Institut für Informatik
Fakultät für Mathematik und Informatik
Abteilung Datenbanken

Privacy-Preserving Smartwatch Health Data Generation For Stress Detection Using GANs

Masterarbeit

vorgelegt von:
Nils Wenzlitschke

Matrikelnummer:
3755701

Betreuer:
Prof. Dr. Erhard Rahm
Lucas Lange

Abstract

Smartwatch health data is increasingly being used in health applications and patient monitoring, including stress detection. This data contains a high level of sensitive personal information about individuals and patients. At the same time, it is burdensome and expensive to collect this data from many individuals. In this thesis, we present an approach to generate privacy-preserving smartwatch data. We test the performance of synthetic sequence data generated by different state-of-the-art GANs: TimeGAN, *conditional GAN* (cGAN), and DoppelGANger. We train and hyperparameter tune these GANs for our use case. Using these GANs, we generate synthetic datasets that learn the statistical distribution of WESAD, a wearable stress detection dataset. To evaluate these synthetic datasets, we present an evaluation method consisting of three evaluation facets: diversity, fidelity, and usefulness. We use a PCA and t-SNE analysis to evaluate the diversity of the dataset, and use different machine learning classifiers to test the indistinguishability of the synthetic and the original data. To evaluate usefulness, we train *convolutional neural network* (CNN) stress detection models on the synthetic data in two different ways. First, we augment the original dataset by the synthetic dataset and evaluate the impact of the new synthetic samples on the training of the stress detection models. We also perform the *Train on Synthetic Test on Real* (TSTR) evaluation procedure, where we train the dataset exclusively on the synthetic data and test on the real data. By augmenting the original dataset with synthetic data generated by the cGAN architecture, we achieve a stress detection accuracy of 0.9115 and can thus improve the baseline approach without synthetic data by 0.0275. When training only on synthetic data generated by the cGAN, the stress detection model achieves an accuracy of 0.910 on the original WESAD dataset, improving the baseline by 0.02. To make the synthetic datasets generated by the GANs privacy-preserving, we apply Differential Privacy (DP) to the cGAN that achieves the best results. We evaluate the results of this DP-cGAN analogously to the GAN evaluation method for $\epsilon = \{0.1, 1, 10\}$. We obtain a stress detection accuracy of 0.811 for the TSTR evaluation with an $\epsilon = 0.1$, decreasing the baseline accuracy of the WESAD dataset by 0.082, while providing high privacy guarantees.

This work demonstrates that GANs, and more specifically DP-GANs, can be used to generate synthetic health data. The synthetic data mimics the statistical distribution and physiological stress response characteristics of the WESAD dataset. It also ensures privacy guarantees with a performance trade-off to improve stress detection results.

Nils Wenzlitschke
3755701

Acknowledgements

Special thanks to Maximilian Ehrhart and Bernd Resch for providing me with code from the paper *A Conditional GAN for Generating Time Series Data for Stress Detection in Wearable Physiological Sensor Data* and tips on how to best train this architecture.

Nils Wenzlitschke
3755701

Contents

List of Figures

List of Tables

1. Introduction

In recent years, deep learning has made a significant difference. The triumph of *Convolutional Neural Networks* (CNN) is initiated by Krizhevsky et al. [1] with the AlexNet in the ImageNet Challenge, which used deep learning methods to outperform conventional approaches. As a result, the topic of deep learning is getting a lot of attention, both in current research and in everyday life.

Deep learning is finding applications in various professional fields such as law enforcement, financial services, and customer service. It is also making critical advances in medicine and healthcare, to the point where AI is better able to detect breast cancer than human doctors [2]. In healthcare, the study of stress is becoming increasingly important, to the extent that stress has been called the health epidemic of the 21st century [3]. To address this problem with advances in deep learning, Gil-Martin et al. [4] show that using a CNN, classification into stressed and non-stressed based on physiological and biological signals is feasible. The researchers trained their CNN model on the WESAD dataset, which consists of physiological and biological signals from 15 subjects [5]. The results show different performance in the stress detection for individual subjects, with some performing very well and others performing poorly.

To solve this problem of high variability, deep learning models often require a larger dataset. Depending on the application field, data is more or less accessible and available. In the healthcare sector in particular, there is often little data available on specific diseases, which in turn come from only a few individuals. Measuring and aggregating new data is often time-consuming and costly. Besides, the data often comes from patient records, which can include information about diseases, health status, and medications. This raises another issue, which is the privacy of this data.

The protection of sensitive personal data is increasingly becoming the focus of social discourse, which also led to the introduction of data protection laws like the General Data Protection Regulation (GDPR)[1] in the EU in 2016. This regulates the handling and processing of personal data. This has led to efforts to anonymize data to support analytics and applications while protecting sensitive data. However, approaches where data is anonymized, supposedly ensuring privacy, have also revealed potential weaknesses to allow re-identification of the person [6]. To overcome the first problem, the scarcity of data, the use of *Generative Adversarial Networks* (GAN) provides a good solution. Thus, new synthetic data points can be generated based on an existing dataset [7].

The newly generated data points can then be used to train the model and are likely to improve the final performance to some extent. Now, it might seem reasonable to think that synthetically generated data also solves the problem of protecting personal data, since it involves people who do not actually exist. However, researchers have shown that the data points generated in this way, and the model trained on them, also leak information about the individual from the distribution of the original dataset [8]. Nevertheless, to solve the privacy problem, different approaches from the field of *differential privacy* (DP) can be implemented. Xie et al. [9] have shown that adding DP to a GAN can ensure that the privacy of the original data distribution is preserved, and no inferences

[1]https://dsgvo-gesetz.de/

can be made about the original distribution, while at the same time generating data points that allow the models to be trained with more data.

The following work addresses the research question of whether it is possible to develop a GAN to learn the statistical distribution of a smartwatch dataset for stress detection and generate synthetic data samples to improve stress detection. For this purpose, we use the WESAD dataset of Schmidt et al. [5]. We will show that it is possible to augment this dataset with synthetic data to increase stress detection performance. Since this dataset contains multivariate time-series data with the two states *stress* and *non-stress*, we first had to determine GAN architectures that can learn these data characteristics. We take existing state-of-the-art time-series GAN architectures, extend them, and train and hyperparameter tune them for our use case, stress detection. We address the difficulty of GAN evaluation and compare the synthetic data generated by the GANs with respect to three different evaluation facets. We perform a visual evaluation to determine the diversity of the data. That is, the ability of the GAN to represent the diversity and the time-dependent evolution of the original data for the two states, *stress* and *non-stress*. The fidelity evaluation is devoted to the statistical attributes of the synthetic data compared to the original data. Answering whether the GANs can reproduce the statistical frequency distribution and correlation within the different signals of the WESAD dataset, depending on the stress state. Furthermore, we test the fidelity by using several binary machine learning classifiers to verify the indistinguishability between the synthetic datasets and the original dataset. Finally, we train the CNN architecture of Gil-Martin et al. [4] to verify the usefulness. Finally, we apply DP to the best of these GAN architectures and can use it to generate privacy-preserving smartwatch data, on which stress detection can be performed with some performance trade-off. We do this to address another research question, whether it is possible to apply DP to the GAN architectures for our use case.

After providing a brief overview of the motivation and goal of this thesis, in Chapter 2 we will introduce background and fundamental concepts to understand the applied methods. We first give an overview of the role of smartwatch health data for stress detection. Then, we present the fundamental idea, functionality, and challenges of GANs, explain DP and elaborate the application of DP to GANs. In Chapter 3, we consider related work on which our approach to synthetic smartwatch generation is built. We introduce the WESAD dataset, related GAN architectures, stress detection for evaluating the synthetic dataset, Privacy in Synthetic Data and *Differential Private GANs* (DP-GAN). Chapter 4 provides a presentation of the methodology we use in this thesis, explaining choices we have made. The chapter offers a comprehensive workflow from preprocessing the data, to the different GAN architectures implemented, to the evaluation of the synthetic data, and the application of DP to our GAN architecture. Following this workflow overview, Chapter 5 explains the detailed experimental setup. This includes technical implementation details from the software and libraries used to the specific GAN architectures to the training parameters. Furthermore, we describe the implementation of the stress detection CNN model for the usefulness evaluation of the synthetic data. Finally, we give an overview of the training setting used to implement DP on a GAN architecture. Chapter 6 evaluates the results of the synthetic dataset generated by the three GAN architectures with respect to evaluation facets we introduced. We consider the diversity, the fidelity as well as the usefulness of the synthetic data. In order to test the usefulness, we train several stress models based on the synthetic GAN datasets and evaluate this in two procedures:

the augmentation of the real dataset by synthetic subjects and the *Train on Synthetic Test on Real* (TSTR) method. Chapter 7 discusses the findings from the previous chapter, highlighting the advantages and disadvantages of GANs in general and specific to the individual architectures. At the same time, it shows the practical implications of our work for research and smartwatch health care data in general. Moreover, we present a user-friendly frontend developed by us for generating the synthetic data using the GAN models trained in this thesis. In Chapter 8 we summarize the presented work and give an outlook on possible further research questions resulting from this work.

2. Background

In this chapter, we explain the fundamental concepts of this thesis. In Section 2.1, we provide an overview of the role of smartwatches in stress detection and its importance. Section 2.2 covers the fundamentals of GANs and how they can be used to generate synthetic data. We introduce two different GAN extensions that are important for further work and the challenges that each architecture presents. Finally, Section 2.3 explains differential privacy and how it can be used to protect the privacy of synthetic data in the context of GANs.

2.1. Understanding Smartwatch Health Data for Stress Detection

In the following section, we describe the role of wearable devices in healthcare and individual use for self-diagnosis and personal analysis. We describe the opportunities and different uses of wearables in healthcare and introduce one area of application: stress detection. We discuss the health consequences of long-term stress and explain the physiological responses of the human body to stress. We then show how stress can be measured with the Empatica E4 Wristband. We also describe how the individual sensors can measure the body's physiological stress responses and how these sensors work. We also present different approaches to stress detection and provide a guide to machine learning-based stress detection based on physiological stress data from the Empatica E4. Finally, we address privacy concerns related to health data.

2.1.1. The role of wearables in healthcare

Piwek et al. [10] point to the increasing use of wearables for self-diagnosis and personal analytics. Wearables have the ability to monitor patients over time to detect chronic diseases [10, 11]. Wearables can also serve as more comprehensive systems of predictive, preventive diagnostics for early detection of health problems. The authors believe that in the future, wearables will be a standard part of the healthcare system, providing both individual and aggregated data to patients, governments, and healthcare providers. This type of data aggregation holds great promise for the future of healthcare, however due to the highly sensitive nature of the data, it requires the development of functional use cases and privacy-compliant handling. One such functional use case of particular interest to both individuals and healthcare systems is stress response detection.

2.1.2. Stress and its health consequences

Stress, a common physiological response to stressful situations, can have significant health consequences. Long-term stress can cause many serious health problems, including headaches, sleep disturbances, increased risk of cardiovascular disease [12, 13, 14], increased susceptibility to infection, slower physical recovery, and decreased mental performance [15]. In addition, as mentioned earlier, wearable devices such as smartwatches offer the ability to measure health conditions, including stress, relatively unobtrusively over time. In order to use smartwatches for stress detection, we must first have an understanding of the nature of stress and the methods by which it can be measured. During a stressful event (e.g., exam, meeting, sport) the body is under great physical or psychological stress and the sympathetic nervous system is activated to meet metabolic demands [16, 17]. To do this, certain processes in the body are accelerated, the *fight-or-flight* response, causing heart rate, blood pressure, and sweat rate to increase [18, 17]. After the stress event, the parasympathetic nervous system engages and initiates rest and repair processes to restore balance. Chronic stress occurs when the sympathetic response is continuously elevated and there is no balance between the sympathetic and parasympathetic systems. It is more important to prevent chronic stress than to treat it, and to make this possible, stress detection systems are available. Unobtrusive wrist devices are able to detect these bodily responses, namely Empatica [19] or the Microsoft Band.

2.1.3. Empatica E4 and its sensors

In particular, the Empatica E4's[2] ability to accurately measure physiological stress indicators has been validated in several studies [20, 21, 22]. This device includes a 3-axis hand-wrist *accelerometer* (ACC_x, ACC_y, ACC_z), an *Electrodermal Activity* (EDA) sensor, an optical skin thermometer measuring the *skin temperature* (TEMP), and a *Photoplethysmography* (PPG) sensor measuring blood volume pulse (BVP), heart rate (HR), and heart rate variability (HRV).

The EDA sensor measures the electrical conductance at the surface of the skin. Because the sympathetic nervous system controls the sweat glands, EDA can measure an indication for physiological arousal [23]. A high electrical conductance is an indicator of an increased level of stress and arousal. This is measured using a tiny amount of current passed between two electrodes in contact with the skin. The conductance is measured in *microSiemens* (μS). The Empatica device measures EDA with a sampling rate of 4Hz (4 times per second) for conductance in the [0.01, 100]μS range [19].

The PPG sensor, which measures blood volume, pulse, and heart rate, works by illuminating arteries filled with blood with green and red light. The arteries reflect the light according to their oxygen-saturated hemoglobin. The sensor then measures the reflected light. BVP is measured at a fixed sampling rate of 64Hz (64 times per second). The BVP signal can be used to calculate HR and HRV. Increasing heart rate and decreasing HRV are indicators of stress [24, 25].

The optical skin thermometer is an infrared thermopile sensor. It captures the infrared radiation emitted by the skin without contact and can generate a voltage from the temperature difference in the sensor components and use this current to determine the skin temperature [26, 19]. It is important to note that skin temperature is not representative of core body temperature. During a stress response, the sympathetic nervous system can constrict blood vessels, called vasoconstriction, which can cause skin temperature to drop. A very low skin temperature is therefore a possible indicator of stress [27]. The Empatica device measures TEMP with a sampling rate of 4Hz (4 times per second). It is calibrated for a temperature range of -40 to 115 °C and measures with an accuracy of ± 0.2 °C. However, it must be noted that the average skin temperature varies between 33.5 and 36.9 °C [28].

[2]https://e4.empatica.com/e4-wristband

The ACC sensors measure the acceleration force in the 3-dimensions X, Y, and Z at 32Hz (32 times per second). The sensor itself does not provide an indicator of moments of stress, but contextualizes the physiological signals, taking into account physical activity and movement.

2.1.4. Methods for stress detection

To detect stress, researchers have developed various methods for stress detection [29]. Mirsamadi et al. [30] detect and analyze stress and emotions based on speech by using recurrent neural networks. Tzirakis et al. [31] detect stress and emotional affection by analyzing video content using deep neural networks.

In addition, researchers have also developed stress detection systems based on the previously presented physiological signals captured by sensor modalities [32]. Therefore, theses signal data is analyzed and used to train various traditional machine learning algorithms such as Decision Tree, Linear Discriminate Analysis (LDA), and K-Nearest Neighbor (KNN). Healey and Picard [33] conducted one of the first studies on stress detection using physiological data. To do this, they first recorded stress data by measuring the relative stress levels of subjects in different driving situations on the road. The measured physiological signals were recorded with an electrocardiogram, electromyography, respiratory rate sensor, and EDA sensors. These data were then processed into 22 hand-crafted features, which then formed the input to the LDA machine learning algorithm for binary classification into stress and non-stress states. Subsequently, other researchers used different combinations of physiological stress data with various traditional machine learning methods.

Siirtola [34] focus his research on stress detection using sensors of commercially available smartwatches. The study found that different combinations of sensors can be used to detect stress. The researcher uses the WESAD dataset [5], which is recorded with the four sensor modalities from the Empatica E4 wrist-worn device. To detect stress, the author uses different combinations of sensor signal data and three different machine learning algorithms: quadratic discriminant analysis, random forest, and LDA. The best stress detection of 87.4 is obtained with a sensor signal combination of TEMP, BVP and HR using LDA.

Subsequently, other researchers applied deep learning approaches to the physiological sensor data to counteract the laborious part of the manual feature extraction and computation [4, 29]. Deep neural networks are able to extract the features themselves from the raw data. For this purpose, both Li et al. [29] and Gil-Martin et al. [4] use a CNN for feature extraction and stress detection.

2.2. Generative Adversarial Network

To generate a medical dataset, Imtiaz et al. use GANs [35]. In 2014, the GAN was introduced by Goodfellow et al. [36]. GAN is a class of machine learning frameworks, which consists of two neural networks that are trained simultaneously. The generative model G that captures the data distribution, and the discriminative model D that calculates the probability if a sample originates from the dataset rather than G. Fig. 2.1 shows the architecture of the original GAN. The generator G aims to create realistic sample data. Then, the discriminator D receives the synthetic-created samples and real samples. The discriminator classify each sample as either real or fake. The generator does not have direct access to the real sample. The interaction with the discriminator is the only way it learns. Both the synthetic samples and the real dataset samples are available to the discriminator. The discriminator receives the error signal based on the ground truth of knowing whether the sample originated from the real dataset or from the generator. The same error signal is then passed through the discriminator to train the generator, resulting in the generation of better synthetic samples. The generator network can be considered as a mapping from a representation vector space, the so-called latent space z, to the representational data space x, e.g., an image. Formally, this can be written as $G : G(z) \rightarrow \mathbb{R}^{|x|}$, where $z \in \mathbb{R}^{|z|}$ and $x \in \mathbb{R}^{|x|}$ depicts the data representation. The discriminator can formally be written as $D : D(x) \rightarrow (0, 1)$, where D describes a mapping from the input data to the probability of belonging to the fake data (close to 0) or to real data (close to 1). G is trained to maximize D's probability of making a mistake. Training takes place in the form of a min-max game, which is defined by the objective function $V(G, D)$.

The original min-max game objective function [37] is:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

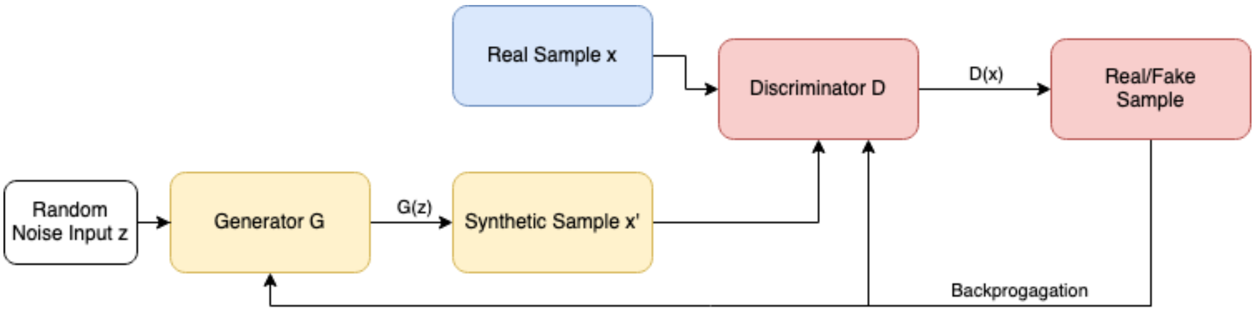


Figure 2.1.: Overview of the fundamental GAN structure: The generator G produces a synthetic sample x' based on a random noise input z. The synthetic samples x', along with real samples x, are the input to the discriminator D, that classifies each sample either as real or fake. Based on this classification, the loss is calculated to update the generator and discriminator via backpropagation.

2.2.1. Challenges

Saxena and Cao [38] highlight several challenges and their solutions in the application and training of GANs. The main challenges are *mode collapse*, *non-convergence and instability*, and metric *evaluation*.

Mode collapse. Mode collapse is one of the main reasons for unstable GAN training. The mode collapse problem is shown in 2.2.

The problem occurs when a generator G, i.e. the mapping function from the latent space to the data space, maps several different inputs $z_1, z_2, \ldots, z_n$ to a very similar or identical output such that the following holds [39]:

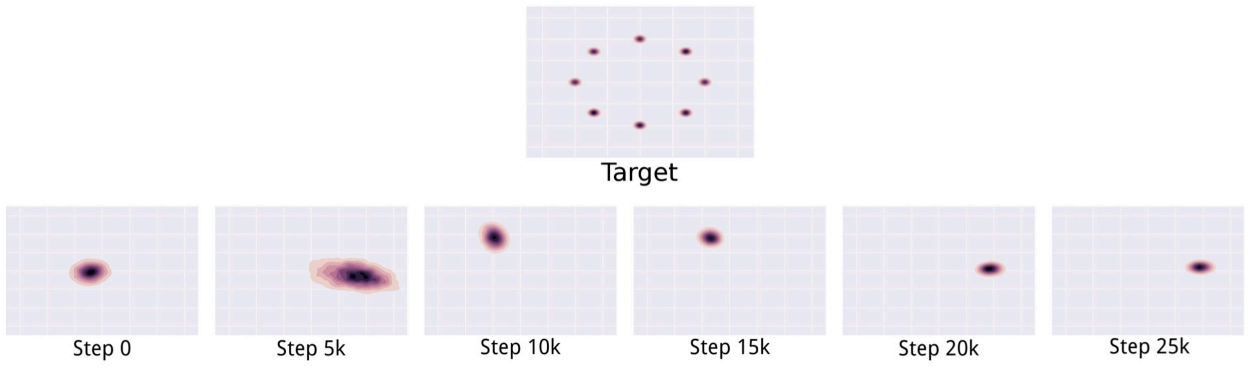


Figure 2.2.: The figure shows the mode collapse problem for a two-dimensional toy dataset. At the top is the target distribution that the GAN should learn. Below is the learned distribution over time during training. Instead of learning the distribution, the model generates the different training modes alternately. However, it is not able to learn the original distribution. Image is taken from [39, 1].

$$G(z_1) \approx G(z_2) \approx \ldots \approx G(z_n)$$

This can happen when the generator generates a sample that fooled the discriminator over and over again, and as a result keep producing this sample. If the training of the generator is not changed by an incentive such as an additional or changed loss function, mode collapse will occur.

Non-convergence and Instability. Since the generator and the discriminator are in a min-max game where they both want to minimize their own loss function and maximize the loss function of the other, GAN training can become unstable. The model parameters are constantly changing, which destabilizes the model and it never converges.

Evaluation. Another challenge in GAN training is the quantitative evaluation of the generative model. While qualitative evaluation (visual inspection) of generated images, audio samples, or time-series can often determine whether a generative model performs well, there are no real established metrics in the field of GANs that can quantify the performance of a generative model. For example, on the one hand a model may achieve supposedly low loss values for generator and discriminator, or come close to the statistical distribution of the original dataset, but the generated samples are poor. On the other hand, supposedly quantitatively poor GANs may produce good samples [39].

To avoid mode collapse and non-convergence, Saxena and Cao [38] summarize to use the appropriate architecture and objective function, and the optimal optimization techniques.

2.2.2. Conditional GAN

The *conditional GAN* (cGAN) is an extension of the GAN framework introduced by Mirza et al. [40]. In this framework, both the generator and the discriminator receive additional auxiliary information such as a class label as input. This auxiliary information allows cGAN to achieve better results when generating multimodal data [41]. Figure 2.3 illustrates the structure of a CGAN. The objective function of a cGAN is:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y)))] \tag{2.2}$$

where the only difference to Equation 2.1 is that the generator and discriminator additionally receive the conditional class label y in the generation function $G(z|y)$ and the discriminator function $D(x|y)$.

In addition to the challenges of GANs already mentioned in Section 2.2.1, there is the additional challenge with cGANs of whether the additional label really affects the training process and the generation of synthetic data.

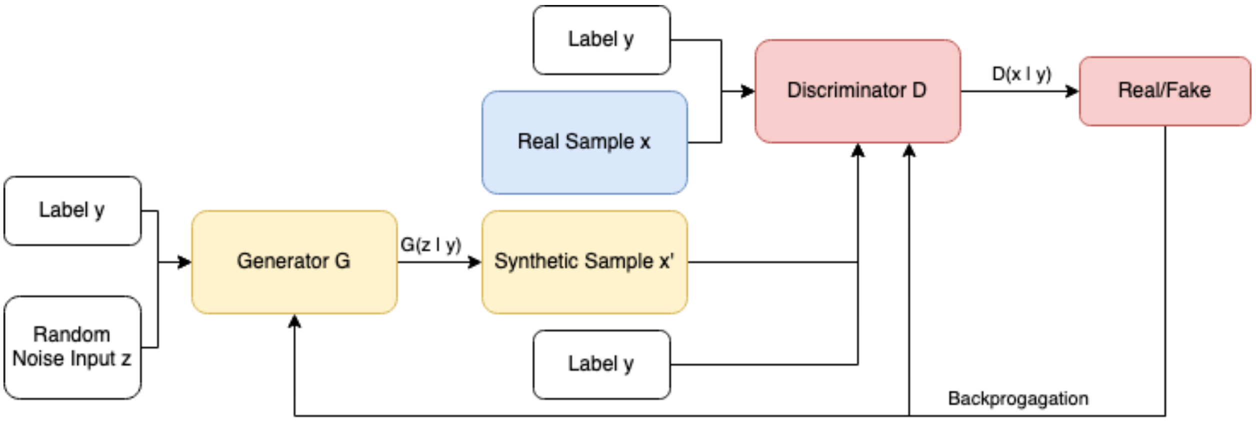


Figure 2.3.: Overview of the cGAN structure: The generator produces a synthetic sample based on a random noise input z and an auxiliary information label y. The synthetic samples, along with real samples, are the input to the discriminator and a corresponding auxiliary information label y, which must classify whether each sample is real or false. Based on this decision, the loss is calculated to update the generator and discriminator via backpropagation.

2.2.3. Time-series GANs

Building on the development of GANs and their success in generating image data, researchers developed *time-series GANs* that can learn and reproduce the distribution of time-series data. Essential to this type of data is that the GAN models reproduce the temporal relationships between data points over time.

Brophy et al. [42] define time-series data as a sequence $xt = x_1, \ldots, x_n$ of vectors which depends on time t for continuous/real and discrete time. The data can be univariate or multivariate, and time-series can be continuous or discrete.

The difference with the canonical GAN is that the first GAN architecture used fully connected neural networks for the generator and discriminator [37]. However, these are not good at learning temporal dependencies like a *recurrent neural networks* (RNN). RNNs, on the other hand, can optimally capture sequential data such as financial data, medical data, text, and speech modeling due to their loop-like structure [42]. The only difficulty is long-term data, which RNNs cannot understand by themselves. For this purpose, Hochreiter et al. [43] introduced the *Long Short-Term Memory* (LSTM) network, a special form of RNN capable of learning temporal dependencies over a longer period of time. In addition, LSTMs offer the advantage of addressing the vanishing and exploding gradient problem to some degree that affects conventional RNNs [44]. In terms of these capabilities, RNNs, and especially in the form of LSTMs, are the fundamental architecture of generator and discriminator for time-series GANs. In addition to the challenges that affect basic GANs and are described in section 2.2.1 time-series GANs have to address the following challenges [45]:

Temporal Dependencies. One of the problems here is the mapping of temporal dependencies, which is addressed by using LSTM to generate the sequences, but not definitively solved.

High Dimensionality. Time-series data, if not exactly univariate, often have highly complex correlations between multivariate features, which the time-series GAN must learn in addition.

Evaluation. The evaluation of time-series GANs is further complicated by the two points mentioned above, the correlation between the multivariate features of the dataset and the temporal dependencies.

2.3. Differential Privacy

Differential Privacy (DP), introduced by Dwork et al. [46], is a mathematical concept for measuring the degree of privacy of an individual in a dataset. Academia and industry, such as Apple [47], Google [48], and US Census [49], have adopted DP as a standard for data protection [50].

Intuitively, any algorithm A applied to a dataset D is differentially private if the output does not indicate whether someone's individual data is present in the dataset or not. Thus, if a data point belonging to an individual were removed from the dataset D resulting in a new dataset D', the change in the output must be statistically indistinguishable. Datasets that differs in only one element are called *neighboring datasets.*

The privacy guarantee is represented by the privacy budget ϵ. This is the measure of privacy loss or privacy preservation. A small ϵ (≤ 1.0) means that the distance between two output probabilities of an algorithm A applied to two neighboring datasets D and D' is small. The closer the ϵ value is to zero, the higher the privacy. Thus, $\epsilon = 0$ means that the algorithm A with two neighboring datasets, corresponding to two different inputs, will produce exactly the same output twice. The value $\epsilon = 0$ guarantees privacy preservation without any loss of privacy. However, this also results in no data accuracy. Thus, the algorithm suffers a performance loss [49]. At the same time, the expression $\epsilon = \infty$ expresses the non-private case [9]. In this case, the privacy loss is maximal with no privacy preserved, but the algorithm A operates with the optima accuracy based on the data. This leads to a trade-off between privacy and performance.

δ is the probability that the privacy loss is not bounded by ϵ. The optimal value for δ is less than the inverse of the dataset size, i.e. $\frac{1}{|D|}$. (ϵ, δ)-DP is given if two datasets that differ by a single entry are statistically indistinguishable, and the following holds:

$$\Pr[A(D) \in S] \leq e^{\epsilon} \Pr[A(D') \in S] + \delta \tag{2.3}$$

where S is the set of all possible results that can be produced by the algorithm A applied to the dataset D and D'.

To implement these privacy guarantees, noise is added. One way to implement this is the *Differential-Private Stochastic Gradient Descent* (DP-SGD) algorithm of Abadi et al. [51]. The algorithm works by computing the gradient at each SGD step for a subset of random data points by first clipping it and then adding noise.

2.3.1. Differential Privacy in GANs

The application of noise in terms of privacy preservation needs to be applied to all neural networks that access the real data. In case of a GAN, the noise is only applied to the discriminator during the training. This comes back to the post-processing property [52] that states that if data has been properly privacy-preserved, further calculations and transformations on this data do not reduce the privacy level. When applied to GANs, this means that applying DP to the discriminator ensures the privacy guarantee for the GAN architecture.

3. Related Work

The following Chapter focuses on related work that underlies our approach to generating synthetic data with different time-series GANs. First, in Section 3.1, we present recent developments in the field of health data generation, as physiological data for stress detection fall into the domain of health and medical data. We consider the importance of high quality data in medical research and discuss issues related to privacy and sensitivity of these data. We also show how synthetic data can provide a remedy and what technologies have been and can be used to generate them.

