256, Leaky ReLU, Batch normalization
		Fully Connected Layer	512, Leaky ReLU, Batch normalization
		Fully Connected Layer	1024, Leaky ReLU, Batch normalization
		Fully Connected Layer	699, Leaky ReLU, Batch normalization
		Reshape Layer	3 x 233
	Discriminator	Input layer	3 x 233
		Fully Connected Layer	512, Leaky ReLU, Batch normalization
		Fully Connected Layer	256, Leaky ReLU, Batch normalization
		Output layer	1, Sigmoid
Activity recognition	Generator	Input vector	128
		Fully Connected Layer	256, Leaky ReLU, Batch normalization
		Fully Connected Layer	512, Leaky ReLU, Batch normalization
		Fully Connected Layer	1024, Leaky ReLU, Batch normalization
		Fully Connected Layer	2056, Leaky ReLU, Batch normalization
		Fully Connected Layer	1200, Leaky ReLU, Batch normalization
		Reshape Layer	6 x 200
	Discriminator	Input layer	6 x 200
		Fully Connected Layer	512, Leaky ReLU, Batch normalization
		Fully Connected Layer	256, Leaky ReLU, Batch normalization
		Fully Connected Layer	128, Leaky ReLU, Batch normalization
		Output layer	1, Sigmoid

[27], which assumes two players. The generator tries to learn the real TS data distribution whereas the discriminator tries to accurately guess whether the fed data is from the original dataset or from the generator. To be victorious in the game, the two players shall compete repeatedly to improve both the generation (maximise resemblance) and the discrimination (minimising the difference).

Mathematically, let $\mathbf{x}$ be a TS window from the dataset distribution p_X, and $\mathbf{z}$ be a random vector. We only consider that $\mathbf{z}$ is from a uniform distribution with a support of $[-1, 1]$, Let G and D be the generative and discriminative models, the generative model takes $\mathbf{z}$ as input and outputs the TS data, $G(\mathbf{z})$, that has the same support as $\mathbf{x}$. Denote the distribution of $G(\mathbf{z})$ as p_G. The discriminative model approximate the probability that the input TS data is drawn from p_X. Ideally, $d(\mathbf{x}) = 1$ if $\mathbf{x} \sim p_X$ and $D(\mathbf{x}) = 0$ if $\mathbf{x} \sim p_G$. The generative and discriminative models can be trained together by solving Equation (1) below:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{real}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}))]. \quad (1)$$

In our work, the architecture for the two parts only comprised of fully connected layers. Depending on the dataset, two architectures were proposed:

The first dataset utilises a generator with five layers, beginning with an input-layer that corresponds to the latent vector z of 32 elements. This is followed by a fully-connected layer of 256 nodes, and three additional fully-connected layers of 512, 1024, and 699 nodes, respectively. Each layer is equipped with a Leaky ReLU activation function and batch normalisation is applied. Finally, the last layer is reshaped to a 3 × 233 node configuration that matches the architecture of the input data. The discriminator, on the other hand, takes as input either real or artificial data that is first reshaped to an 899-node fully connected layer. This layer then passes through two additional fully-connected layers of 512 and 256 nodes, respectively, both of which are equipped with leaky ReLU activation functions and batch normalisation. The output node is a single node with a Sigmoid activation function, responsible for indicating whether the data is real or artificial.

For the second dataset, the same architectures are used, with only differences being a latent vector size of 128 nodes and the inclusion of an additional fully-connected layer of 2056 nodes after the 1024-layer. This is followed by a 1200 node layer that is reshaped to 6 × 200 instead of the 899-node layer in the previous dataset. The discriminator for this dataset begins with a 6 × 200 input that is flattened to 1200 nodes, followed by three additional fully-connected layers of 512, and 256, 128 nodes, respectively.

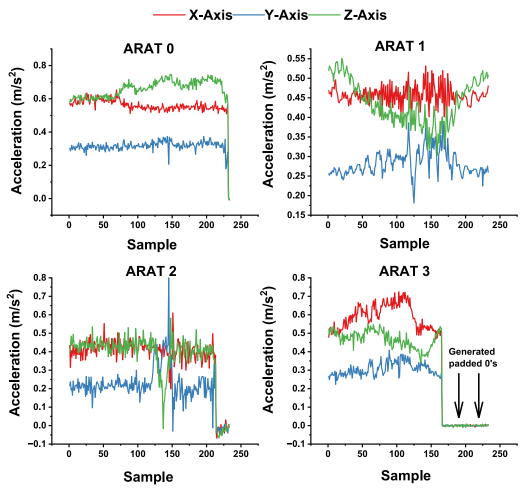
The architectural details of these setups are summarised in Table III.

B. Generated Data From the GAN Model

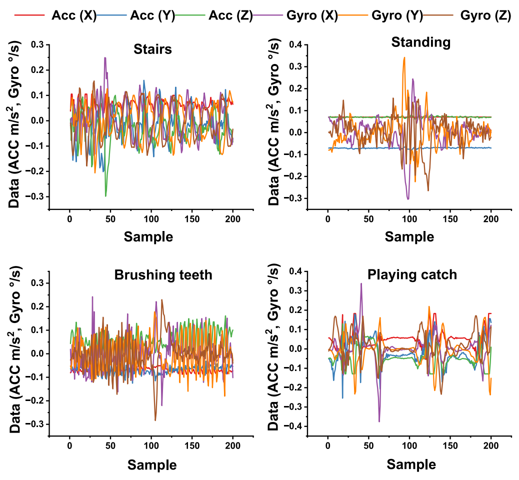
1) ARAT Dataset: It has been observed in Section II-A that the dataset is imbalanced and has a limited number of TS segments. This makes training of the deep neural networks challenging. To address this issue without the need for additional data collection, a GAN-based data augmentation technique has been proposed to generate synthetic data. This approach can help overcome the problem of imbalanced data, and its applicability may extend to other TS-related research studies.

As TS segments have varying lengths, direct application of deep learning algorithms is not feasible. These algorithms require input streams of equal length, which can be achieved by padding the segments with zeros to match the length of the longest segment which is 233 samples (time acquisition). This approach is a well-established technique in TS analysis and has been used in several studies as a simple and effective way to achieve equal-length input data [28]. Moreover, in [29], zero-padding was used in their GAN-based approach for generating synthetic TS data to handle variable-length TS data.

