BALANCING PRIVACY AND ACCURACY IN IOT USING DOMAIN-SPECIFIC FEATURES FOR TIME SERIES CLASSIFICATION

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Pranshul Lakhanpal

June 2023

COMMITTEE MEMBERSHIP

TITLE: Balancing Privacy and Accuracy in IoT using Domain-Specific Features for Time Series Classification

AUTHOR: Pranshul Lakhanpal

DATE SUBMITTED: June 2023

COMMITTEE CHAIR: Sumona Mukhopadhyay, Ph.D.
Assistant Professor of Computer Science

COMMITTEE MEMBER: Franz Kurfess, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Joydeep Mukherjee, Ph.D.
Assistant Professor of Computer Science

ABSTRACT

Balancing Privacy and Accuracy in IoT using Domain-Specific Features for Time Series Classification

Pranshul Lakhanpal

ε-Differential Privacy (DP) has been popularly used for anonymizing data to protect sensitive information and for machine learning (ML) tasks. However, there is a trade-off in balancing privacy and achieving ML accuracy since ε-DP reduces the model's accuracy for classification tasks. Moreover, not many studies have applied DP to time series from sensors and Internet-of-Things (IoT) devices. In this work, we try to achieve the accuracy of ML models trained with ε-DP data to be as close to the ML models trained with non-anonymized data for two different physiological time series. We propose to transform time series into domain-specific 2D (image) representations such as scalograms, recurrence plots (RP), and their joint representation as inputs for training classifiers. The advantages of using these image representations render our proposed approach secure by preventing data leaks since these image transformations are irreversible. These images allow us to apply state-of-the-art image classifiers to obtain accuracy comparable to classifiers trained on non-anonymized data by exploiting the additional information such as textured patterns from these images. In order to achieve classifier performance with anonymized data close to non-anonymized data, it is important to identify the value of ε and the input feature. Experimental results demonstrate that the performance of the ML models with scalograms and RP was comparable to ML models trained on their non-anonymized versions. Motivated by the promising results, an end-to-end IoT ML edge-cloud architecture capable of detecting input drifts is designed that employs our technique to train ML models on ε-DP physiological data. Our classification approach ensures the privacy of indi-

viduals while processing and analyzing the data at the edge securely and efficiently.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my thesis advisor, Dr. Sumona Mukhopadhyay, for her invaluable insight, guidance, and knowledge throughout the research process. Her expertise and support have been instrumental in shaping this thesis and overcoming challenges along the way.

I am also grateful to the members of my thesis committee, for their valuable feedback and suggestions that greatly enhanced the quality of this work.

I would like to extend a heartfelt appreciation to my parents for their unwavering support and the opportunities they have provided me. Their encouragement and belief in my abilities have been a constant source of motivation.

I would also like to thank my colleague, Asmita, for collaborating with me while writing our first conference paper. Her assistance and contribution have been truly invaluable, and I am grateful for her support throughout this journey.

I would like to thank Andrew Guenther, for uploading this template on Git Hub.

Lastly, I would like to thank all my friends and well-wishers who have offered their encouragement and understanding during this undertaking.

Without the collective support and guidance of these individuals, this thesis would not have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Background and Motivation

Recently, there has been an increase in the use of physiological time series data collected from IoT devices for machine learning (ML) classification tasks [13]. However, it is well-known that using IoT devices enabled by the cloud to collect and process data, and then training ML models is susceptible to privacy breaches. Protecting privacy is critical in healthcare that involves physiological data but there has been little progress in meeting this demand [72, 6]. ε-Differential Privacy (DP) is a mathematical framework that ensures privacy protection by introducing noise into data [31]. ε is an important parameter that quantifies the degree of privacy protection offered by a differentially private method. It determines the trade-off between data utility and privacy. A stronger privacy guarantee is associated with a smaller ε value which limits the amount of information about any particular data that can be deduced. On the other hand, a greater ε value permits a significantly higher level of privacy loss, enabling more precise data analysis or outcomes.

A key advantage of using ε-DP is its resilience against linkage attacks that occur on anonymized data [63]. However, adding too much noise may deteriorate the quality of the ε-DP anonymized data which limits the usability of the anonymized data and leads to poor generalization and performance of ML models. Most of the existing ML techniques trained using ε-DP exhibit lower accuracy than non-DP [15, 29, 75]. Therefore, there is a need for privacy-preserving techniques in healthcare that guarantee privacy to patients without compromising the performance of the classifier. To

this end, this thesis suggests a differential privacy IoT edge-cloud architecture that uses state-of-the-art image classifiers trained on domain-specific features that ensures privacy. **The objective of this research is to develop a technique to improve the performance of ML classifiers for time series that has been anonymized with ε-DP**. Physiological time series (signals) such as Electroencephalogram (EEG) recordings are measured in the time domain [71] using non-invasive methods which are not high-resolution recordings and tend to pick up a lot of noise. Since these measurements are not very useful in their raw form, the data is often processed using the appropriate signal-processing technique to obtain signals with useful information. Since physiological time series are generated from dynamical systems, this work proposes to transform the physiological time series into dynamical features that are represented by recurrence plots (RP) and scalogram images. The benefit of using such image representations are (a) exploit the merit of state-of-the-art image classifiers that extract patterns from the textures of these images (b) these domain-specific images capture the dynamical properties of the time series data that are inherently generated from dynamical systems and (c) these image representations are irreversible transformations which prevent data leakage by delinking the anonymized data to original time series data. Results from comparative classification performance with DP anonymized time series vs anonymized image representation of time series show that the image representations as the input data features for classification worked best with data anonymization under small values of ε. The suggested IoT-ML architecture reduces the amount of processing and analysis required by using edge devices, such as smartphones or wearable technology.

1.2 Research Objective and Questions

This thesis focuses on finding a balance between data anonymization and data utility for improving the classification of physiological time series data with ε-DP and proposing a modular edge-cloud IoT architecture that can be used for collecting, cleaning, processing, anonymizing, training, and deploying ML models for classification of time series. Based on the literature, the following are the existing limitations that this thesis is trying to address.

- In our review, we weren't able to find any studies exploring the effect of anonymizing different feature types on the performance of ML classifiers with ε-DP data.
- Past research does not explore the possibility of a universal value of ε that can be used for anonymizing data in a way that provides an adequate privacy guarantee and performs at par with non-anonymized data in ML tasks.
- There are no robust techniques for time series classification with ε-DP time series such that the ML classifier's performance is close to the classification with non-anonymized ε-DP time series.

The goal of this research is to answer the following research questions (RQ):

1. **RQ 1: Is it possible to achieve ML classification performance with DP anonymized time series data close to non-anonymized time series?**
2. **RQ 2: Does there exist an ε value for time series data that can guarantee classifier accuracy close to non-anonymized time series and the image representation of time series?**

3. **RQ 3: Is it possible to design robust ML software architecture for sequential time series data from IoT devices that can ensure flexibility, extensibility, and scalability?**

The RQs are described in detail below:

RQ 1: You can balance the privacy and usability levels of your data using a positive value named ε (epsilon) when using ε-Differential Privacy (ε-DP) [12]. This work will use Laplace Noise to add noise to our dataset to make it $\varepsilon-$Differentialy Private.

The noise added to our dataset will have a Laplace distribution centered around 0 with a sensitivity of 1. The noise will be scaled according to the value of $1/\varepsilon$.

The value of ε ranges from 0 to infinity. So, if ε is small, more privacy is preserved, but data accuracy worsens. If ε is large, privacy will be worse but data accuracy will be preserved [12].

This study used different features to train the ML models, such as Raw Time Series, scalograms, recurrence plots, and Multi-modal Data (combining scalograms and recurrence plots), and compare the performance of these models under different values of ε and explore if our ML models trained with anonymized data can achieve accuracies as close to ML models trained with non-anonymized data.

RQ 2: Since DP affects different datasets in diverse ways [58], this thesis will explore possible values of ε that are relatively low and the noisy data can still provide the same level of performance when compared to data without any noise when training machine learning models.

RQ 3: ML is not only focused on various intricate models, but it also includes multiple stages that contribute to the overall performance of the model [48]. All

these steps are necessary for the ML pipeline to accept raw data and then transform it into something meaning full from which information can be inferred. The data is collected, stored, cleaned, and pre-processed to isolate important features from the data.

In scenarios where limited bandwidth is available and data size far surpasses the capabilities of resources available to run the ML pipeline smoothly, organizations opt for fog computing and edge-cloud computing to distribute the workload across the cloud [50]. This thesis aims to propose a flexible and scalable IoT-ML architecture where IoT devices could easily be integrated and every part of the ML pipeline can be divided into individual components that can be easily switched with different components depending on the IoT device or the data.

1.3 Thesis Contribution

The aim of this work is to develop a robust classifier for ε DP time series such that the classifier yields performance close to non-DP data. To address these research gaps, this thesis explores the possibility of suggesting a universal value or a range of ε values that can be used for applying DP on physiological datasets and still perform comparably to non-DP data with ML models. Since DP affects different datasets in diverse ways [58], this thesis also explores the possibility of finding a feature representation of time series data that performs similarly with and without DP applied to it. This study addresses these problems from a new perspective by transforming physiological time series into domain-specific image representations for classification. This study transforms the time series data into scalograms [23], recurrence plots (RP), and a third feature that stacks both scalograms and RP together to create a joint representation [7].

This work also developed a DP-enabled ML-IoT architecture that can apply DP and evaluate our trained ML models. Our IoT architecture is designed to work with both raw time series and image data. Our focus is on finding a balance between time series data anonymization and accuracy in training ML models on physiological time series data that is generated from IoT devices.

- **RQ 1: Is it possible to achieve ML classification performance with DP anonymized time series data close to non-anonymized time series data?**

  This study proposes to transform physiological time series into domain-specific features represented by RP and scalogram images. This allowed us to use state-of-the-art image classifiers to extract patterns from the textures of these images. Results from comparative classification performance with DP anonymized time series vs. non-anonymized image representation of time series show that the image representations as the input data features worked best with data anonymization for classification under small values of ε.

- **RQ 2: Does there exist an ε value for time series data that can guarantee classifier accuracy close to non-anonymized time series and the image representation of time series?**

  This thesis experimented with different values of ε and used them to anonymize the time series data and its image representation using Laplace noise. Results demonstrate that ε values in the range 0.25 - 0.30 generated ML performance close to that of the non-anonymized data.