In Section 3.3, we briefly discuss the initial challenges faced in the development of time-series GAN architectures, and how these challenges have been addressed by different techniques in research. This discussion leads us to the three main architectures that we have used or extended in this work: TimeGAN, cGAN, and *DoppelGANger GAN* (DGAN).

Finally, in Section 3.5 we review related work on privacy in synthetic data and applying DP to GAN architectures to enhance the privacy guarantees of our synthetic datasets.

3.1. Using Synthetic Data for Analysis in Medical Research

Medical and healthcare research has a growing need for high quality data. In particular, electronic health record data provide valuable real-world evidence about health conditions of individuals, leading to new hypotheses, and methods. These datasets, based on individual clinical trials, offer great analysis potential in meta-studies.

However, a major problem lies in the sensitive nature of these data. Each data point contains extensive information about an individual person and their health status. Therefore, this data is highly sensitive, and many complex privacy requirements make it difficult to access [53].

Kokosi and Harron [53] provide an overview of different approaches for generating synthetic data. The goal of all these approaches is to create a dataset that mimics the properties of the original dataset at the individual level, taking into account sample size and matching in the respective columns. Due to the often complex and high-dimensional data in medical research, machine learning methods are mainly used. These are able to reproduce the distribution of the data and the correlations within the data points well. The machine learning methods include Bayesian networks, which use sampling techniques to synthetically create the dataset, or deep learning methods such as GANs. GANs are becoming increasingly popular for creating synthetic data [54, 55, 56]. A significant benefit and use case of synthetic datasets in medical research is their potential for preliminary development and testing of code for hypotheses before accessing the real dataset. Researchers can leverage synthetic datasets to evaluate and refine their methods, improving subsequent analysis. Another advantage is the preservation of anonymity that synthetic data provide to some degree. Each sample cannot be specifically associated with an individual, but is a probabilistic sample from a mimicked statistical distribution of the original dataset. This results in less sensitive patient information being directly accessible. However, the supposed preservation of anonymity is limited by the mere synthetic generation, as we will discuss later. Furthermore, as noted by Azizi et al. [57], there is an increasing demand to expand the availability of research data for secondary analysis. For this, synthetic datasets can be used to share with other researchers for reproducibility without providing the real dataset to verify models [53].

Gonzales et al. [58] review the state of synthetic data in healthcare in 2023. They show that synthetic data can be used in a variety of ways and can often bridge data gaps. They also predict that there will be more synthetic datasets, more tools for generating synthetic data, and more users or researchers using these tools and datasets. They list a number of uses for synthetic data in healthcare, such as simulation and prediction [59], testing hypotheses, methods, and algorithms [60], epidemiologic studies for public health [61], education and training [62], and linkage testing [63].

3.2. An Overview of the WESAD Dataset: Physiological and Motion Signals for Stress and Affect Detection

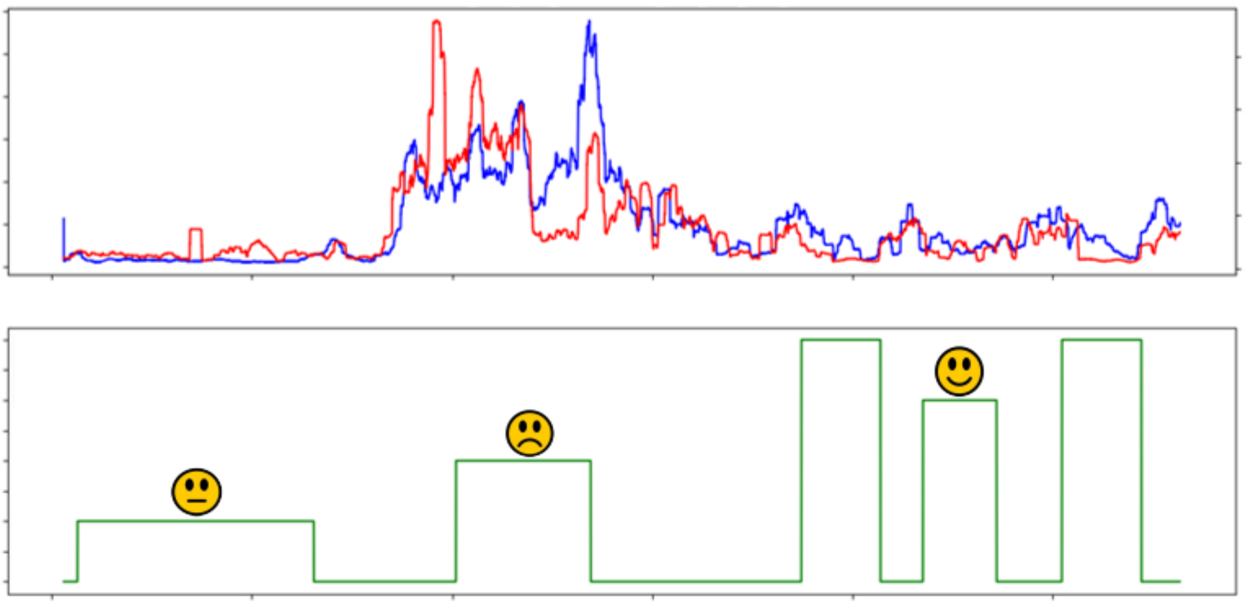


Figure 3.1.: Example: Graphical representation of subject data over the course of the different lab sessions. the different episodes are indicated by the labels: Neutral, Stress, Amusement. [5].

Schmidt et al. [5] introduce the publicly available **WE**arable **S**tress and **A**ffect **D**ataset (WESAD)[3] dataset for wearable stress and affect detection. The dataset includes physiological and motion signals from 15 subjects (12 males and 3 females). The signals are recorded using both a RespiBAN device on the chest and an Empatica E4 wristband in a laboratory setting. It includes several sensor modalities measured by the RespiBAN, such as: respiration, body temperature, three-axis body acceleration, electrocardiogram (ECG), electromyogram, and electrodermal activity (EDA) sampled at 700 Hz. The Empatica E4 device also recorded ECG, EDA and additionally blood volume pulse (BVP), arm temperature (TEMP) and three-axis hand acceleration (ACC_x, ACC_y, ACC_z). Unlike the RespiBAN, these signals were measured at different sampling rates and downsampled

[3]https://ubicomp.eti.uni-siegen.de/home/datasets/

to 1 Hz for further experiments in this work. The dataset covers three different affective states (neutral, stress, amusement) as shown in Figure 3.1. To obtain these signals, all 15 subjects went through 5 phases of a laboratory protocol during a ninety-minute session.

This protocol consists of the following stages:

- Baseline phase: the participant remains in a neutral position and reads a magazine for 20 minutes. This can be done standing or sitting. This way, the researchers try to induce a neutral affective state.
- Amusement phase: the subject watches 11 funny videos for a duration of 6 minutes
- Stress phase: The stress phase is induced by exposing the subject to the Trial Social Stress Test (TSST) for 10 minutes. In Schmidt et al.'s [5] TSST, the test consists of two parts, a "public speaking" unit and a mental arithmetic task, each lasting 5 minutes duration in front of a three-people panel, which they were told were supposed to be human resource specialists from the research facility.
- Meditation phase: an expert guides the subject back to neutral affective mood using a breathing protocol in a seated position with eyes closed.
- Recovery phase: the subject is informed that the panel members are "normal" researchers, the sensors are synchronized with a double-tap gesture, and the sensors are removed.

Although the dataset includes self-reports from the subjects, these are not considered in this thesis. Only the wristband signals measured by the Empatica E4 are used in the following experiments, as the work focuses on smartwatch data.

3.3. GAN Architectures

3.3.1. Time-Series Based GAN

After the GAN was introduced by Goodfellow et al. [36] in 2014, it was essentially used and known for media generation [64, 65]. The first fundamental proposal for time-series GAN happened in 2016 when Jiwoong et al. [66] introduced the recurrent GAN for image generation. Building on these foundations and the initial introduction of recurrent GANs in computer vision, Esteban et al. [67] introduced a Recurrent GAN (RGAN) and a Recurrent Conditional GAN (RCGAN) for continuous data generation in 2017. Until then, only one other GAN, the C-RNN-GAN by Mogren et al. [68] aimed at generating continuous-valued sequences, which were polyphonic classical music sequences. Both GANs by Esteban et al. [67] focus on the generation of real-valued multidimensional time-series data in order to use this synthetic data to support applications based on medical data. The architecture of the two GANs is similar in that both use recurrent neural networks for the generator and discriminator. For the RCGAN, additional conditional auxiliary information is given to both neural networks to control the generation of the sequences. To evaluate the two GANs and their corresponding synthetic sequences, the researchers introduce the novel method of the *Train on Synthetic Test on Real* (TSTR) evaluation framework. They generate a synthetic dataset, train a classification model on this synthetic dataset, and evaluate the performance of this model on a real test set. This evaluation method is also used in the following thesis. Based on this evaluation, the authors train the RCGAN on two classification applications. A digit classification task on the MNIST dataset and an early warning system on a medical dataset of 17,000 patients from an intensive care unit. The results show little degradation in model performance, and the authors conclude that the synthetically generated dataset is suitable for training models for the classification tasks.

Due to the challenges inherent in training GANs, and in particular time-series GANs, time-series GANs mostly employ LSTM neural networks to overcome the vanishing gradient problem often encountered in conventional RNN architectures [44]. This is also true for the three previously presented approaches, RGAN, RCGAN, and C-RNN-GAN, which implement the recurrent aspect using two LSTMs for generator and discriminator. In 2019, Yoon et al. [69] propose the TimeGAN, an architecture that utilizes the traditional unsupervised GAN training and a supervised learning approach. By combining these approaches, the authors preserve the temporal dynamics within the time-series data and respect the ground truth relationships between the multivariate sequential variables over time. The researchers were able to improve upon the state-of-the-art time-series GANs (RCGAN and C-RNN-GAN). They provide a novel GAN architecture consisting of four network components, including an embedding function, a recovery function, a sequence generator, and a sequence discriminator. The embedding function translates real sequences and random vectors into the latent space, where the adversarial network consisting of the sequence generator and the sequence discriminator learns the temporal dynamics in the latent space representation of the sequence. Finally, the recovery function reconstructs the feature space back into the latent space. These functions are trained simultaneously over different loss functions for each function itself.

3.3.2. cGAN

In 2014, Mirza and Osindero [40] provide an architecture named conditional GAN. With this architecture, the authors extend the canonical GAN by Goodfellow et al. [36] by feeding the generator a random prior noise z and an additional auxiliary information y such as class labels. The discriminator receives the auxiliary information y and the sample x to be discriminated. To add the additional information, the random noise z and the auxiliary information y are combined into a joint hidden representation for the generator.

Building on this idea, Ehrhart et al. [70] propose a data augmentation technique using a cGAN architecture to handle the imbalance in a physiological measurement dataset in a stress detection use case. The authors are inspired by existing time-series GAN approaches. However, instead of using LSTM networks for the generator and discriminator as in previous research, they use an LSTM for the generator and a Fully Convolutional Network (FCN) for the discriminator to improve the extraction of features from the physiological signal. This architecture is coupled with a diversity term that addresses the challenges of unstable cGAN training. The diversity term is a mathematical term derived from the distance between two generated samples. The authors introduce this term into the loss computation to train the generator's ability to generate different samples with different random noise inputs and not fall into mode collapse.

The imbalanced ground truth time-series data is collected with low-cost wearable sensors, the Empatica E4[4], in a previously controlled laboratory data collection campaign, where subjects were exposed to audio stimuli to induce moments of stress.

Methodologically, the authors used an LSTM as a generator and a (FCN) as a discriminator. These two networks together form the GAN for the generation of time-series. To employ the conditional property of the network, the researchers feed the stress label of the respective time-series sample into the GAN as auxiliary information, making it a cGAN.

Remarkably, by adding the synthetic data to the stress detection evaluation, the stress detection model based on an LSTM increases the detection of stress moments by 19.05% in recall and 11.03% in F1-score compared to the baseline LSTM without the synthetic data samples.

In summary, the paper by Ehrhart et al. [70] provides an architecture for augmenting a dataset by adding synthetic samples based on a small and imbalanced dataset. The researchers are conducting these experiments in the healthcare domain, using health data from smartwatches and the same application, stress detection based on wearable sensors. The data they use is also collected with an Empatica E4, the device used as an aggregation tool for the WESAD dataset.

Since this approach, the nature of the data, and the domain strongly match our research goals, in the following work we will implement the cGAN architecture as one of the GAN architectures to synthetically extend the WESAD dataset. We will also present an adaption that extends the existing approach to include DP.

3.3.3. DGAN

Lin et al. [71] propose the DGAN architecture. The research question that the authors address is whether it is possible to create high-fidelity, easily generalizable synthetic datasets for the domain of network applications. The goal is to provide a toolkit that facilitates the collection and sharing of data using data-driven techniques. Thus, their paper takes advantage of recent advances in GANs to learn high-fidelity representations of high-dimensional relationships in datasets.

In addition, the authors explore the privacy properties of their GAN architecture and conclude that further research is needed. The key challenge seems to be the shrinking fidelity while providing sufficient privacy guarantees. They formulate the problem of destroying the fidelity of different test datasets to have hard-to-interpret guarantees, while still being vulnerable to privacy leaking attacks.

The architecture of the DGAN differs from canonical approaches, in three main aspects. The authors use a decoupled generation of metadata and measurements by using an auxiliary discriminator. This allows them to condition the generated measurements based on the generated metadata. The authors address the mode collapse problem by adding the fake metadata to capture the min/max values on each generated sample. To capture the temporal correlations and generate representative time-series samples, a batch RNN generator is used.

[4]https://www.empatica.com/research/e4

The paper elaborates an approach that promises generalization to different data domains and achieves performance improvements over other existing GAN architectures, with 43% better fidelity than baseline models. At the same time, the authors show a further advantage over other state-of-the-art approaches: The training time is significantly lower (1.7 times faster than RCGAN and 15.2 times faster than TimeGAN) compared to existing time-series GANs under the same conditions and with the same dataset. Finally, their paper elaborates and critiques key theoretical aspects for time-series data that are to receive privacy guarantees via DP.

In the following work, we will use this GAN to generate synthetic data based on the WESAD dataset. At the same time, we will use Lin et al.'s consideration of user privacy as a starting point for an elaboration and outlook on DP in the use of time-series GANs.

3.4. Evaluating GANs using Stress Detection

The GANs we use require a comprehensive evaluation. Since we want to generate a synthetic stress/non-stress dataset based on the WESAD dataset, we need to evaluate the performance of this dataset in a real application scenario, stress detection. To accomplish this, we draw on the work of Gil-Martin et al. [4] who developed a stress detection model based on the WESAD dataset. They propose a deep learning architecture based on a Convolution Neural Network (CNN). Furthermore, they investigate various signal preprocessing techniques to generate the input to the CNN. They evaluate this architecture both for training on individual subsets of the WESAD dataset (physiological, motion) over the respective recording modalities (wristband, chest-band) and all data points together and in different classification experiments. In the experiment of interest, they report an accuracy of 92.7 and an F1-score of 92.55 for the binary classification of stress and non-stress situations based on the wrist dataset. For the following work, we use a previously implemented solution of the proposed approach of Gil-Martin et al. by Sülzle and Wenzlitschke [5]. However, this implementation only achieves accuracy and F1-score of 0.87.

3.5. Privacy in synthetic data

In the broader context of our work on generating synthetic datasets using GANs, the following section discusses the topic of privacy in synthetic data and how privacy guarantees can be applied to GAN architectures. Considering the privacy guarantees attributed to the outcome of data synthesis and conventional GANs, one might think that additional efforts to implement DP are obsolete [53, 72, 73]. Therefore, the idea is repeated that data synthesis poses little privacy and anonymity risk to of the original data because the synthetic samples are not identical to the samples of the original distribution in terms of an explicit one-to-one match [74].

However, researchers have shown that the assumption of privacy guarantees based on a synthetic dataset is not valid. It has been shown that generative models and GANs in particular can leak

[5]https://github.com/peasypi/Stress-Detection-From-Wearables

information about the original training patterns from a synthetic dataset [75, 76, 77]. Thus, generative sequence models can suffer from unintentional memorization of training data, such that a given synthetic sample is identically similar to a training sample [78]. This weakness of a generative model can be exploited by attackers using membership inference attacks to learn whether a data point known to the attackers is present in the original data [79, 74]. The attacker uses the output data from the machine learning models and applies statistical or machine learning techniques to determine whether certain information was present in the generative models training set. Membership inference is a privacy violation because the target has not necessarily given consent.

3.5.1. Differential Privacy GANs

Researchers have subsequently addressed this issue with the introduction of DP on GANs. DP aims to protect the fundamental privacy of a dataset [45]. Building on the fundamental idea of DP of Dwork et al. [46], Xie et al. [9] introduce the DPGAN. This GAN is a proposal to make the training data of a GAN differential private. They show that this approach guarantees (ϵ, δ)-DP. The implementation of DP is achieved by applying noise to the gradients during training.

A similar effort in terms of applying DP to GAN architectures to generate smartwatch datasets is implemented by Imtiaz et al. [35]. They use a boundary-seeking GAN (BGAN) architecture and DP to generate a Fitbit-smartwatch-based smart healthcare dataset. They test this architecture in three different privacy settings and show that their approach is able to learn categorical and numerical data points for highly diverse tabular data distributions. The Laplacian noise and the corresponding ϵ are chosen depending on the attribute category, such that $\epsilon = 0.2$ is chosen for categorical or static attributes, which require higher privacy settings, and $\epsilon = 0.5$ is chosen for so-called behavioral attributes.

Regarding the different previous related works, in the following thesis we will perform the feasibility study of the DP by applying noise to the gradients, as we will explain in the further thesis. A review of related work shows that optimal training of a GAN and finding the appropriate hyperparameters is a challenge in itself.

4. Methodology

The following chapter introduces our methodology for applying GANs to the WESAD dataset to generate synthetic datasets. An overview of our workflow for generating synthetic data can be found in Figure 4.1 and is described in detail in the following sections. Section 4.1 outlines the preprocessing steps taken to prepare the time-series signal data for the following tasks. In Section 4.2, we discuss the dataset characteristics that influence the selection of suitable GAN architectures. We introduce the three implemented GAN architectures, TimeGAN, cGAN, and DGAN, providing an overview of the architectures, explaining design decision and give an overview over the corresponding workflow. Evaluating GANs in general, and time-series GANs in particular, presents different challenges. In Section 4.3, we address these challenges, propose our evaluation method decisions, and explain why a combination of these methods are indicating how well our GANs perform on the WESAD test dataset.

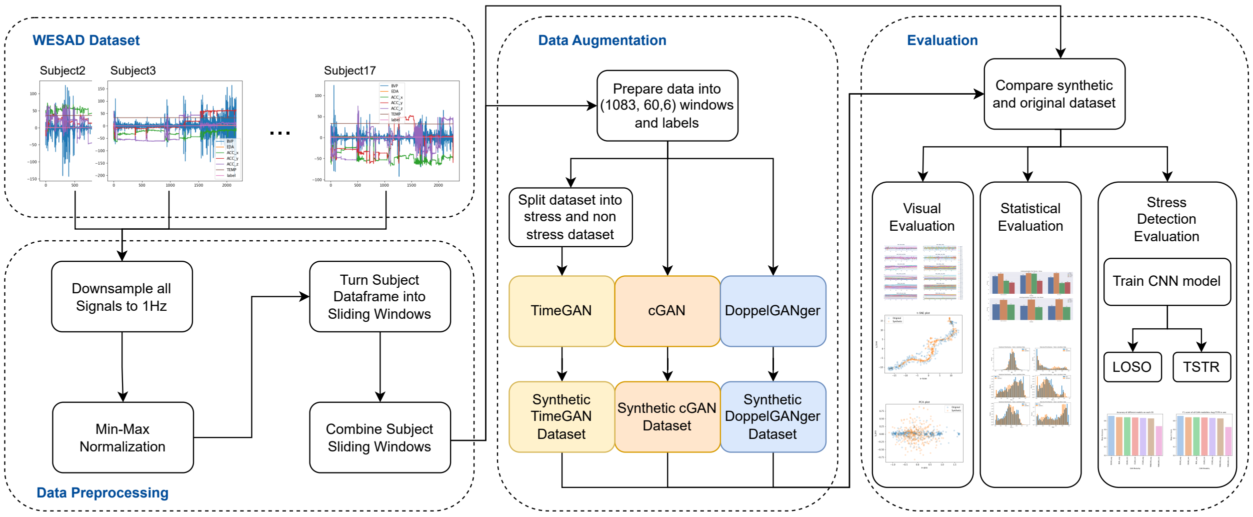

WESAD Dataset
Subject2
Subject3
Subject17
...
Data Preprocessing
Downsample all Signals to 1Hz
Min-Max Normalization
Turn Subject Dataframe into Sliding Windows
Combine Subject Sliding Windows
Data Augmentation
Prepare data into (1083, 60,6) windows and labels
Split dataset into stress and non stress dataset
TimeGAN
cGAN
DoppelGANger
Synthetic TimeGAN Dataset
Synthetic cGAN Dataset
Synthetic DoppelGANger Dataset
Evaluation
Compare synthetic and original dataset
Visual Evaluation
Statistical Evaluation
Stress Detection Evaluation
Train CNN model
LOSO
TSTR

Figure 4.1.: Shown is our workflow for generating synthetic samples to improve stress detection based on the WESAD dataset. In the first step, we load and preprocess the WESAD dataset. Then we train the GAN models for data augmentation. Each model generates a synthetic dataset. Finally, we evaluate the different datasets.

4.1. Preprocessing

Preprocessing is a crucial step in working with time-series signal data. In the following work, we have adopted the signal preprocessing of Gil-Martin et al. [4]. First, the Empatica E4 measurement signals were recorded at different sampling rates. Therefore, they need to be resampled to a unified sampling rate and aligned for further processing to ensure that each point in time has corresponding samples in each signal. All signals are downsampled to a consistent sampling rate of 1Hz using the Fourier method. Thus, despite the reduction in data points per second, the crucial non-stress/stress dynamics can still be captured, while training of the GANs is greatly accelerated.

Second, the labels are adjusted so that the labels *baseline* and *amusement* are labeled as *non-stress* labels. The data points associated with the other labels are removed from the dataset. This is done because we are only considering the stress and non-stress data points that are relevant to binary stress detection. A first exploratory data analysis on the dataset showed that the signal data collected during the amusement phase in the laboratory is crucial for stress detection because there is an overlap in some signal characteristics, such as BVP or ACC, for amusement and stress. In order to develop a stress detection system that can also robustly discriminate between amusement and stress situations, we perform this label adaptation. Since the subjects' signals were measured sequentially in different phases of the laboratory session, we can remove the data points from the meditation and recovery phase without potentially changing the trends of the signals. After relabeling, there are 23,186 *non-stress* labels and 9,966 *stress* labels.

Third, the signals are *min-max normalized* to the range [0,1], eliminating the scaling difference among the signals while still capturing the relationships within the data. In addition, the normalization has a great impact on the subsequent GAN training, as it helps the model to converge faster during training, thus shortening the time for the model to learn optimal weights. Figure 4.2 shows the dataset for *Subject 4* of the WESAD dataset after the resampling, relabeling, and normalization for the different signals and the corresponding stress label.

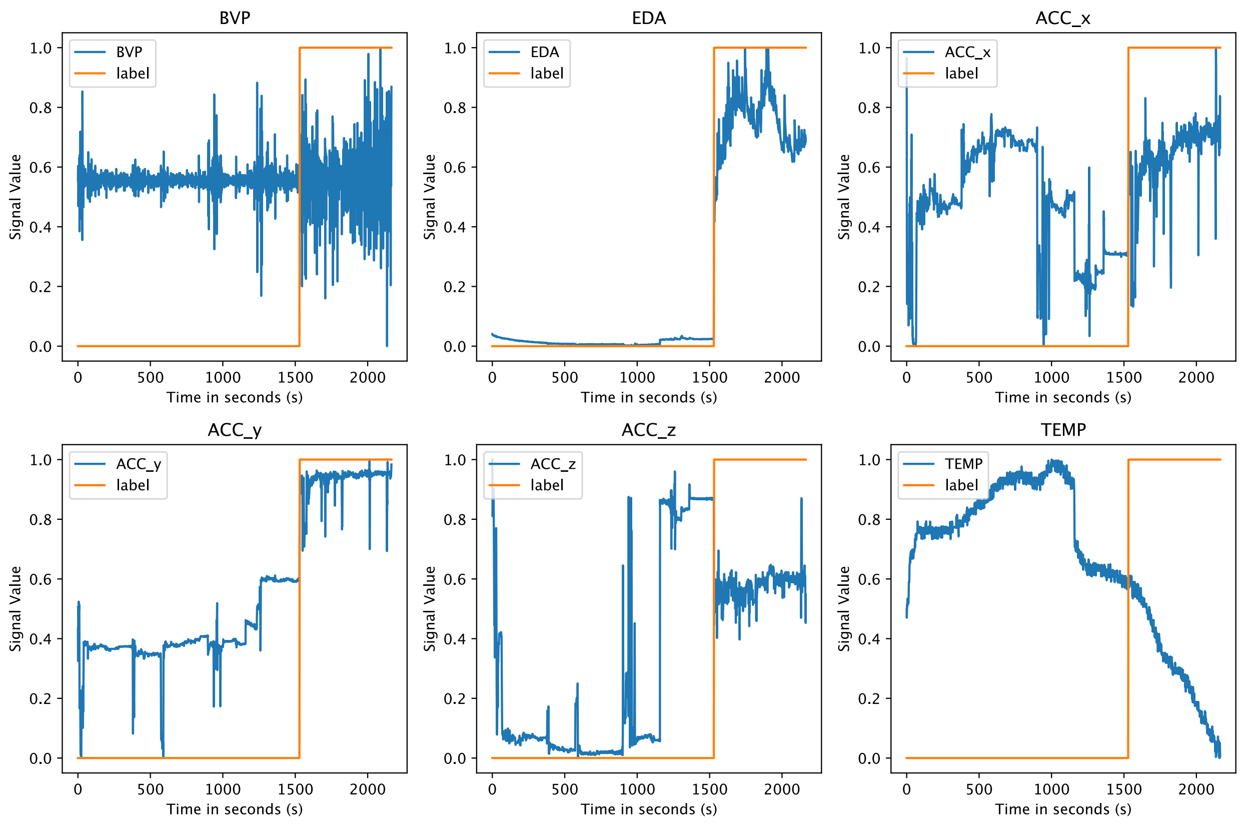


Figure 4.2.: Plot of individual signal data for *Subject 4* after resampling to 1Hz, relabeling to non-stress/stress data, and normalization with the respective stress states.

After this preprocessing, we plot a correlation matrix to validate the relationship between the respective signals. As the correlation matrix in Figure 4.3 shows, there is a strong positive correlation between the EDA signal and the stress label, with a value of 0.77. This indicates that as the EDA, i.e. the conductivity of the skin, increases, so does the stress level. TEMP has a moderate negative correlation with the stress label of -0.35. Both correlations are consistent with the understanding of physiological responses to stress that we presented in background Section 2.1.

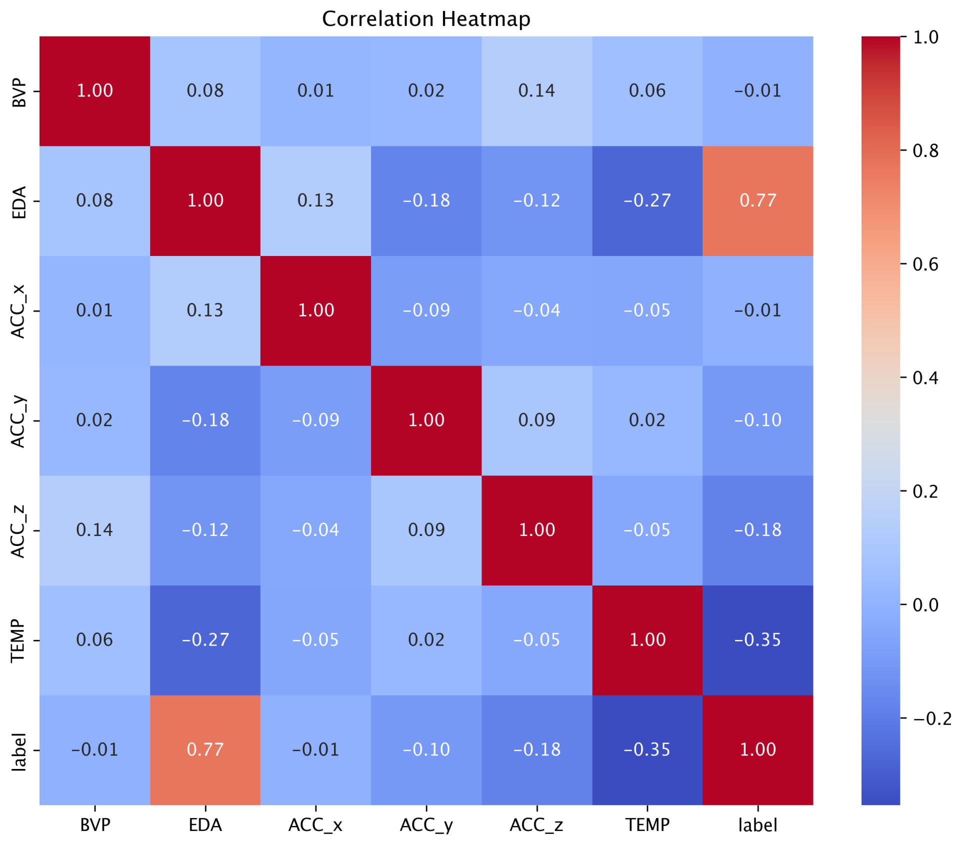


Figure 4.3.: Shown is the correlation matrix over the min-max normalized and downsampled to 1Hz signals of the WESAD dataset.