After padding, the segments were normalised to the range of [0, 1]. Unlike image generation, where the label of the



(a) ARAT dataset.



(b) WISDM dataset.

Fig. 2. Some of the generated TS segments for the datasets.

generated image can be visually recognised, in the TS domain, it is difficult to associate each generated window of data with its corresponding real-domain counterpart. Therefore, a separate GAN has been trained on each class of the data, with each class trained separately, and the data for each class generated independently.

Generating synthetic data separately for each class also avoids potential biases that could arise from the padding process. By training the GAN separately for each class, the generated data will have the same padding and distribution as the original data for that class. This is important because the padding process may introduce a bias in the generated data if performed on the entire dataset together.

After conducting multiple trials with different latent vector sizes, a size of 32 data points was determined to be the most suitable. The criterion for determining the optimal latent vector size was based on the quality of the generated data, as increasing the latent vector size beyond 32 did not significantly improve it.

Adam optimiser [30] with a momentum of 0.5 and learning rate of 2×10^{-4} are empirically selected for both the generator and discriminator red by testing various GAN models and trials, and binary cross-entropy was used for their compilation. The selection of learning rate was empirical and involved testing various values. The aim was to identify a value that would strike a balance between convergence speed and over fitting avoidance. The choice of using the Adam optimizer with a momentum of 0.5 and binary cross-entropy for compilation is a common practice in many GAN architectures. This decision was also informed by empirical testing.

For each class 640 epochs were required for training. This number was selected by monitoring the degradation of the discriminator loss. Moreover, Google Colab with 32 GB RAM and a Nvidia T4 GPU was used, moreover, the code was developed in Keras and TensorFlow.

Figure 2a showcases several synthetic TS segments generated from different ARAT categories using the proposed GAN model. The generated data share the same $n \times 3 \times t$ dimensional structure as the input data, with X, Y, and Z axes representing triaxial data. The generated TS segments exhibit curvature patterns similar to those observed in the real data segments, as can be seen in the plot. The GAN model can generate multivariate TS data from any input segment, as evidenced by the triaxial data generated by the model. Furthermore, the model has learned to generate the zeros that were padded to the segments to achieve equi-length time windows, as demonstrated in the rightmost plot of Figure 2a. A thresholding method was employed to determine the end of the signal, where the amplitude decreases below a specific level. This method involved identifying three consecutive data points with amplitudes equal to or less than a threshold value of 0.05 on all axes.

However, during experimentation, the GAN model did seem from visual inspection to exhibit mode collapse, which caused it to generate synthetic data for some portions of the input segments while neglecting others.

2) Activity Recognition Dataset: In accordance with the methodology outlined in Section II-B, the dataset was partitioned into 10-second segments consisting of 200 readings each using non-overlapping sliding windows. Subsequently, each data segment was labelled with the activity label that appeared most frequently, resulting in a structured dataset of dimensions $16854 \times 6 \times 200$. In order to generate synthetic data from this dataset, a GAN model was employed as detailed in Section III-A. Prior to applying the GAN, the TS segments were normalised. The hyperparameters used in the GAN training process were similar to those employed previously, with a latent vector size of 128, the Adam optimizer with a momentum of 0.5 and learning rate of 5×10^{-4}, and binary cross-entropy. The GAN training process was repeated for a total of 20k epochs. Figure 2b displays the TS segments generated by the GAN model, with each class visually identified through comparison with the real dataset. It is worth noting that the generated data appear to exhibit curvature patterns that resemble those observed in the real data segments. However, it is also observed that some classes of the generated segments appear to have been omitted, indicating a potential occurrence of mode collapse.

While visual inspection can be informative, it is not sufficient to draw accurate conclusions. Further required analysis is presented below.

C. Analysis of GAN Generated Data

We verify the effect of mode collapse on both datasets using two techniques:

1) A similarity study is conducted between the generated segments and the original ones. This is achieved by correlating the generated signal with every original signal and calculating the similarity between them using an objective method. The original signal with the highest similarity is considered the parent signal that spawned the segment. To ensure the robustness of the objective method against noise or small vibrations, we consider either Dynamic Time Warping or the LCSS [31]. LCSS has been proven to be more resilient under noisy conditions and can work with data of different lengths, hence, it has been chosen for this study. Then, the number of generated data for each class is computed to show the distribution of the generated dataset over the parent classes.
2) The generated data is used to train a classifier, and its performance is evaluated on the real dataset used as a validation set. Classification and other metrics are computed to show whether the generated data contains sufficient information to differentiate between the classes of the original dataset. Before classification, the TS data is encoded into images using Gramian Angular Field (GMAF) and fed into a ResNet-18 classifier. This pipeline has been shown to yield promising results for TS classification of the same dataset in previous works [25], [26], [32].

1) LCSS Algorithm: One of the very first applications of LCSS algorithm has been for string matching [33]. Later contributions worked on the extension of LCSS and it has been widely used for measuring the similarity of two TS with different lengths focusing on similar parts between two TS [33], [34], [35].

The basic core method of LCSS is dynamic programming that applies similarity-based searching from machine region both in time and space to keep away from distant or degenerating regions.

For LCSS operation, let's define **a** and **b** as finite discrete TS. a_1^p is associated with the first TS as **a** with a discrete time index varying between 1 and p. In a similar manner, b_1^q is associated with the second TS **b** with a discrete time index varying between 1 and q. Additionally, a_i, b_i represent the i^{th} sample of TS a and b, respectively. A recursive algorithm has been formulated to provide a solution to the LCSS [31] as given in Equation (2):

$$\mathrm{LCSS}_{\delta,\epsilon}(\mathbf{a}_1^p,\mathbf{b}_1^q) = \begin{cases} 0 & \text{if} \quad p<1 \quad or \quad q<1, \\ 1+\mathrm{LCSS}_{\delta,\epsilon}(\mathbf{a}_1^{p-1},\mathbf{b}_1^{q-1}) & \text{if} \begin{cases} d_{LP}(a_p,b_q)<\epsilon \ and \\ \mid p-q\mid<\delta, \end{cases} \\ Max\begin{cases} \mathrm{LCSS}_{\delta,\epsilon}(\mathbf{a}_1^{p-1},\mathbf{b}_1^{q}) \\ \mathrm{LCSS}_{\delta,\epsilon}(\mathbf{a}_1^{p},\mathbf{b}_1^{q-1}) \end{cases} & otherwise \end{cases} \tag{2}$$

where p and q represent the lengths of TS a and b, respectively, meanwhile, d_{LP} $(a_p - b_q)$ take any L_P -norm of the $(a_p - b_q)$.