- **RQ 3: Is it possible to design a robust ML software architecture for sequential time series data from IoT devices that can ensure flexibility, extensibility, and scalability?**

This work produced a robust DP-ML-IoT architecture that fully utilizes the potential of its edge and cloud components. This architecture is an extension of the architecture proposed by [40]. This architecture is designed to accommodate sequential data such as time series and image data whereas the architecture in [40] was developed for structured data. In the design proposed in this work, the edge is used for data generation, cleaning, preprocessing, feature engineering, and data anonymization of raw time series data. Another major change this work introduced in the architecture was the evaluation of the model on the cloud which saved time and computation power on the edge device. The anonymized data is stored in a cloud storage solution instead of the edge because of resource limitations. The cloud storage is also constantly monitored for data drifts and if detected the ML model is automatically updated and the latest model is sent to the edge. This architecture is designed in a way to ensure that each module is independent of the others which allows future users to modify each component to make it compatible with their desired data. Only the best and fully vetted model is then sent to the edge where it is used to make inferences.

1.4 Thesis Organization

This research work is structured as follows. Chapter 2 presents the background and research related to the field. Chapter 3 presents the framework of the architecture and methodology. Chapter 4 explains the methodology behind training the models and establishing the baseline performance of each feature type. Chapter 5 presents the methodology of how noise is added to make the data ε-Differentially Private and compares the performance of models of different features with different ε values, and Chapter 6 discuss the results of this work and answers the research question asked in this thesis. Finally, Chapter 7 presents a conclusion of the work done in this thesis

and presents the next steps to improve the architecture based on the current research trends.

Chapter 2

BACKGROUND AND RELATED WORK

2.1 Background

2.1.1 EEG and Wearable Headbands

Electroencephalogram, or EEG is a method to measure a person's brainwaves [19]. Used in psychophysiological research, it is a neuroimaging style of monitoring and recording the electrical activity of the brain in the form of signals measured in microvolts [22]. Raw EEG signals contain five major brain waves distinguishable by their frequency: Delta δ with a frequency between 0.5 Hz - 4 Hz (state of meditation, deep sleep or coma), Theta θ with a frequency between 4 Hz - 8 Hz (sleep or daydreaming), Alpha α with the frequency between 8 Hz - 13 Hz (calm and deep relaxation), Beta β with the frequency between 13 Hz - 30 Hz (conscious, logical-analytical thinking), and Gamma γ with a frequency between greater than 30 Hz (higher processing tasks, cognitive functioning) [53, 1]. Figure 2.1 shows the different waves present in EEG separated on the basis of their frequencies.

These brainwaves can be recorded by two broadly used methodologies, invasive and non-invasive. Invasive recordings are obtained by intracranially implanting the electrodes, whereas non-invasive EEG recordings are obtained by the electrodes attached to the scalp surface [16]. Since implanted electrodes are much closer to the brain than scalp electrodes, they record brain signals with higher amplitudes and spatial resolution than scalp EEG [16].

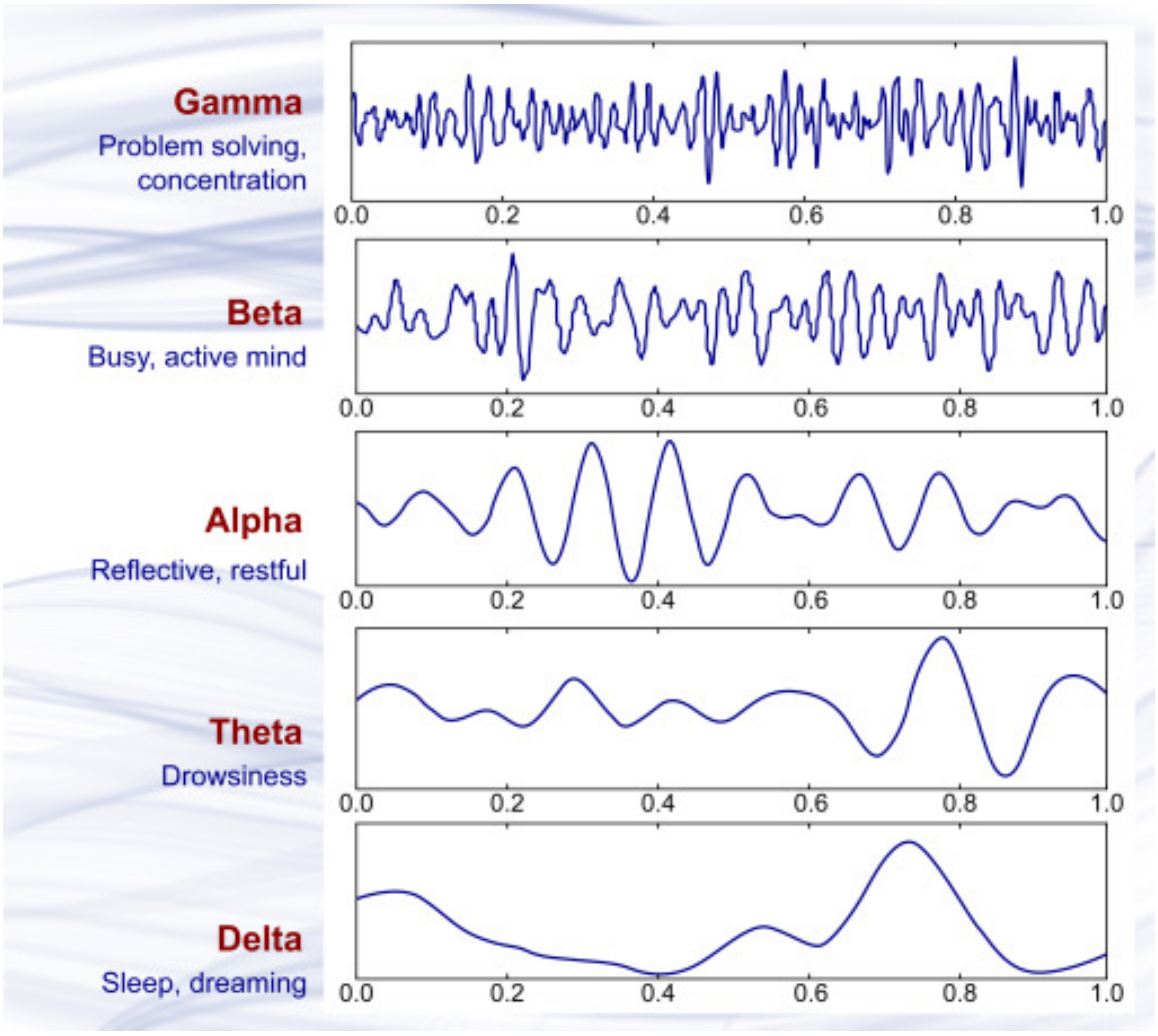

Figure 2.1: Brain wave samples with dominant frequencies belonging to different frequency bands [5]

Generally, it is observed that portable EEG headbands utilize non-invasive method to record brain waves, and these wearable EEG devices have a lower number of electrodes or channels [22] in comparison to medical-grade EEG which have 64 or more electrodes or channels [60]. Consumer-grade EEG headsets like NeuroSky, Muse, Emotiv, and OpenBCI have 1 channel, 4 channels, + 5 or 14 channels, and 8 – 16 channels respectively.

2.1.2 Feature Engineering: Transforming Time Series to Scalogram and Recurrence Plot Images

2.1.2.1 Time Frequency Analysis

Time-Frequency Analysis (TFA) is a method used in digital signal processing to evaluate a signal in the time domain, for us EEG signal, in both the time and frequency domains. TFA is the representation of the power intensity of a particular frequency at a particular point of time in a signal [22]. One can also detect the presence of different bands of frequency of the EEG signal, i.e. Delta, Gamma, Alpha, Beta, and Theta, by applying TFA on the signal.

There are various techniques to perform TFA on a signal such as short-time Fourier transform, wavelet transform, quadratic time-frequency transforms, advanced wavelet transforms, and adaptive time-frequency transforms [64]. This study used the Morlet Wavelet Transform to get the time-frequency representation of the EEG signals. There are numerous methods for analyzing data with respect to time, including autocorrelation, crosscorrelation, stationarity, seasonal adjustment/decomposition, singular spectrum analysis, and count series analysis. Similarly, there exist many frequency domain analyses to analyze data with respect to the frequency domain, such as Fourier analysis, wavelet analysis, and Laplace transform analysis, assuming the time series is stationary. Time-frequency analysis (TFA), on the other hand, simultaneously analyzes both time and frequency and is especially useful for non-stationary time series with time-varying trends, irregular cycles, time-varying periodic cycles, and other time-dependent characteristics.

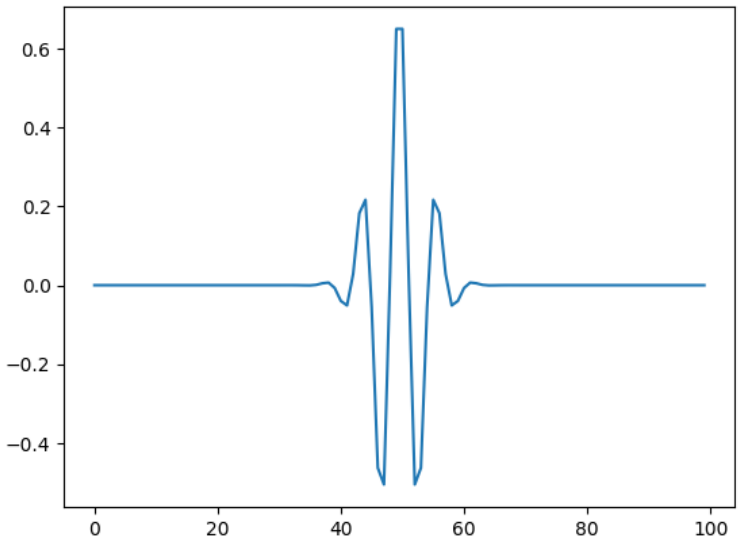

Figure 2.2: Morlet's wavelet

2.1.2.2 Morlet Wavelet Transformation

As described by authors Francois Tadel, et al. [74] Complex Morlet wavelets are very popular in EEG/MEG data analysis for time-frequency decomposition.

They have the shape of a sinusoid, weighted by a Gaussian kernel, and they can therefore capture local oscillatory components in the time series. An example of this wavelet is shown in Figure 2.2.