Given that the dataset consists of ninety-minute sessions per subject, we divide these long sessions into smaller, more manageable input windows. We transfer each subject's ninety-minute session into 60-second windows. In addition, as Dzieżyc et al. [80] did in their work on the WESAD dataset for research on an end-to-end deep learning approach for affect recognition, we introduce a sliding window of 30 seconds. This sliding window overlaps with the previous and next window for 30 seconds, providing more contextual information for each window. Additionally, sliding windows increases the amount of data on which the GANs are trained. We examine the stress/non-stress labels within each window. To assign a label for a window, we determine the present majority class in the 60- second window, analogous to Dzieżyc et al.[80]. Finally, we concatenate both the 60-second windows and the associated stress class labels from all subjects into a training dataset for GAN training. Figure 4.4 shows ten random sequences from the set of preprocessed sliding windows. These are shown for the individual signals for both the non-stress and stress states. The distribution of the signal sequences is characteristic for the respective stress state and corresponds to the theoretical basis on the physiological stress responses of the body from Chapter 2.1, e.g., TEMP: non-stress high / stress low; EDA: non-stress low / stress high; or BVP, which has a higher frequency in the stress state than in the non-stress state.

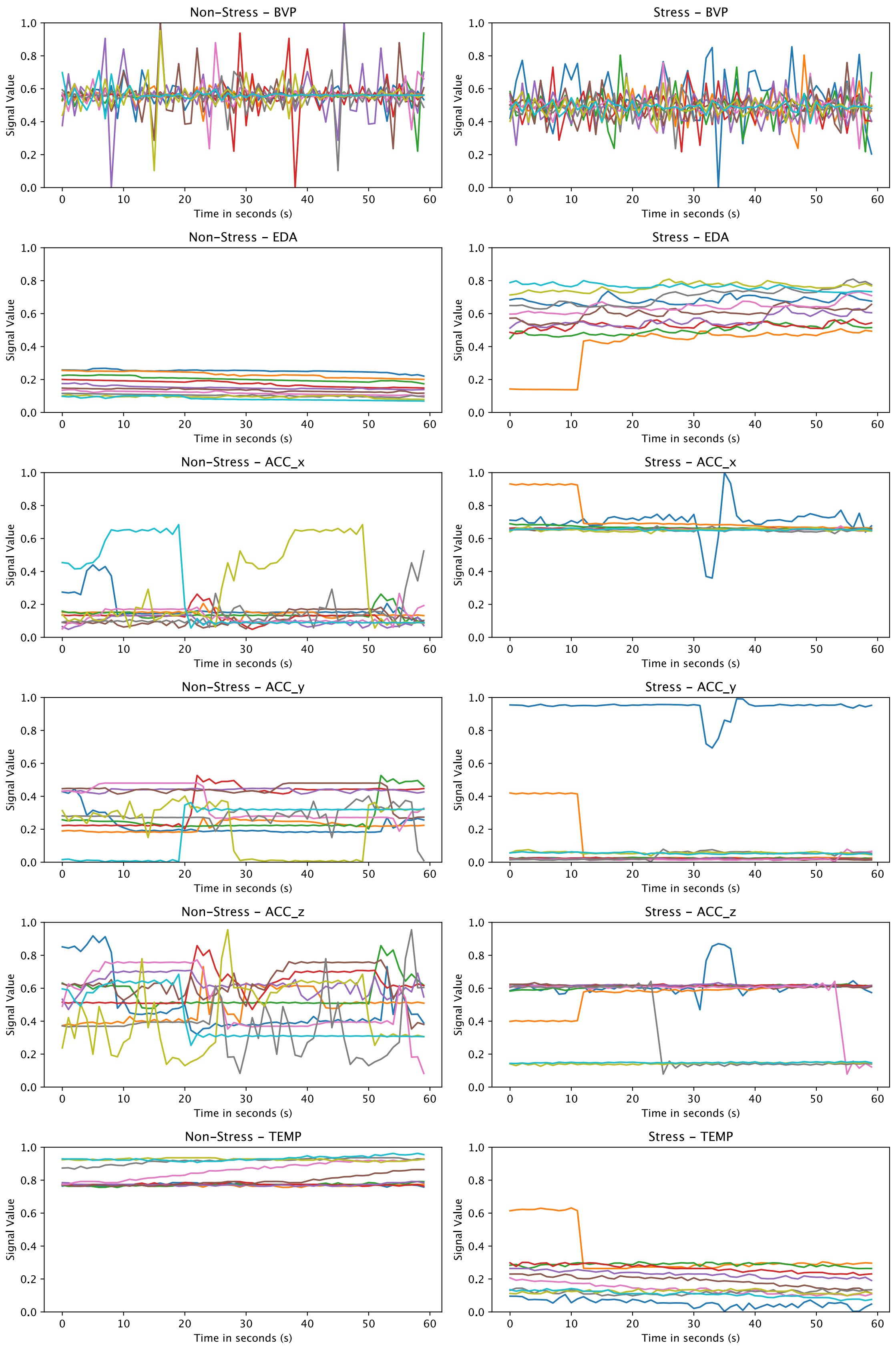


Figure 4.4.: Shown are 10 preprocessed window sequences, split into non-stress and stress samples, for each signal.

4.2. Model Architectures Overview

Data is essential for deep learning tasks, and although the goal of this work and of training a GAN is to generate synthetic data, existing datasets are necessary for this endeavor. It is equally important to understand the nature of the data and how to prepare it for the task. After bringing our signal data to a consistent sampling frequency in the previous step, it is important to determine the proper model architecture for this preprocessed data. In the research field of GANs, researchers have developed numerous architectures for each specific data form since the original paper by Goodfellow et al. [36], the more significant it is to understand the distinguishing characteristics of the corresponding data to find a proper fitting architecture. The given WESAD dataset has three main characteristics that are critical for architecture selection.

1. Each time point across all 6 signals has a stress/non-stress label.
2. It is continuous time-dependent data recorded over a time interval.
3. Each signal correlates with the other signals.

Based on these data characteristics, we have selected three different GAN architectures that address these criteria.

4.2.1. TimeGAN Overview

We introduce TimeGAN as a baseline model to measure the expected improvements over the other GANs. While TimeGAN is able to capture continuous time dependence and multi-correlation within the data, it cannot be fed additional information such as the stress label to generate conditional data. This means that TimeGAN must be trained twice to generate a synthetic dataset that covers both conditions. Once for the data non-stress labeled data, and once for the stress labeled data, our workflow for preprocessing the WESAD dataset for the TimeGAN training can be seen in Figure 4.5. This, and the fact that Lin et al. [71] show that TimeGAN is one of the GANs with the longest training time, makes this approach more cumbersome in the training process than the following ones, which build on this approach.

In order to capture not only the distribution of features within each time point, but also potentially complex dynamics over time, the authors introduce a novel mechanism that combines an unsupervised GAN with a controlling supervised training from the field of autoregressive models. The proposed model consists of four components: an embedding function, a recovery function, a sequence generator, and a sequence discriminator. In Figure 4.6 these components, the functions and the respective losses can be seen. The objective of this model is to simultaneously learn to encode features, generate samples, and iterate over time. The embedding and recovery functions convert the sequences from feature space to latent space and back. In this latent space, the adversarial network can learn the temporal dynamics underlying the data using these low-dimensional representations. The adversarial network, consisting of a generator and a discriminator, then outputs synthetic samples into this latent embedding space. To generate a sequential time-dependent sample, the generator takes a tuple of static and temporal random vectors and outputs a synthetic latent code.

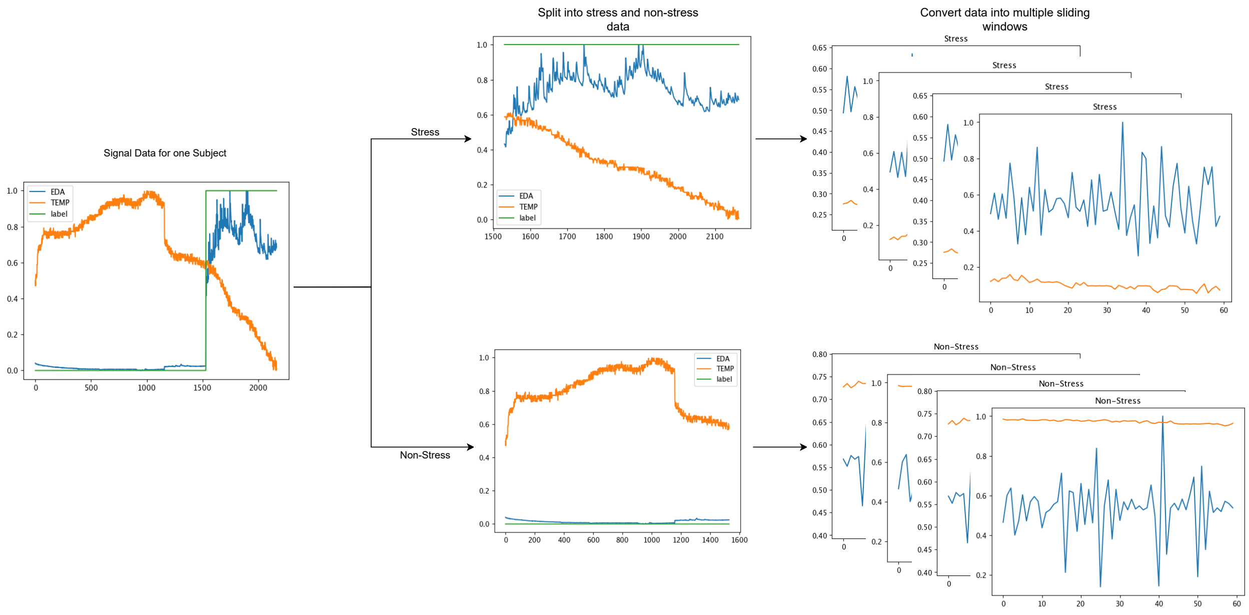

Signal Data for one Subject
EDA
TEMP
label
Stress
Non-Stress
Split into stress and non-stress data
Convert data into multiple sliding windows
Stress
Non–Stress

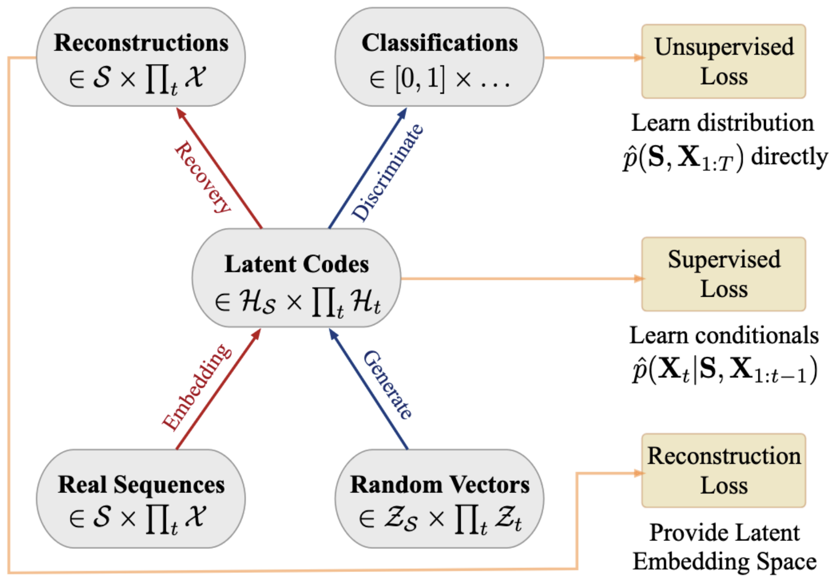
Figure 4.5.: TimeGAN preprocessing workflow to capture the conditional characteristics of the WESAD dataset

The TimeGAN implementation uses RNNs for both the generator and the discriminator. It also trains an additional neural network that maps time-series sequences to a vector embedding space. While Lin et al. [71]. state that this learning of an additional embedding space leads to a network that models long time-series poorly, this is addressed in the TimeGAN by segmenting long datasets into chunks and evaluating this model on the generation of new time-series with this fixed length. Because we do not want to have a model that generalizes over different domains of varying length in the original data but rather augments the data for a specific dataset, we also work with fixed length windows of the WESAD dataset, and will evaluate our approach on these chunks of the dataset.

4.2.2. cGAN Overview

The cGAN is a type of GAN that takes into account additional information such as class labels to generate labeled data. The architecture we use for the cGAN is based on the work of Ehrhart et al. [70]. The authors claim that their architecture is capable of generating labeled synthetic samples that are statistically similar to the original dataset. Using the cGAN, they are able to generate synthetic data that is indistinguishable from the original dataset. They then use this data to train a neural network to detect stress moments. The proposed cGAN consists of two neural networks, a generator that aims to produce synthetic stress or non-stress samples that are similar to samples drawn from the real dataset, thus fooling the discriminator, and a discriminator that learns to distinguish between real stress or non-stress moments and synthetic samples. As noted in the description of the data criteria, the dataset is based on time-dependent variables. The architectural workflow can be seen in Figure 4.7.

First, the conditional generator, implemented as a Stacked-LSTM, receives a latent space and auxiliary information, the class label (stressed/non-stressed), based on this input it generates a fake



Reconstructions $\in \mathcal{S} \times \prod_t \mathcal{X}$
Classifications $\in [0,1] \times \dots$
Unsupervised Loss
Learn distribution $\hat{p}(\mathbf{S}, \mathbf{X}_{1:T})$ directly
Recovery
Discriminate
Latent Codes $\in \mathcal{H}_\mathcal{S} \times \prod_t \mathcal{H}_t$
Supervised Loss
Learn conditionals $\hat{p}(\mathbf{X}_t | \mathbf{S}, \mathbf{X}_{1:t-1})$
Embedding
Generate
Real Sequences $\in \mathcal{S} \times \prod_t \mathcal{X}$
Random Vectors $\in \mathcal{Z}_\mathcal{S} \times \prod_t \mathcal{Z}_t$
Reconstruction Loss
Provide Latent Embedding Space

Figure 4.6.: TimeGAN Overview of components, functions, and their losses. The embedding and reconstruction functions learn to map from and into the latent space. Meanwhile, the generator and discriminator learn through min-max training based on the embedded sequence samples. Image taken from Yoon et al. [69].

sequence with a corresponding label. Second, the FCN receives this fake sequence with label and a real sequence with the belonging label and discriminates the sample as real or fake. Third, using backpropagation, the FCN or LSTM adjusting their weights in alternating epochs, until optimal weights are found within training. The authors address the time-dependence using a generator which is implemented by a LSTM taking care of the temporal context. Thus, the neural network predicts the next time step, based on the previous and present time state, making them suitable for processing temporal, and sequential data. Besides, using the LSTM, the author stating they are more robust to the problem of vanishing or exploding gradients, which often occurs for comparable generator architectures implemented by other temporal neural network architectures, for instance RNNs.

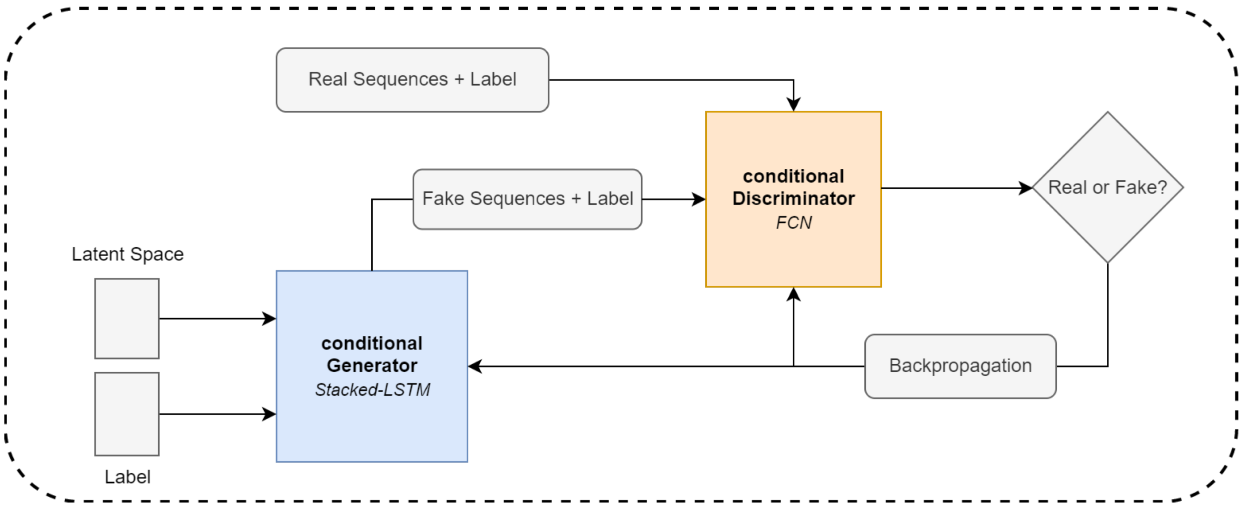


Figure 4.7.: Ehrhart et al.'s overview of original cGAN structure. Image taken from [70]

Ehrhart et al. uses a temporal FCN as implementation for the discriminator. The researchers state it outperforms the recurrent discriminator. It consists of convolutional layers which extract features from the time-series signal. The second data criteria foresees that each point in time has a corresponding class label. Ehrhart et al. also working with stress and non-stress moment data, extending their proposed GAN architecture by feeding vector-based conditional information to their neural network. Before concatenating the vector and the corresponding matrix X the conditional information vector ystress(n) fed into a 2D categorical embedding layer and upsampled to the shape of X via a linear dense layer. Using the categorical embedding, a mapping is learned by the neural network setting the stress label and the sequence into relation. Thus, the generator can be better controlled by adding conditional information.

We use this architecture, determine the optimal parameters for the GAN based on our dataset using a hyperparameter tuning. For the generator, we solely use the LSTM architecture proposed by Ehrhart et al. [70] as initial experiments have shown that the generated samples closely resemble the original signal windows, while the discriminator's loss tends to reach a plateau early on. To address this issue, we implement two other discriminator architectures, an LSTM, as well as a transformer architecture for comparison to determine the optimal setup of this GAN.

Ehrhart et al. [70] introduce a diversity term to counteract mode collapse and reward the model for generating diverse samples. Also during training, after a certain number of epochs, the *classifier-two-sample test* (CTST) is performed. Here, a binary classifier on unseen test data decides whether it belongs to the original dataset or is part of the synthetic dataset. The closer the accuracy of this binary classifier is to 0.5, the better the cGAN generation works. A low accuracy (0.5) means that the binary classifier is not able to assign the data to the correct origin, better than guessing. These values are recorded during training and hyperparameter sweeping to give an indication of the best GAN architecture.

4.2.3. DGAN Overview

Lin et al. [71] introduce a solution to existing shortcomings of time-series GANs with respect to fidelity with the DGAN. Since the canonical GAN relies on fully-connected multi-layer perceptrons, this is not sufficient to capture long-time relationship and correlations between data points of a time-series. To capture these long-time relationships and generate them synthetically, they use RNNs, to be more precise LSTM. They rely on batch generation by the RNNs to generate longer time-series, improving both the efficiency and the capture of temporal relationships. Furthermore, to incorporate additional information, Lin et al. [71] propose an architecture with multiple generators. The researcher's intuition is to decouple this conditional generation task into two tasks, the generation of metadata and the generation of continuous measurements, i.e., time-series sequence samples based on this metadata. There is a separate generator for each of the different task. Thus, a single generator is used for the generation of metadata, i.e., categorical data, and a separate one for continuous data, in our case the values that our sequence represents for a signal. An auxiliary discriminator is used in addition to a general discriminator. The auxiliary discriminator has the task of learning the output of the metadata. This additional discriminator allows the network to learn the metadata which strongly influences the characteristics of the sequence measurements.

In our case, this is the network being fed the stress or non-stress label to produce the associated output.

To address the mode collapse, they introduce even one more generator, which treats the min and max of each time-series as metadata. First, the MLP generator generates fake metadata, second this metadata is then the input for the min/max generator. Third, the metadata from the metadata generator and the min/max is the input for the sequence generation RNN. This architecture can be seen in Figure 4.8

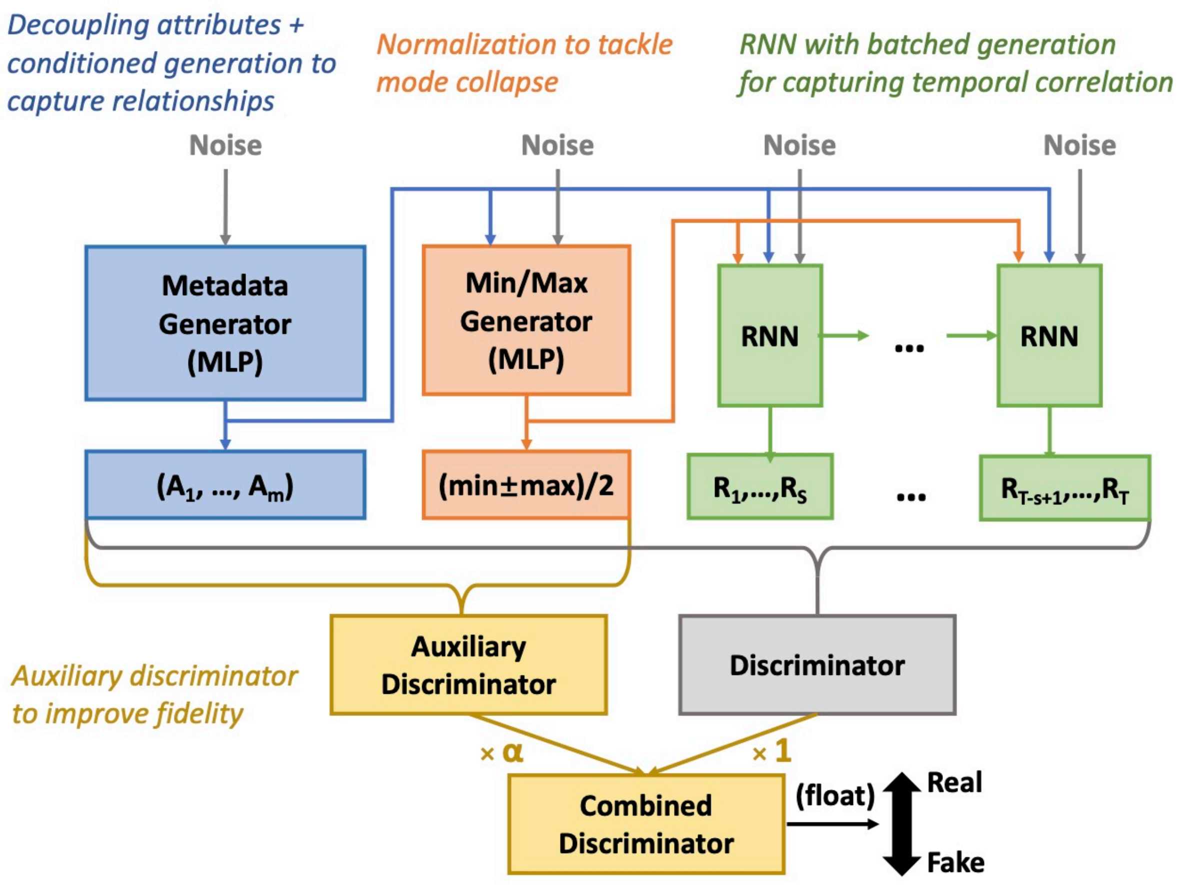


Figure 4.8.: Overview of architecture of the DGAN shows the multitude of generators and discriminators. Figure taken from [71].

At the same time, the loss of both discriminators is merged to improve the training of the generators. By combining these techniques, Lin et al. [71] indicate to address the relationship between the different attributes, the problem of mode collapse and the generation of high-fidelity synthetic time-series data. This approach also offers the advantage that a trained model can be further refined, and by flexibly changing the metadata, can generate synthetic data for a different use case.

4.3. Evaluation

The evaluation process is one of the key challenges in training and using GANs. While in many areas of neural network application, certain evaluation metrics have become established within the research field. This is true for the area of GANs that involve computer vision such as image generation, but not for time-series GANs. In terms of GAN evaluation, it needs to be distinguished between qualitative and quantitative metrics. Qualitative metrics refer to subjective human judgment and are often used in fields where visual quality, realism, or aesthetics are critical, such as computer vision. These qualitative metrics are difficult to quantify; on the other hand, quantitative metrics are primarily concerned with capturing objective measurements, such as accuracy, performance, or statistical properties of a sample.

While in case of computer vision, the most representative qualitative metric is the human annotation of the visual quality. In terms of time-series data, it is way harder to evaluate it on visual quality. Brophy et al. [42] state the comparatively low publication density as the reason the field has not yet been able to agree on evaluation metrics.

Another problem in finding a uniform evaluation metric can be traced back to the different application domains of time-series GANs. For example, a time-series GAN used in the financial sector will have to meet different requirements than a GAN used in the healthcare sector. This makes both the uniform metric definition and the creation of a benchmark dataset difficult. Moreover, different researchers care about different things. While one researcher is particularly interested in the consistency of the statistical distribution, the next researcher places a high value on high quality samples.

Contrary to the field of computer vision GANs, where there are many comprehensive benchmark datasets such as ImageNet [81] and CIFAR-10 [82], this does not exist for the field of time-series GANs. One of the problems is the generalizability of the GANs and possible privacy problems of these data, as in the case of medical GANs that rely on clinical personal data. These two issues make it more difficult for the scientific field to compare approaches and results.

While we can address the issue of inconsistent evaluation metrics by picking and using promising approaches, in this work we neglect to determine the performance of our approach on a benchmark dataset. This is not the goal of this work, as we will extend an existing dataset and validate it for the particular use case.

In order to evaluate our results comprehensively, we choose different metrics. For this choice of evaluation metrics, we decide on the three desiderata Yoon et al. [69] propose on their evaluation on TimeGAN:

- Diversity: samples should be as distributed as the real data points
- Fidelity: samples should be indistinguishable from real points
- Usefulness: samples should be as useful as the real dataset when used in the same application context (i.e., training on synthetic data, testing on real data).

While the three evaluation approaches address various facets of the synthetic data, they all examine the nature of the synthetic data compared to the original WESAD dataset. We perform the *diversity* and *fidelity* evaluation on both the synthetic and original datasets, comparing the *non-stress* and *stress* data, respectively. In this way, we can ensure that the conditional property is learned by the GANs and that they can generate both stressed and unstressed data according to the original data.

4.3.1. Diversity

Assessing *diversity* means assessing the ability of the GAN to understand the data distribution in its entirety. The visual examination of individual sample sequences from each signal type plays a crucial role in answering the following questions: Visual inspection of individual sample sequences from each signal type plays a crucial role in answering the following questions: Do the synthetic sequences show a similar signal progression, a similar signal profile as the real sequences in the 60-second windows? Can the model generate different sequences per signal type, or are they prone to mode collapse as described in the background Section 2.2.1? Furthermore, to examine the diversity, visualizations are useful that reduce the dimensions of the dataset and make them comparable in a two-dimensional space. In this context, we apply a Principal Component Analysis (PCA) and a t-Distributed Stochastic Neighbor Embedding (t-SNE) analysis.

PCA. PCA is a statistical technique for simplifying and visualizing a dataset. It converts many correlated statistical variables into principal components. PCA is able to identify the principal components in which the data differ while preserving the coarser structure of the data.

t-SNE. t-SNE is another method for visualizing high-dimensional data. Each data point is assigned a position in a two-dimensional space. This reduces the dimension while maintaining significant variance. Unlike PCA, it is not good at preserving the location of distant points, but can better represent the equality between nearby points.

4.3.2. Fidelity

Fidelity as defined by Yoon et al. [69] measures the indistinguishability between the synthetic and the real dataset. Furthermore, we determine the fidelity by examining the statistical properties of the individual signals in the synthetic dataset or the real dataset. Therefore, we analyze the statistical distributions of the signal values over the entire laboratory session time course and compare this with the statistical distribution of the values that form the synthetic dataset. An optimal synthetic dataset completely mimics the distribution of the real dataset and is therefore indistinguishable from the real one. We also evaluate the fidelity by performing indistinguishability tests. This goes beyond the statistical distribution comparison, and we get more information about how well the synthetic dataset mimics the real one, and how well the synthetic data can fool trained machine learning models. For this test, we train several binary classifiers to distinguish between real and synthetic data. This follows the intuition that, when the synthetic data closely mimics the real data, the classifier struggles distinguishing and tends to an optimal accuracy of 0.5. Upfront, we partition the synthetic and real data based on the stress label to evaluate the fidelity on each label. We then label the real data '1' and the synthetic data '0'. The data is then combined and split into train and test dataset with 80:20 ratio. Over this data, we train a variety of classifiers such as logistic regression, K-Nearest Neighbors (KNN), and Naive Bayes. We use different classifier algorithms to ensure that the result obtained is robust, and not based on an insufficiency or bias of a single classifier. The performance is then assessed by calculating the accuracy of the predictions of the remaining 20% test dataset.

4.3.3. Usefulness

To test the third criterion, the *usefulness* of a dataset, we use a CNN for stress detection. We provide further details on the preprocessing for the CNN stress detection model in Section 4.4. To measure usefulness, we perform two different training procedures with the synthetic data.

LOSO. On the one hand, we use the generated dataset to train the model in the *Leave-One-Subject-Out* (LOSO) procedure. We train 15 different models, each using 14 of the 15 subjects from the WESAD dataset as training data and evaluating them on the remaining subject so that each subject is the test data once. To evaluate our synthetic data, we generate stress and non-stress sequences per GAN model with the size of an average subject. We use this data in the LOSO training procedure as a *synthetic subject*, having 15 subjects plus 1 synthetic subject, and evaluate the influence of the additional synthetic data on the LOSO evaluation of the other subjects. To compare the influence, we use the results of Sülzle and Wenzlitschke[6] and can compare the accuracy, recall, precision, and F1-score with and without the synthetic dataset in the LOSO procedure.