Two parameters are used in LCSS to introduce flexibility in controlling the matching regions in time (δ) or space (ϵ). In the end, the similarity of the two times-series is measured using the output of the LCSS including a normalising factor associated with the lengths of input times-series as shown in:

$$S_{\delta,\epsilon}(\mathbf{a},\mathbf{b}) = \frac{\mathrm{LCSS}_{\delta,\epsilon}(\mathbf{a}_1^p,\mathbf{b}_1^q)}{min(p,q)} \tag{3}$$

Based on the above definition, the returned values by the LCSS vary from 0 to 1, the highest value is related to a situation when the two TS fully match, and vice-versa. The values of δ and ϵ are taken from the work in [31], which concluded that their best values are:

$$\epsilon = 0.5 \times (min(std(a), std(b)) \tag{4}$$

where std is the standard deviation of a and b,

$$\delta = round(0.1 \times n) \tag{5}$$

where n is $min(length(a, b))$

Figure 3 shows two different generated signals (blue) and their associated parent signal (red) and the corresponding similarity of 0.93 and 0.71 using the LCSS algorithm.

2) GMAF: The Gramian matrix is a matrix-based encoding method that converts TS data into images by using polar coordinates as a representation of the data. Each component of this matrix is either the addition of the sines of the polar angles (GASF) or the difference of their cosines (GADF). The time increases as the location shifts from the top left to the bottom right, thus maintaining the temporal dependence of the TS. This feature allows the polar-coordinates to be converted using the transformation principle back to the original TS data.

In this work, the used GASF are summerised as follows:

- First, using the linear standardisation equation, re-scale the data to the range [0, 1] (or [−1, 1]). 6:

$$\hat{x_i} = \frac{x_i - min(X)}{max(X) - min(X)}, \tag{6}$$

- After that, using equations 7 and 8, the data is translated into its polar coordinates form.

$$\phi = arccos(\hat{x_i}), \quad -1 \le \hat{x_i} \le 1, \;\; \hat{x_i} \in X, \tag{7}$$

$$r = \frac{t_i}{N}, \quad t_i \in \mathbb{N}. \tag{8}$$

- Finally, we sum the cosines of the polar angles to get GASF representation as follows:

$$\begin{aligned} GASF &= cos(\phi_i + \phi_j) \\ &= \hat{X}^T \cdot \hat{X} - \sqrt{I - \hat{X}^2}^{\,T} \cdot \sqrt{I - \hat{X}^2} \end{aligned} \tag{9}$$

where X represents the components of the TS X, I is the unit vector following the transformation to polar coordinates, and t is the time stamp index. Figure 4 shows the GMAF encoded images for the three axes of segments from ARAT 0, ARAT 1, ARAT 2 and ARAT 3 categories.

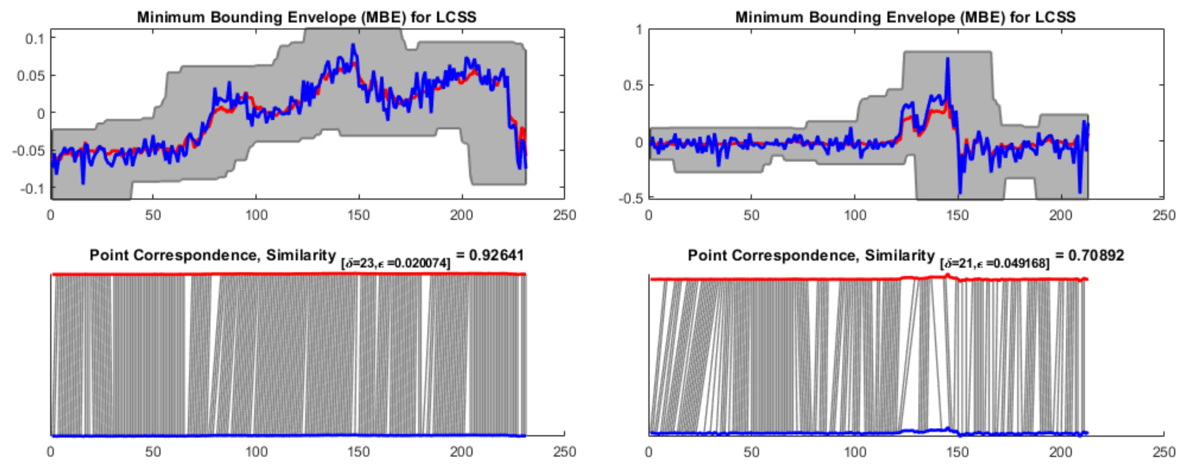


Fig. 3. LCSS applied to two spawned TS data.

TABLE IV
THE NUMBER OF GENERATED SEGMENTS PER CLASS SEEN BY LCSS ALGORITHM FOR GAN FOR ARAT 0

Segment	1	2	3
GAN	23	18	1459

TABLE V
THE NUMBER OF GENERATED SEGMENTS PER CLASS SEEN BY LCSS ALGORITHM FOR GAN FOR ARAT 1

Segment	1	2	3	4	5	6
GAN	0	1055	445	0	0	0

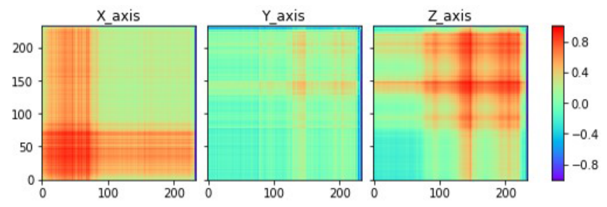


Fig. 4. An encoding example of an ARAT 0 TS segment into its GASF.

3) Results for the ARAT Dataset: For the ARAT dataset, 1500 segments are generated for each ARAT category. To determine the parent segment for each generated segment, LCSS, as described in Section 2, is used to find the segment with the highest similarity, as explained in Subsection III-C. The results for each ARAT class are presented in Tables IV, V, VI, and VII respectively. These tables highlight the bias in the generation of the data. For instance, Table IV indicates that 97.3% of the generated data corresponds to segment 3, whereas segments 1 and 2 only account for 1.53% and 1.2%, respectively. A similar bias is observed in other classes where certain segments are not generated at all, indicating mode collapse.