Contrary to the standard short-time Fourier transform, wavelets have a variable resolution in time and frequency. For low frequencies, the frequency resolution is high but the time resolution is low. It is the opposite for high frequencies. Therefore wavelets are able to provide both frequency and temporal precision [74].

2.1.2.3 Scalogram

A scalogram is a visual way of representing the signal strength, or "loudness", of a signal over time at various frequencies present in a particular waveform. Not only

can one see whether there is more or less energy at, for example, 2 Hz vs 10 Hz, but one can also see how energy levels vary over time. Scalograms are basically two-dimensional graphs, with a third dimension represented by colors. Figure 2.3 visualizes an example of a wave and its representation as a scalogram.

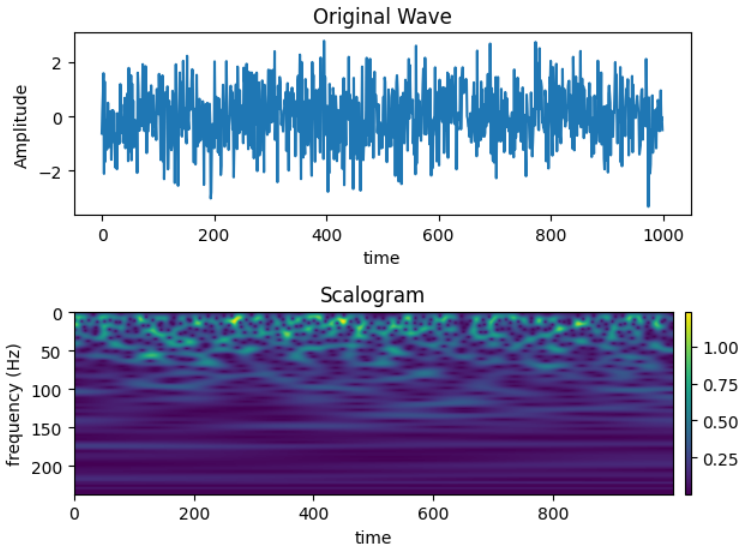

Figure 2.3: Scalogram

Scalograms are created using a technique called the Continuous Wavelet Transform (CWT) that indicates how strong different frequencies of a signal are over time. A Morlet wavelet was used to perform CWT after which Fast Fourier transformation (FFT) was applied to the Morlet wavelet and the signal. The output received after applying FFT is convoluted and an inverse Fourier transform is applied to it which gives a time-domain representation of the data.

2.1.2.4 Recurrence Plot

A recurrence plot is a graphical representation of a recurrence analysis. It is a two-dimensional plot that visualizes the recurrence of states or patterns in a time series of data. In a recurrence plot, the vertical and horizontal axes represent the time points

of the time series, and each point in the plot represents a recurrence of a state or pattern in the time series.

Recurrence plots can provide useful insights into the behavior of a system, such as its periodicity, stability, and complexity. They can be used to identify patterns or states in the EEG data that may be related to specific brain activity, such as seizures [73], sleep stages [76], or cognitive states.

Let us assume the physiological time series $\{x_t\}$ of length T denoted by:

$$\{x_t | t = 1, 2, \ldots, T\} \tag{2.1}$$

Then the recurrence plot between two data points x_i and x_j would be denoted as:

$$R_{i,j} = \Theta\left(\varepsilon - \|x_i - x_j\|\right) \tag{2.2}$$

where the Θ is known as the Heaviside function that compares any two points of the trajectory and is defined as follows:

$$\Theta\left(x\right) = \begin{cases} 1, if x \geq 0, \\ 0, otherwise \end{cases} \tag{2.3}$$

where $\|\cdot\|$ is the norm and ε is the threshold. If the norm is less than ε, it is regarded as a recurrence point and is indicated as a black dot else it is shown as a white dot. ε was set to 0 in the experiments.

Figure 2.4 visualizes an example of a wave and its representation as a recurrence plot.

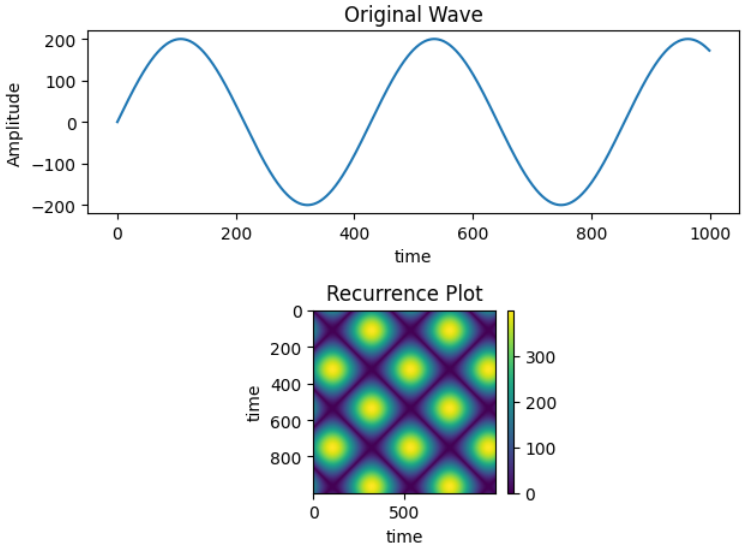

Figure 2.4: Recurrence Plot

2.1.3 Machine Learning

Machine learning (ML) is an application of AI that enables computer systems to learn and improve from experience without being explicitly programmed [3]. These algorithms help the system learn specific tasks training them on some examples or data. ML algorithms can commonly be divided into four categories: supervised, unsupervised, semi-supervised, and reinforcement learning [14]. These algorithms can be used to perform a variety of tasks such as protection, clustering, classification, etc. This thesis has used ML for time-series classification.

2.1.3.1 Supervised

Supervised Learning is a subset of ML that uses labeled data to train algorithms that are used to classify data or predict outcomes accurately. Labeled data means that each input in the training dataset is implied to have a specified output, known as

the class label. In simple terms given a training sample, supervised learning trains a function f with the goal that $f(x)$ predicts the true label on unseen data. Two major applications of supervised learning are to solve classification and regression problems. Classification models/algorithms are mapping functions (f) that map input variables (X) to discrete output variables (y). The output variables are often called labels or categories. Regression models are mapping functions (f) that translate continuous output variable (y) to input variable (X) pair. A real value, such as an integer or floating point value, is the continuous output variable [18].

2.1.3.2 Unsupervised

Unsupervised Learning is a subset of ML that does not require the use of labeled data to train the model. These algorithms analyze and cluster unlabeled datasets. These algorithms identify hidden patterns or data clusters without the assistance of a human. These algorithms are generally used for clustering, dimensionality reduction, and exploratory data analysis.

2.1.3.3 Semi-Supervised

As the name suggests, semi-supervised machine learning is a combination of supervised and unsupervised learning techniques. It uses both unlabeled data and labeled data, combining the advantages of both supervised and unsupervised learning without the difficulties associated with obtaining a lot of labeled data. It really shines in situations where labeled data is difficult to obtain, such as in medical imaging or financial fraud detection.

2.1.4 Deep Learning

Deep Learning is a subfield of machine learning that involves training artificial neural networks with multiple layers to learn complex patterns in data. It is inspired by the structure and function of the human brain, where neurons are interconnected and work together to process information.

2.1.4.1 Convolutional Neural Network

Convolutional Neural Networks (CNNs) [43] are a type of artificial neural network that is widely used in computer vision tasks, such as image and video recognition, object detection, and segmentation. A typical CNN may contain the following layers:

- **Convolutional Layer:** A convolutional layer is a basic component of a CNN that performs convolution by multiplying and summing filter elements with an input image. It takes a 3D tensor input volume and applies filters to produce a set of feature maps. Each filter is a matrix of weights applied to a small region of the input volume, resulting in feature maps that capture local patterns and structures, such as edges or textures.

- **Pooling Layer:** A pooling layer is used in CNNs to reduce the spatial dimensions of feature maps produced by a convolutional layer. It operates independently on each feature map, retaining important information while reducing the size and computational cost. Max pooling is the most common type, using a fixed-size window to take the maximum value within it. The layer helps prevent overfitting and introduces translation invariance. Stacking multiple convolutional and pooling layers allows CNN to learn complex representations.

- **Fully Connected Layer:** A fully connected layer is a type of neural network layer where each neuron receives input from every neuron in the previous layer. Each connection has a weight that is learned during training, and the output of each neuron is computed by applying an activation function to a weighted sum of the inputs. Fully connected layers are commonly used for classification tasks, particularly in convolutional neural networks where they are used to map the learned features to the desired output. They can also be used in other types of neural networks, such as feedforward or recurrent networks.

CNNs have lately demonstrated state-of-the-art performance on challenging tasks with very little or no data feature engineering. Instead, they rely on feature learning on raw data, which is why they are used in this work.

2.1.4.2 LSTM

LSTM stands for Long Short-Term Memory, which is a type of recurrent neural network (RNN) architecture designed to handle sequential data and address the vanishing gradient problem in traditional RNNs [47]. Due to its ability for capturing long-term relationships, handle variable-length sequences, resilience to noisy data, model complicated temporal patterns, and analysis of multivariate time series, LSTM networks are ideally suited for time series data. In addition to processing sequences of various lengths, filtering out the noise, and capturing complex patterns, LSTMs solve the vanishing gradient problem.

The original LSTM described in [47] consists of a memory cell and three multiplicative gating units: the input gate, the forget gate, and the output gate. These gates control the flow of information through the memory cell and allow the model to selectively store, update, and retrieve information over time.

2.1.4.3 Self-Attention

The Transformer model's core element is the self-attention layer, which was first mentioned in the [78]. It enables each element of a sequence, such as time series data, to pay attention to and consider how important other components are. Capturing long-range relationships, enabling parallel computing, managing variable-length sequences, and resilience to positional information are some advantages of the self-attention layer. Models can successfully identify patterns and temporal correlations in time series data by using self-attention, which enhances their performance in tasks like predictions and detecting anomalies.

2.1.5 Gated Recurrent Unit

Gated Recurrent Unit (GRU) is a specific kind of recurrent neural network (RNN) layer [24]. To address the challenge of capturing long-term dependencies in sequential input, the GRU was designed. It uses gating techniques, such as reset and update gates, to regulate the flow of information inside the network.

The GRU layer decides how much new information should be retained using the update gate and lets the model to selectively keep or reject data from the prior hidden state depending on the reset gate. This minimizes the vanishing gradient problem and enables the model to represent long-term relationships with accuracy.