Train-test leakage is a problem that occurs when instances from the test set unintentionally influence the training set, which can lead to an overly optimistic performance of the trained model. This problem is also part of our LOSO training procedure, since the data from each subject that we excluded for evaluation was part of the training dataset for the GANs.

To address the train-test leakage problem, we trained a GAN without *subject 14* as an exemplary case. Subject 14 performed very poorly in comparable work with the stress detection model. By separately training a GAN without this subject and training with the same, we can test the generalizability of our data.

TSTR. On the other hand, the TSTR framework is commonly used in the synthetic data domain. TSTR means that the model is fully trained on the synthetic data and then evaluated only on the real dataset. Thus, the real dataset is not part of the training process and the generalizability of the synthetic dataset can be better represented. We apply TSTR to stress detection on the 15 subjects. For this, we first generate a synthetic dataset per GAN model with the size of the original WESAD dataset. We then train a CNN model for each synthetic dataset. We then evaluate this model on each of the subjects. The final performance of the model is given by the averaged accuracy, recall, precision, and F1-score over all subjects, as in the LOSO method. The metrics we use to evaluate the stress detection model are accuracy, precision, recall and F1-score.

Accuracy determines the ratio of correctly classified stress, true positives (TP), and non-stress, true negatives (TN), windows to the entirety of all predictions. While this measure can provide an indication of the performance of the model, for unbalanced training datasets, accuracy alone is not sufficient to provide a comprehensive picture of the model's performance.

[6]https://github.com/peasypi/Stress-Detection-From-Wearables

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision considers the true positives (TP), i.e., all correctly classified stress windows, in relation to all samples classified as stress windows. In other words, how often does the model classify stress without predicting false positives (FP) (windows falsely identified as stress).

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall is a measure that determines the ratio of correctly classified stress windows to the total number of stress windows. A high recall value means that the model is capturing a large portion of the stress moments, possibly at the cost of a higher occurrence of FP but less false negatives (FN).

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-score combines both precision and recall, which are important for our stress detection model, to determine the balance between these metrics. The F1-score is the harmonic mean and is particularly useful for unbalanced datasets, such as the WESAD dataset, in the distribution of stress labels.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

4.4. Preprocessing for Stress Detection with a CNN Model

We conduct the usefulness evaluation using a CNN Model by Gil-Martin et al. [4]. The researchers use a 2-dimensional CNN to perform stress detection on the WESAD dataset. The input to this neural network is a further processing form of the windows we described in Section 4.1. Here, the windows of the form `(number of samples, length of window in seconds, number of signals)` are transformed via *Fast Fourier Transformation* (FFT) into a frequency dependent representation. FFT is an algorithm that efficiently calculates the Fourier transform to convert a time-dependent signal into its frequency components, which form the original signal.

The preprocessing workflow is illustrated in Figure 4.9 by dividing the 60-second windows into further subwindows of different lengths depending on the signal type and a sliding window of 0.25 seconds. The different length of these subwindows results from the unique frequency spectrum characteristics of each signal type. Gil-Martin et al. [4] provide these values for the frequency range and the corresponding sub-window length, which can be seen in Table 4.1.

The upper bound of the frequency range multiplied by the sub-window length results 210 which is a fixed input size for the CNN. The FFT is then applied to these subwindows. This means that these subwindows are transformed into frequency spectra. The frequency spectra of the subwindows are then averaged along all subwindows in a 60-second window to finally obtain a frequency spectrum per 60-second window, resulting in an average spectrum having the same number of 210 frequency points. These 210 frequency points then become the input to the CNN model.

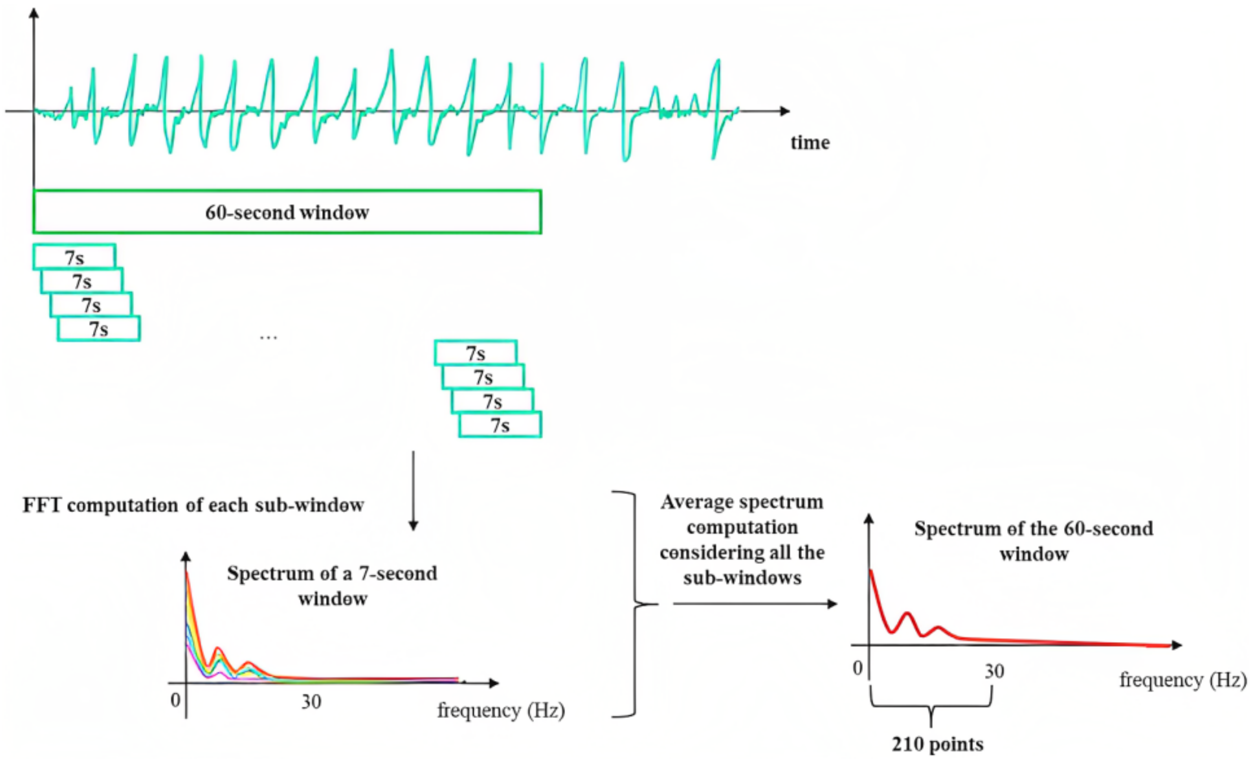

time
60-second window
7s
7s
7s
7s
...
7s
7s
7s
7s
FFT computation of each sub-window
Spectrum of a 7-second window
0
30
frequency (Hz)
Average spectrum computation considering all the sub-windows
Spectrum of the 60-second window
0
30
frequency (Hz)
210 points

Figure 4.9.: Preprocessing workflow for CNN stress detection and image taken by Gil-Martin et al. [4].

Signal	Frequency range	Sub-window length	Number of inputs to CNN
Accelerations (X, Y, Z)	0–30 Hz	7 seconds	210
BVP	0–7 Hz	30 seconds	210
EDA	0–7 Hz	30 seconds	210
TEMP	0–6 Hz	35 seconds	210

Table 4.1.: Details for subwindow length per signal taken from Gil-Martin et al. [4].

4.5. Applying Differential Privacy to GANs

One of the main challenges in applying DP to GANs is to produce high quality samples that convey high information value with respect to the domain without giving the linkage of a sample with individual data [46, 42]. To examine DP in the following, we draw on the approach of Xie et al. [9] and Liu et al. [83]. Both approaches introduce the same concept, a DP guarantee via applying noise to the gradient. The researchers provide a proof for the (ϵ, δ)-differential privacy guarantee. They can guarantee DP by adding noise gradients to the optimizer during the training phase, and evaluate this approach over multiple dataset, showing they are able to produce high quality data samples.

The generator only receives the feedback from the discriminator, while the discriminator is accessing the real data [83]. That indicates that only the discriminator needs to apply to DP. DP is used only for the cGAN discriminator, since the cGAN gave the best results during the hyperparameter tuning. In order to determine the necessary noise multiplier, i.e. the factor by which the noise must be added in order to guarantee a certain privacy level, we must first determine the following parameters:

δ is chosen to be less than the inverse of the size of the dataset, as described in the background Section 2.3. In our case, we have, 1083 sliding windows. But only 545 of them are unique windows, so δ needs to be smaller than $\frac{1}{545}$, resulting in $\delta = 10^{-4}$. For good privacy guarantees, the privacy budget should be $\epsilon \leq 1$. To investigate the trade-off between privacy and quality, we tested different values for the privacy budget $\epsilon = \{0.1, 1, 10, \infty\}$ in our DP-cGAN experiments. Furthermore, the calculation of the noise depends on the number of epochs and the size of the microbatches. For more details, see the experimental setup chapter Section 5.6.

5. Experimental Setup

The following chapter outlines the experimental setup for this thesis and describes the technical details we decided to use. The chapter consists of the following sections: In Section 5.1 we provide an overview of the tools, libraries, and frameworks used to implement the different GANs. Section 5.2 describes the signal processing on the WESAD dataset that we did to obtain the training dataset for the GAN training. Section 5.3 explains the architectures for each GAN in terms of implementation details of the individual building blocks, generator and discriminator, regarding the number and type of different layers. In Section 5.4 we explain the training processes common to all GANs, but also highlight the design of the specific training iterations for each GAN. Finally, in Section 5.5 we present the further preprocessing, architecture, and training process for the stress detection model, which we will use in the following to evaluate the different GAN architectures.

5.1. Environment

For the implementation, we use Python as our programming language of choice, working in multiple Jupyter notebooks. The Jupyter notebooks represent the individual steps taken to preprocess the data, define the hyperparameter tuning, and train the model. Subsequently, the data is generated and then evaluated using the visual and statistical metrics specified in Section 4.3. Finally, the stress detection evaluation model is used to retrieve the evaluation metrics in a use case setting to obtain the usability evaluation dimension. We use Keras[7] with TensorFlow[8] for the implementation of TimeGAN and cGAN, and PyTorch[9] for the DGAN architecture, since it comes with the gretel-ai synthetics implementation[10] that we use. The hyperparameter sweep for the cGAN and the TimeGAN are performed using WandB[11]. To show the applicability of DP, we also use the tensorflow-privacy[12] library to replace the optimizer with a DP optimizer.

The source code is available in a Git repository[13] for easy access and reproducibility.

5.2. Signal preprocessing

For the signal preprocessing of the GANs, we follow the methodology of Gil-Martin et al. [4] described in Section 4.1. Based on the implementation of Sülzle and Wenzlitschke[14], we first process each subject of the WESAD dataset individually. The data is downsampled to 1Hz, relabeled to *non-stress* and *stress* labels, and *min-max-normalized* into the range of [0,1]. We then turn this data into 60-second windows with an overlap of 30 seconds. Subsequently, the individual 60-second

[7]Available on GitHub: https://github.com/keras-team/keras
[8]Available on GitHub: https://github.com/tensorflow/tensorflow
[9]Available on GitHub: https://github.com/pytorch/pytorch
[10]Available on GitHub: https://github.com/gretelai/gretel-synthetics
[11]See: https://wandb.ai/
[12]Available on GitHub: https://github.com/tensorflow/privacy
[13]Available on GitHub: https://github.com/geheim01/Privacy-Preserving-Smartwatch-Health-Data-Generation-Using-DP-GANs
[14]Available on GitHub: https://github.com/peasypi/Stress-Detection-From-Wearables

windows and their corresponding labels from the subjects are merged into the training dataset. Resulting in data with a shape of `(1083, 60, 6)`, where `1083` relates to the number of windows with a length of `60` seconds and `6` signals (BVP, EDA, ACC_x, ACC_y, ACC_z, TEMP). `320` of these windows are *stress*-labeled and `763` are *non-stress* labeled.

5.3. GAN Architectures

As just described, the data after preprocessing has the shape `(1083, 60, 6)`. After this signal preprocessing, the data is split into *stress* and *non-stress* data, and then split into an 80:20 train/test split for further training and evaluation in the training process. This data after signal preprocessing serves as input for training the different GAN networks.

While we describe three different GAN architectures in detail below, it should be emphasized that hyperparameter tuning was only performed on the TimeGAN and the cGAN. We only applied DP to the cGAN because it gave the best results in the exploratory training trials and was therefore the most promising. The code for the cGAN architecture was provided to me by Ehrhart et al. [70]. We adapted and hyperparameter tuned this architecture for our use case.

5.3.1. TimeGAN Implementation

We use the TimeGAN implementation[15] which was used as a baseline approach in the Hide and Seek Privacy Challenge [84]. This implementation was originally developed by Yoon et al. [69] and subsequently ported for use in TensorFlow 2[16]. We will use this implementation for the following experiments.

A crucial difference to the other two GANs used here is that the TimeGAN cannot capture the conditional aspect. To counteract this, we first divide the dataset into *stress* and *non-stress* data, and then train two TimeGANs to deal with the two conditions. Then, to obtain a synthetic dataset that mimics the WESAD dataset, we generate samples with each of the two models and merge them.

Generator The generator is an RNN implemented with Gated Recurrent Units (GRU) with a tanh activation function, 10 hidden units, and 3 layers. These GRU layers followed by a dense layer with 10 hidden units and a sigmoid activation function. For the optimizer, the Adam optimizer is used. The loss is calculated using the *binary cross-entropy* loss function.

Discriminator The discriminator is built analogously to the generator with an RNN consisting of three GRU layers, except that the output dense layer has only 1 unit and no activation function. The Adam optimizer and the *binary cross-entropy* loss function are used for training. The embedding network and the recovery network are implemented with the same architecture as the generator and the discriminator, where the embedding network follows the generator network and the recovery network follows the discriminator architecture.

[15]Available at: https://bitbucket.org/mvdschaar/mlforhealthlabpub/src/master/app/hide-and-seek
[16]Available on GitHub: https://github.com/mcps5601/TimeGAN-tensorflow2

5.3.2. cGAN Implementation

The cGAN architecture is based on the work of Ehrhart et al. [70]. In the following, we describe the implementation details, including the configuration of the architecture and the optimal hyperparameters, which we determined through a hyperparameter sweep via WandB[17].

The architecture of the cGAN consists of an LSTM in the generator. For the discriminator, we implemented two different neural networks, an FCN and an LSTM architecture. These three architecture can be seen in Figure 5.1.

Generator. The generator first feeds the generated random noise into the neural network through an input layer and reshaped to add an additional channel. Then the categorical input, i.e. the *stress/non-stress* class, is embedded by an embedding layer, put through a dense layer and concatenated with the reshaped noise. This is then put through two LSTM layers. The input sequence length is 60, we use 64 hidden units for the LSTM layers.

FCN Discriminator. For the discriminator, we tried two architectures: FCN and LSTM. The FCN discriminator takes the conditional information and the stress/non-stress signal sequence as input. Then, the conditional information is preprocessed in the manner of embedding and dense layer, and then concatenated with the signal sequence. The concatenated matrix is then passed through several Conv1D, BatchNorm, and ReLU blocks to be averaged and pooled at the end. For the three Conv1D layer we apply [32,64,32] filters, with kernel size of [8,5,3] with the respective layer. Finally, the *binary cross-entropy* loss function is applied to determine whether the sample is drawn from the real data distribution or synthetically generated by the generator.

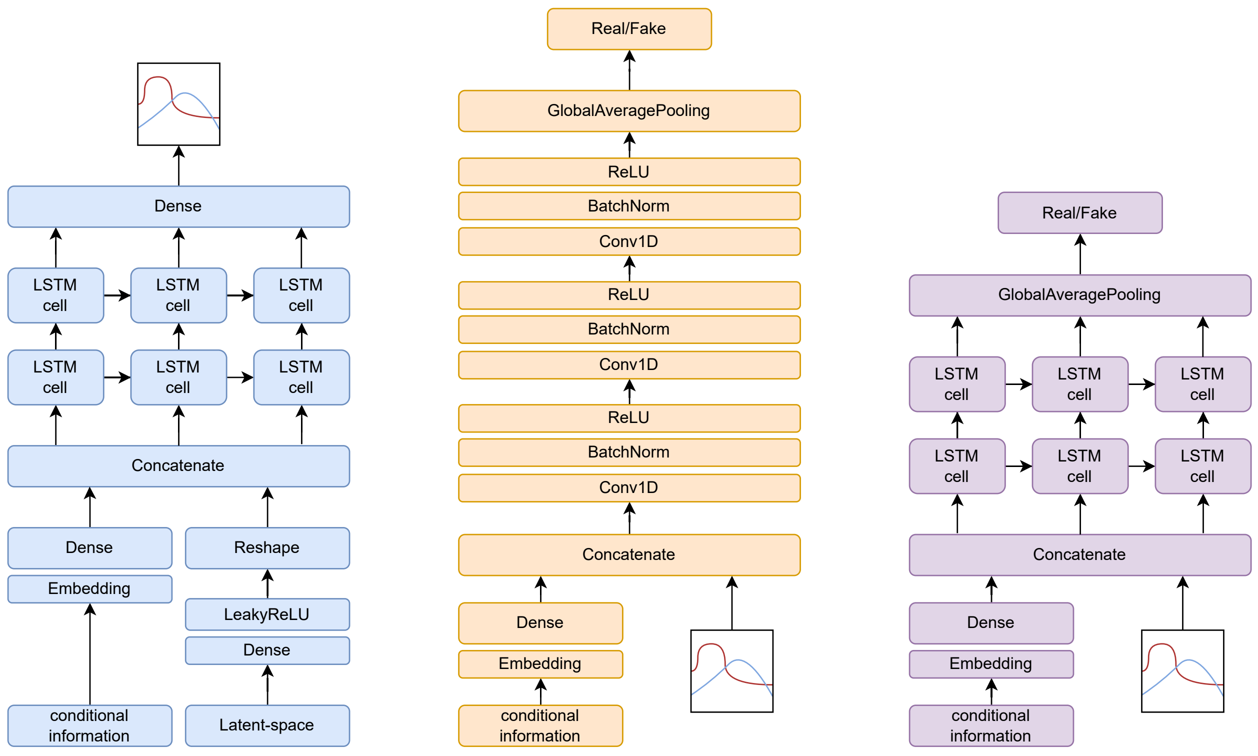
LSTM discriminator. The LSTM discriminator transforms the input in the same way as the FCN discriminator, up to the point where the conditional information and the input signal sequence are concatenated. This is followed by two LSTM layers, which are also followed by a global average pooling layer, which *binary cross-entropy* loss function. For the LSTM layers, we choose a hidden size of 64 units.

5.3.3. DGAN Implementation

We use the DGAN implementation that comes with the gretel synthetics library for time-series synthetic data[18]. The DGAN network consists of several generator and discriminator networks. The metadata generator and the min/max generator are implemented with multilayer perceptrons (MLPs) with 2 hidden layers and 100 units per layer. The measurement generator, which generates the continuous data, is implemented with a 1-layer LSTM with 100 units. The softmax activation function is used for the metadata generator output. We use the sigmoid activation function for the measurement generator because we normalize the data to [0,1]. The discriminator and the auxiliary discriminator are implemented with a 4 hidden layer MLP with 200 hidden units per layer.

[17]https://wandb.ai/

[18]Available on GitHub: https://github.com/gretelai/gretel-synthetics/tree/master/src/gretel_synthetics/timeseries_dgan



(a) conditional LSTM Generator (b) conditional FCN Discriminator (c) conditional LSTM Discriminator

Figure 5.1.: Generator and discriminator architectures for the cGAN. Image based on Ehrhart et al. [70] adjusted by LSTM discriminator.

5.4. GAN Training

To ensure the reproducibility of our results, we set a random seed for the GAN trainings to 42. Furthermore, the training of the GANs follows a similar abstract pattern: the generator is first fed random noise, from which it generates a sample. Then, the discriminator classifies the sample as a stress or non-stress situation. This process is repeated in a training loop over many iterations, where the generator learns to generate better samples based on the discriminator loss, while the discriminator learns to discriminate better between real and fake data over time. The details are explained below.

5.4.1. TimeGAN Training

The TimeGAN consists of four main components: Embedding and recovery function, and the discriminator and generator network, as can be seen in Figure 4.6. These are trained by three loss functions: *Reconstruction Loss* ($\mathcal{L}_R$), *Unsupervised Loss* ($\mathcal{L}_U$), and *Supervised Loss* ($\mathcal{L}_S$). Initially, these loss functions are trained separately, and later combined.

The training consists of three parts. First, the embedding network is trained for 2,000 iterations. It learns to map the higher-dimensional feature space into a lower dimensional latent embedding space with $\mathcal{L}_R$:

$$\mathcal{L}_R = \mathbb{E}_{x_{1:T}\sim p}\left[\sum_t \|x_t - \tilde{x}_t\|_2\right] \tag{5.1}$$

$\mathcal{L}_R$ is determined by computing the distance between the reconstructed data $\tilde{x}_{1:T}$ and the original data $x_{1:T}$ using the L2 norm. $x_{1:T}$ represents a temporal sample sequence from our real data distribution p. In order for the embedding and reconstruction functions to learn the embedding space correctly, the goal is to minimize this loss in order to obtain accurate reconstructions.

$$\mathcal{L}_U = \mathbb{E}_{x_{1:T}\sim p}\left[\sum_t \log y_t\right] + \mathbb{E}_{x1:T\sim\hat{p}}\left[\sum_t \log(1-\hat{y}_t)\right] \tag{5.2}$$

$\mathcal{L}_U$ is used to train the generator and discriminator in a min-max game. Therefore, the predicted likelihoods of y_t and $1-\hat{y}$ originating from the real distribution p or the synthetic distribution $\hat{p}$ are compared.

$$\mathcal{L}_S = \mathbb{E}_{x_{1:T}\sim p}\left[\sum_t \|h_t - g_X(h_{t-1}, z_t)\|_2\right] \tag{5.3}$$

$\mathcal{L}_S$ is used to train the generators' ability to capture the temporal dynamics within the sequence data points. $\mathcal{L}_S$ is calculated using the L2 norm as the distance between the latent space h and the synthetic latent space generated by the generator. This generator receives as input the latent space h_{t-1} of a temporally preceding state and a random noise variable z_t.

In the second training round, only $\mathcal{L}_S$ is trained. Finally, the four components are jointly trained with their respective loss functions for 6,000 iterations. Thus, the generator is trained twice as often as the discriminator, which helps the generator to produce better samples.

We set the batch size to 32, and for all four component neural networks, we utilize the Adam optimizer with a learning rate of 10^{-3}.

5.4.2. cGAN Training

The cGAN consists of two networks, the generator G and the discriminator D, which are optimized simultaneously. To compute the loss, the min-max adversarial loss function from the original GAN paper [36] is used and adapted below.

$$\min_G \max_D = \mathbb{E}_{x,y}[\log(D(x|y))] + \mathbb{E}_{z,y}[1 - \log(D(G(z|y)))] \tag{5.4}$$

Therefore, the discriminator tries to maximize the log probability of correctly classifying the real samples and fake samples from the time-series data. At the same time, the generator tries to minimize the probability classifying the generated samples as fake. The generator generates these sequence samples from a latent space z with a Gaussian distribution $N(\mu, \sigma^2)$. A novel concept in this training loop is the introduction of the diversity term. As mentioned before, a key challenge in

GAN training is the production of the same samples over and over again, leading to mode collapse. An example of a mode collapse that occurred during hyperparameter tuning and could be resolved by adjusting the hyperparameters is shown in Figure 5.2.

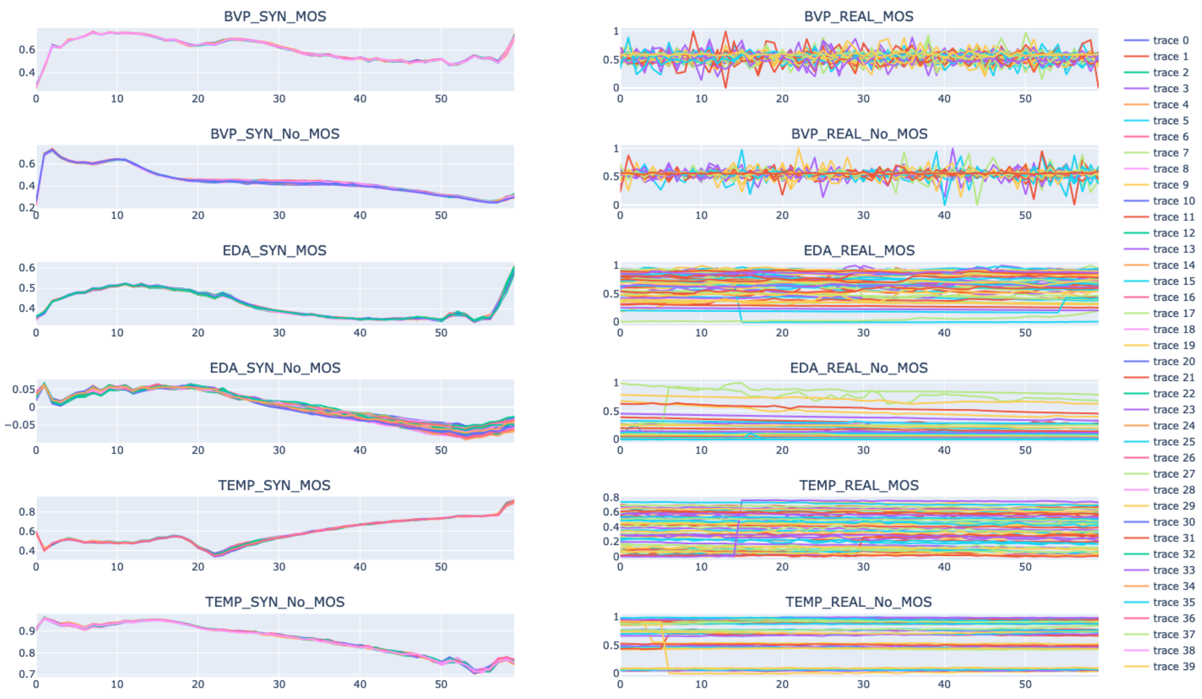


Figure 5.2.: Mode collapse, which occurred during hyperparameter tuning of the cGAN model. The generated synthetic sequences are shown on the left for *stress* (MOS) and *non-stress* (No_MOS), which tend to be all the same, while the sequences from the original WESAD dataset for the BVP, EDA, and TEMP signal types are shown on the right.

To counteract this problem, the generator produces two samples from different latent spaces. The distance between these two samples produces the diversity term in this training step. Ehrhart et al. [70] define the diversity term as:

$$\max_G f(G) = \mathbb{E}_{z_1,z_2}\left[\frac{\|G(z_1,y) - G(z_2,y)\|}{\|z_1 - z_2\|}\right] \tag{5.5}$$

This approach follows the basic idea that if the noise samples are different but the generated sequence sample is the same, the diversity term is 0. This diversity term is then factored into the gradient calculation for the generator, which results in the generator also trying to maximize this diversity term.

The introduction of the diversity term leads to this adapted version of the loss function:

$$\min_G \max_D f(G,D) - \lambda f(G) \tag{5.6}$$

where $f(G, D)$ represents the original objective function and $f(G)$ represents the diversity term. The λ term defines the importance of the diversity term.

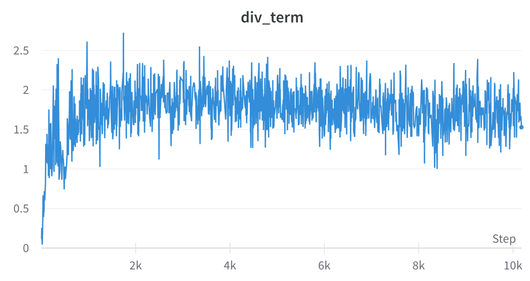
For our experiments, we followed the diversity term $\lambda = 8$ chosen by Ehrhart et al. [70]. Figure 5.3 shows the evolution of the diversity term values during training. Initially, these values increase and then fluctuate between 1.5 and 2 as the training time progresses, which, according to initial exploratory investigations, is indicative of good sample sequence diversity.

We used multiple optimizers in a hyperparameter sweep WandB[19], resulting in the Adam optimizer with a learning rate of 0.0002 and a beta value of 0.5 as the best setting. The training is done in different training iterations, where several runs with different settings are performed.

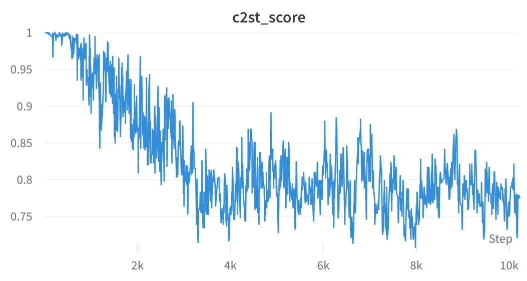
We determined these optimal performing GANs in consideration of plotting signal sequences, t-SNE and PCA plots. Furthermore, we inspected the CTST score and the diversity term after each 100 epochs (Figure 5.3).

The best performing cGAN-LSTM was trained for 1599 epochs and a batch size of 64 reaching a CTST score of 0.745. The best performing cGAN-FCN was trained for 2848 epochs with a batch size of 64 achieving a CTST score of 0.642

The CTST score is implemented by a binary classifier, a KNN Classifier, to compare the similarity of the synthetic data with the original data during training and to detect progress or degradation in the training of the GAN. Figure 5.3 shows this evaluation metric for training the best cGAN model. The lower this value is, the better the GAN is able to generate the synthetic dataset that mimics the original dataset. The worse the performance of the classifier in distinguishing between the two datasets.



(a) Diversity Term



(b) CTST Score

Figure 5.3.: Shown are the two training metrics, the diversity term and the CTST score, for the cGAN in the wandb training environment, which illustrate the learning progress of the model over time.