After this, we use the total generated dataset of 6000 segments (1500 for each class) to train the classifier described in Section 2. The resulting dataset after the data encoding was 6000 $\times$ 3 $\times$ 233 $\times$ 233 (three channels 233 $\times$ 233 images). Figure 4 shows GMAF encoded images for ARAT 0 segment.

The images were fed to a pre-trained ResNet-18, and the training process followed cyclical learning rate as suggested by Smith in [36], which has been proven effective in previous studies [25], [26], [32], [37]. The model was trained for 20 epochs using the dataset, and the original data encodings were used as a validation set. Two metrics were employed for evaluating the performance of the model: Accuracy and F1-score weighted by class. The latter metric was chosen as it takes into account the performance of each class.

The F1-score weighted by class is calculated using the following equation:

$$F1_{weighted} = \frac{\sum_{i=1}^{n} w_i F1_i}{\sum_{i=1}^{n} w_i}$$

where n represents the total number of classes, w_i is the weight assigned to class i, and $F1_i$ is the F1-score for class i. The weights w_i are defined based on the class distribution in the validation dataset.

Based on the results obtained, it was found that the classification accuracy of the model is 63%, indicating that the model is struggling to accurately classify some of the classes. This is further supported by the F1-score weighted by class of 0.58. These results suggest that the generated dataset used for training the model did not provide sufficient information about the original dataset used for validation and metric computation. The observed mode collapse in the generated dataset may have led to the poor performance of the model on some classes.

4) Results for the Activity Recognition Dataset: The second dataset was processed in a similar manner as the first one. A total of 90000 TS (TS) segments were generated using the GAN, and the LCSS method was used to find the similarities with the parent segments. The results are presented in the distribution bar chart shown in Figure 5.

From the chart, it is evident that the generated data suffers from a significant bias, with some classes having over 20000 segments while some others have none. This generated dataset was then used to train the same classifier used for the first dataset. A training dataset of 90000 $\times$ 6 $\times$ 200 $\times$ 200 (six channels of 200 $\times$ 200 images) was used, with 800 segments from each class taken as a validation set, resulting in a total of 14400 segments. The trained model achieved an accuracy of 48.73% and an F1-score weighted by class of 0.45. These results indicate that, similar to the first dataset, the model is

TABLE VI
THE NUMBER OF GENERATED SEGMENTS PER CLASS SEEN BY LCSS ALGORITHM FOR GAN OR ARAT 2

Segment	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
GAN	5	0	42	51	12	99	9	2	0	0	0	0	2	26	3	40	40	97	138
Segment(cont)	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
GAN	7	19	17	158	71	44	190	39	3	19	1	75	261	0	1	0	7	0	22

TABLE VII
THE NUMBER OF GENERATED SEGMENTS PER CLASS SEEN BY LCSS ALGORITHM FOR GAN FOR ARAT 3

Segment	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
GAN	0	28	6	0	0	0	13	3	7	122	10	80	0	51	14	49
TS-SGAN	69	52	26	73	35	65	61	43	12	118	14	65	50	13	44	16
Segment (cont)	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
GAN	24	28	66	55	2	162	191	77	68	29	129	271	0	13	2	

experiencing mode collapse and that not all the information has been captured while generating the synthetic data.

To address this issue, we propose incorporating a SN into the GAN architecture, which is described in Section IV-A

IV. TS-SGAN TO TREAT MODE COLLAPSE

The data generated from the GAN in Section III suffers from mode collapse which results on the data not being heterogeneous. We propose to solve this by adding a SN, this was inspired by the work in computer vision of Allahyani et al. in [38] and adapting it into the TS domain.

In [39] the SN was initially introduced for the use in the tasks involving face and signature verification. Two sub-networks with common weights make up SN [40]. SN compares the characteristics from the pair networks using Euclidean distance while learning the features from each sub-network. As a result, during the training, the network aims to increase distance between feature pairs (latent data) when they are from separate classes while reducing it when they are from the same class. SNs have been normally employed frequently for the re-identification task because of this attribute as the job's objective is to determine how similar two sequences are to one another [41], [42]. The associated verification loss as given in Equation (10).

$$Loss_{ver}(L_i, L_j) = \begin{cases} \frac{1}{2}\|L_i - L_j\|^2 & i = j \\ \frac{1}{2}\max(m - \|L_i - L_j\|, 0)^2 & i \neq j, \end{cases} \tag{10}$$

where m is the margin, and L_i and L_j are the latent data for the i^{th} and j^{th} TS data sequence. They correspond the output of the last layers on the i^{th} and j^{th} branch before being fed into the similarity metric function.

A. TS-SGAN

Our suggested method for reducing the mode collapse problem is to combine the SN with the modelled GAN architecture to construct a time-series Siamese GAN (TS-SGAN). This will add an additional layer that will learn to differentiate between the different segments of the input layer and try to spawn more heterogeneous data. As shown in Figure 6, the TS-SGAN is divided into two components; the first component consists of a generator G and a discriminator $D1$, which are the fundamental configuration in every GAN design. G produces artificial data $\hat{X}$ using random noise vectors z as input. The discriminator $D1$ accepts its inputs from the real data X and the created data $\hat{X}$ so that the networks outputs a probability of the data to be real. The second part of the network is to be responsible about the heterogeneity of the generated data. It is made up of a SN and a $D2$ discriminator. The SN in the TS-SGAN architecture finds similarity in a batch of data, it generates the similarity of the entire batch for real data (S) as well as the created data $\hat{S}$; this serves as an additional layer to spot mode collapse. The inputs to $D2$ are S and $\hat{S}$. If the data in the batch are heterogeneous, the $D2$ identifies the similarity as true similarity. In any other case, the $D2$ identifies it as bogus similarity thus indicating mode collapse.

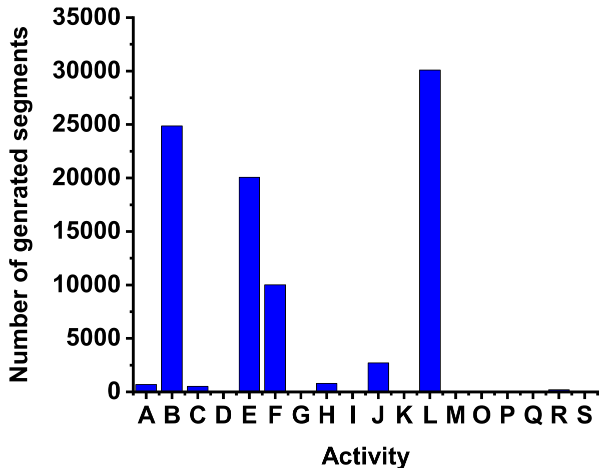


Fig. 5. Distribution of the GAN generated data per classes.