The GRU's unit is capable of capturing long-range relationships, managing variable-length sequences, and demonstrating computing efficiency are advantages for time series data and sequential tasks. GRU layers are an alternative to conventional RNNs and LSTM units in applications such as sequence modeling [25], natural language processing [9], speech recognition [68], and time series analysis [85].

This study uses GRUs with a combination of Residual Convolutional Networks to extract local patterns, capture long-term dependencies, and model temporal dynamics for the time series data.

2.1.6 Differential Privacy

Differential privacy is a mathematical framework that provides a formal notion of privacy for individuals in a dataset while allowing for accurate statistical analysis of the data [45]. The concept of differential privacy was first introduced by Dwork et al. in 2006 as a response to the increasing concern over the privacy risks associated with the growing use of data in various fields, including healthcare [31].

The primary goal of differential privacy is to protect the privacy of individuals whose data is included in a dataset while still allowing for accurate statistical analysis. This is achieved by adding random noise to the data before it is analyzed, which makes it difficult for an attacker to determine the presence or absence of an individual in the dataset [32].

Differential privacy has several important properties that make it an attractive approach for privacy-preserving data analysis. One of the main properties is that it provides a mathematical guarantee of privacy protection, meaning that it is possible to quantify the level of privacy protection provided by a differential privacy mechanism. Additionally, differential privacy is flexible and can be applied to various types of data and analysis methods [32].

To implement differential privacy, various technologies, and tools can be used, including differentially private algorithms, privacy-preserving data management systems, and programming languages and libraries that support differential privacy. Addition-

ally, researchers may use simulations or real-world datasets to evaluate the effectiveness of differential privacy.

2.1.6.1 Differential Privacy Types and Mechanisms

Differential privacy can broadly be classified into the following types:

- **ε-Differential Privacy:** According to [31] ε-DP is a rigorous mathematical formulation of privacy guarantees for statistical inquiries and analysis of data. Any algorithm is considered to be ε-DP if it meets the following conditions:

  For any neighboring datasets D and D', where there is just one element that differs, an algorithm M satisfies ε-DP if $S \subset Range(M)$:

  $$Pr[M(D) \in S] \leq exp(\varepsilon) \times Pr[M(D) \in S] \tag{2.4}$$

  In this equation the probability that the algorithm M outputs a result in subset S on any dataset D is represented by $Pr[M(D) \in S]$. The inequality makes sure that the ratio of these probabilities, where ε is a privacy parameter, is constrained by $exp(\varepsilon)$.

  It guarantees that the presence or absence of an individual data point will not materially affect the result or the information obtained by an attacker and gives a quantitative level of privacy protection.

- **(ε, δ)-Differential Privacy:** (ε, δ)-Dp is an extention of (ε)-DP. Along with the constrains of privacy leakage allowed by ε it relaxes the strict privacy guarantee of ε-DP by allowing a small probability δ $(0 \leq \delta \leq 1)$ of a privacy breach [31].

  Any algorithm is considered to be (ε, δ)-DP if it meets the following conditions:

For any neighboring datasets D and D', where there is just one element that differs, an algorithm M satisfies ε-DP if $S \subset Range(M)$:

$$Pr[M(D) \in S] \leq exp(\varepsilon) \times Pr[M(D) \in S] + \delta \tag{2.5}$$

This means that the output of algorithm M on dataset D and D' are almost as likely to occur with the ratio of probabilities bounded by $exp(\varepsilon)$, except with a small probability δ of a breach in privacy.

- **Concentrated Differential Privacy (CDP):** In [34] the authors introduced an new DP framework called Concentrated Differential Privacy which offers tighter restrictions on privacy leakage as compared to ε-DP. It addresses the problem of cumulative privacy losses when multiple requests are made to a DP method.

  Instead of focusing on the privacy loss of each individual request, CDPs goal is to limit the privacy loss throughout an entire series of requests. By accounting for the composition of numerous queries, it provides better assurances and lessens the effects of sequential composition.

Differential privacy techniques are a core method for protecting privacy while allowing for accurate data analysis in knowledge management. These techniques involve adding noise to data to make it more difficult to identify individual records or extract sensitive information.

The Laplace mechanism is a commonly used differential privacy technique that involves adding noise to the output of a query. This technique adds noise that follows a Laplace distribution with a scale determined by the sensitivity of the query. The Laplace mechanism is often used for answering numeric queries such as mean or sum queries [32].

Another differential privacy technique is the exponential mechanism, which is often used for non-numeric queries. The exponential mechanism involves selecting a query result that minimizes the difference between the true result and a perturbed version of the result. The perturbation is based on the sensitivity of the query and is determined by the exponential distribution [32].

The Gaussian mechanism is another commonly used differential privacy technique that is often used for differentially private machine learning. This technique involves adding Gaussian noise to the data to make it more difficult to identify individual records or extract sensitive information. The scale of the noise is determined by the sensitivity of the query or the model [32].

This study uses ε-DP and adds Laplace noise to perturb the input data before being used to train the machine learning models.

2.1.7 Cloud

Talking in terms of computing, the cloud refers to the network of remote servers that are connected and can be accessed over the internet [2]. A third-party provider, such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP), often owns and runs these servers. Users may remotely store, administer, and process data and applications thanks to cloud infrastructure, which in return eliminates the need for hardware and infrastructure on-site.

Through the use of multiple on-demand services and resources, cloud computing enables its users to expand their computing resources as required. Common cloud services consist of Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [51]

Since cloud services do not require on-premise hardware it provides a lot of benefits as well:

- **Scalability:** Based on user resource requirements cloud services can be easily scaled up or down without significant infrastructure investments.
- **Cost Efficiency:** The users only pays for the resources they actually used and also save the cost of hardware and software purchases and maintenance, as those are taken care of by the cloud provider.
- **Reliability:** Cloud providers boast high availability and redundancy across multiple data centers, ensuring data and applications are accessible even in the event of hardware failures or disasters.

The way companies and people manage and use computing resources has been transformed by cloud computing. As it offers flexibility, scalability, and cost-effectiveness in the deployment and management of IT infrastructure and services, it has established itself as an essential component in a number of sectors.

2.1.8 Data Drift

Data drift refers to the change in the statistical properties of the data over time. Ir happens when the characteristics, distribution, or relationships of the data pivot from the initial or the expected state [66].

It is an important concept in ML and data analysis because these models are generally trained on historical data, assuming that future data will follow the same patterns. However, if the underlying data-generating process changes, the trained model may become less accurate or fail to perform well.

Data drift can be caused due to several reasons such as a shift in the population, a change in the data collection process, a change in end-user behavior, etc [11]. For example, as a population grows older their preferences also keep changing thus it would be hard for a recommendation system to give relevant recommendations if the model was never updated with the change in population preferences

Thus timely detection of data drift is a very important task that can ensure model reliability. Drift detection involves monitoring and comparing the incoming data to the reference or training data. Statistical techniques can be used to assess the presence and significance of data drift.

Detecting data drift is crucial to ensure the reliability and effectiveness of models. It involves monitoring and comparing the incoming data to the reference or training data. Statistical techniques, such as hypothesis testing, control charts, or distribution comparisons, can be used to assess the presence and significance of data drift [66].

There are several statistical methods that can be used to detect data drift, we have decided to use the Kolmogorov-Smirnov test.

2.1.8.1 Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (KS) test is a statistical test used to compare two datasets or to compare a dataset to a theoretical distribution [17]. It helps determine if the two datasets or the dataset and the theoretical distribution come from the same underlying population or if they differ significantly.

The KS test is based on the maximum difference between the cumulative distribution functions (CDFs) of the two datasets being compared [17]. The test calculates a test

statistic, denoted as D, which represents the largest vertical distance between the two CDFs.

We typically follow these steps to perform the KS test:

1. Formulate hypotheses: Define the null hypothesis (H_0) as the assumption that the data at different time points come from the same distribution. The alternative hypothesis (H_a) suggests that there is a significant difference between the distributions, indicating data drift.

2. Split the data: Divide collected data into subsets based on time points or intervals of interest.

3. Calculate Empirical Cumulative Distribution Functions (ECDFs): Compute ECDFs for each data subset, representing proportions of data points less than or equal to a value. It estimates the distribution of the data at each time point

4. Compare ECDFs: Use KS test to compare ECDFs, calculating maximum vertical difference (D) for test statistic

5. Determine critical value: Obtain critical value based on chosen significance level (alpha) from KS test table or statistical software.

6. Compare test statistic to critical value: If D exceeds the critical value, reject H_0 in favor of H_a, indicating significant differences and data drift.

The KS test assumes that every point in the dataset is continuous, which means that there are no gaps or discontinuities in the data values and that the data points in the dataset are independent of each other. Since our dataset comprises multiple subjects the recorded signals for each subject are independent from each other. And signals are continuous by nature so we can use the KS test to detect drift.

2.2 Related Work

2.2.1 Classification of EEG

This study analyzed previous work that use machine learning to solve problems closely related to the dataset being used in this study. In [35] the authors propose using a Deep Neural Network (DNN) with a Wavelet Packet Transform (WPT) to identify deceit information based on brain EEG signals. The proposed approach achieved a 95% classification accuracy and outperformed traditional learning methods.

[8] proposes a new method for analyzing (EEG) signals for eye state prediction, using Self-Organizing Map (SOM) clustering and Deep Belief Network (DBN) approaches. The proposed method was able to significantly improve the accuracy of the EEG prediction and outperformed other classification techniques, such as Linear SVM, Gaussian Naïve Bayes, and Decision Trees.

Chaudhary et. al., [23] propose a deep learning approach to detect intentionally poor performance on a cognitive-motor integration (CMI) task by analyzing EEG data. The proposed model converts the EEG signal to scalogram images and used those images to train their Deep Learning models, they achieved an accuracy of 98.71% in differentiating between maximal effort and intentionally poor task performance. They use deep learning models with CNN, LSTM, and Attention layers.

This study [57] presents a new deep learning method for effectively classifying arrhythmia using 2-second segments of 2D recurrence plot images of ECG signals. The method was tested using publicly available ECG databases, and achieved high testing accuracies in the first and second classification stages, respectively, after five-fold cross-validation. The method was found to be superior to other studies that used 1D ECG data, confirming that converting ECG signals into 2D segments enhances the

accuracy of arrhythmia classification. The study provides an advanced methodology for detecting and discriminating between different arrhythmia types.