5.4.3. DGAN Training

The DGAN consists of several generators and discriminators. One discriminator classifies the generated sequence samples, and an auxiliary discriminator distinguishes the additional metadata output. Both discriminators are trained simultaneously with the following loss function:

[19]See: wandb.ai

$$\min_{G} \max_{D_1,D_2} \mathcal{L}_1(G, D_1) + \alpha\mathcal{L}_2(G, D_2) \tag{5.7}$$

where $\mathcal{L}_i, i \in \{1, 2\}$ is the Wasserstein loss of the original and auxiliary discriminators. The loss function for both discriminators is implemented via the Wasserstein loss because it improves training stability and avoids mode collapse.

For the optimizer, the Adam optimizer is used with a learning rate of 10^{-3} for both the generator and the discriminator and a batch size of 552.

5.5. Stress Detection Evaluation Model

In this section, we describe the steps to test our third evaluation metric, usefulness in a real-world scenario. To do this, we draw on architecture from Gil-Martin et al. [4] for stress detection of wearable smartwatch data on the WESAD dataset. Gil-Martin et al. propose a CNN architecture that detects the stress of the respective signals based on the frequency spectrum. We use Sülzle and Wenzlitschke's implementation[20] of the architecture and signal preprocessing in the following step and extend the evaluation with the TSTR framework.

Before we can train the stress detection model, we need to further process our synthetic datasets. These datasets consist of multiple windows with the shape `(1083,60,7)`. This time-dependent data must be transformed into the frequency spectrum as described in 4.4.

5.5.1. Stress Detection Signal Preprocessing

The following signal processing builds on the previous preprocessing of the WESAD dataset in 60-second windows from Section 5.2. These windows are the training data for the GAN training, but must be further processed to serve as the training data for the stress detection. It is important to understand that the following processing is applied to both the newly generated synthetic data from the GANs and the 60-second windows from WESAD dataset.

The stress detection signal processing input consists of `(1083,60,6)` windows, each labeled with the corresponding *stress* or *non-stress* state. We divide each window into subwindows. The subwindow length depends on the signal type, as shown in Table 4.1 in Section 4.4, with a sliding window of 0.25. To subsequently transform these time-dependent subwindows into the frequency spectrum, we use the FFT provided by SciPy[21]. We apply the FFT to the time-dependent subwindows and obtain a subwindow of the signal in the frequency spectrum. By varying the frequency ranges, we can ensure that the regions of interest of each signal type are captured. Depending on the frequency range, we adjust the subwindow length to obtain a constant input shape of 210 data points. Subsequently, the frequency spectrum subwindows of each 60-second window are averaged to obtain an average frequency spectrum per 60-second window.

[20]Available on GitHub: https://github.com/peasypi/Stress-Detection-From-Wearables
[21]See: https://scipy.org

To avoid possible errors in dealing with missing frequencies in the higher range caused by GAN generated data, we pad the FFT subwindows with additional 0s to the upper limit 210.

5.5.2. Architecture

The architecture of the CNN model for stress detection consists of an input layer with an input shape of `(6,210,1)`, corresponding to the number of signals, the number of inputs from preprocessing in the FFT representation, and a single channel. Figure 5.4 illustrates the implemented architecture. The input layer is followed by three blocks consisting of Conv2D layers, Dropout Layer and a 2DMaxPooling for extracting and learning the features from the data. The Conv2D layers has 64 filters with the ReLU activation function and a kernel size of `(1,3)`. The Dropout layer has a dropout rate of 0.3 and the 2DMaxPooling layers has a pool size of `(1,2)`.

These feature learning Conv2D layer blocks are followed by two blocks of Dense layers with 128 and 64 units and the ReLU activation function to learn the classification into the *stress* and *non-stress* class. Each dense layer is followed by a Dropout layers with a dropout rate of 0.3. The output layer is a Dense layer with two units corresponding to the class number and the sigmoid activation function. We have chosen the sigmoid function over the softmax function because it is a binary classification into *stress* and *non-stress* class. Although the softmax function is also suitable for binary classification, the sigmoid function simplifies the output of the classes and is therefore more computationally efficient.

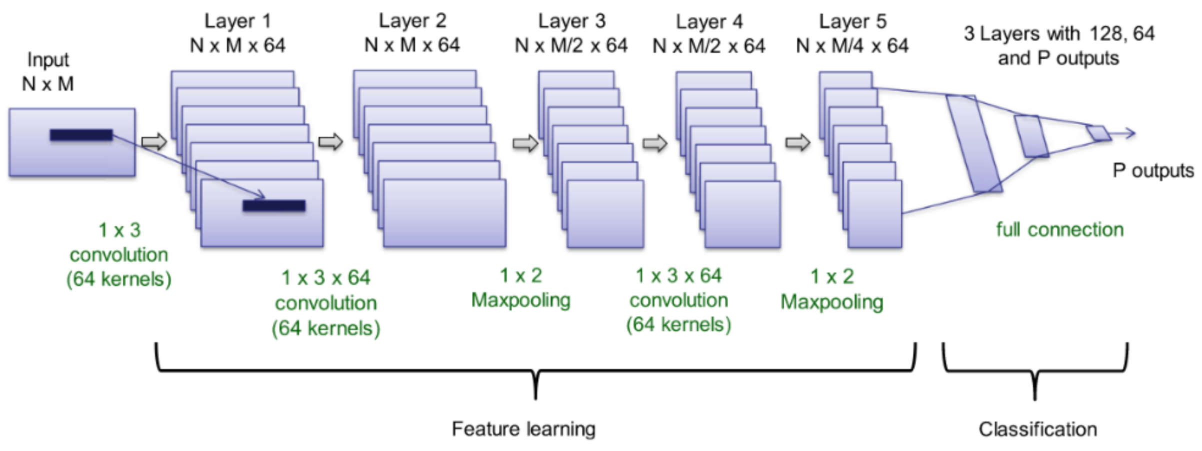


Figure 5.4.: CNN Architecture for the Stress Detection Model. Image by Gil-Martin et al. [4]

5.5.3. Training

Training is performed using the LOSO cross-validation method. Here, the model is trained such that iteratively X-1 subjects form the training data, while the resulting model is evaluated on the remaining subject. Thus, as a baseline approach on the unmodified WESAD dataset, we obtain 15 models based on 15 subjects. This LOSO procedure is used because stress data can vary greatly among different subjects, and thus the generalizability of a model can be better verified by testing it on individual stress data it has not seen before.

Since the classes in the dataset are unbalanced as described in Section 4.1, there are many more *non-stress* labeled windows in the original WESAD dataset than stress labeled windows, we use class weights for training this binary classifier. We determine the class weights using the Keras fit function. We set the class weights for the *non-stress* class to 1 and for the *stress* class we use the ratio of the *non-stress* class to the amount of *stress* class instances. The training is done over 100 epochs with a batch size of 50. As optimizer, we use RMSprop with a default learning rate of 10^{-3}. As loss function, we use the *binary cross-entropy*. In training, we do not use a validation set, but evaluate the final trained model on the remaining subject. We monitor the model performance during training using accuracy, recall, and precision. In addition, we trained the model with a model checkpoint callback, where only the best model based on its loss is stored, and an early stopping callback, where a model was stored if its loss did not improve after 10 epochs.

The training for the augmentation evaluation case follows this procedure for the WESAD dataset. To address the problem of train-test leakage, we additional train GAN models on the WESAD dataset without *Subject 14* for exemplary case. In doing so, we add synthetic subjects. These can either correspond to one subject or multiple. The LOSO training procedure remains the same, except that each model now contains additional subjects in the training dataset. To determine the optimal size of synthetic augmentation, we add subjects between the size of a single subject and 100 subjects.

For the TSTR evaluation case, training is performed on the synthetic data only. We first shuffle the synthetic data and then split it into a training and a validation set with the ratio 80:20. In contrast to the LOSO procedure, the training and validation data remain the same. As a result, the TSTR training method returns a single model per synthetic dataset.

5.5.4. Evaluation

In the LOSO procedure, the models are evaluated on each withheld test subject. For each subject we calculate accuracy, precision, recall Score and F1-score. Accuracy specifies the proportion of correct stress and non-stress detections. Recall quantifies the number of correctly detected stress situations (True Positive) out of all stress situations (Actual Stress). Precision determines the number of correct stress detections (True Positive) with regard to all supposed stress detections (Predicted Stress). The F1-score is the harmonic mean of precision and recall, and thus balances these metrics. We then average the scores of the different subjects to obtain the performance measures accuracy and F1-score around stress detection over the entirety of the WESAD dataset. The evaluation of the TSTR method is analogous to the LOSO method in that we test the model obtained by TSTR training on each subject individually and then average the results to obtain an overall performance.

5.6. Introduction of Differential Privacy to the cGAN Architecture

To introduce the DP criteria, we use the vectorized DP Adam optimizer from the TensorFlow privacy library[22] for the discriminator in the cGAN architecture. The cGAN architecture remains as described in section 5.3.2.

Training is performed over 1600 epochs with a batch size of 8. We use an L2 norm clip of 1 for the vectorized DP Adam optimizer, the microbatch size is set to the batch size of 8, δ is 10^{-4}, and the learning rate is 10^{-3}. We compute the noise multiplier as a function of the target ϵ and the above parameters using the TensorFlow privacy function *compute noise*. We choose the noise to obtain the privacy budget with target $\epsilon = \{0.1, 1, 10, \infty\}$. $n = 545$ corresponds to the number of unique windows in the dataset and not to the number of windows we train the GAN model with *1083*, since these are created by the sliding windows and thus contain already seen points. In addition, the number of epochs is multiplied by 2 to calculate the noise, since each data point in a training is seen twice due to the overlapping sliding windows.

[22]available on GitHub: https://github.com/tensorflow/privacy

6. Results

In the following chapter, we will present the results from the experiments. As described in Section 4.3 on evaluation methods, we present the performance of the different GANs under consideration of three evaluation facets: *diversity*, *fidelity*, and *usefulness*.

We evaluate the synthetic datasets that result from the different GAN trainings. Thus, we generate a synthetic dataset with each model in the same size as the original WESAD dataset. First, in Section 6.1 we examine the *diversity* evaluation metric. Therefore, we plot several synthetic and original WESAD sequences to assess whether the diversity of the original dataset can be reproduced. We also investigate how diverse the distribution of the synthetic data is compared to the original dataset by performing PCA and t-SNE analysis. In Section 6.2, we analyze the *fidelity* of the synthetic datasets. To do this, we compare the correlation matrices of the datasets to understand whether the underlying multivariate dependence between the different signal types and the stress condition has been reproduced in the synthetic datasets. We then plot the statistical frequency distribution over the different signal types compared to the WESAD dataset. We also perform an indistinguishability test over several machine learning classifiers to examine how indistinguishable the synthetic datasets are from the original dataset. Finally, in Section 6.3 we show the *usefulness* of the generated sequences by the result of training a stress detection model based on the different synthetic datasets, compared by TSTR and LOSO training methods. In the following, we explicitly examine the difference between the LSTM and FCN discriminators presented in Section 5.3.2 for the cGAN model. For the *diversity* and *fidelity* evaluation, we only considered the cGAN-LSTM because the results were very close, and we only wanted to show the fundamental differences between the GAN architectures. An evaluation of different privacy guarantees follows in Section 6.4.

6.1. Diversity: Visual Results

By visualizing multiple sample sequences of the synthetic dataset, it is possible to examine both the diversity, i.e., the ability of the GAN to learn the full spectrum of the original data distribution, and one of the most common problems of GANs, mode collapse. A mode collapse is made visible by the fact that the network is only capable of producing one sample or a small subset of possible outcomes over and over again. The GAN will output synthetic samples sequences with very low diversity.

6.1.1. Plotting signal sequences

We can investigate whether and to what extent the time dependence and profile of the different signal types can be reproduced by plotting sample sequences from the different datasets. Figure 6.1 shows 50 original sequences from the WESAD dataset and synthetic sequences generated by the GAN models for each signal type for the *stress* condition. Figure 6.2 shows analogously the *non-stress* sequences. The models are capable of generating more than a few identical sample sequences, so we can see from this visualization that the models has not encountered the usual difficulty of mode collapse. Moreover, across all signal types, the *stress* windows show different signal tendencies than the *non-stress* windows, as can be seen in Figure 6.1 compared to the equivalent Figure 6.2 with *non-stress*. This means that the models are able to learn the conditional aspect, given by additional class information during training, and to reproduce it using these samples. These tendencies can be seen very well in the case of EDA, as described in Chapter 2, a typical stress response is an increase in the skin conductance level, while this value tends to be lower in *non-stress* states. This learned difference can be seen across all GAN types.

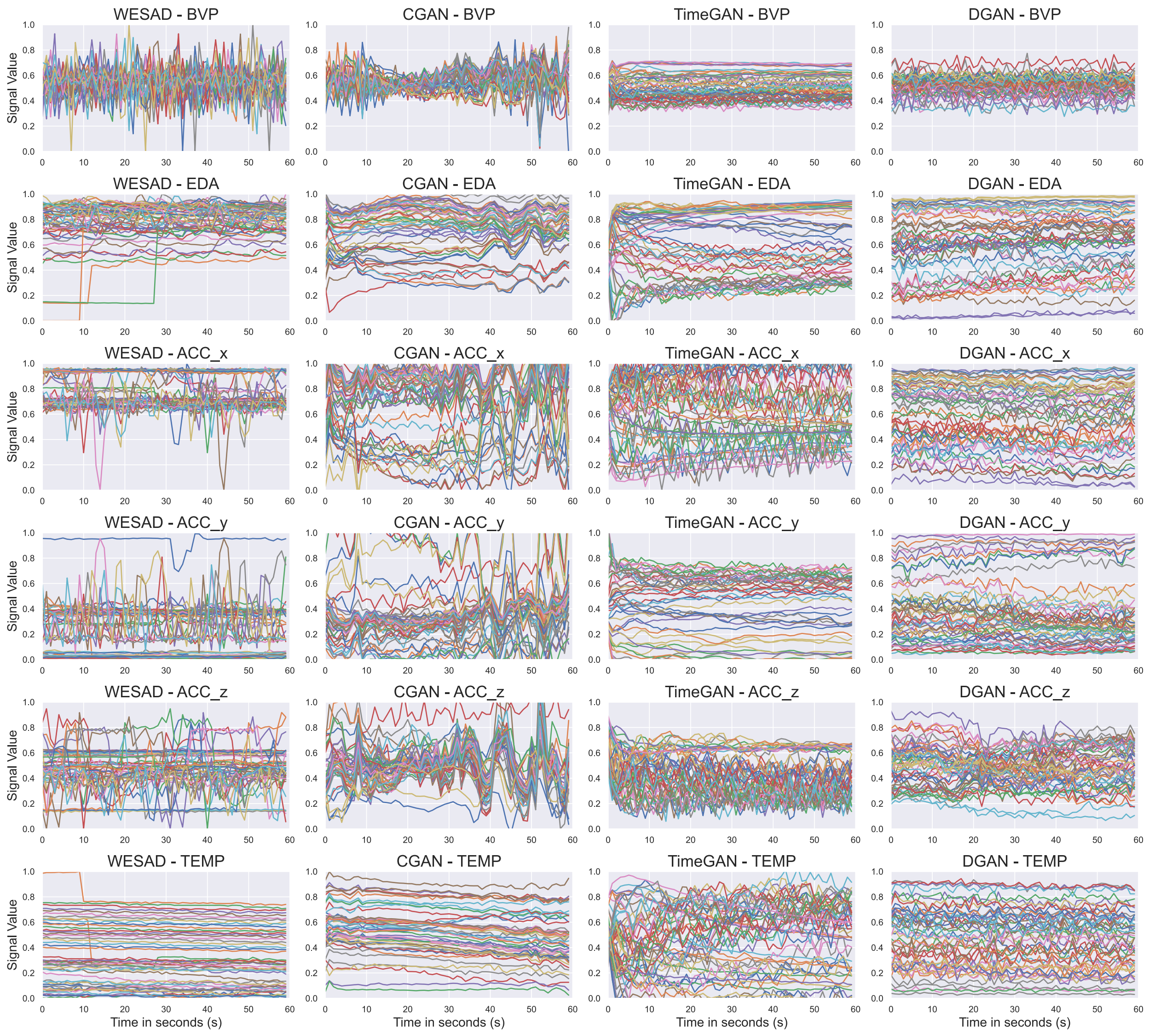


Figure 6.1.: Shown are 50 *stress* sequences from the original WESAD dataset and synthetic sample sequences generated by the different GAN models. These samples are plotted per row for each signal type: BVP, EDA, ACC_x, ACC_y, ACC_z, and TEMP. The columns show the corresponding dataset. This overview allows comparing the ability of the models to mimic the temporal dynamics of the original dataset.

cGAN. Furthermore, we can see that the cGAN signal-typical characteristics are well reproduced across the different samples of the cGAN. For example, the BVP signal for the stress state shows abrupt changes within a condensed time frame, which represents the signal profile of the BVP signal. Only the increasing amplitude and frequency in the mid-region of the signal sequences are not replicated. The EDA as well as the TEMP signal show a more steady signal profile over time.

The EDA stress samples are primarily generated in the upper range [0.3,1] as in the original, representing the stress state very well, while the non-stress EDA samples correctly mimic the lower range [0,0,4].

TimeGAN. The TimeGAN captures the range of different stress signals reasonably well. The EDA signal is properly captured in its range and signal profile. However, for the other signal types, the stress signal profile shows a random pattern with high amplitude and constant frequency, which does not represent the actual signal. This is even more true for the non-stress sequences, which partially cover the ranges (EDA, BVP) but generate even more random signal profiles, such as the ACC and TEMP signals.

DGAN. The DGAN is able to reproduce the range moderately well, can mimic the signal profile in jagged courses well for BVP and the ACC signal sequences, but fails to reproduce the more steady progression of EDA and TEMP.

6.1.2. PCA and t-SNE Analysis

For the second visual analysis, we examine the t-SNE and PCA plots in Figure 6.3 for the *stress* dataset, which shows the result of the PCA and t-SNE analysis for the different synthetic datasets generated by the GANs. The PCA and t-SNE plots for the *non-stress* data can be seen in Figure 2 in the Appendix 8.1. This figure gives a good overview of the learned structure and the distribution of the models over the original data. We do not get this information from the previous plot of sequences. It shows how well the diversity of the original dataset has been mimicked and whether synthetic data point clusters form outside of it, indicating that the model has not learned the full diversity of the WESAD dataset. Conversely, it can show that the models may not have learned certain outliers of the real dataset.

cGAN. It can be seen that the synthetic dataset resembles the distribution of the original dataset in many data points of the PCA representation and mimics the diversity. However, the t-SNE plot shows that the local similarity between the two classes is clearly separated in the *stress* dataset. This leads to the assumption that the cGAN has not learned the full diversity of the original dataset, which is reinforced by the sequence plots. The *non-stress* show much more similarity in the distribution of synthetic and original data, leading to the assumption that the cGAN learned the distribution of *non-stress* data better.

TimeGAN. The PCA analysis shows that the TimeGAN synthetic *stress* data covers the distribution partly in a cluster. The *non-stress* data is clustered around the original distribution in both plots. This also reinforces what is seen in the sequence plots, where the range is correctly mimicked, but the signal profile does not match and tends to have a strong amplitude

DGAN. The DGAN, on the other hand, looks very good in both plots for the *stress* and *non-stress* dataset. The synthetic data in the PCA plot is almost identical to the original data. There are only a few outliers. This impression is also confirmed by the t-SNE plot, which shows much greater similarities in diversity compared to the cGAN and TimeGAN plots.

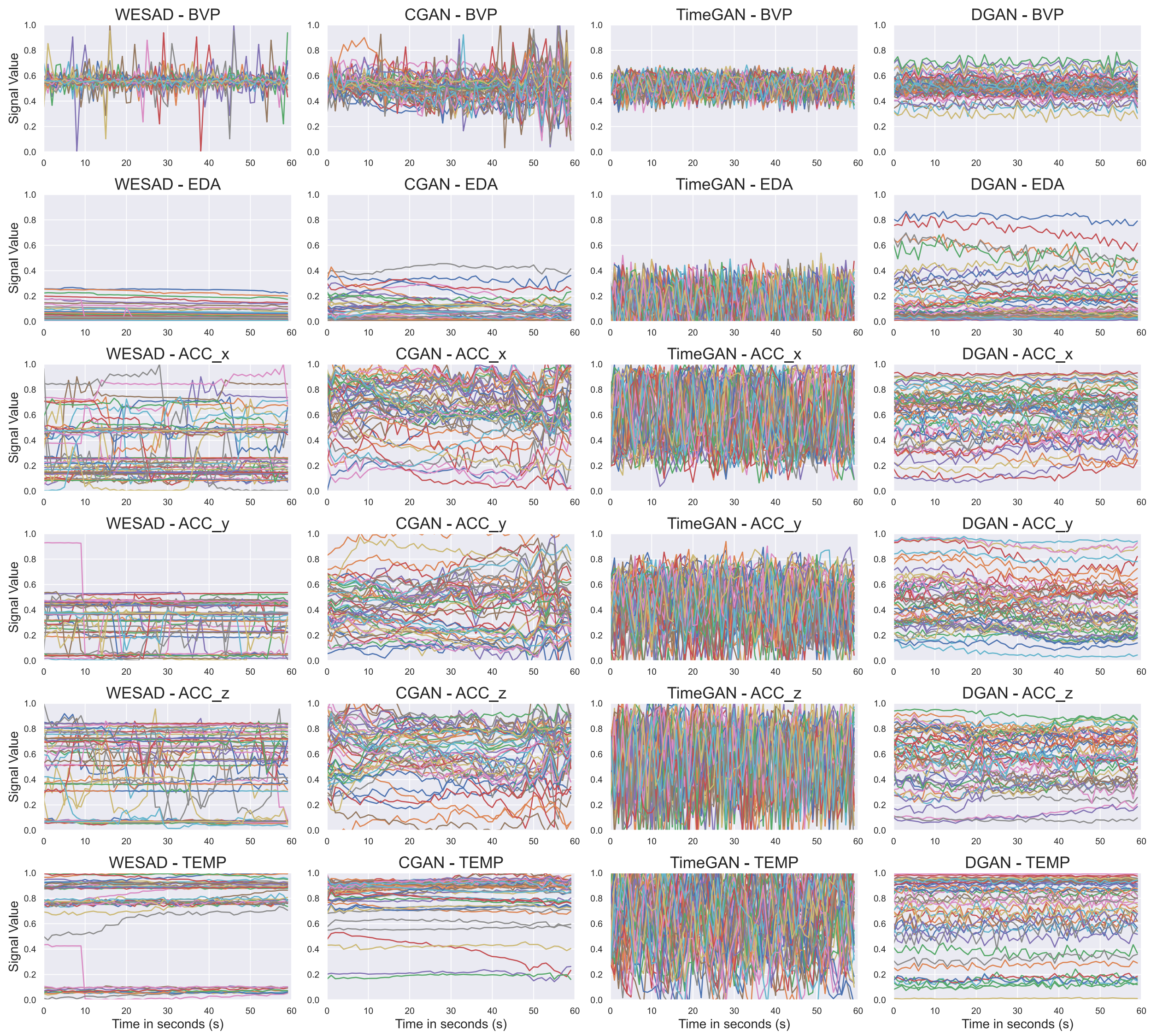


Figure 6.2.: Shown are 50 *non-stress* sequences from the original WESAD dataset and synthetic sample sequences generated by the different GAN models. These samples are plotted per row for each signal type: BVP, EDA, ACC_x, ACC_y, ACC_z, and TEMP. The columns show the corresponding dataset. This overview allows comparing the ability of the models to mimic the temporal dynamics of the original dataset.

6.2. Fidelity: Statistical Results

The examination of the fidelity gives an indication of whether our models have learned the fundamental statistical distribution. If so, they should be partially indistinguishable from the original data. In order to test this, we compare the correlation matrices of the original WESAD dataset and the synthetic datasets generated by the different GANs. We then look at the statistical distribution of the synthetic datasets in comparison to the original dataset. Finally, we evaluate the results of several machine learning classifiers, which are supposed to distinguish the synthetic datasets from the real one.

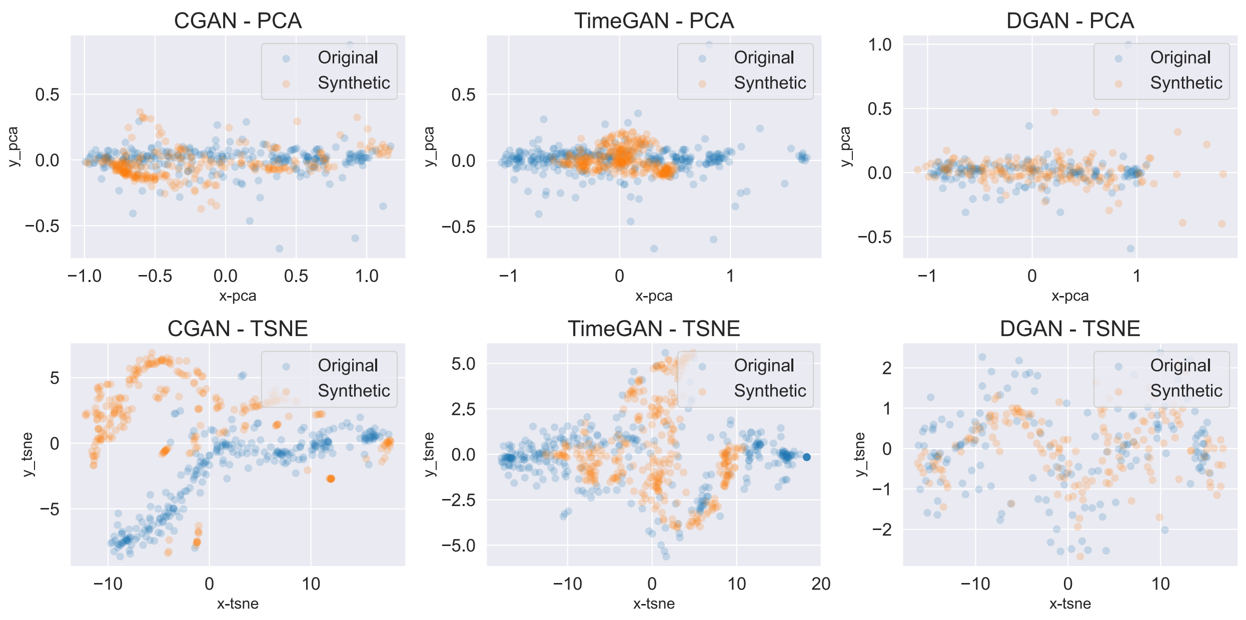


Figure 6.3.: Stress: The plots show the relative distribution of the synthetic GAN *stress* datasets (orange) compared to the original *stress* dataset (blue) in the two-dimensional feature space to see how well the GAN models mimic the original distribution.

6.2.1. Correlation analysis

In Figure 6.4 we compare the correlation matrices for the different synthetic datasets. Relevant correlations from the original dataset are between EDA and the *stress* label, with a strong positive correlation of 0.78 and a weak negative correlation between TEMP and the *stress* label of -0.35. The ACC signal correlation values of the WESAD dataset are in the negligible range, indicating no significant correlation between these signals and the *stress* state.

cGAN. The cGAN is able to moderately reproduce the correlation matrix of the WESAD dataset. The correlation between EDA and a stress moment is 0.91, indicating a strong positive relationship. Thus, the cGAN learned the strong positive relationship between EDA and the *stress* label of the original dataset. , the cGAN can only reproduce the ACC_z *stress* label relationship. The correlation between TEMP and *stress* label is -0.56, indicating a moderately negative relationship. This shows that the cGAN can also correctly capture the correlation for TEMP and *stress* label, but overestimates it. In terms of inter-signal correlations, the cGAN overestimates these, such as the correlation for BVP and EDA and EDA and TEMP. In conclusion, the cGAN is able to reproduce the most important trends. This varies depending on the type of signal being reproduced.

DGAN. The DGAN is able to reproduce the original correlations of the WESAD dataset very well, both between the signal types and the *stress* label, and the correlations between the signals, both in trend and in the correlation value itself. The correlation values differ only in a small range ($\pm 0.05 - 0.10$). For example, the strong positive correlation of EDA and the *stress* label comes very close to the original dataset, with a value of 0.74. The same is true for the slightly negative correlation between TEMP and *stress* label, with -0.32 and -0.35, respectively. In summary, the DGAN reproduces the correlations of the original dataset best.

TimeGAN. TimeGAN can reproduce well the correlation between EDA and *stress* label and also TEMP and *stress* label. On the other hand, it invents weak positive correlations like e.g., between ACC_z and EDA or even moderate negative correlations like e.g., between EDA and TEMP or EDA and ACC_y which are not present in the original dataset.