In order to implement the TS-SGAN, we need to merge the function inherited from the GAN as well as the function responsible for the heterogeneity of the output data. The first part was demystified in Section III and given by Equation (1). The latter one, which is responsible for treating mode collapse by varying the GAN output a heterogeneity principle for both G and $D2$, The role of $D2$ is to discriminate between the heterogeneity in the real TS data and the heterogeneity in the created one. Very similar to the Nash equilibrium game principle between G and $D1$ to generate realistic Ts data, we can view the Nash equilibrium between G and $D2$ as a mean to generate heterogeneous data [43].

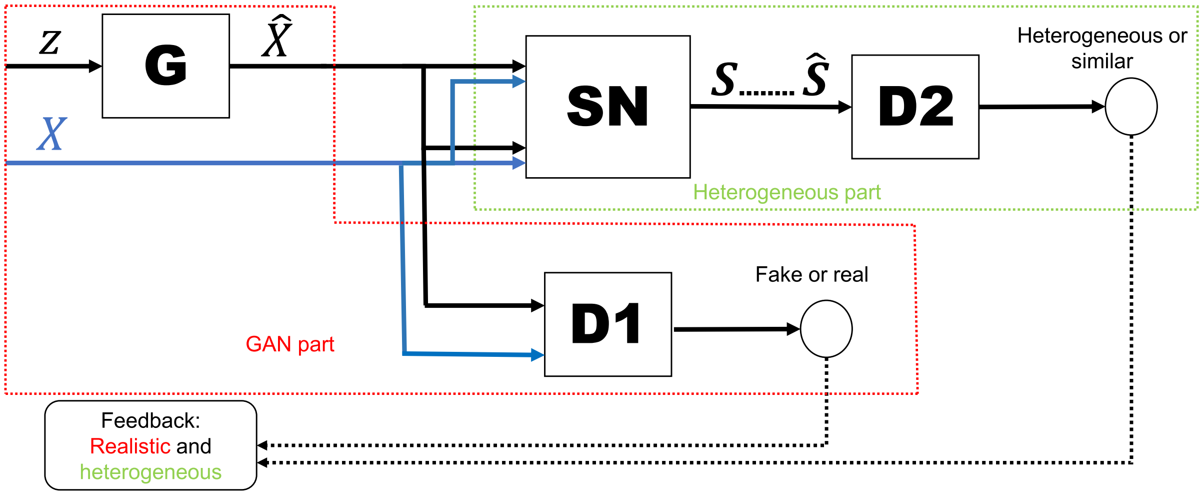


Fig. 6. Siamese GAN flowchart.

TABLE VIII
SIAMESE NETWORK AND DISCRIMINATOR 2 ARCHITECTURES

Network	Layer Type	Number of Nodes, Activation Function, Batch Normalization
Siamese Network Architecture	Input Layer	3 x 233 or 6 x 200
	Convolutional Layer	4, Tanh, Same Padding
	Convolutional Layer	16, Tanh, Same Padding
	Flatten Layer	11184
	Output Layer	1, Linear
Discriminator 2 Architecture	Input Layer	3 x 233 or 6 x 200
	Fully Connected Layer	128, Leaky ReLU, Batch Normalization
	Output Layer	1, Sigmoid

The SN in the TS-SGAN takes a pair of real data denoted as p_{real} and outputs their similarity S. Simultaneously, it takes another pair from the generated data denoted as p_{fake} which also generates their similarity $\hat{S}$. The role of $D2$, then, is to discriminate between the heterogeneity of p_{real} and p_{fake}. Finally, G role is to generate p_{fake} TS data that possesses p_{real} heterogeneity. Consequently, G in this part, tries to minimise the coast function while $D2$ tends to maximise it. The process is given in Equation (11).

$$\begin{aligned}\min_{G}\max_{D_2} & V(D_2, G) \\ &= \mathbb{E}_{\boldsymbol{x1,x2}\sim p_{\text{real}}}[\log D_2(SN(x_1, x_2))] \\ &\quad + \mathbb{E}_{\boldsymbol{z1,z2}\sim p_z(z)}[\log(1 - D_2(SN(G(z_1), G(Z_2))))]. \quad (11)\end{aligned}$$

The architecture of the heterogeneity part of our proposed TS-SGAN is shown in Table VIII. The SN comprises of two 2D CNN layers with a 3×3 kernel including similar padding and *tanh* activation function with successively 4 and 16 channels, followed by a flatten function before outputting the similarity value. $D2$ comprises a simple MLP layer of 128 nodes with the output node responsible for generating either bogus for mode collapse or real for the heterogeneous data.

Coupling this last part with the GAN part from Section III gives us the TS-SGAN architecture. The corresponding overall Loss function of the TS-SGAN is given in Equation (12).

$$\begin{aligned}\min_{G}\max_{D_1,D_2} & V(D_1, D_2, G) \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{real}}(\boldsymbol{x})}[\log D_1(\boldsymbol{x})] \\ &\quad + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D_1(G(z))] \\ &\quad + \mathbb{E}_{\boldsymbol{x1,x2}\sim p_{\text{real}}}[\log D_2(SN(x_1, x_2))] \\ &\quad + \mathbb{E}_{\boldsymbol{z1,z2}\sim p_z(z)}[\log(1 - D_2(SN(G(z_1), G(Z_2))))]. \quad (12)\end{aligned}$$

B. Algorithmic Process

Figure 6 shows the flowchart of the proposed TS-SGAN, as discussed earlier, it comprises G and $D1$ for the GAN part responsible for generating realistic data, and SN and $D2$ responsible for generating heterogeneous data. G takes the latent data vector Z as input and outputs the bogus TS data, while, $D1$ takes instances of real TS data and the bogus TS data, and outputs the probability that the input is real. The SN takes two batches from a dataset DTS that contains both bogus $\hat{X}$ and real data X and outputs the similarity between them ($\hat{S}$ for bogus data and S for real data). Finally $D2$, takes S and $\hat{S}$ and produce the probability of heterogeneity.