This study generates a recurrence plot of the raw time series data and compares the performance of both non-DP and DP recurrence plots.

In [7] the authors propose two efficient multimodal fusion frameworks, called Multimodal Image Fusion (MIF) and Multimodal Feature Fusion (MFF), for ECG heartbeat classification. They convert ECG signals into three types of images using Gramian Angular Field (GAF), Recurrence Plot (RP), and Markov Transition Field (MTF). They combine the input images to create a three-channel input image to be used for their deep-learning models. This is what they call Multimodal Image Fusion. They achieved classification accuracy of 99.7% and 99.2% on arrhythmia and MI classification, respectively which was a huge performance improvement for them when compared to using individual modalities.

This thesis intends to follow their study to create its inputs by combining the different channels of the data to get a multi-channel output that will be used to train its model.

In [70] the authors propose a novel recurrent neural network architecture called ChronoNet for automated analysis of EEG data to identify abnormal or normal brain activity, which is the first step in diagnosing neurological conditions.

This thesis uses ChronoNet for performing classification tasks on its datasets.

2.2.2 DP for Time Series and EEG/Physiological Data

In [58, 82], the authors discovered that training the ML model with DP affects the performance of the ML model while preserving the data's privacy. They also show that the effect of DP varies from one dataset to another. While the overall quality of

the data is reduced, it is still able to convey the information it was intended to [33]. Thus, DP is a promising approach to data privacy.

Debie et al. [29] used EEG data from 9 subjects to perform motor imagery classification performance. In their study, they used Generative Adversarial Networks (GAN) to generate and classify EEG data. They trained two GAN models, one with privacy and the other without privacy. Their model achieves data privacy by limiting the maximum influence of any single user while the model is training and adding noise to the model gradients. The results indicated that the model without privacy worked slightly better than the model with privacy.

Similarly, Ziller et al. [86] demonstrated that complex-valued neural networks could be trained with rigorous privacy and excellent utility. In their experiments, they trained a complex-valued neural network to detect Left Bundle Branch Block (LBBB) using ECG data and achieved almost the same accuracy on models trained with and without DP data. In another experiment, they used the SpeechCommands dataset to train a Convolutional Neural Network (CNN) on complex spectrogram data. Each waveform signal was transformed to a complex-valued 2-D spectrogram and used ζ-DP-SGD to train a complex-valued CNN. In this experiment, they did notice a drop in accuracy score when DP was applied to the data.

In [75] they trained state-of-the-art CNNs on high-quality chest radiographs to predict chest radiographs diagnosis with non-DP deep CNNs and DP models. They found that there was a very slight decrease in performance in the privacy-preserving models while providing high user privacy and model fairness [75].

In [81] the author proposed some ways to secure sensitive EEG data. The paper explores a lack of trusted third-party data centers to store and publish DP data. They then discuss how users can use their DP mechanism to anonymize private data

before releasing them to an untrusted server. This however might degrade the utility of the data thus the user will need to decide between utility and privacy.

In [29, 86, 75] the authors propose a privacy-protecting ML mechanism that applies DP on the gradients or the weights of their ML models however this study proposes a mechanism that applies DP to its input data rather than apply it to the model gradients.

2.2.3 DP for Images

There has been a rise in the popularity of the Differentially Private Stochastic Gradient Descent (DP-SGD) [4] technique that is used to modify the gradients used in stochastic gradient descent. Similar to other uses of DP in ML, DP-SGD's performance also deteriorates as privacy is increased [28]. The authors in [28] proposed an improved version of DP-SGD with careful hyper parameterization to improve model performance in pre-trained models such as Wide-ResNet, NFNet-F3, and ImageNet used for image classification on the CIFAR-10 dataset. Although they contribute significantly towards improving privacy-protected ML models for image classification, there still remains the question of how to protect/anonymize the image data.

In [21] the authors propose a privacy-preserving face recognition framework called Privacy using Eigenface Perturbation (PEEP). PEEP works by adding Laplace noise to the Eigenfaces of a face image before it is shared with a central server. This makes it difficult for an attacker to learn anything about the individual in the image, even if they have access to the entire dataset of Eigenfaces. In their experiments, they showcased that PEEP gave a classification accuracy of around 70% - 90% with an MLPClassifier when $\varepsilon = 4$ on the CelebA dataset.

2.2.4 IoT Architecture for DP and ML

In [84], the authors introduced a classification system for IoT devices that protects privacy in an edge-computing setting. Their technique adds noise to extracted features in the cloud in order to address the privacy issue when ML models are released from the cloud to the edge nodes. According to experimental findings on several datasets, their approach successfully balances utility and privacy protection. This study only uses the cloud for model training and evaluation, and all the data processing such as data cleaning and feature extraction

In [30] the authors propose a fog-assisted healthcare system to maintain the blood glucose level. Their system displayed improved performance in terms of energy efficiency, prediction accuracy, computational complexity, and latency compared to cloud computing healthcare systems.

In [44] the authors propose a novel fog-based healthcare system for Mechanized Diagnosis of Heart Diseases using ML algorithms. The proposed system employs IoT devices, cloud/fog computing platforms, and ML algorithms to predict heart disease using ECG data. The author also suggests that further improvements can be made by incorporating blockchain and applying the fog model to real-time data.

In [77] they propose a wearable sensor-based system that uses a Recurrent Neural Network (RNN) for activity prediction. The system utilizes multiple healthcare sensors and an edge device, and the trained RNN outperforms traditional methods on a publicly available dataset. The proposed approach could be applied in healthcare services for real-time analysis and predicting human activities in smartly controlled environments. The authors have proposed a proposes a three-tier sensor-edge-cloud architecture to reduce the overload on the cloud by using an edge machine with a

GPU that performs the model training and prediction as an improvement to the current system.

In [80] the authors proposed an edge-based differential data collection scheme for wireless sensor networks (WSNs) that addresses privacy concerns in sensor-cloud systems. The scheme stores sensitive data in a hierarchical storage system, reducing communication costs and improving storage efficiency. However, storing the data in a hierarchical manner leaves the data less useful.

In [40] the authors propose a privacy-preserving edge-cloud ML architecture based on DP strategies. The proposed architecture protects data privacy while maintaining ML accuracy, using a three-tier sensor-edge-cloud computing framework. This study proposed an architecture similar to [40] however the major difference is that models trained in this work are evaluated on the cloud rather than on the edge as that saves more time. This architecture is also geared towards time series and image data whereas their architecture only focused on categorical and discreet data.

2.2.5 Domain-Specific Features

The reason to convert one-dimensional time series to two-dimensional image data was to highlight and capture the regional trends that would otherwise be scattered over time to improve time series classification with DP. This also enabled us to use state-of-the-art image classification deep learning models for classification [39]. This study is conducted with four different features, namely raw time series, scalograms, recurrence plots (RP), and a joint representation, called multi-modal representation, comprising RP and scalograms stacked together. Scalograms and RP images are well-known techniques that capture the dynamical properties of dynamical systems [56].

Scalograms are pictorial feature representations that denote the strength of various frequencies of a signal over time [23].

2.2.6 Data Set and Experimental Tasks

The experiments in this thesis are conducted using two datasets:

1. **Sabotage Data**: EEG signals are recorded of patients performing simple tasks. The patients were given specific instructions to either complete the task with maximum speed and accuracy (true effort condition) or deliberately perform poorly while still completing the trials (sabotage condition) [23].

2. **EEG Alpha Waves dataset**: This dataset contains EEG recordings of subjects for an eye open/closed experiment [20]. From here on this dataset will be referred to as the alpha wave or the eye dataset.

2.3 Summary

This chapter discussed introductory content and provided a background for the proposed methodology. It also discussed relevant work done in the field of EEG signal classification, making physiological signals differentially private, DP for images and ML, and deferentially private ML-IoT architectures, and introduced the datasets that are being used. This chapter points out that not many studies have applied DP to time series from sensors and Internet-of-Things (IoT) devices. This study aims to fill in this knowledge gap.

Chapter 3

SYSTEM DESIGN AND METHODOLOGY

3.1 System Design for DP-ML-IoT

The traditional end-to-end ML pipeline used in the industry has been described in Figure 3.1 [67]. We follow the same procedure described in the framework. For our implementation, each step is divided into different sections and is completely independent of each other to make our framework more modular and customizable. Dividing each component into individual components reduces dependencies and opens the door to cross-platform models, it also makes it easier for the user to scale individual components and add more functionality into the pipeline without affecting the existing components.

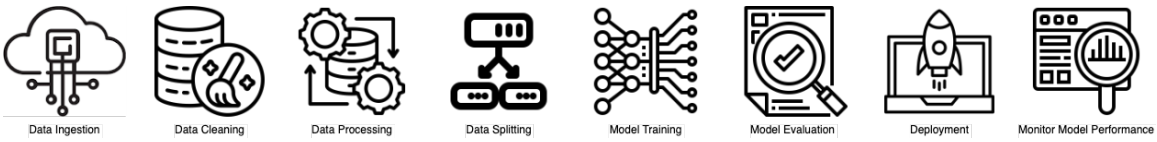

Figure 3.1: End-To-End ML Pipeline [67].

Our proposed ML pipeline is applicable across different workflows and can be used over both edge and cloud. Each process in this workflow is running independently and only data is shared across various modules. Each module can be customized and deployed on individual microservices and can be systematically called in any manner.

This chapter explains how the DP-ML-IoT architecture, visualized in Figure 3.2 which is an end-to-end ML pipeline utilizing both edge and the cloud was designed and explains each step of the architecture.