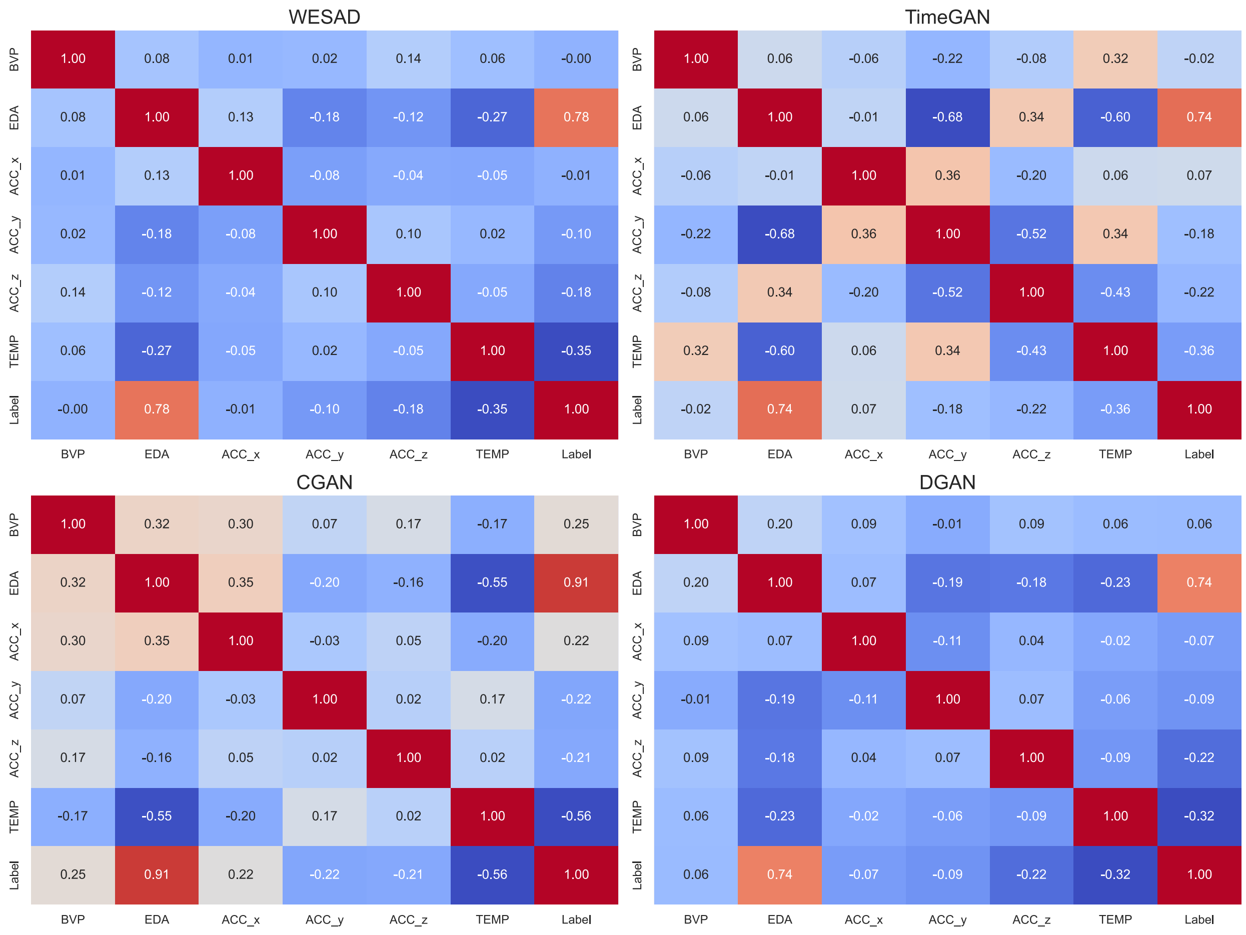


Figure 6.4.: Shown are the correlation matrices of the original WESAD dataset, as seen in Figure 4.3, and the synthetic datasets generated by the different GAN architectures. This figure shows whether the correlations between the respective signals and the stress state could be adequately learned by the GAN.

6.2.2. Statistical distribution

The evaluation of the statistical distribution provides an overview of the statistical feature differences between the original dataset and the synthetic datasets, based on which the indistinguishability of a synthetic dataset can be decided. Figure 6.5 shows the frequency distribution of the different synthetic datasets compared to the distribution of the original dataset for all signals in the stress state. The respective Figure 3 for the synthetic *non-stress* dataset can be seen in the Appendix 8.1. The frequency distributions are shown in the normalized range [0,1] of the datasets.

cGAN. It can be seen that the cGAN matches the statistical distribution of the original dataset well for most signal types. However, for the ACC signals in particular, it has difficulty mapping certain value ranges. We can see that the cGAN cannot properly generate the low value ranges of the TEMP signal in the stress state. While it seems to better capture the higher values range in the *non-stress* state.

TimeGAN. The TimeGAN does not perform much worse than the cGAN in the stress. However, it can be seen that the original distribution is distinguishable from the TimeGAN stress dataset. The same applies for the non-stress statistical distribution.

DGAN. The DGAN is a better representation of the original dataset than the cGAN with respect to the TEMP signal and the low range of values in the EDA. At the same time, it has similar difficulties in generating the ACC values.

In summary, the synthetic datasets generated by the cGAN and DGAN captures the characteristic statistical frequencies. However, the synthetic datasets are still distinguishable from the original dataset, not reproducing the identical frequency distribution for signal types.

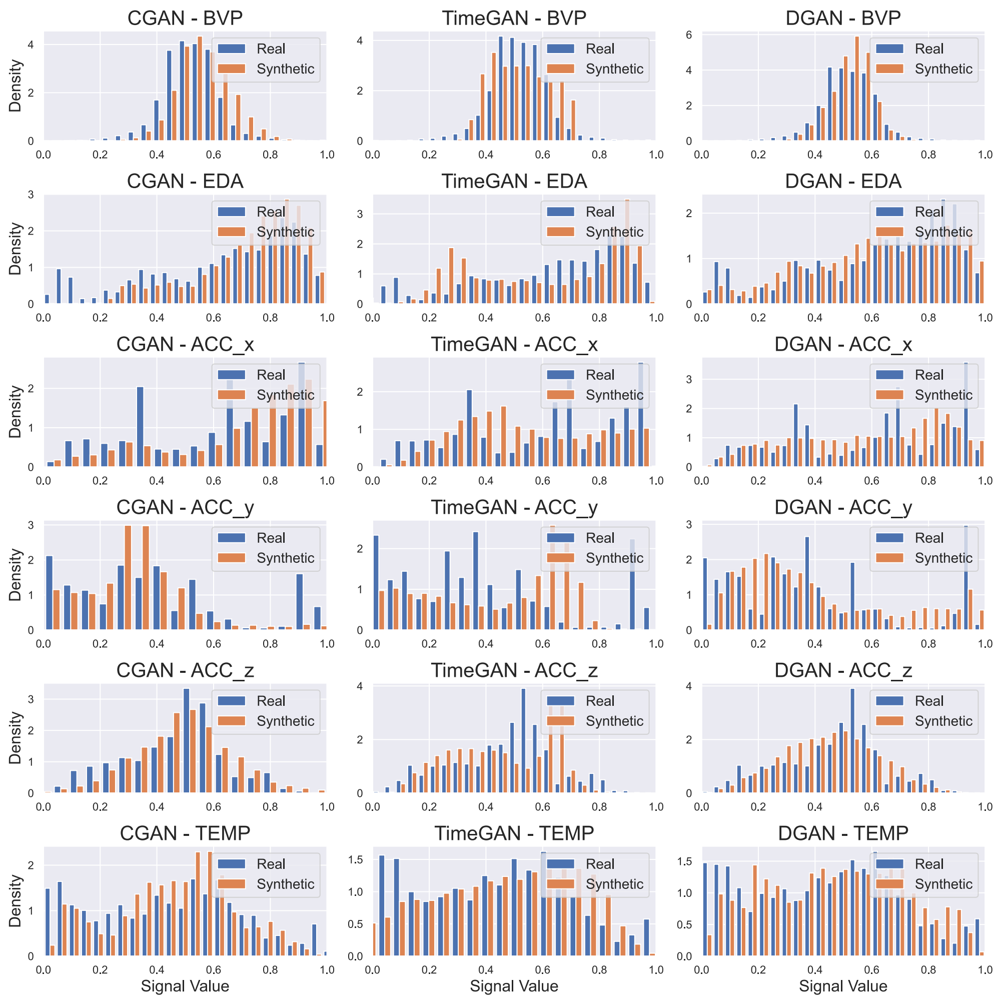


Figure 6.5.: Shown are the distributions of the stress signals from the original WESAD dataset and the synthetic signals generated by the different GAN architectures. For each signal type, the real distribution from the WESAD dataset, shown in blue, is compared to the synthetic signal, shown in orange. The density of the normalized signals is plotted in the range [0,1].

6.2.3. Indistinguishability testing

Figure 6.5 shows our evaluation by indistinguishability testing using five machine learning classifiers: Logistic Regression, KNN, Naive Bayes, Support Vector Machine (SVC), and a Multilayer Perceptron (MLP). The figure shows the accuracy achieved by each binary classifier in classifying a sequence from a synthetic dataset or original dataset. It is important to note that a high accuracy means that the classifier is often able to distinguish the synthetic from the real sample. The optimal accuracy for this test is 0.5. If a classifier has an accuracy of 0.5 in a binary classification problem, it is equivalent to chance, similar to a coin toss. This would indicate that, the synthetic dataset is indistinguishable from the original dataset. We test the classifiers on both *stress* and *non-stress* data individually.

cGAN. The cGAN synthetic data is consistently recognized as synthetic data for the *stress* label with a high accuracy of 0.83 to 1 across all classifiers, and the non-stress data, which can be identified with an accuracy of 0.73 to 0.9, less frequently. This indicates that the sequences generated by the cGAN are easy to distinguish from the original dataset and thus do not identically mimic the WESAD dataset.

DGAN. We find that the DGAN performs decisively better. Thus, the accuracy across the different classifiers is moderate low between 0.66 and 0.79 for the *stress* data and 0.62 to 0.86 for the *non-stress* data. These values indicate that the binary classifiers have moderate to strong difficulty in correctly identifying the origin of the DGAN generated synthetic sequences and the original sequences.

TimeGAN. The TimeGAN performs much more volatile, while it achieves the lowest result on the Naive Bayes classifier for *stress* sequences of 0.64. The other classifiers recognize the stress sequences with an accuracy of 0.94 to 0.99. For the *non-stress* data, the accuracy of the different classifiers is similarly unstable. Thus, it achieves moderate accuracy from 0.66 to 0.7 for the Logistic Regression, KNN and Naive Bayes, but the synthetic samples are detected by the SVC and MLP classifiers with an accuracy of 1.0.

The synthetic data that most consistently fooled the machine learning classifier was generated by the DGAN, followed by the inconsistent results for the TimeGAN, and finally the cGAN, which was consistently identified with high accuracy.

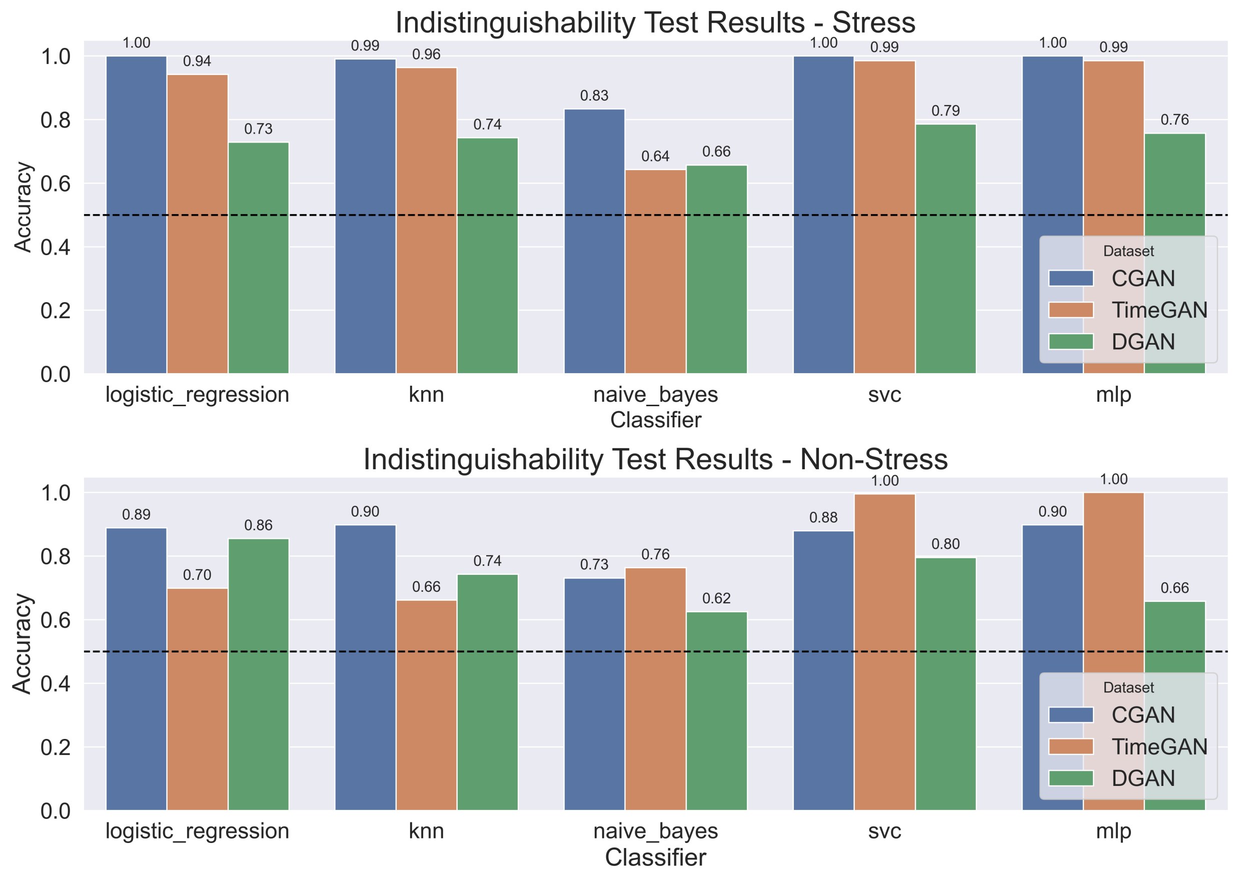


Figure 6.6.: Indistinguishability testing results comparing the performance of multiple binary classifiers: Logistic regression, KNN, Naive Bayes, Support Vector Machine (SVC), and a Multi Layer perceptron (MLP). Each bar shows one accuracy result on a specific dataset. The optimal accuracy is 0.5, which is represented by the dashed black line.

6.3. Usefulness: Stress Detection

We examine stress detection results for two different experiments. First, we evaluate the augmentation of the initial data with the synthetic data. Second, we apply the TSTR framework, where we train the stress detection model based only on the respective synthetic datasets and then evaluate it on the real data.

6.3.1. Augmentation

With this evaluation approach, we want to test whether augmenting the WESAD dataset with a synthetic subject increases stress detection over the other subjects. Table 6.1 depicts results over all subjects for the different GAN architectures, with an additional synthetic subject added to the original WESAD dataset. The synthetic subject has the size of an average subject with multiple 60-second stress windows with the shape `(72, 60, 7)`. To perform this evaluation, we applied the LOSO procedure and averaged the metrics subsequently obtained.

Among the evaluated GAN architectures, the model cGAN-FCN model achieves the highest accuracy with 0.912, followed by the cGAN-LSTM with 0.896. Both the DGAN and the TimeGAN

Augmented	cGAN-LSTM	cGAN-FCN	TimeGAN	DGAN	original
S2	0.971	0.941	0.824	0.971	0.971
S3	0.794	0.794	0.765	0.765	0.824
S4	0.971	0.971	0.971	1.000	0.971
S5	0.971	0.971	0.914	0.971	1.000
S6	1.000	0.971	0.971	0.943	0.943
S7	0.657	0.829	0.657	0.829	0.943
S8	0.914	0.971	0.914	0.943	0.943
S9	1.000	1.000	0.971	0.971	1.000
S10	0.944	0.944	0.972	0.944	0.972
S11	0.861	0.778	0.833	0.583	0.806
S13	1.000	1.000	1.000	0.917	0.972
S14	0.639	0.806	0.556	0.528	0.278
S15	0.972	0.944	0.972	0.972	0.972
S16	1.000	1.000	1.000	1.000	1.000
S17	0.750	0.750	0.500	0.500	0.667
Average	0.896	**0.912**	0.855	0.856	0.884

Table 6.1.: Overview of experiment results for GAN models for the augmentation case with adding one single synthetic subject to the real dataset. Depicted are the accuracy results achieved by training the stress detector using synthetic data over the individual subjects.

achieve an accuracy that is worse than the baseline model. This means that augmented data actually worsens the result in this case. When considering the other metrics such as F1-score, precision and recall, these metrics are similar to the accuracy on the corresponding GAN datasets as can be seen in detail in Table 6.2. Overall, these findings suggest that augmenting the dataset can improve the stress detection model, choosing the proper architecture such as cGAN-FCN or cGAN-LSTM. Moreover, the cGAN architectures succeed over the other models.

Augmented	Accuracy	F1 Score	Precision	Recall
cGAN-LSTM	0.896	0.895	0.895	0.895
cGAN-FCN	**0.912**	**0.912**	**0.912**	**0.913**
TimeGAN	0.855	0.855	0.855	0.857
DGAN	0.856	**0.855**	0.854	0.856
original	0.884	0.882	0.885	0.878

Table 6.2.: Overview of experiment results for GAN models for the augmentation case with adding one single synthetic subject to the real dataset. Depicted are the average results over all metrics achieved by training the stress detector using synthetic data. REAL represent the original baseline stress detection model trained on the 15 subject WESAD dataset.

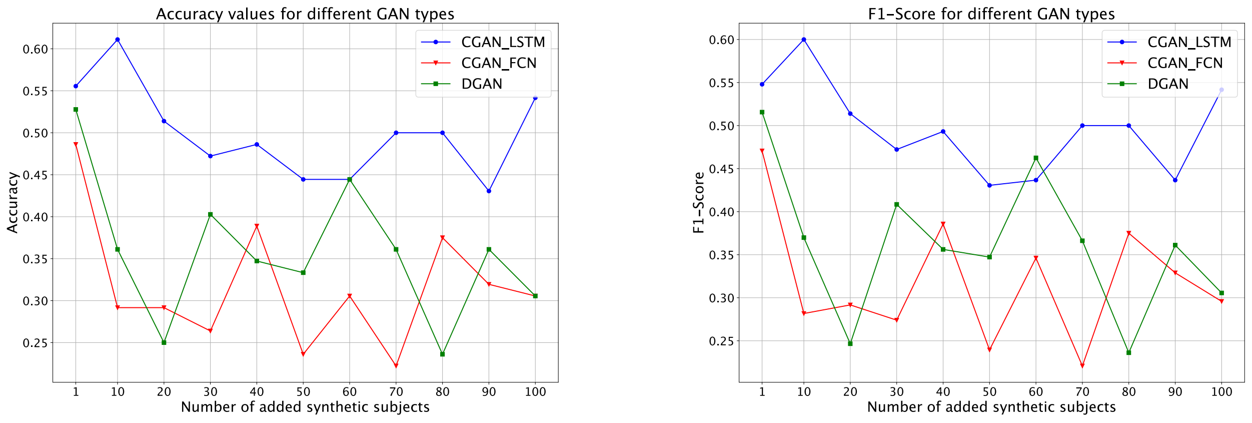
This approach, as described in 4.3, is prone to train-test leakage. Therefore, it is difficult to argue that the addition of synthetic data improves the generalizability of stress detection. To address this issue, we separately trained a GAN dataset without *Subject 14* to assess the generalizability of our synthetic data and show that the overall explainable improvement is not solely due to a possible train-test leakage. We only trained the cGAN with FCN and LSTM discriminator and the DGAN models on the WESAD dataset without *Subject 14*. We made this decision because of the significant time required to train the TimeGAN models, for both cases *stress* and *non-stress*, and accordingly did not include it in this evaluation. The results for stress detection on the augmentation by 1 synthetic subject that was generated by a GAN model that did not previously see *Subject 14* are in Table 6.3.

The use of synthetic data by the models that have not yet seen *Subjects 14* in the GAN training process shows that the addition of a single synthetic subject already significantly improves performance to the original stress detection accuracy of 0.278. Thus, the addition of one synthetic subject generated by the cGAN-LSTM model achieves an improved accuracy of 0.556 in the stress detection.

Subject	cGAN-LSTM	cGAN-FCN	DGAN	original
S14	**0.556**	0.486	0.417	0.278

Table 6.3.: Accuracy of cGAN-LSTM, cGAN-FCN, and DGAN compared to baseline on *Subject 14* in the augmentation evaluation. It is important to note that the GANs were trained on a WESAD dataset without *Subject 14* and then tested separately to address the training test leakage problem.

One motivation for this work is to address the scarcity of health data. We investigated how many synthetic subjects we could add to improve stress detection for *Subject 14*. We added synthetic subjects in the range [1,100] in increments of 10. The results are shown in Figure 6.7 for the cGAN-LSTM, cGAN-FCN, and DGAN. For the cGAN-LSTM, the best accuracy is achieved by adding 10 synthetic subjects. The accuracy of stress detection can be increased from 0.278 in the original stress detection model to 0.611. Adding more synthetic subjects does not improve stress detection performance for the cGAN-LSTM. The poorer performing cGAN-FCN and DGAN does not benefit from the augmentation by more synthetic subjects, and reaches a maximum accuracy value of 0.486, and 0.528 for one additional subject. The F1-score values are almost identical to the accuracy values for all GAN architectures.



(a) Accuracy (b) F1-score

Figure 6.7.: Comparison of accuracy and F1-score for the three GANs: cGAN-LSTM (blue), cGAN-FCN (red), and DGAN (green). The x-axis represents the number of synthetic subjects added to the stress detection training for the left out *Subject 14*. The y-axis shows the achieved values for accuracy and F1-score using this stress detection.

6.3.2. TSTR

For the TSTR evaluation method, we trained one model per synthetic GAN dataset. We then evaluated these models on each subject in turn. By training only on synthetic data and testing on real data, we can test whether the data is suitable for the real-world application of stress detection. The results suggest that they are: Table 6.4 shows the averaged values across all 15 subjects for the various metrics. The cGAN-LSTM performs best with an accuracy of 0.906 and an F1-score of 0.906. This is worse than the result obtained by data augmentation in section 6.3.1, but better than the original result, suggesting that it both mimicked and improved upon the distribution of the original dataset. Looking more closely at the results for individual subjects in Table 6.5, it is clear that they have improved, as previously poorly performing subjects such as *Subject 14* were detected much better.

The DGAN is also capable of generating synthetic data that, when used as training data for a stress detection model, performs 0.11 better than the stress model trained on the original data.

However, the TimeGAN achieved an accuracy of 0.3 and performed significantly worse than the baseline model. This is due to the fact that the stress model trained on the TimeGAN data classifies almost every sequence sample of the WESAD dataset as stress.

TSTR	Accuracy	F1 Score	Precision	Recall
cGAN-LSTM	**0.910**	**0.912**	**0.913**	**0.910**
cGAN-FCN	0.872	0.869	0.874	0.864
TimeGAN	0.300	0.300	0.300	0.300
DGAN	0.904	0.903	0.903	0.903
original	0.893	0.890	0.892	0.888

Table 6.4.: Overview of experiment results for GAN models for the TSTR case where the synthetic dataset comes in size of the WESAD dataset. Depicted are the average results over all metrics achieved by training the stress detector using synthetic data. REAL represent the original baseline stress detection model trained on the 15 subject WESAD dataset. DGAN stands for DGAN.

TSTR	cGAN-LSTM	cGAN-FCN	TimeGAN	DGAN	original
S2	0.897	0.765	0.309	0.941	0.941
S3	0.783	0.739	0.304	0.739	0.797
S4	1.000	0.943	0.286	1.000	1.000
S5	0.958	0.901	0.310	1.000	0.986
S6	0.986	0.972	0.310	0.986	1.000
S7	0.873	0.958	0.169	0.958	0.986
S8	0.972	0.972	0.282	0.986	0.986
S9	0.986	0.958	0.286	0.986	0.972
S10	0.986	0.986	0.315	0.986	0.986
S11	0.875	0.708	0.320	0.736	0.833
S13	0.986	0.861	0.320	1.000	0.986
S14	0.556	0.667	0.375	0.694	0.278
S15	0.986	0.972	0.319	0.972	1.000
S16	0.958	0.944	0.319	0.931	0.958
S17	0.863	0.740	0.274	0.644	0.685
Average	**0.910**	0.872	0.300	0.904	0.893

Table 6.5.: Overview of experiment results for GAN models for the TSTR evaluation. Depicted are the accuracy results achieved by training the stress detector using synthetic data over the individual subjects.

6.4. Evaluating first attempts on differential privacy

According to Jordon et al. [84], there is a natural trade-off between fidelity of synthetic data and the provided privacy guarantees. As the fidelity of the synthetic samples increases, the guaranteed privacy decreases. Jordon et al. [84] ask *how private can the data be made while still maintaining fidelity and usefulness.*

We followed this idea and tested the cGAN for different privacy guarantees $\epsilon = \{0.1, 1, 10, \infty\}$. We evaluate these analogous to the previous evaluation on the different GAN architectures in the three facets *diversity*, *fidelity*, and *usefulness*. As a GAN architecture, we use the best performing GAN from the previous evaluation, the cGAN-LSTM, and apply the different privacy guarantees to it.

However, it should be noted that the results of the training could not be optimally hyperparameter tuned depending on the epsilon value. Instead, the number of epochs, which becomes a hyperparameter in the training of GANs, was used to calculate the noise. Thus, this value is fixed, and if it is changed, the noise and thus the rest of the training process changes with it. In addition, this fixed calculation of the noise does not allow early stopping, because otherwise the noise calculation and the associated privacy guarantee would not be allowed.

6.4.1. Diversity evaluation on the DP-cGAN

Figure 6.8 presents 50 stress sequence samples per signal type for the different privacy guarantees. The plot shows that the signal profile could not be captured for all signal types by all DP-cGAN models. The DP-cGAN with $\epsilon = 0.1$ covers the whole range [0,1] for all signals, it is therefore diverse in its sample sequences, but cannot capture the signal profile, only for BVP where it captures the erratic, jumping trend. The DP-cGAN with $\epsilon = 1$, on the other hand, is less diverse and occurs mainly for all signal types outside the range [0,1]. The sequences of the DP-cGAN with $\epsilon = 10$ are also less diverse, while this one covers the primary range of [0.6,1] for the EDA. The same is true for ACC_x and ACC_y, where the primary range of [0.6,1] and [0.1,0.5] is well captured and reproduced.

These plots suggest that the models are not capable of reproducing visual similar sequences. The training requires further hyperparameter tuning and especially training over more epochs to initially obtain the [0,1] range for all signal types, as in the case of DP-cGAN $\epsilon = 1$.

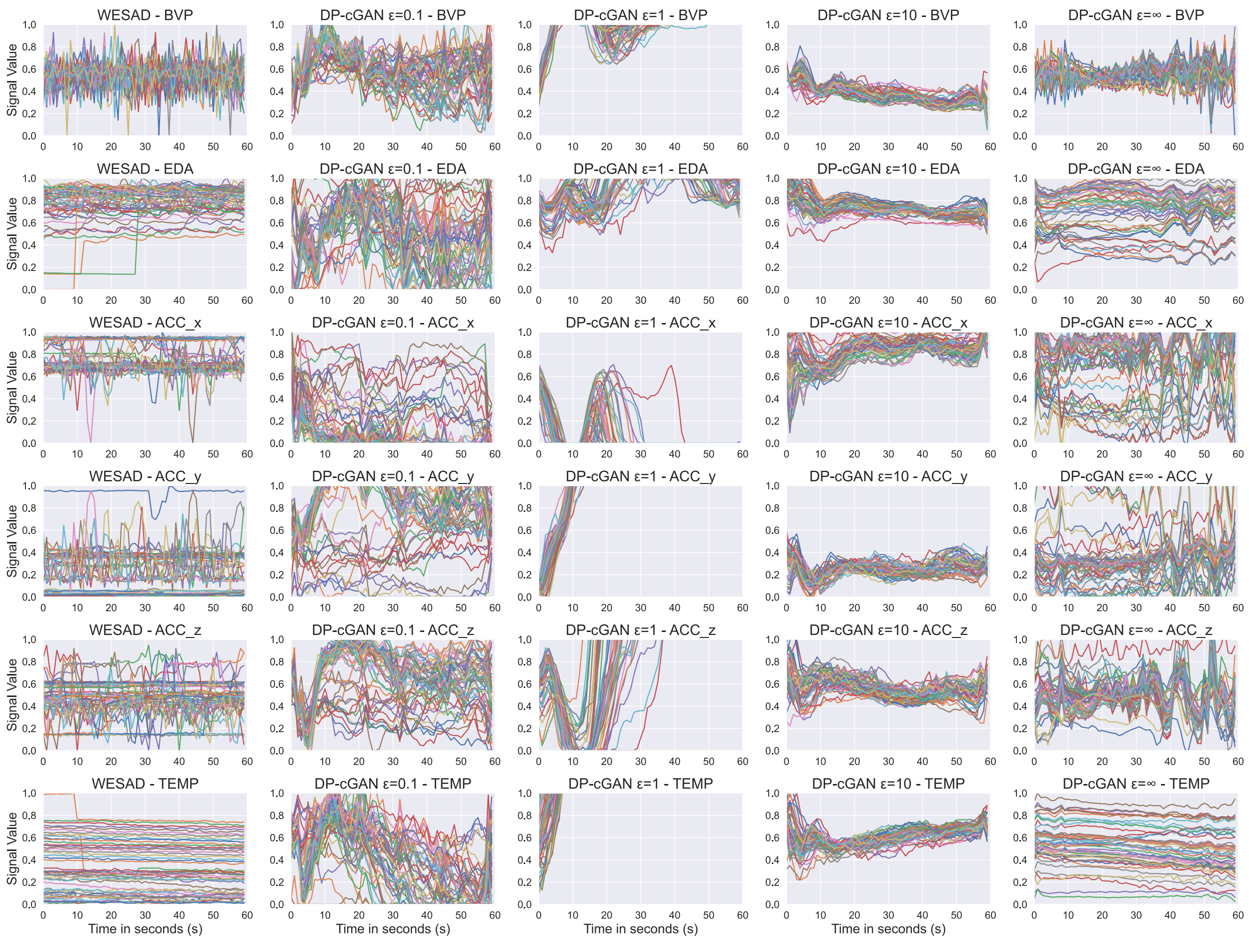


Figure 6.8.: Shown are 50 *stress* sequences from the original WESAD dataset and synthetic sample sequences generated by the different DP-cGAN models. These samples are plotted per row for each signal type: BVP, EDA, ACC_x, ACC_y, ACC_z, and TEMP. The columns show the corresponding dataset. This overview allows comparing the influence of privacy guarantees to the generation of samples.