Hence, the steps to produce heterogeneous and realistic data using the TS-SGAN are:

- The SN is trained on the real dataset, to learn differentiate between the segments.
- A batch of latent data z is fed to G in order to generate bogus data $\hat{X}$.
- Real data is divided into two parts X_1 and X_2 and are fed to SN in order to produce S.
- Bogus data is divided into two parts $\hat{X}_1$ and $\hat{X}_2$ and are fed to SN in order to produce $\hat{S}$.
- $D2$ takes S and $\hat{S}$ to produces heterogeneity probability and $D1$ takes X and $\hat{X}$ to produces how real probability.
- The process is repeated until p_{bogus} converges to p_{real}, and the parameters of SN, G, $D1$ and $D2$ are updated according to the loss function provided in Equation (12).

The pseudo-code of the TS-SGAN is given in Algorithm 1.

C. Experimental Results

In this section, we present the experimental results of our proposed TS-SGAN model, which was trained using the

Algorithm 1 TS-SGAN Algorithm to Produce Realistic Heterogeneous TS Data

```
Input      : Dataset of real TS data and the number
             of iterations as $I$ for SN.
Output     : G that produces realistic and
             heterogeneous TS data.
Parameters: $\delta$, $\theta$, $\Gamma$ which are parameters of the
             architectures $D1$, $D2$, $G$, are
             consecutively initialised.
Sample of noise data $[Z_1 \ldots\ldots Z_m]$.
Sample of bogus TS data $[\hat{X}_1 \ldots\ldots \hat{X}_m]$.
DTS sample containing both bogus $\hat{X}$ and real data X.
while iterations $< I$ do
    Train SN on DTS $(\hat{X} \cup X)$
end
while $\hat{X}$ not converge to $X$ do
    Generate Z $[Z_1 \ldots\ldots Z_n]$
    $G(Z)$ random batch of latent TS data $[\hat{X}_1 \ldots\ldots \hat{X}_n]$
    X random batch of real TS data $[X_1 \ldots\ldots X_n]$
    Split latent TS in two:
      $\hat{X}^1 = [\hat{X}_1 \ldots\ldots \hat{X}_{\frac{n}{2}}]$, $\hat{X}^2 = [\hat{X}_{\frac{n}{2}} \ldots\ldots \hat{X}_n]$
    Split real TS in two:
      $X^1 = [X_1 \ldots\ldots X_{\frac{n}{2}}]$, $X^2 = [X_{\frac{n}{2}} \ldots\ldots X_n]$
    $S = SN(X^1, X^2)$, $\hat{S} = SN(\hat{X}^1, \hat{X}^2)$
    Update $\delta$, $\theta$, $\Gamma$ using Equation (12).
end
```

procedure described in Section IV-A and Section IV-B. Specifically, the pre-processing and configuration used for training the GAN were also used for training TS-SGAN. Moreover, the same analysis that was conducted in Section III-C, which involved finding the parent segment for the generated data using LCSS and training the same classifier on the generated data, followed by using the real data as a validation set.

1) ARAT Dataset: Initially, the SN was trained for 32 epochs to acquire the ability to distinguish between the distinct segments contained in the dataset. After that, the hyperparameters employed in Section III-C3 for GAN training were utilised, to train the GAN namely, 640 epochs, the Adam optimizer with a momentum of 0.5, binary cross-entropy loss function, a latent vector size of 32, and a learning rate of 2×10^{-4}.

Upon conducting a visual inspection of the generated data, it was observed that the spawned time chunks exhibited a comparable quality to those generated by the model trained in Section III-A. Notably, the model learned to generate the padded zeroes that were added to ensure uniform signal length, which is a consequence of retaining the GAN component of TS-SGAN. However, it is evident that the TS-SGAN model effectively incorporates all dataset segments in generating novel data. Furthermore, a thorough visual examination reveals

TABLE IX
THE NUMBER OF GENERATED SEGMENTS PER CLASS SEEN BY LCSS ALGORITHM FOR GAN AND TS-SGAN FOR ARAT 0

Segment	1	2	3
TS-SGAN	492	500	508

TABLE X
THE NUMBER OF GENERATED SEGMENTS PER CLASS SEEN BY LCSS ALGORITHM FOR GAN AND TS-SGAN FOR ARAT 1

Segment	1	2	3	4	5	6
TS-SGAN	339	206	292	200	214	249

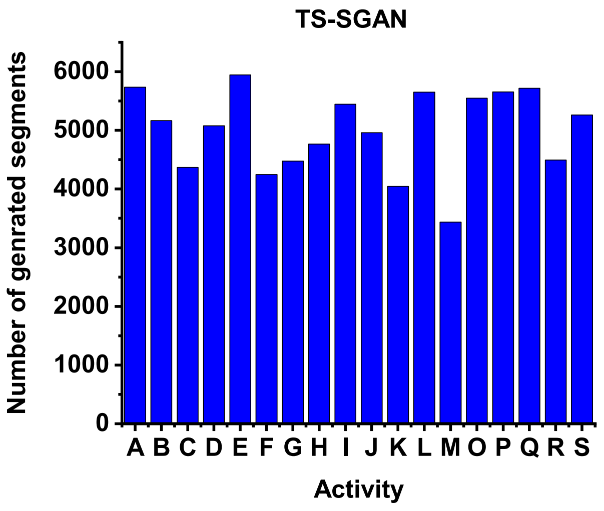


Fig. 7. Distribution of the TS-SGAN generated data for different classes.

that the model does not suffer from mode collapse, this is substantiated below.

Following the same procedure used in Section III-A, we generate 1500 segment per class, results are shown in Tables IX, X, XI, and XII respectively.

The tables clearly demonstrate that the TS-SGAN-generated data are uniformly distributed across various segments and classes of the dataset, which was not the case with the GAN-generated data. Furthermore, training the same classifier as in Section III-C3 on the TS-SGAN-generated data yields an accuracy of 98.2% and a weighted F1-score of 0.99, surpassing the GAN model's performance by 35% in accuracy and 0.41 in F1-score per class. These results indicate that the TS-SGAN model is not prone to mode collapse and effectively captures all relevant information while generating synthetic data for this dataset.

2) Activity Recognition Dataset: Initially, the SN was trained on the authentic dataset consisting of 14400 segments, for a total of 150 epochs, with the aim of acquiring the ability to distinguish between the various classes.