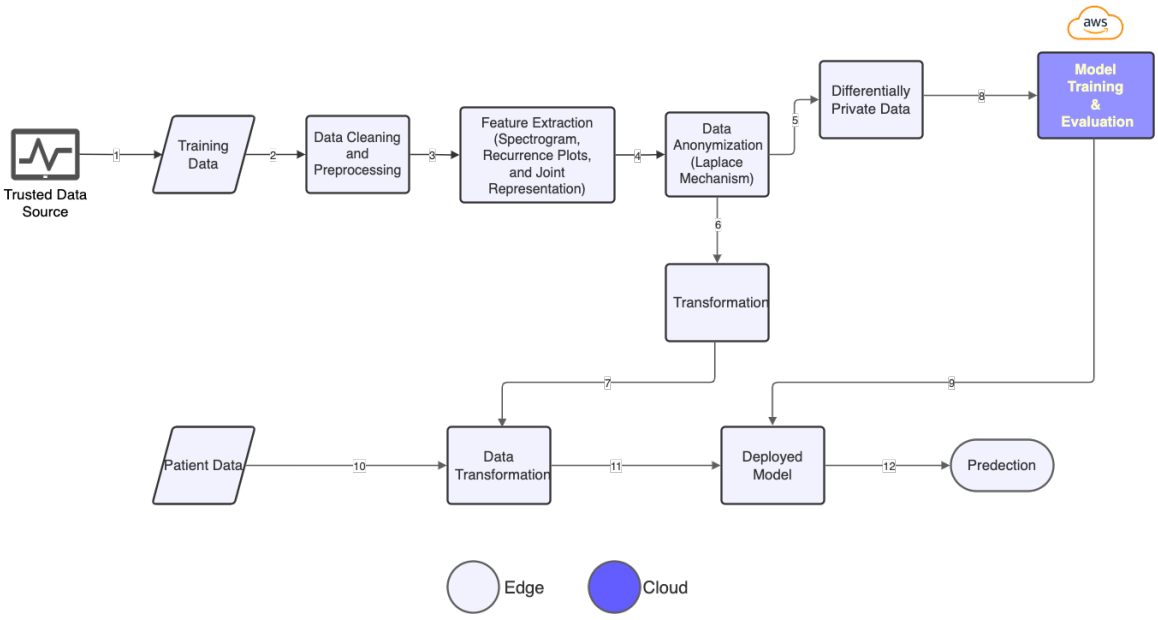

Figure 3.2: ε-DP capable IoT-ML architecture

3.1.1 Edge

Data ingestion is the key component of the edge. In a typical edge-cloud pipeline, the edge is made up of a range of IoT sensors and devices that provide/generate data in huge quantities. This study uses the edge as a platform that can collect and process our data before it feeds the data to the cloud for model training purposes. This work assumes that the edge is a device with limited resources (computer or a mobile device). Edge being a low-powered device comes with certain drawbacks such as storing a large amount of data might not be possible scaling up the storage for which more hardware is required, and training of complex models is also not possible on the edge as it would require substantial computation power which could end up costing thousands of dollars to the user. All these problems can easily be solved using cloud services.

Despite edge having some drawbacks it might be necessary to keep the data close to the edge when dealing with classified or critical information. This study is working

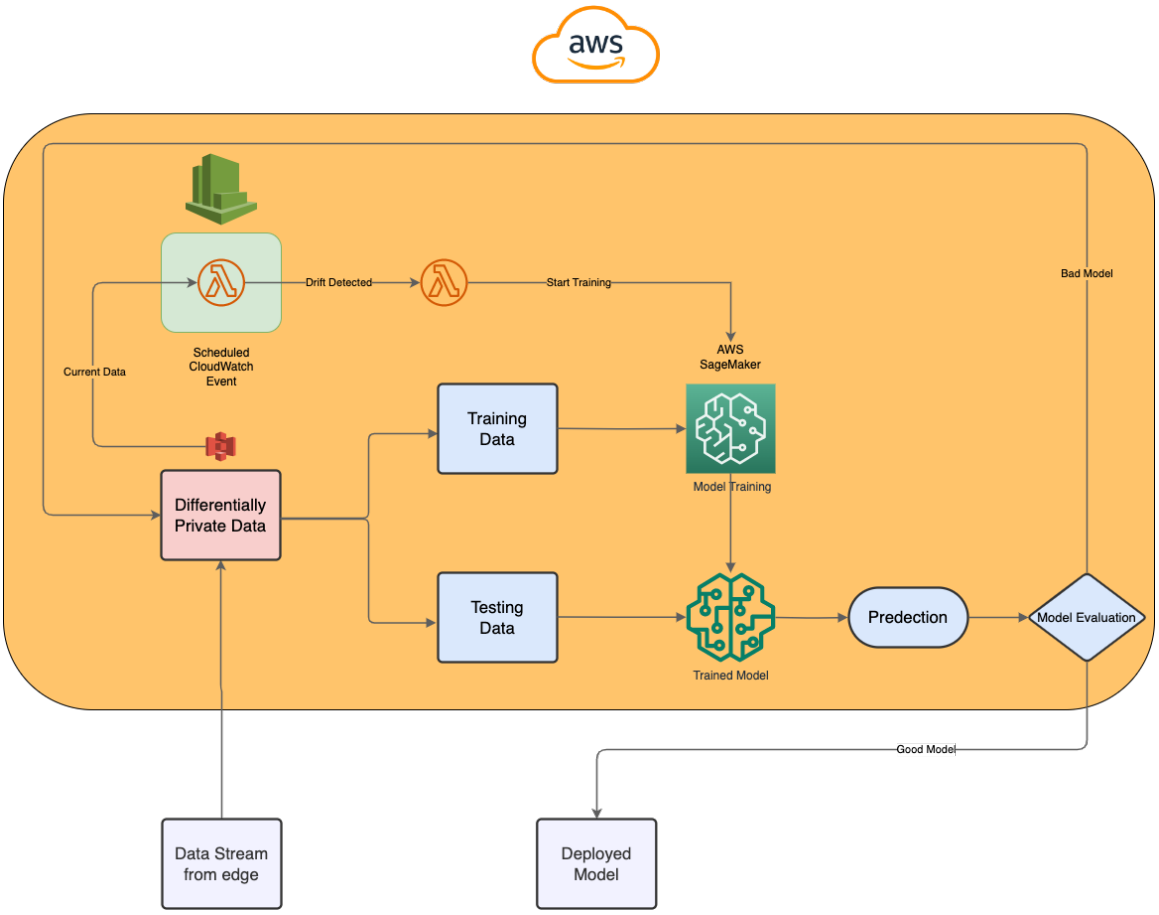

Figure 3.3: Cloud Architecture with drift detection

with healthcare data, and since healthcare data is considered private and sensitive information, it preprocess and anonymized the data at the edge layer before the data is sent to the cloud for further steps. The edge is used for all the data ingestion, cleaning, preprocessing, feature engineering, and Local-DP application on the data, after which the data is uploaded to the cloud for data archival and model training.

3.1.2 Cloud

This study uses the cloud for some of the most important parts of our ML pipeline, namely data storage/archiving and ML model training. This study uses Amazon Web Services (AWS), which offers affordable, scalable, and efficient cloud solutions,

and uses S3 (Simple Storage Service) to archive our current and reference data and AWS SageMaker on-demand instance to train our ML models. AWS CloudWatch and Lambda functions were used to monitor drift on the training data. A lambda function was created that pulls the current data stored in the S3 bucket. This study uses the Evidently AI library in Python to calculate if drift occurred or not. To calculate if drift occurred or not in this work the ks_stat_test from evidently was used, which performs a Kolmogorov-Smirnov test on the current data and the reference data. If a drift was detected the current version of data is stored as reference data in a separate S3 bucket and calls another lambda function that calls the SageMaker instance that trains the ML model with the latest available data. If no drift is detected the lambda function is terminated. The lambda function is configured to be a scheduled call triggered every 24 hours using AWS CloudWatch.

3.1.3 Data Flow

The workflow of this architecture in Figure 3.2 is described as:

After the data has arrived from a trusted data curator (1) the data is streamed into the edge layer. The raw data (2) is cleaned and preprocessed (3) as mentioned in the previous section. Feature extraction (4) is then performed on the cleaned data, which is sent for data anonymization using Laplace Mechanism. A pre-selected value for ε is used based on our research and use that value to generate Laplace noise which will be added to our data (5). All the optimal transformation and epsilon values are then sent stored for later use (6). The anonymized data is then sent to the cloud (8) where it is stored in an AWS S3 bucket from where the data will be sent to AWS Sagemaker where our model will be trained and evaluated based on the predetermined metrics on an AWS SageMaker 'ml.p3.2xlarge' instance which uses NVIDIA Tesla V100 GPU with 8 vCPU and 61 GB memory. The model with the least amount of overfitting is

then chosen, i.e. accuracy difference between the test set and train set is less than 5% and the testing accuracy is at least 80%. If the model meets those criteria it is deployed to the edge layer (9). The optimal data transformations (7) are then used to transform raw data (11) collected from the patient (10) and then fed to the trained ML model (12) to make inferences. As seen in Figure 3.2, each step in the model training is modular and can easily be removed and replaced by a new method. It should also be noted that only model training and evaluation are being carried out at the cloud and the rest of the processes are carried out at the edge, which ensures data privacy.

3.1.4 Advantages of Edge-Cloud Architecture

The biggest advantage is the price and time that the user will save by opting to use the cloud. Setting up hardware for a large ML pipeline will be very expensive and difficult to maintain but with cloud services, the user only pays for the time the service was used for and does not have to deal with maintenance thus the user ends up saving their time and money. The other advantage is that it allows us to build an end-to-end ML pipeline, given that it provides a huge stack of services to deal with enormous data from disparate sources. AWS's S3 buckets also provide an unlimited storage service for data objects which is very useful for users dealing with a high volume of data. There is a dedicated service provided by AWS that can be customized to detect drift in our ML pipeline, CloudWatch can be used to detect and deal with drifts that may occur anywhere in our ML pipeline. This thesis aims to automate the creation of S3 buckets, and classifier training using AWS Sagemaker service and keep track of data drift using AWS CloudWatch.

3.2 Methodology

This work uses the sabotage and the alpha wave datasets and then performs data cleaning, data processing, and feature extraction. The final result of our previous steps was then fed into our Deep Learning model. These methods are thoroughly discussed in the following sections.

3.2.1 Data Collection

This study has used two separate datasets, namely the Sabotage Data and the Alpha wave dataset. These datasets are described in Section 2.2.6.

3.2.2 Data Collection Tasks

This section describes the task done by the participants during data collection.

3.2.2.1 Cognitive Motor Integration Task

To collect data for the Sabotage dataset, subjects were given a computer-based visuo-motor skill assessment task where vision and action are decoupled. The task required the integration of spatial and cognitive rules and thus required cognitive-motor integration (CMI). The task requires one to move the index finger of their dominant hand along the touch screen of the tablet to move a cursor (white dot, 5 mm diameter) from a central location to one of four peripheral targets (up, down, left, or right relative to center) as quickly and as accurately as possible. The authors of this data collected the EEG data from the Muse headband while participants performed this task.