The same conclusion is reinforced by the visual representation of the sequences in the PCA and t-SNE plots in Figure 6.9. On the one hand, the DP-cGAN with $\epsilon = 0.1$ in the PCA analysis shows that the synthetic sequences are clustered around the distribution of the original dataset, while some data points overlap. On the other hand, the synthetic dataset may somehow represent the diversity of the original dataset by covering most of the blue original data points. The DP-cGAN with $\epsilon = 1$ has all its values far from the original dataset clustered, for both PCA and t-SNE analysis. This visual information is consistent with the impression from the signal sequence in Figure 6.8 that the DP-cGAN is not capable of producing similar synthetic sequence samples. The synthetic sequences of the DP-cGAN with $\epsilon = 10$ are clustered in one spot over the range of the original dataset in the PCA analysis and cannot represent the diversity of the original dataset well. The same can be seen in the t-SNE analysis, where the data points are strictly separated. The sequences of DP-cGAN with $\epsilon = \infty$ in the PCA plot align quite well with the original data points in the PCA analysis and are thus able to reproduce the diversity of the dataset, but the t-SNE analysis can clearly separate the original and synthetic data.

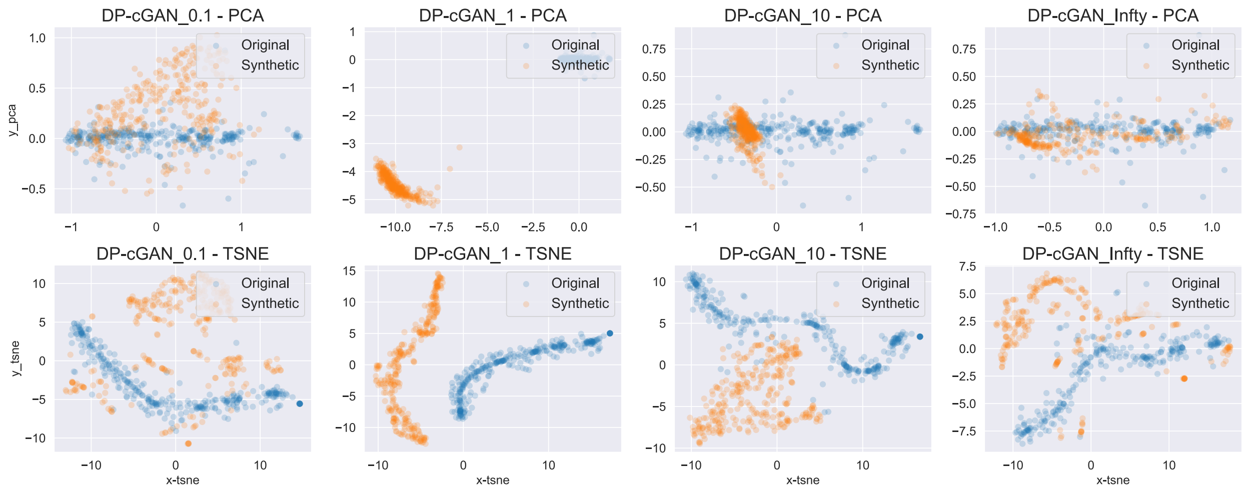


Figure 6.9.: The figure shows the PCA and t-SNE analysis to compare the synthetic stress dataset with the original stress WESAD dataset. The four different privacy guarantees $\epsilon = \{0.1, 1, 10, \infty\}$ are shown per column. The plots show the relative distribution of the synthetic GAN datasets (orange) compared to the original dataset (blue) in the two-dimensional feature space to see how well the GAN models mimic the original distribution.

6.4.2. Fidelity evaluation on the DP-cGAN

To determine the *fidelity* for the DP-cGAN, we look at the statistical distribution of the synthetic datasets generated by the DP-cGAN. Figure 6.10 shows the distributions for the different signal types in the stress state. First, the point already made in the *diversity* evaluation shows that some DP-cGANs generate signal sequences outside the range [0,1]. This is especially true for the DP-cGAN with $\epsilon = 1$, which generates clearly distinguishable data. The DP-cGAN with $\epsilon = 0.1$ generates more data points around near the mode of the signal distribution. The DP-cGAN with $\epsilon = 10$, is also pretty distinguishable from the original dataset.

6.4.3. Usefulness evaluation on the DP-cGAN

To determine the *usefulness* of the different privacy guarantees $\epsilon = \{0.1, 1, 10, \infty\}$, we trained the stress detection model as described in Section 5.5 on the synthetic datasets of the DP-cGAN models. The results were obtained using the TSTR method. The result of this experiment can be seen in Table 6.6.

$\epsilon = 0.1$. The model surprisingly performs better than the models with $\epsilon = \{1, 10\}$ compared to the baseline, the stress detection model trained on the original dataset. The stress detection model trained on privacy-preserved dataset achieves an accuracy of 0.811 This is less expected with this high privacy guarantee of $\epsilon = 0.1$.

$\epsilon = 1$. What is interesting about the result of this DP-cGAN model is that it performs worse than the DP-cGAN with $\epsilon = 0.1$ and and only slightly better than a naive majority classifier. This is a classifier that can serve as a baseline and intuitively always yields the majority class result, in our case the non-stress class classification. In our case, applied to the WESAD dataset, this classifier achieves an accuracy of 0.705, while the DP-cGAN model achieves an accuracy of 0.708 and an F1-score of 0.708.

However, the analysis of the sample sequences in Figure 6.8 shows that this model produces many more uniform samples, mainly outside the range [0,1], and does not learn the time-dependent signal profile of the signals better than $\epsilon = 0.1$, which explains this worse performance. Nevertheless, this model still performs better than the majority classifier.

$\epsilon = 10$. The stress detector model trained on the DP-cGAN data with $\epsilon = 10$ achieves an accuracy of 0.780 and an F1-score of 0.780.

These results of the *usefulness* evaluation show that it is possible to use a synthetic dataset with privacy guarantees to train a stress detection model. With a surprisingly low value of $\epsilon = 0.1$, a good stress detection performance in terms of accuracy and F1-score can be obtained. Here, the stress model has only seen synthetic data with privacy guarantees. However, these results do not support the claim *the higher the privacy guarantees, the more the performance decreases*, not confirming the trade-off between privacy and performance.

However, it is also evident that the results of the previous visual and statistical evaluation for the DP-cGAN with $\epsilon = 1$ are confirmed here in its poor performance. It only achieves an accuracy of 0.709 because the stress detection model trained on this data classifies the *non-stress* class for almost every test sequence. At the same time, interestingly, the DP-cGAN with $\epsilon = 0.1$ achieves a good utility performance, although the visual representation of the stress sequences does not suggest this. Only the representation of the *non-stress* sequence data in the Appendix 8.1 in Figure 1 gives an indication of the successful reproduction of the non-stress state of a range for the BVP signal, which is initially reproduced in its central fluctuation, and the TEMP signal, which occurs mainly in the upper range [0.5,1].

TSTR	Accuracy	F1 Score	Precision	Recall
original	0.893	0.890	0.892	0.888
DP-cGAN-$\epsilon = 0.1$	0.811	0.812	0.811	0.813
DP-cGAN-$\epsilon = 1$	0.708	0.708	0.708	0.708
DP-cGAN-$\epsilon = 10$	0.778	0.780	0.775	0.785
DP-cGAN-$\epsilon = \infty$	0.901	0.900	0.901	0.900

Table 6.6.: Overview of DP experiments in the TSTR training setting, where the synthetic dataset comes in size of the WESAD dataset. Depicted are the average results over all metrics achieved by training the stress detector using the different privacy guarantees $\epsilon = \{0.1, 1, 10, \infty\}$ on the cGAN-LSTM architecture. Original represents the baseline stress detection model trained on the 15 subject WESAD dataset.

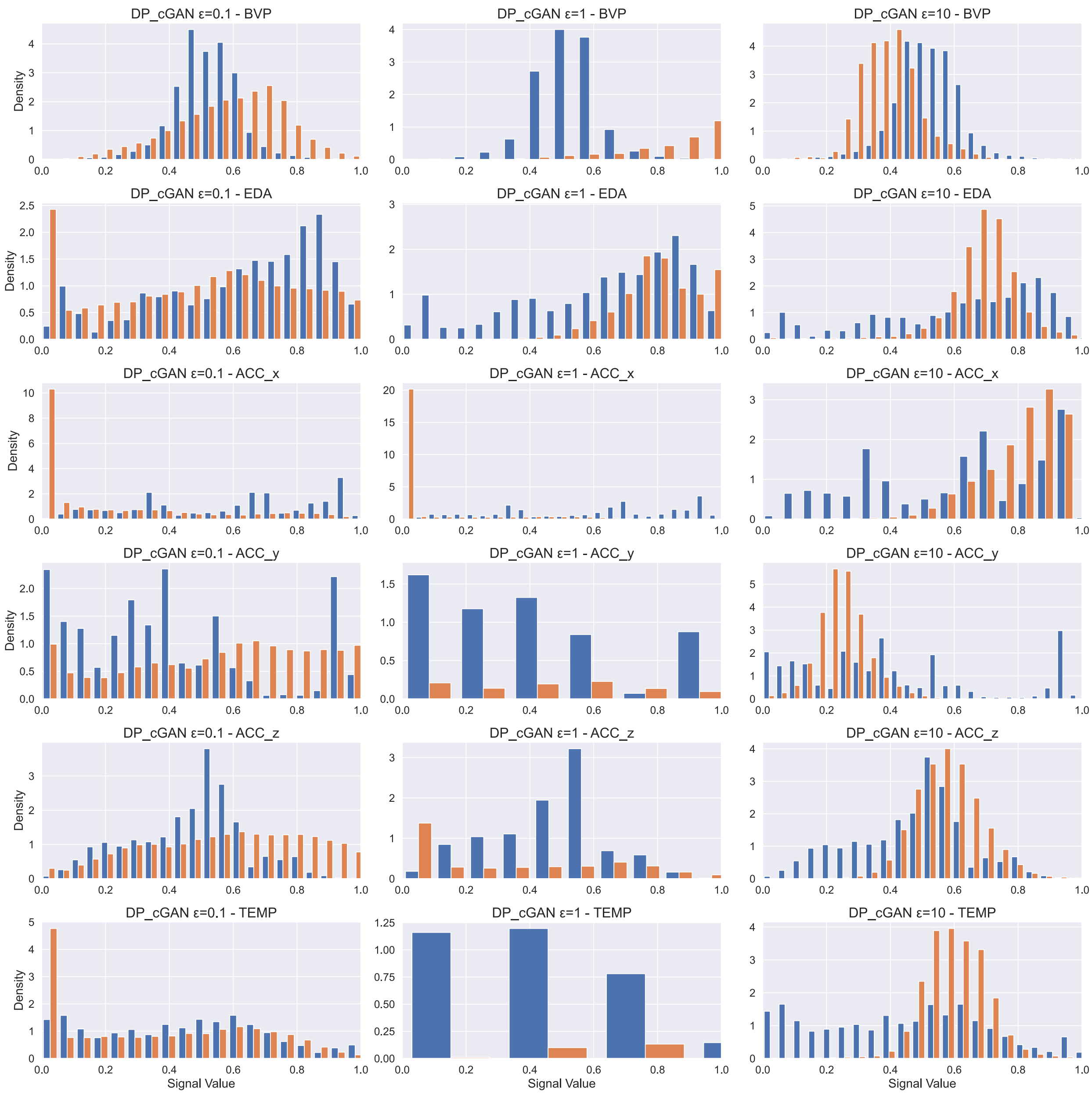


Figure 6.10.: Shown are the distributions of the stress signals from the original WESAD dataset and the synthetic signals generated by DP-cGAN with different privacy guarantees $\epsilon = \{0.1, 1, 10\}$. For each signal type, the real distribution from the WESAD dataset, shown in blue, is compared to the synthetic signal, shown in orange. The density of the data is not plotted in the range [0,1] because some DP-cGANs have not learned this range and plot values outside this range.

7. Discussion

In the following chapter, we will examine the results of the previous chapter. We will highlight several advantages of GANs in general that have been shown by our experiments, but also difficulties related to our processes of training, evaluation, and ensuring differential privacy. Then, we further consider the impact of the GAN models and our synthetic datasets on the smartwatch use case and stress detection on smartwatches in particular.

7.1. Advantages and Limitations of GANs

In this work, we were able to train several GANs that are able to learn the statistical distribution features of the original WESAD dataset. This allows us to generate synthetic data at different sizes. Our evaluation shows that both generating a synthetic dataset at the scale of the initial dataset in the TSTR evaluation setting and generating a synthetic subject for augmenting the original dataset improved the baseline stress detection results based on the WESAD dataset.

However, the models do not all perform equally well. The performance of TimeGAN is only slightly worse than the other models at 0.873, but it degrades the baseline model, which is not useful for the augmentation case. For the TSTR evaluation, it only achieves a result of 0.3, as it classifies all sequences of the WESAD dataset as stress. This is probably due to the poorly trained non-stress data, which, as in Section 6.1, poorly represent the diversity of the original dataset. This is a difficulty that has been confirmed in the use of TimeGAN, namely the training of the TimeGAN model twice for the split datasets to capture the conditional aspect. It ended up producing good stress data, while it failed in learning the non-stress data distribution.

The results of the evaluation via the LOSO procedure requires further verification. Since the synthetic datasets were trained on the complete WESAD dataset, it is possible that certain improvements in accuracy result from a train-test-leakage. To address this and verify the improvement based on the synthetic data augmentation, we trained an additional GAN model for the cGAN-LSTM, cGAN-FCN and DGAN that, unlike the previous models, did not see *Subject 14* of the WESAD dataset in training, as a case study. We then performed the LOSO stress detection training on the respective synthetic datasets and were able to significantly improve the stress detection with all GAN models on this subject.

We then incrementally added another 10 synthetic subjects until we reached 100. We wanted to see if adding more synthetic unseen data would increase performance and when accuracy might reach a tipping point and level off. We found that the highest accuracy in stress detection was achieved after adding 10 synthetic subjects. We were able to show that the addition of 10 more synthetic subjects improved the baseline accuracy from 0.278 to 0.611. The addition of more synthetic subjects ensured that the accuracy always remained better than in the case of no synthetic data. This suggests that the ratio of real data to synthetic data may be critical. Since we have a dataset of 15 real subjects, adding 10 more synthetic subjects based on them is already a large proportion of the total dataset.

Furthermore, it is clear that we cannot add as many subjects as we want until we reach a desired accuracy.

Thus, we can show that the generation of synthetic data can improve the performance of the stress detection model without the need to measure new real data in the laboratory, which is very time-consuming and expensive.

In addition to data augmentation, GANs provide the possibility of ensuring DP. This can be a key advantage among researchers who want to share datasets for reproducibility or further research. This is especially valuable when the original dataset contains highly sensitive data, such as patient medical data. Thus, given the right DP parameter, a certain level of privacy is guaranteed, while other research institutions can work on the synthetic equivalent of a privacy-protected dataset. The evaluation of the application of DP to the cGAN from Section 6.4 has shown that the extension of an existing cGAN architecture in the discriminator with a DP optimizer is both an easily integrable and successful introduction of DP to a GAN architecture. We could not confirm the trade-off between privacy and performance, as our DP-cGAN with $\epsilon = 0.1$ produced better synthetic data for the use case of stress detection than the DP-cGAN with $\epsilon = \{1, 10\}$. On the other hand, we found that the fixed calculation of the noise parameter based on the number of epochs in a highly fluctuating training between generator and discriminator leads to difficulties in determining the optimal model.

While GANs provide these significant benefits, there are also difficulties in handling and training these GANs. The training of our GAN models has suffered from well-known GAN training difficulties, such as mode collapse, i.e., the learned mapping of noise to one or a few output samples, or the difficulty of estimating the performance of a GAN during training. We were only able to adapt to these problems by performing a time-consuming hyperparameter sweep for each GAN. Only in the training of the DGAN we had no difficulties with the mode collapse.

Considering the evaluation metrics we used, we were able to estimate the performance via the visual and statistical output of the samples. However, this form of evaluation is time-consuming and otherwise cumbersome, since across all epochs this output must be examined in correlation with the losses of the generator and discriminator. It is also difficult to determine the optimal number of epochs, since GANs tend to train in equilibrium from a point in time above generator loss and discriminator loss, which degrades the output, making the determination of the optimal epoch another hyperparameter to optimize. The training stopping conditions by e.g., early stopping is difficult to implement if the actual evaluation of a synthetic dataset can only be determined in the usefulness evaluation case, i.e., the use of our dataset for stress detection. Besides, this raises the question of how the synthetic datasets we generate should look like. Should the GANs primarily learn the statistical distribution and correlations within the original data particularly well, as the case of the DGAN shows, or should it improve the performance in the use case of stress detection, as the cGAN does. Since in our case, the research question is whether the performance of the stress detection can be improved by adding synthetic data. With this question in mind, further evaluation metrics can be developed. Thus, an optimal evaluation pipeline would trigger the training of the stress detector model during the training after a certain number of epochs. This would be automatically evaluated to produce a usefulness metric measure after this epoch in terms of accuracy or F1-score.

Another limitation of GANs and the resulting dataset are ethical concerns. A synthetic dataset learns only the underlying distribution of statistical features from the original dataset. This also means that a synthetic dataset is always as biased as the underlying dataset. Researchers show that popular machine-learning datasets that are the fundament for important machine learning insight can contain different biases [85]. This affects both computer vision task datasets such as ImageNet [81] and OpenImages [86], which are affected by representation bias. Additionally, writes Mehrabi et al. [87] about existing representation biases in knowledge base, which are used as a basis for various Natural Language Processing (NLP). Thus, biases which are present in datasets are replicated via the synthetic dataset. This leads to gender or demographic imbalances, which then leads to unfairness in machine learning systems build on these datasets.

In terms of the WESAD dataset we used, this means that we reproduce the inequality in the gender distribution, 12 males and 3 females, and even amplify it when generating additional data points over the size of the dataset. For example, it can be seen in Table 6.1 that generating a synthetic dataset does not achieve the same improvement across genders. Thus, while for the females in the dataset *Subject 8* and *Subject 11* average stress detection performance worsen for certain GANs, the male of *Subject 14* improves significantly. Thus, the average improvement on female subjects for the cGAN-LSTM architecture is 0.013, while for male character it is 0.018. The same applies for the DGAN model, which has an average improvement of 0.02, while the female stress detection decreases by 0.04. However, these changes cannot solely be attributed to gender, as other subjects also showed reduced performance after the augmentation of the dataset by synthetic data. This should be further investigated in future studies.

7.2. Insights Regarding Practical Smartwatch Users

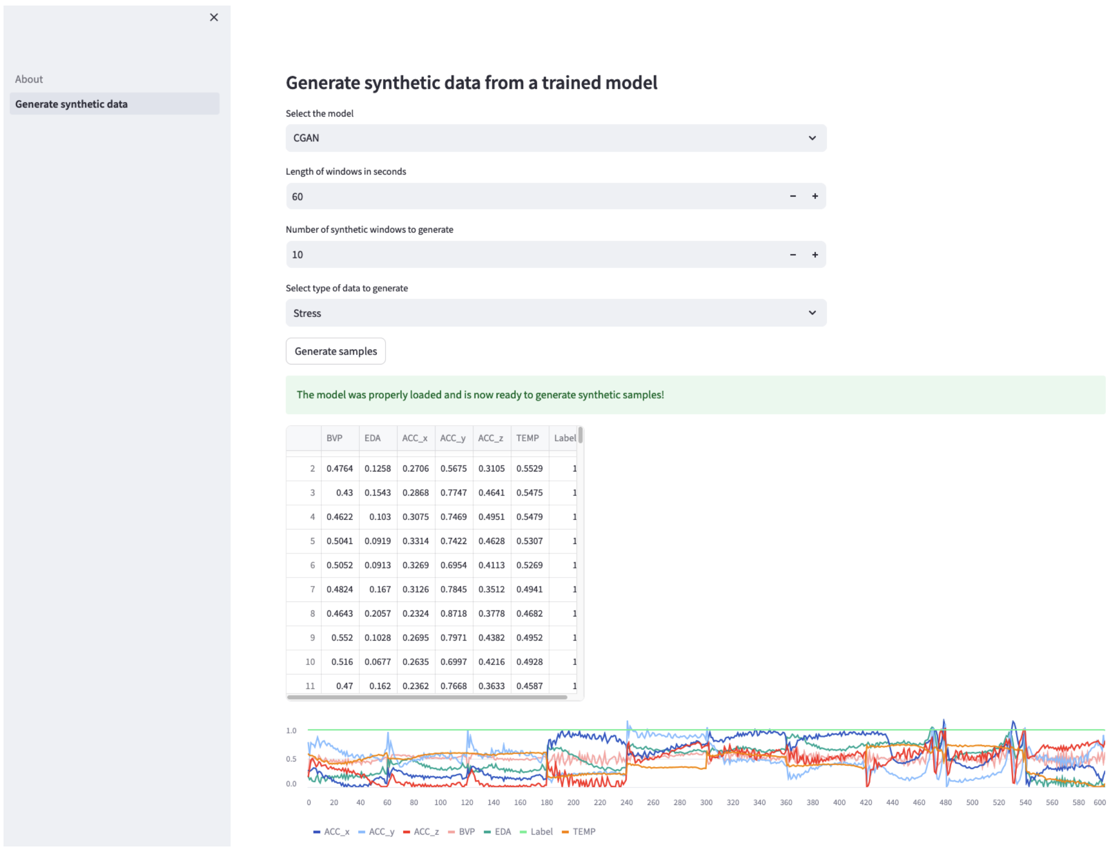
In the following, we consider the implications for our application area of wearable smartwatch stress data. Thus, synthetically generating the WESAD dataset means enabling data augmentation that primarily saves time and costs beyond one and a half hours and research personnel for 15 subjects. It thus offers the possibility of taking a small dataset as a starting point and augmenting it with synthetic samples. At the same time, our TSTR experiments have shown that this very use of a synthetic dataset is able to improve the generalizability of the final stress model. As a consequence, it is possible to place small parts of the population for new datasets in comparable lab situations as the subjects in the WESAD laboratory scenario, in order to subsequently improve stress detection decisively.

Furthermore, we were able to achieve a stress detection performance with a high privacy guarantee of 0.811 . Using DP, this could mean that for further Empatica E4 stress detection data collections, data that is actually privacy-preserving could be transferred from research institution to research institution. These data could accelerate specific research areas in the form of synthetic privacy-preserving data. For example, stress research could develop better approaches based on bigger benchmark datasets and address the individuality and difficult generalizability of stress through collaborative research on larger stress datasets. These datasets might not be publicly available without applying DP because of their sensitive individual health data

Improved stress detection rolled out via many commercial wearable smartwatch devices, such as the Fitbit or the Apple Watch, could provide crucial medically relevant information on a person's health status through support via the additional synthetic data. As chronic long-term stress is an increasingly common cause of illness, depression, and absenteeism, the ability to expand the dataset and generate synthetic samples from it provides a small service of education and measurement to curb stressful events.

7.3. Development of a Frontend Interface for Data Synthesis using GANs

To make the results of this work more easily available, we have developed a prototype frontend for generating synthetic stress and non-stress data based on the WESAD dataset for two model types, cGAN and the DP-cGAN. To implement the frontend, we used the Streamlit[23] library. Figure 7.1 shows the user interface of the application, which is currently running on the user's localhost. The user can select one of the GAN models cGAN and DP-cGAN with different $\epsilon = \{0.1, 1, 10\}$ privacy guarantees, from a drop-down menu. Then the number of synthetic samples to be generated can be specified, and the length of the sequence in seconds, which can be selected in the next input field. As a reminder, the WESAD dataset has a total of 545 unique 60-second sequences over the 15 subjects, with each subject containing an average of 36 60-second windows. Another option is to select whether to generate stress only, non-stress only, or both. Clicking the *Generate samples* button will generate the desired synthetic dataset, while providing basic statistical information about the data distribution of each signal and plotting the windows of the dataset. The data can then be downloaded in *.csv* format.



Generate synthetic data from a trained model

Select the model: CGAN

Length of windows in seconds: 60

Number of synthetic windows to generate: 10

Select type of data to generate: Stress

The model was properly loaded and is now ready to generate synthetic samples!

	BVP	EDA	ACC_x	ACC_y	ACC_z	TEMP	Label
2	0.4764	0.1258	0.2706	0.5675	0.3105	0.5529	1
3	0.43	0.1543	0.2868	0.7747	0.4641	0.5475	1
4	0.4622	0.103	0.3075	0.7469	0.4951	0.5479	1
5	0.5041	0.0919	0.3314	0.7422	0.4628	0.5307	1
6	0.5052	0.0913	0.3269	0.6954	0.4113	0.5269	1
7	0.4824	0.167	0.3126	0.7845	0.3512	0.4941	1
8	0.4643	0.2057	0.2324	0.8718	0.3778	0.4682	1
9	0.552	0.1028	0.2695	0.7971	0.4382	0.4952	1
10	0.516	0.0677	0.2635	0.6997	0.4216	0.4928	1
11	0.47	0.162	0.2362	0.7668	0.3633	0.4587	1

Figure 7.1.: Shown is the user-friendly interface of the frontend application that helps the generation of synthetic stress and non-stress data across different GAN models. Drop-down menus are used to customize the data synthesis parameters. The GAN type, the number of synthetic sample sequences, the desired sequence length, and the stress type can be selected.

[23]https://streamlit.io

8. Conclusion

We have developed and trained several GAN models capable of generating synthetic datasets based on the original WESAD dataset. For this purpose, we first considered the characteristics of the dataset to develop appropriate GANs. We implemented several state-of-the-art GAN architectures to generate multiple synthetic datasets: TimeGAN, cGAN, and DGAN. Not all GANs performed equally well, both in handling the training and in tuning the hyperparameters until a stable training was established. For example, TimeGAN training takes significantly more time than cGAN and DGAN training.

We have shown that augmenting the WESAD dataset with synthetic samples can improve stress detection. We were able to improve the performance of a CNN stress detection model by adding one, several, or 100 synthetic subjects to the LOSO training, answering our first research question. We were also able to show in the TSTR evaluation that training on the synthetic generated data and testing on the original WESAD dataset improved the generalizability of the stress detection model across the subjects on average.

To compare the different GAN models, we applied an evaluation method that considers three different facets of the synthetic data. We evaluated the *diversity* of the synthetic samples visually based on their similarity to the original dataset. This allowed us to demonstrate that the GANs could reproduce typical stress characteristics for the EDA and TEMP signals, with reference to domain knowledge from stress research. In addition, PCA and t-SNE analysis were performed to determine the similarity in *diversity* between the synthetic and original data.

We looked at the statistical distribution and correlation within signals to determine the *fidelity*. Thus, we assess how well the synthetic generated data is indistinguishable from the original data using various binary machine learning classifiers in an indistinguishability test. Finally, we investigated the *usefulness* of this synthetic data by training CNN stress detection models. From these metrics, we were able to show that although synthetic sample sequences were visually or statistically more similar to the original data, this did not necessarily mean a better result for the stress detection use case. This made training and hyperparameter tuning more difficult.

Finally, we have shown that DP can be used to preserve the privacy of the synthetic data, thus answering our second research question. To do this, we extended the cGAN model with a DP optimizer for the discriminator. We were able to show that this privacy-preserving data achieved good results in stress detection under the privacy-performance trade-off, while preserving privacy to a high degree. At the same time, we have also shown the difficulties in training DP in GAN, which is mainly in finding the optimal balance of GAN in training.

8.1. Future Research Directions

Based on the results of this work, further research questions and tasks arise to improve the synthetic generation of smartwatch data. In particular, the difficulty of determining the optimal model in the training process has been demonstrated. To address this, the development of an end-to-end training pipeline that triggers the training and evaluation of the stress model after a certain number of epochs during GAN training to better track performance would improve the training of these GAN models for specific use cases.

As suggested by Ehrhart et al. [70] in their future work, the implementation of the Wasserstein loss for cGAN model would be another possible improvement on the cGAN training. This also applies to this work and the cGAN model used here.

Another important research that follows this work is the investigation of the synthetic data for their stated DP guarantees. Among other things, performing an unintentional memorization attack on the various synthetic DP datasets by placing a canary inside the synthetic dataset would be a simple, meaningful attack. [78]

Another interesting investigation is the work and training of GANs using fewer signal types of the WESAD dataset. More different signals mean for the GANs to learn more complex relationships and correlations between these signals. Furthermore, Siirtola et al. [34] and Gil-Martin et al. [4] have shown for the WESAD dataset that stress detection can work better with fewer signal types, which makes the explicit generation of more sequences of these signal types interesting.

Since it showed that synthetic datasets reproduce the biases in the underlying original dataset, further work could address the imbalance in demographics, ethnicity, and gender within the original dataset. Thus, synthetic samples could be generated to augment the original dataset, counteracting this imbalance. This approach could be inspired by the work on FairGANs by Xu et al. [88] to generate a fair dataset for smartwatches.

Bibliography

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems.* Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: `https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[2] Scott Mayer McKinney et al. "International evaluation of an AI system for breast cancer screening". In: *Nature* 577.7788 (2020), pp. 89–94.

[3] George Fink. "Stress, definitions, mechanisms, and effects outlined: Lessons from anxiety". In: *Stress: Concepts, cognition, emotion, and behavior.* Elsevier, 2016, pp. 3–11.

[4] Manuel Gil-Martin et al. "Human stress detection with wearable sensors using convolutional neural networks". In: *IEEE Aerospace and Electronic Systems Magazine* 37.1 (2022), pp. 60–70.

[5] Philip Schmidt et al. "Introducing wesad, a multimodal dataset for wearable stress and affect detection". In: *Proceedings of the 20th ACM international conference on multimodal interaction.* 2018, pp. 400–408.

[6] Arvind Narayanan and Vitaly Shmatikov. "Robust de-anonymization of large sparse datasets". In: *2008 IEEE Symposium on Security and Privacy (sp 2008).* IEEE. 2008, pp. 111–125.

[7] Mohammad Abufadda and Khalid Mansour. "A Survey of Synthetic Data Generation for Machine Learning". In: *2021 22nd International Arab Conference on Information Technology (ACIT).* IEEE. 2021, pp. 1–7.