Subsequently, utilising the identical parameters from Section III-C4, the TS-SGAN was trained, and 90000 segments were generated. The class distribution per partition is illustrated in Figure 7. It is observed that akin to the findings obtained for the ARAT dataset, the TS-SGAN-generated data is uniformly distributed among the classes, as opposed to what was observed for the GAN-generated data.

Furthermore, upon training the previously established classifier on the generated data and then using the authentic data

TABLE XI
THE NUMBER OF GENERATED SEGMENTS PER CLASS SEEN BY LCSS ALGORITHM FOR GAN AND TS-SGAN FOR ARAT 2

Segment	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
TS-SGAN	57	33	22	47	65	73	25	40	45	50	25	32	45	28	50	54	31	57	10
Segment(cont)	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
TS-SGAN	44	19	28	60	10	85	64	21	29	29	35	74	32	50	30	35	6	35	25

TABLE XII
THE NUMBER OF GENERATED SEGMENTS PER CLASS SEEN BY LCSS ALGORITHM FOR GAN AND TS-SGAN FOR ARAT 3

Segment	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
TS-SGAN	69	52	26	73	35	65	61	43	12	118	14	65	50	13	44	16
Segment (cont)	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
TS-SGAN	26	20	24	45	127	30	87	43	62	16	107	53	11	74	19	

as a validation set, an accuracy of 90.8% with an F1-score per class of 0.90 was obtained. This represents a substantial improvement of 42.07% in accuracy and 0.45 in F1-score as compared to the GAN-generated data.

D. Results Summary

In this study, we introduce a framework for generating realistic and heterogeneous multivariate TS data. Our approach leverages two datasets related to post-stroke rehabilitation assessment: (1) a small, unbalanced dataset containing data for a popular rehabilitation assessment scale, and (2) a larger activity recognition dataset. We generated data that closely resembled the original data and contained enough information to enable near-perfect classification by a classifier.

Initially, we employed a vanilla GAN to generate the multivariate TS data. We trained the model separately on each category in the first dataset, as it is not feasible to predict the label from visual inspection of the signal. In the case of the second dataset, we trained the model on the entire dataset, as it was easier to distinguish between classes. While the generated data met the realism criteria, further inspection revealed that many segments from the dataset were not included in the generated data. This observation suggested that the model suffered from mode collapse, failing to capture the heterogeneity of the true distribution. To confirm this hypothesis, we used the LCSS algorithm to compute the similarity between the generated data and the original segments. Results on both datasets confirmed the presence of mode collapse. Additionally, we trained a classifier on the generated data and evaluated it using the real data as a validation set, and the classification results were poor due to mode collapse.

To address the issue of mode collapse, we proposed a new method using the TS-SGAN. This involved adding a layer that learns the heterogeneity between different input structures and uses this information to characterise all modes, enabling the generator to spawn more diverse data. We achieved this by adding an SN that first discriminates between the dataset elements and then generates similarity, which is sent to a second discriminator in the network. The resulting generator did not suffer from mode collapse, and the generated data were heterogeneous, as evidenced by the distribution of the generated data across both datasets and the excellent classification results when training the classifier on the newly generated data.

This study lays the groundwork for future research endeavours. Our intention is to extend the application of the TS-SGAN model to other types of GAN architectures, such as conditional GANs and cycle GANs, and assess their efficacy on various TS datasets. While our focus has been on post-stroke rehabilitation, we believe that this model has the potential to be applied to other TS domains with further research and development. Additionally, conducting a thorough analysis of the generated data, including their variability, and performing comprehensive comparisons with the original datasets, could serve as valuable groundwork for future research.

V. CONCLUSION

In the field of post-stroke tele-rehabilitation assessment, the collection of large amounts of data from wearable sensors is crucial. This data should be both realistic and diverse in order to facilitate the development of effective models for assessing patients' actions. Data augmentation in the TS domain is an important step towards the development of more efficient assessment models that can generalise better to real-life circumstances. While GANs have been shown to generate meaningful data, they are prone to mode collapse, limiting their effectiveness. In this paper, we proposed a new model called TS-SGAN, which addresses the mode collapse issue of vanilla GANs by incorporating a second discriminator and a SN. The resulting model was able to generate more diverse and realistic data, which improved the classification performance of the activity recognition dataset from 48.73% to 90.8% and from 63% to 98.2% for the ARAT dataset. These results demonstrate the potential of the TS-SGAN model to enhance the quality of TS data and improve the accuracy of classification models in post-stroke tele-rehabilitation assessment.

ACKNOWLEDGMENT

For the purpose of open access, the authors have applied a creative commons attribution (CC BY) license to any author accepted manuscript version arising. The Action Research Arm Test (ARAT) data came from the Ministry of Education Singapore grant 2010MOE-IF-005.

REFERENCES

[1] B. H. Dobkin, "Rehabilitation after stroke," *New England J. Med.*, vol. 352, no. 16, pp. 1677–1684, Apr. 2005.

[2] R. C. V. Loureiro, W. S. Harwin, K. Nagai, and M. Johnson, "Advances in upper limb stroke rehabilitation: A technology push," *Med. Biol. Eng. Comput.*, vol. 49, no. 10, pp. 1103–1118, Oct. 2011.

[3] L. Santisteban, M. Térémetz, J.-P. Bleton, J.-C. Baron, M. A. Maier, and P. G. Lindberg, "Upper limb outcome measures used in stroke rehabilitation studies: A systematic literature review," *PLoS ONE*, vol. 11, no. 5, May 2016, Art. no. e0154792.

[4] K. E. Laver, Z. Adey-Wakeling, M. Crotty, N. A. Lannin, S. George, and C. Sherrington, "Telerehabilitation services for stroke," *Cochrane Database Systematic Rev.*, no. 1, pp. 1–81, 2020.

[5] T. Johansson and C. Wild, "Telerehabilitation in stroke care—A systematic review," *J. Telemedicine Telecare*, vol. 17, no. 1, pp. 1–6, Jan. 2011.

[6] M. H. Lee, D. P. Siewiorek, A. Smailagic, A. Bernardino, and S. B. I. Badia, "Opportunities of a machine learning-based decision support system for stroke rehabilitation assessment," 2020, *arXiv:2002.12261*.