3.2.2.2 Eye State Task

Each participant underwent one session consisting of ten blocks of ten seconds of EEG data recording. Five blocks were recorded while a subject was keeping his eyes closed (condition 1) and the others while his eyes were open (condition 2). The two conditions were alternated. Before the onset of each block, the subject was asked to close or open his eyes according to the experimental condition. The experimenter then tagged the EEG signal using the in-house application and started a 10-second countdown of a block.

3.3 Data Cleaning and Preprocessing

The input data is read from a file. The data is then separated based on the labels. The separated data is then converted into an MNE [42] data structure. The Alpha Wave dataset was recorded at a sampling frequency of 512 Hz. Hence, it is resampled to 256 Hz which is the same as the sabotage dataset. The data is then divided into small fixed-length windows containing 64 data points with a 50% overlap which are 0.25 s long and have an overlap of 0.125 s. After going through this process the raw time series data of shape 64 x N is obtained, where N is the number of channels of the headset through which the data was recorded ($N = 4$ for the sabotage dataset and $N = 16$ for the alpha wave dataset).

3.4 Feature Engineering

After obtaining the raw time series data the next step in this pipeline is feature engineering. The raw time series data is transformed into 2D grayscale images to help improve the performance of classification tasks for time series data. Our goal in

converting one-dimensional time series to two-dimensional image data was to highlight and capture the regional trends that would otherwise be scattered over time. This also enabled us to use state-of-the-art image classification Deep Learning models for our problem [39].

This study is using four different features: raw time series, scalograms, recurrence plots, and multi-modal features (scalograms and recurrence plots stacked together).

3.4.1 Scalograms

This study explains how scalograms are made in Section 2.1.2.3. To create scalograms in this study, each window of the time series data is taken, and the cwt function from ssqueezepy library [61] is applied to each window to perform continuous wavelet transform (CWT) on them. CWT indicates how strong different frequencies of a signal are over time. The resultant scalograms were of the shape 162 x 64 x N, so they are downsized to be the shape 64 x 64 x N. The performance of the ML models trained with and without the downsampled scalograms have very minute differences.

3.4.2 Recurrence Plot

This study explains how RPs are created in section 2.1.2.4. To create RP from the raw time series data, the RecurrencePlot.transform function from the pyts [38] library is used. After applying this transform function, 2D RP of size 64 x 64 is created for each channel. The recurrence plots for all N channels are stacked together and the resultant shape of our dataset became 64 x 64 x N which was consistent with the shape of our scalograms.

3.4.3 Multi-Modal

To create the multimodal dataset, the first step is to create scalograms and recurrence plots as described before and then combine the two datasets by stacking/concatenating the channels and the resultant dataset is of the shape 64 x 64 x 2N.

3.5 Data Anonymization

After generating the images ε-DP is then applied to the images. The Laplace function found in the numpy.random package in Python is used to implement the Laplace mechanism for ε-DP with the value of ε ranging from 0.01 to 0.99. The noise generated through that function follows a Laplacian distribution which is centered at 0 with a sensitivity of 1. The noise is then added to the training data. The inputs with varying levels of noise added to them and inputs with no noise added to them ($\varepsilon = \inf$) are visualized in Figure 3.4 and Figure 3.5.

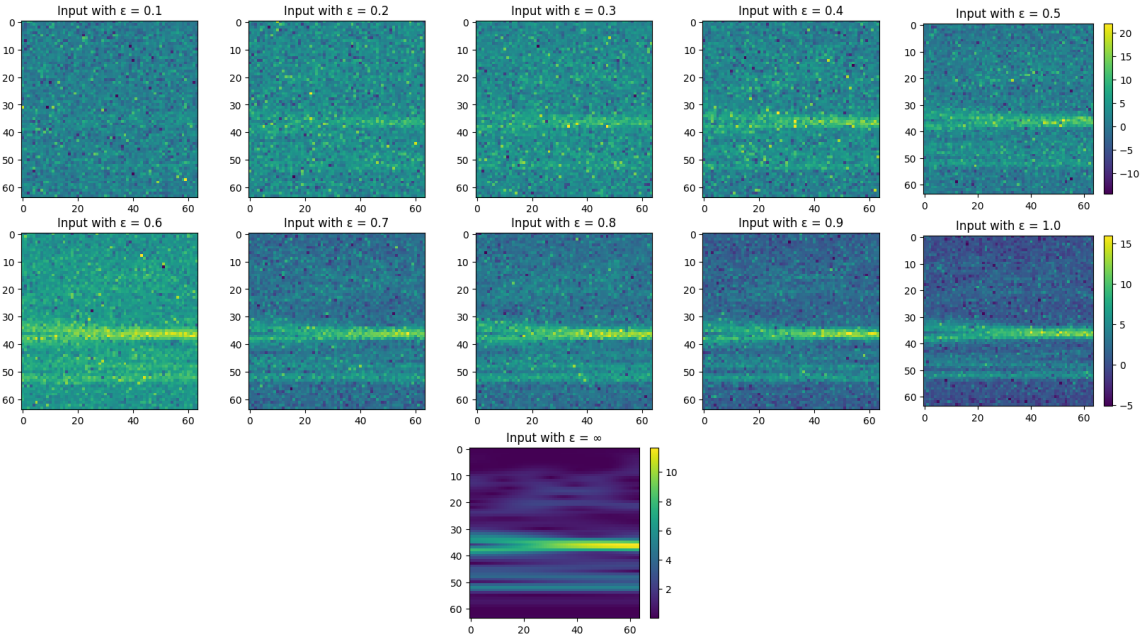

Figure 3.4: Scalograms with varying levels of noise added to them

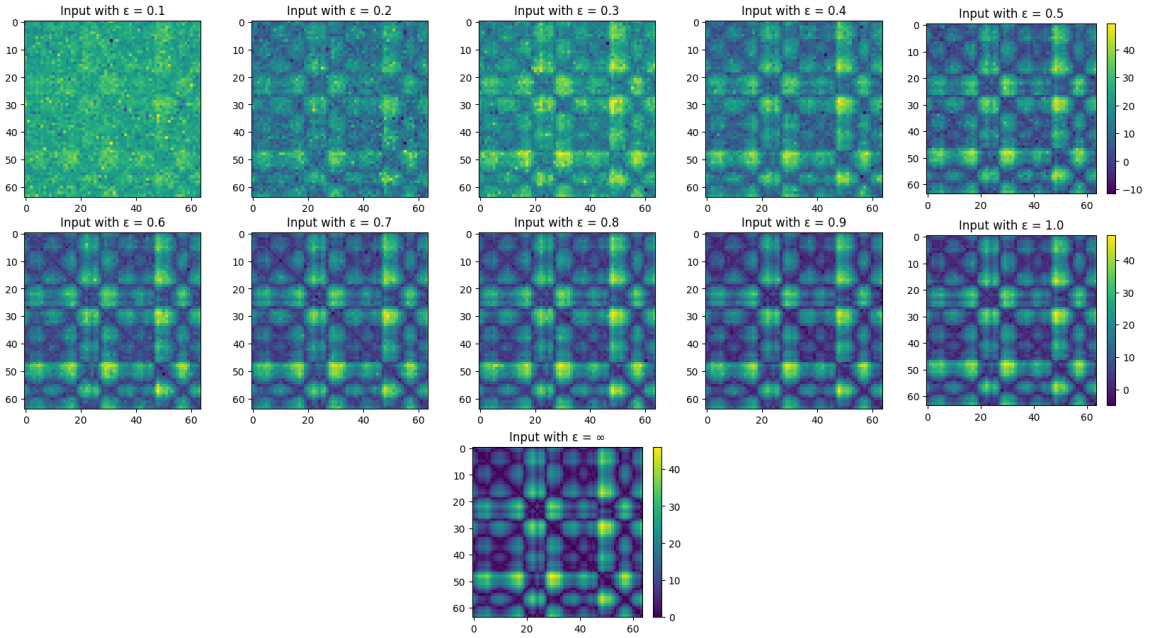

Figure 3.5: Recurrence Plots with varying levels of noise added to them

3.6 Summary

This chapter discussed the datasets used for this study and the tasks performed by the subjects while the data was being collected. It also described our data preprocessing steps and feature engineering steps to clean our data and convert the raw time series data to images and anonymize them. It also explains different parts of our DP-IoT-ML edge-cloud architecture.

Chapter 4

MODEL TRAINING AND ESTABLISHING BASELINES

4.1 Introduction

This section discusses the DL model and the different metrics used in this work to evaluate our models.

4.2 Evaluation Metrics

4.2.1 Evaluation Metrics for Binary Classification

To evaluate the proposed approach in this study, several experiments were performed which involved pre-processing and anonymizing time series and the consequent image representations using different values of ε. Nested five-fold cross-validation (CV) was used to train the ML models with and without DP. The classification metrics used in this study is accuracy as described below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Where,

- TP = True positive, which is the number of correctly predicted positive class values

- TN = True negative, which is the number of correctly predicted negative class values
- FP = False positive, which is the number of incorrectly predicted positive class values
- FN = False negative, which is the number of incorrectly predicted negative class values

4.2.2 Evaluation Metrics for Comparing Anonymized and Non-Anonymized Data

This study also compared how different the anonymized inputs were from the non-anonymized inputs using Peak Signal Noise Ratio (PSNR) and cosine similarity. PSNR was used for quality measurement between the original non-anonymized and anonymized images and used cosine similarity to find similar patterns in the images. Mathematically these metrics are defined as follows:

$$PSNR\,(I1, I2) = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}(I1, I2)}\right) \tag{4.2}$$

where,

$$MSE\,(I1, I2) = \frac{1}{m \cdot n}\sum_{i=1}^{m}\sum_{j=1}^{n}(I1(i,j) - I2(i,j))^2 \tag{4.3}$$

- I1 = noise-free image representation of Time series
- I2 = I1 + noise.
- MAX represents the maximum possible pixel value of the images

- MSE(I1, I2) represents the mean squared error between images I1 and I2
- m and n represent the dimensions of the images I1 and I2, respectively
- I1(i, j) and I2(i, j) represent the pixel values at position (i, j) in images I1 and I2, respectively

$$Similarity = \cos(\theta) = \frac{\sum_{k=1}^{n}(x_k \cdot y_k)}{\sqrt{\sum_{k=1}^{n}(x_k)^2} \cdot \sqrt{\sum_{k=1}^{n}(y_k)^2}} \tag{4.4}$$

where,

x_k and y_k are the corresponding elements of vectors x and y at index k and $x_k \cdot y_k$ is calculating the dot product between the two vectors. Vectors x and y are created by flattening image $I1$ and image $I2$. Image $I1$ is the image representation of our time series data and $I2$ is $I1$ + noise. The index k denotes the position of the images in the dataset.