[8] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. "Deep models under the GAN: information leakage from collaborative deep learning". In: *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security.* 2017, pp. 603–618.

[9] Liyang Xie et al. "Differentially private generative adversarial network". In: *arXiv preprint arXiv:1802.06739* (2018).

[10] Lukasz Piwek et al. "The Rise of Consumer Health Wearables: Promises and Barriers". In: *PLoS Medicine* 13 (2016).

[11] Christine E. King and Majid Sarrafzadeh. "A Survey of Smartwatches in Remote Health Monitoring". In: *Journal of Healthcare Informatics Research* 2 (2018), pp. 1–24.

[12] Roland Rosmond and Per Björntorp. "Endocrine and metabolic aberrations in men with abdominal obesity in relation to anxio-depressive infirmity". In: *Metabolism* 47.10 (Oct. 1998), pp. 1187–1193. DOI: `10.1016/s0026-0495(98)90321-3`. URL: `http://dx.doi.org/10.1016/S0026-0495(98)90321-3`.

[13] Bruce S. McEwen. "Stress and the Individual". In: *Archives of Internal Medicine* 153.18 (Sept. 1993), p. 2093. DOI: `10.1001/archinte.1993.00410180039004`. URL: `http://dx.doi.org/10.1001/ARCHINTE.1993.00410180039004`.

[14] George P. Chrousos. “The Concepts of Stress and Stress System Disorders”. In: *JAMA* 267.9 (Mar. 1992), p. 1244. DOI: 10.1001/jama.1992.03480090092034. URL: http://dx.doi.org/10.1001/JAMA.1992.03480090092034.

[15] Suzanne C. Segerstrom and Gregory E. Miller. “Psychological Stress and the Human Immune System: A Meta-Analytic Study of 30 Years of Inquiry.” In: *Psychological Bulletin* 130.4 (2004), pp. 601–630. DOI: 10.1037/0033-2909.130.4.601. URL: http://dx.doi.org/10.1037/0033-2909.130.4.601.

[16] A. Angeli et al. “The overtraining syndrome in athletes: A stress-related disorder”. In: *Journal of Endocrinological Investigation* 27.6 (June 2004), pp. 603–612. DOI: 10.1007/bf03347487. URL: http://dx.doi.org/10.1007/BF03347487.

[17] Martin Gjoreski et al. “Continuous Stress Detection Using a Wrist Device: In Laboratory and Real Life”. In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. UbiComp ’16. Heidelberg, Germany: Association for Computing Machinery, 2016, pp. 1185–1193. ISBN: 9781450344623. DOI: 10.1145/2968219.2968306. URL: https://doi.org/10.1145/2968219.2968306.

[18] W.B. Cannon. “The Wisdom of the Body”. In: *Nature* 133.3351 (Jan. 1934), pp. 82–82. DOI: 10.1038/133082a0. URL: http://dx.doi.org/10.1038/133082a0.

[19] Maurizio Garbarino et al. “Empatica E3 - A wearable wireless multi-sensor device for real-time computerized biofeedback and data acquisition”. In: *Proceedings of the 4th International Conference on Wireless Mobile Communication and Healthcare - "Transforming healthcare through innovations in mobile and wireless technologies"*. ICST. 2014. DOI: 10.4108/icst.mobihealth.2014.257418. URL: http://dx.doi.org/10.4108/icst.mobihealth.2014.257418.

[20] Cameron McCarthy et al. “Validation of the Empatica E4 Wristband”. In: *2016 IEEE EMBS International Student Conference (ISC)*. Ottawa, ON, Canada: IEEE, May 2016, pp. 1–4. ISBN: 978-1-5090-0935-0. DOI: 10.1109/EMBSISC.2016.7508621. (Visited on 07/09/2023).

[21] Angela A. T. Schuurmans et al. “Validity of the Empatica E4 Wristband to Measure Heart Rate Variability (HRV) Parameters: A Comparison to Electrocardiography (ECG)”. In: *Journal of Medical Systems* 44.11 (Nov. 2020), p. 190. ISSN: 0148-5598, 1573-689X. DOI: 10.1007/s10916-020-01648-w. (Visited on 07/09/2023).

[22] Luca Menghini et al. “Stressing the Accuracy: Wrist-worn Wearable Sensor Validation over Different Conditions”. In: *Psychophysiology* 56.11 (Nov. 2019). ISSN: 0048-5772, 1469-8986. DOI: 10.1111/psyp.13441. (Visited on 07/09/2023).

[23] Michael E Dawson, Anne M Schell, and Diane L Filion. “The electrodermal system”. In: *Handbook of psychophysiology* 2 (2007), pp. 200–223.

[24] Laurie Kelly McCorry. “Physiology of the Autonomic Nervous System”. In: *American Journal of Pharmaceutical Education* 71.4 (Sept. 2007), p. 78. ISSN: 0002-9459, 1553-6467. DOI: 10.5688/aj710478. (Visited on 07/09/2023).

[25] Giorgos Giannakakis et al. “Review on Psychological Stress Detection Using Biosignals”. In: *IEEE Transactions on Affective Computing* 13.1 (Jan. 2022), pp. 440–460. ISSN: 1949-3045, 2371-9850. DOI: 10.1109/TAFFC.2019.2927337. (Visited on 07/09/2023).

[26] Rubén Usamentiaga et al. "Infrared Thermography for Temperature Measurement and Non-Destructive Testing". In: *Sensors* 14.7 (July 2014), pp. 12305–12348. ISSN: 1424-8220. DOI: 10.3390/s140712305. (Visited on 07/09/2023).

[27] E R Nadel, R W Bullard, and J A Stolwijk. "Importance of Skin Temperature in the Regulation of Sweating." In: *Journal of Applied Physiology* 31.1 (July 1971), pp. 80–87. ISSN: 8750-7587, 1522-1601. DOI: 10.1152/jappl.1971.31.1.80. (Visited on 07/09/2023).

[28] William Bierman. "THE TEMPERATURE OF THE SKIN SURFACE". In: *Journal of the American Medical Association* 106.14 (Apr. 1936), p. 1158. ISSN: 0002-9955. DOI: 10.1001/jama.1936.02770140020007. (Visited on 07/09/2023).

[29] Russell Li and Zhandong Liu. "Stress Detection Using Deep Neural Networks". In: *BMC Medical Informatics and Decision Making* 20.S11 (Dec. 2020), p. 285. ISSN: 1472-6947. DOI: 10.1186/s12911-020-01299-4. (Visited on 07/14/2023).

[30] Seyedmahdad Mirsamadi, Emad Barsoum, and Cha Zhang. "Automatic Speech Emotion Recognition Using Recurrent Neural Networks with Local Attention". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA: IEEE, Mar. 2017, pp. 2227–2231. ISBN: 978-1-5090-4117-6. DOI: 10.1109/ICASSP.2017.7952552. (Visited on 07/14/2023).

[31] Panagiotis Tzirakis et al. "End-to-End Multimodal Emotion Recognition Using Deep Neural Networks". In: *IEEE Journal of Selected Topics in Signal Processing* 11.8 (Dec. 2017), pp. 1301–1309. ISSN: 1932-4553, 1941-0484. DOI: 10.1109/JSTSP.2017.2764438. (Visited on 07/14/2023).

[32] Alberto De Santos Sierra et al. "A Stress-Detection System Based on Physiological Signals and Fuzzy Logic". In: *IEEE Transactions on Industrial Electronics* 58.10 (Oct. 2011), pp. 4857–4865. ISSN: 0278-0046, 1557-9948. DOI: 10.1109/TIE.2010.2103538. (Visited on 07/14/2023).

[33] J.A. Healey and R.W. Picard. "Detecting Stress During Real-World Driving Tasks Using Physiological Sensors". In: *IEEE Transactions on Intelligent Transportation Systems* 6.2 (June 2005), pp. 156–166. ISSN: 1524-9050. DOI: 10.1109/TITS.2005.848368. (Visited on 07/09/2023).

[34] Pekka Siirtola. "Continuous Stress Detection Using the Sensors of Commercial Smartwatch". In: *UbiComp/ISWC 2019- - Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers* (2019), pp. 1198–1201. DOI: 10.1145/3341162.3344831.

[35] Sana Imtiaz et al. "Synthetic and private smart health care data generation using GANs". In: *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE. 2021, pp. 1–7.

[36] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[37] Ian J. Goodfellow et al. "Generative Adversarial Networks". In: (2014). DOI: 10.48550/ARXIV.1406.2661. (Visited on 07/12/2023).

[38] Divya Saxena and Jiannong Cao. "Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions". In: *ACM Computing Surveys* 54.3 (Apr. 2022), pp. 1–42. ISSN: 0360-0300, 1557-7341. DOI: `10.1145/3446374`. (Visited on 06/27/2023).

[39] Ian Goodfellow. *NIPS 2016 Tutorial: Generative Adversarial Networks.* Apr. 2017. arXiv: `1701.00160 [cs]`. (Visited on 07/11/2023).

[40] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets.* 2014. arXiv: `1411.1784 [cs.LG]`.

[41] Antonia Creswell et al. "Generative Adversarial Networks: An Overview". In: *IEEE Signal Processing Magazine* 35.1 (Jan. 2018), pp. 53–65. ISSN: 1053-5888. DOI: `10.1109/MSP.2017.2765202`. (Visited on 07/12/2023).

[42] Eoin Brophy et al. "Generative adversarial networks in time series: A systematic literature review". In: *ACM Computing Surveys* 55.10 (2023), pp. 1–31.

[43] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667, 1530-888X. DOI: `10.1162/neco.1997.9.8.1735`. (Visited on 07/13/2023).

[44] Sepp Hochreiter. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions". In: *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* 6 (1998), pp. 107–116.

[45] Eoin Brophy et al. "Generative adversarial networks in time series: A survey and taxonomy". In: *arXiv preprint arXiv:2107.11098* (2021).

[46] Cynthia Dwork. "Differential privacy". In: *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33.* Springer. 2006, pp. 1–12.

[47] Abhradeep Guha Thakurta, San Jose, and Andrew H Vyrros. "LEARNING NEW WORDS". In: ().

[48] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. "RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security.* Scottsdale Arizona USA: ACM, Nov. 2014, pp. 1054–1067. ISBN: 978-1-4503-2957-6. DOI: `10.1145/2660267.2660348`. (Visited on 07/17/2023).

[49] John M Abowd et al. "The Modernization of Statistical Disclosure Limitation at the U.S. Census Bureau". In: ().

[50] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. *DP-CGAN: Differentially Private Synthetic Data and Label Generation.* Jan. 2020. arXiv: `2001.09700 [cs, stat]`. (Visited on 07/17/2023).

[51] Martin Abadi et al. "Deep learning with differential privacy". In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security.* 2016, pp. 308–318.

[52] Cynthia Dwork, Aaron Roth, et al. "The algorithmic foundations of differential privacy". In: *Foundations and Trends® in Theoretical Computer Science* 9.3–4 (2014), pp. 211–407.

[53] Theodora Kokosi and Katie Harron. “Synthetic data in medical research”. In: *BMJ Medicine* 1.1 (2022). DOI: `10.1136/bmjmed-2022-000167`. eprint: `https://bmjmedicine.bmj.com/content/1/1/e000167.full.pdf`. URL: `https://bmjmedicine.bmj.com/content/1/1/e000167`.

[54] Changhee Han et al. “GAN-based synthetic brain MR image generation”. In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE. Apr. 2018. DOI: `10.1109/isbi.2018.8363678`. URL: `http://dx.doi.org/10.1109/ISBI.2018.8363678`.

[55] Maayan Frid-Adar et al. “GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification”. In: *Neurocomputing* 321 (Dec. 2018), pp. 321–331. DOI: `10.1016/j.neucom.2018.09.013`. URL: `http://dx.doi.org/10.1016/j.neucom.2018.09.013`.

[56] Youssef Skandarani, Pierre-Marc Jodoin, and Alain Lalande. “GANs for Medical Image Synthesis: An Empirical Study”. In: *Journal of Imaging* 9.3 (Mar. 2023), p. 69. DOI: `10.3390/jimaging9030069`. URL: `http://dx.doi.org/10.3390/jimaging9030069`.

[57] Zahra Azizi et al. “Can synthetic data be a proxy for real clinical trial data? A validation study”. In: *BMJ Open* 11.4 (2021). Ed. by et al. ISSN: 2044-6055. DOI: `10.1136/bmjopen-2020-043497`. eprint: `https://bmjopen.bmj.com/content/11/4/e043497.full.pdf`. URL: `https://bmjopen.bmj.com/content/11/4/e043497`.

[58] Aldren Gonzales, Guruprabha Guruswamy, and Scott R Smith. “Synthetic data in health care: a narrative review”. In: *PLOS Digital Health* 2.1 (2023), e0000082.

[59] Marc MBV Rouppe Van der Voort, Frits GG van Merode, and Bart HJJM Berden. “Making sense of delays in outpatient specialty care: A system perspective”. In: *Health policy* 97.1 (2010), pp. 44–52.

[60] Che Ngufor et al. “Mixed effect machine learning: A framework for predicting longitudinal change in hemoglobin A1c”. In: *Journal of biomedical informatics* 89 (2019), pp. 56–67.

[61] Wayne TA Enanoria et al. “The effect of contact investigations and public health interventions in the control and prevention of measles transmission: A simulation study”. In: *PloS one* 11.12 (2016), e0167160.

[62] Ted Laderas et al. “Teaching data science fundamentals through realistic synthetic clinical cardiovascular data”. In: *bioRxiv* (2017), p. 232611.

[63] Katie Harron et al. “Linking data for mothers and babies in de-identified electronic health data”. In: *PloS one* 11.10 (2016), e0164667.

[64] John T. Guibas, Tejpal S. Virdi, and Peter S. Li. “Synthetic Medical Images from Dual Generative Adversarial Networks”. In: *CoRR* abs/1709.01872 (2017). arXiv: `1709.01872`. URL: `http://arxiv.org/abs/1709.01872`.

[65] Tero Karras, Samuli Laine, and Timo Aila. “A Style-Based Generator Architecture for Generative Adversarial Networks”. In: *CoRR* abs/1812.04948 (2018). arXiv: `1812.04948`. URL: `http://arxiv.org/abs/1812.04948`.

[66] Daniel Jiwoong Im et al. “Generating images with recurrent adversarial networks”. In: *CoRR* abs/1602.05110 (2016). arXiv: `1602.05110`. URL: `http://arxiv.org/abs/1602.05110`.

[67] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. “Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs”. In: (2017). arXiv: `1706.02633 [stat.ML]`.

[68] Olof Mogren. “C-RNN-GAN: Continuous recurrent neural networks with adversarial training”. In: *arXiv preprint arXiv:1611.09904* (2016).

[69] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. “Time-series generative adversarial networks”. In: *Advances in neural information processing systems* 32 (2019).

[70] Maximilian Ehrhart et al. “A Conditional GAN for Generating Time Series Data for Stress Detection in Wearable Physiological Sensor Data”. In: *Sensors* 22.16 (2022), p. 5969.

[71] Zinan Lin et al. “Using gans for sharing networked time series data: Challenges, initial promise, and open questions”. In: *Proceedings of the ACM Internet Measurement Conference*. 2020, pp. 464–483.

[72] Jerome P Reiter. “New approaches to data dissemination: a glimpse into the future (?)” In: *Chance* 17.3 (2004), pp. 11–15.

[73] Noseong Park et al. “Data synthesis based on generative adversarial networks”. In: *arXiv preprint arXiv:1806.03384* (2018).

[74] Ziqi Zhang, Chao Yan, and Bradley A. Malin. “Membership Inference Attacks against Synthetic Health Data”. In: *Journal of Biomedical Informatics* 125 (Jan. 2022), p. 103977. ISSN: 15320464. DOI: `10.1016/j.jbi.2021.103977`. (Visited on 07/23/2023).

[75] Kin Sum Liu et al. “Performing Co-membership Attacks Against Deep Generative Models”. In: *2019 IEEE International Conference on Data Mining (ICDM)*. Beijing, China: IEEE, Nov. 2019, pp. 459–467. ISBN: 978-1-72814-604-1. DOI: `10.1109/ICDM.2019.00056`. (Visited on 07/23/2023).

[76] Dingfan Chen et al. “GAN-Leaks: A Taxonomy of Membership Inference Attacks against Generative Models”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. Oct. 2020, pp. 343–362. DOI: `10.1145/3372297.3417238`. eprint: `1909.03935` (cs). (Visited on 07/23/2023).

[77] Jamie Hayes et al. “LOGAN: Membership Inference Attacks Against Generative Models”. In: *Proceedings on Privacy Enhancing Technologies* 2019.1 (Jan. 2019), pp. 133–152. ISSN: 2299-0984. DOI: `10.2478/popets-2019-0008`. (Visited on 07/23/2023).

[78] “The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks”. In: ().

[79] Michael Backes et al. “Membership Privacy in MicroRNA-Based Studies”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 319–330. ISBN: 9781450341394. DOI: `10.1145/2976749.2978355`. URL: `https://doi.org/10.1145/2976749.2978355`.

[80] Maciej Dzieżyc et al. “Can we ditch feature engineering? end-to-end deep learning for affect recognition from physiological sensor data”. In: *Sensors* 20.22 (2020), p. 6535.

[81] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115 (2015), pp. 211–252.

[82] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: ().

[83] Yi Liu et al. “PPGAN: Privacy-preserving generative adversarial network”. In: *2019 IEEE 25Th international conference on parallel and distributed systems (ICPADS)*. IEEE. 2019, pp. 985–989.

[84] James Jordon et al. “Hide-and-Seek Privacy Challenge: Synthetic Data Generation vs. Patient Re-identification”. In: *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*. Ed. by Hugo Jair Escalante and Katja Hofmann. Vol. 133. Proceedings of Machine Learning Research. PMLR, June 2021, pp. 206–215. URL: `https://proceedings.mlr.press/v133/jordon21a.html`.

[85] Ninareh Mehrabi et al. “A survey on bias and fairness in machine learning”. In: *ACM computing surveys (CSUR)* 54.6 (2021), pp. 1–35.

[86] Ivan Krasin et al. “Openimages: A public dataset for large-scale multi-label and multi-class image classification”. In: *Dataset available from https://github. com/openimages* 2.3 (2017), p. 18.

[87] Ninareh Mehrabi et al. “Lawyers are dishonest? quantifying representational harms in commonsense knowledge resources”. In: *arXiv preprint arXiv:2103.11320* (2021).

[88] Depeng Xu et al. “Fairgan: Fairness-aware generative adversarial networks”. In: *2018 IEEE International Conference on Big Data (Big Data)*. IEEE. 2018, pp. 570–575.

Declaration of Authorship

Ich versichere, dass ich die vorliegende Arbeit mit dem Thema:

„Privacy-Preserving Smartwatch Health Data Generation For Stress Detection Using GANs“

selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann.

Leipzig, den 26.07.2023

[illegible]

Nils Wenzlitschke

Nils Wenzlitschke
3755701

Appendix

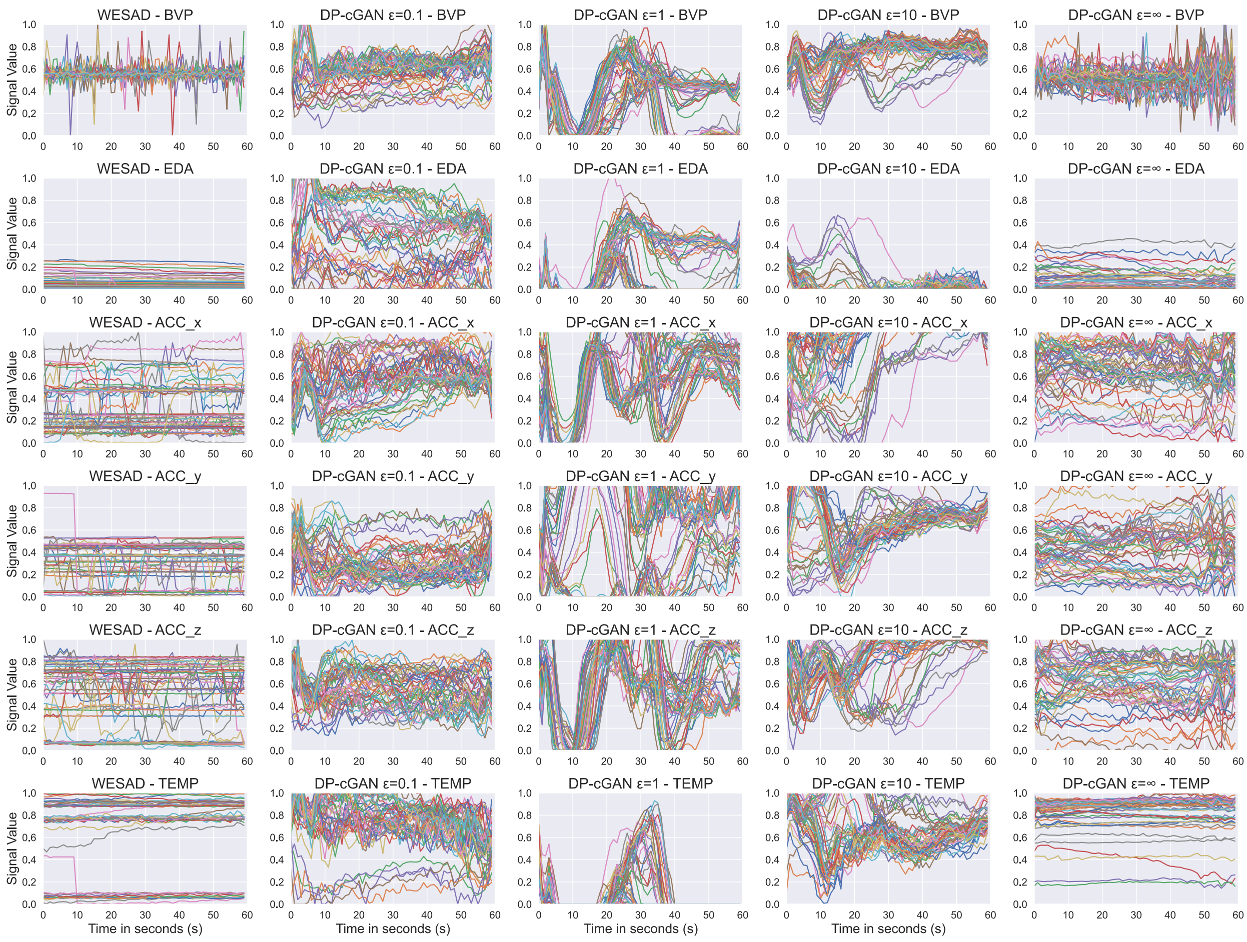


Figure 1.: Shown are 50 *non-stress* sequences from the original WESAD dataset and synthetic sample sequences generated by the different DP-cGAN models. These samples are plotted per row for each signal type: BVP, EDA, ACC_x, ACC_y, ACC_z, and TEMP. The columns show the corresponding dataset. This overview allows comparing the influence of privacy guarantees to the generation of samples.

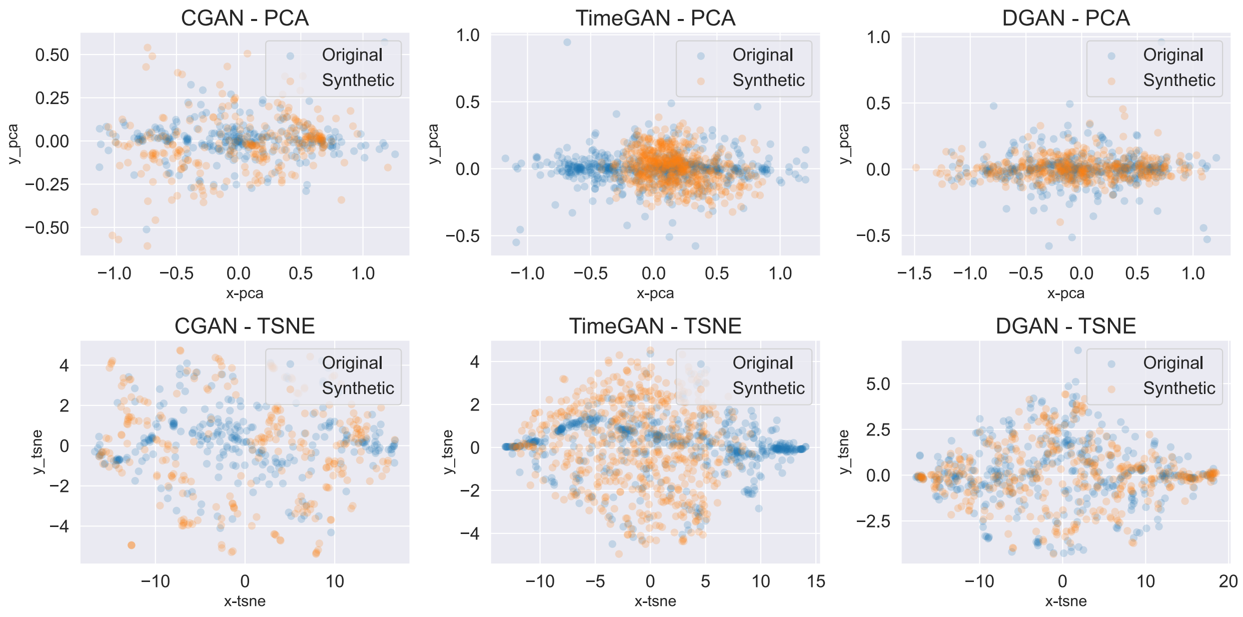


Figure 2.: Non-Stress: The figure shows the PCA and t-SNE analysis to compare the synthetic *non-stress* dataset with the original *non-stress* WESAD dataset. The four different GAN models are shown per column: cGAN, TimeGAN, and DGAN. The plots show the relative distribution of the synthetic GAN datasets (orange) compared to the original dataset (blue) in the two-dimensional feature space to see how well the GAN models mimic the original distribution.

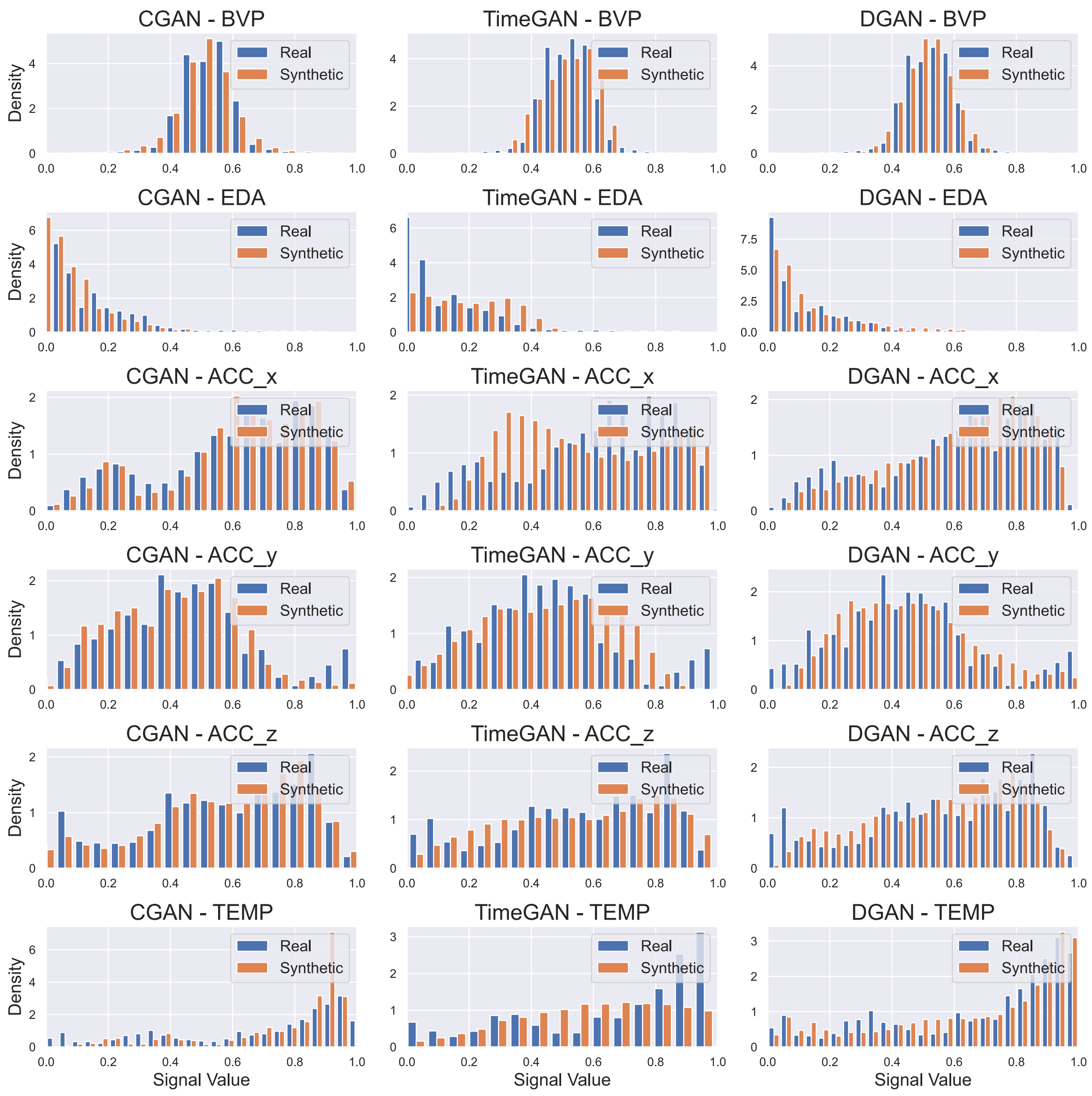


Figure 3.: Shown are the distributions of the *non-stress* signals from the original WESAD dataset and the synthetic signals generated by the different GAN architectures. For each signal type, the real distribution from the WESAD dataset, shown in blue, is compared to the synthetic signal, shown in orange. The density of the normalized signals is plotted in the range [0,1].