[7] S. Campagnini, C. Arienti, M. Patrini, P. Liuzzi, A. Mannini, and M. C. Carrozza, "Machine learning methods for functional recovery prediction and prognosis in post-stroke rehabilitation: A systematic review," *J. NeuroEng. Rehabil.*, vol. 19, no. 1, pp. 1–22, Dec. 2022.

[8] I. Boukhennoufa, X. Zhai, V. Utti, J. Jackson, and K. D. McDonald-Maier, "Wearable sensors and machine learning in post-stroke rehabilitation assessment: A systematic review," *Biomed. Signal Process. Control*, vol. 71, Jan. 2022, Art. no. 103197.

[9] X. Song, S. S. Van De Ven, L. Liu, F. J. Wouda, H. Wang, and P. B. Shull, "Activities of daily living-based rehabilitation system for arm and hand motor function retraining after stroke," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 30, pp. 621–631, 2022.

[10] A. Kaku, A. Parnandi, A. Venkatesan, N. Pandit, H. Schambra, and C. Fernandez-Granda, "Towards data-driven stroke rehabilitation via wearable sensors and deep learning," in *Proc. Mach. Learn. Healthcare Conf.*, 2020, pp. 143–171.

[11] D. R. Brillinger, *Time Series: Data Analysis and Theory*. San Francisco, CA, USA: SIAM, 2001.

[12] T.-C. Fu, "A review on time series data mining," *Eng. Appl. Artif. Intell.*, vol. 24, no. 1, pp. 164–181, Feb. 2011.

[13] D. Kairy, P. Lehoux, C. Vincent, and M. Visintin, "A systematic review of clinical outcomes, clinical process, healthcare utilization and costs associated with telerehabilitation," *Disability Rehabil.*, vol. 31, no. 6, pp. 427–447, Jan. 2009.

[14] Y. Cao et al., "Recent advances of generative adversarial networks in computer vision," *IEEE Access*, vol. 7, pp. 14985–15006, 2019.

[15] A. Borji, "Pros and cons of GAN evaluation measures," *Comput. Vis. Image Understand.*, vol. 179, pp. 41–65, Feb. 2019.

[16] Q. Wen et al., "Time series data augmentation for deep learning: A survey," 2020, *arXiv:2002.12478*.

[17] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, Jan. 2018.

[18] J. Adler and S. Lunz, "Banach Wasserstein GAN," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–10.

[19] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton, "VEEGAN: Reducing mode collapse in GANs using implicit variational learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[20] I. O. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel, and B. Scholkopf, "AdaGAN: Boosting generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10.

[21] H. Thanh-Tung and T. Tran, "Catastrophic forgetting and mode collapse in GANs," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–10.

[22] T. K. M. Lee, K. Leo, S. Sanei, E. Chew, and L. Zhao, "Triaxial rehabilitative data analysis incorporating matching pursuit," in *Proc. 25th Eur. Signal Process. Conf. (EUSIPCO)*, Aug. 2017, pp. 434–438.

[23] G. M. Weiss, "WISDM smartphone and smartwatch activity and biometrics dataset," *UCI Mach. Learn. Repository, WISDM Smartphone Smartwatch Activity Biometrics Dataset Data Set*, vol. 7, pp. 133190–133202, Sep. 2019.

[24] D. Carroll, "A quantitative test of upper extremity function," *J. Chronic Diseases*, vol. 18, no. 5, pp. 479–491, May 1965.

[25] I. Boukhennoufa, X. Zhai, V. Utti, J. Jackson, and K. D. McDonald-Maier, "Encoding sensors' data into images to improve the activity recognition in post stroke rehabilitation assessment," in *Proc. Int. Conf. Pattern Recognit. Artif. Intell.* Cham, Switzerland: Springer, 2022, pp. 114–123.

[26] I. Boukhennoufa, X. Zhai, V. Utti, J. Jackson, and K. D. McDonald-Maier, "A comprehensive evaluation of state-of-the-art time-series deep learning models for activity-recognition in post-stroke rehabilitation assessment," in *Proc. 43rd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Feb. 2021, pp. 2242–2247.

[27] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 139–144.

[28] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[29] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series generative adversarial networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[31] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multi-dimensional time-series with support for multiple distance measures," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2003, pp. 216–225.

[32] I. Boukhennoufa, X. Zhai, K. D. McDonald-Maier, V. Utti, and J. Jackson, "Improving the activity recognition using GMAF and transfer learning in post-stroke rehabilitation assessment," in *Proc. IEEE 19th World Symp. Appl. Mach. Intell. Informat. (SAMI)*, Jan. 2021, pp. 000391–000398.

[33] G. Das, D. Gunopulos, and H. Mannila, "Finding similar time series," in *Proc. Eur. Symp. Princ. Data Mining Knowl. Discovery*. Cham, Switzerland: Springer, 1997, pp. 88–100.

[34] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proc. 18th Int. Conf. Data Eng.*, 2002, pp. 673–684.

[35] S. Sanei, D. Jarchi, and A. G. Constantinides, *Body Sensor Networking, Design and Algorithms*. Hoboken, NJ, USA: Wiley, 2020, pp. 216–225.

[36] L. N. Smith, "Cyclical learning rates for training neural networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 464–472.

[37] I. Boukhennoufa, Z. Altai, X. Zhai, V. Utti, K. D. McDonald-Maier, and B. X. W. Liew, "Predicting the internal knee abduction impulse during walking using deep learning," *Frontiers Bioeng. Biotechnol.*, vol. 10, pp. 1–9, May 2022.

[38] A. Murray and D. B. Rawat, "On the performance of generative adversarial network by limiting mode collapse for malware detection systems," *Sensors*, vol. 22, no. 1, p. 264, 2021.

[39] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah, "Signature verification using a 'Siamese' time delay neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 6, 1993, pp. 1–8.

[40] M. Cen and C. Jung, "Fully convolutional Siamese fusion networks for object tracking," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 3718–3722.

[41] B. G. V. Kumar, G. Carneiro, and I. Reid, "Learning local image descriptors with deep Siamese and triplet convolutional networks by minimizing global loss functions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5385–5394.

[42] X. Ning, K. Gong, W. Li, L. Zhang, X. Bai, and S. Tian, "Feature refinement and filter network for person re-identification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 9, pp. 3391–3402, Sep. 2021.

[43] R. B. Myerson, "Nash equilibrium and the history of economic theory," *J. Econ. Literature*, vol. 37, no. 3, pp. 1067–1082, Sep. 1999.