Figure 4.1 visualizes the change of PSNR of the datasets with increasing values of ε. Similarly, Figure 4.2 visualizes how the cosine similarities of the datasets increase as the privacy budget increases.

4.3 Deep Learning

The following CNNs were implemented and their performance was compared with each other using the classification metrics defined in Section 4.2.

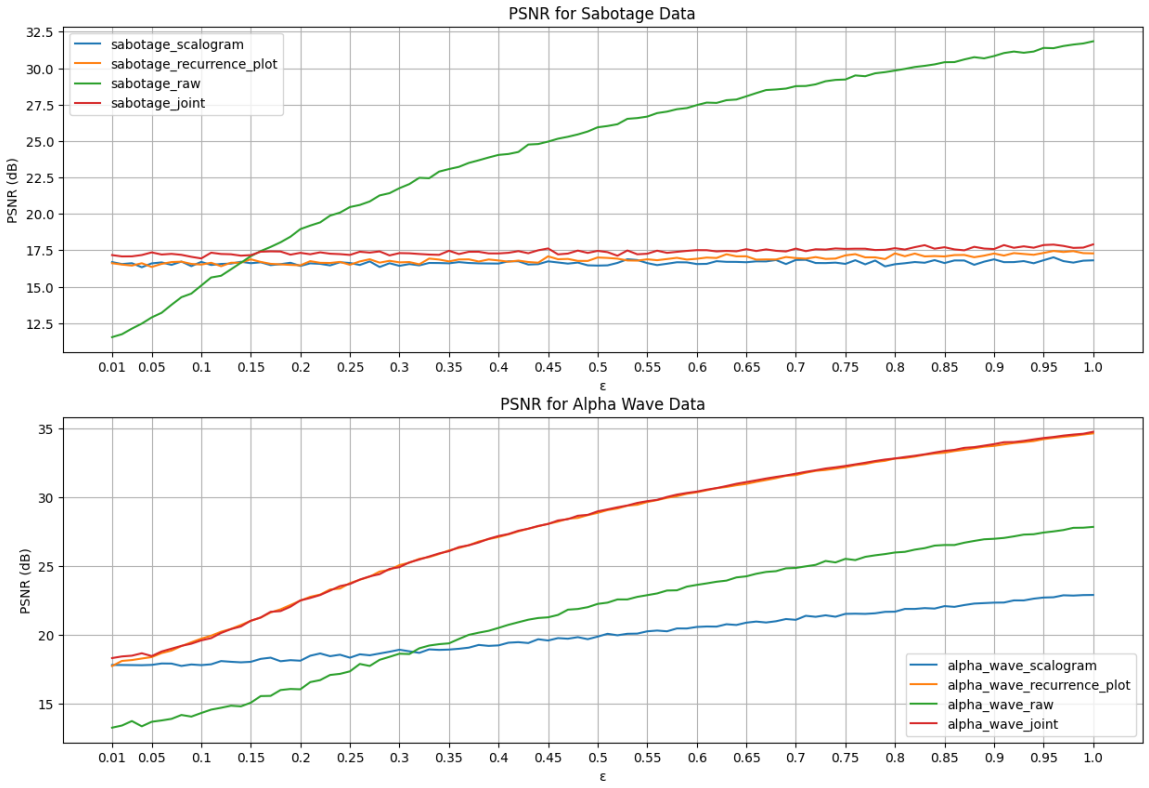

Figure 4.1: PSNR between anonymized and non-anonymized data for different values of ε

4.3.1 Birectional LSTM

This work implemented an LSTM model where the input was passed to two consequent bidirectional LSTM layers with 20 units each. The third layer was a SeqSelfAttention layer with a sigmoid activation function. The fourth layer was another bidirectional LSTM layer with 20 units. The final layer was the dense layer with a sigmoid activation function.

4.3.2 Multi-channel CNN-LSTM With Self Attention

This work uses the multi-channel CNN-LSTM with self-attention (MC-CNN-LSTM-Att) model [23]. The input was passed into a Conv2D layer with 32 filters of size 5x5,

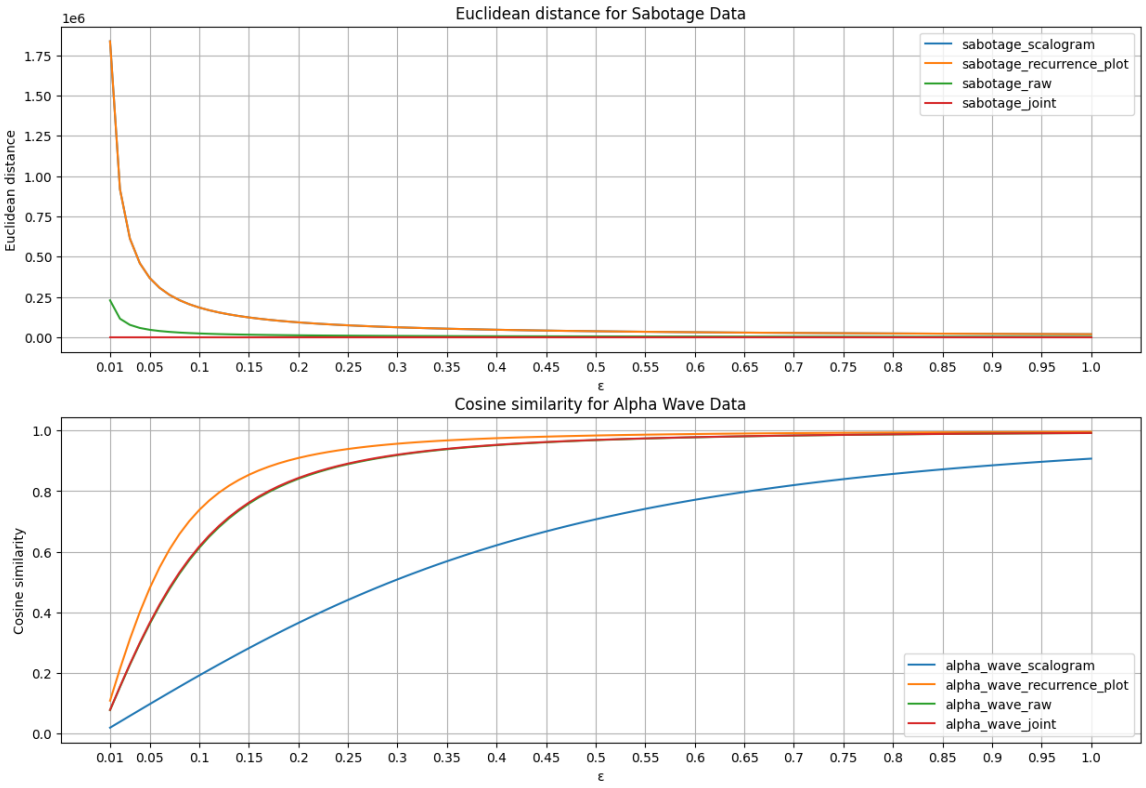

Figure 4.2: Cosine Similarities between anonymized and non-anonymized data for different values of ε

and a maxpool layer of size 2x2. The output received from that was fed to another Conv2D layer with 32 filters of size 3x3 and then to a maxpool layer. To prevent overfitting, a dropout layer was added, the output was flattened which was passed to two LSTM layers, and self-attention was applied and a dense layer was added followed by the softmax function.

4.3.3 ChronoNet

ChronoNet [70] is a recurrent neural network (RNN) architecture designed to work efficiently with physiological time series signals. Its architecture is inspired by the latest advancements in the field of image classification. ChronoNet is constructed by sequentially arranging several 1D convolution layers, which are then followed by

deep-gated recurrent unit (GRU) layers. Each 1D convolution layer employs multiple filters with lengths that exponentially differ, and the stacked GRU layers are densely interconnected in a forward-flowing fashion.

This work uses the image representation of time series data. Since ChronoNet can only work with raw time series data and this study is working with images ChronoNet had to be modified to better suit the needs of this study. So, for this study, changes were made to the existing ChronoNet architecture, by replacing all the 1D convolution layers with 2D convolution layers as they are more suited for image classification tasks.

4.3.4 Modified ChronoNet

This study modified the ChronoNet model to make it compatible with 2D image data input. Figure 4.3 illustrates the modified ChronoNet model. The input is passed to three Conv2D layers of size (2,2), (4,4), and (8,8) respectively with 32 filters of size 2x2 each. The resultant outputs from the three Conv2D layers are concatenated together. This process is repeated two more times and then a batch normalization layer is added to stabilize and standardize the input. A GRU layer with 32 filters is added whose output is then passed to another GRU layer with 32 filters after which the result from both those layers is concatenated and passed onto the third GRU layer. The output from all three GRU layers is concatenated together and passed onto the fourth GRU layer with 32 filters. The output from the fourth GRU is then passed onto the softmax function for classification.

4.4 The Training Procedure for Training and Testing

In this study the data was divided into training and testing data the splits were in the ratio of 70% and 30%, respectively. Five-fold cross-validation was used for the model training for 30 epochs for each fold. The models were validated using validation data within each fold.

This study first collected baseline accuracies for each model trained on non-anonymized data for each dataset. Results for the baseline tests are discussed in the following section

4.5 Results

4.5.1 Baseline Results

This study first compared the performance of all the DL models described in the previous section and found that ChronoNet performed the best. For both sabotage and eye datasets Table 6.1 describes that Chrononet achieved the highest accuracy of 87.48% and 87.29% with raw time series and scalograms respectively.

For the sabotage data, the CNN model with scalograms and recurrence plots didn't give a good performance baseline. The accuracy achieved was 51.36% and 57.04% respectively. Similarly, the bidirectional LSTM model with scalograms and recurrence plots fusion gave 49.01% and 53.46% accuracy respectively.

Table 1 describes the baseline scores achieved for each modality using ChronoNet.

Table 4.1: Baseline Scores for Each Feature
Baseline Accuracies for both datasets

	Sabotage Detection Data		**Alpha Wave Data**	
	Training	**Testing**	**Training**	**Testing**
Raw Time Series	96.75	87.48	90.58	85.72
Scalograms	**87.23**	**85.00**	94.25	87.29
Recurrence Plot	92.99	86.64	**89.08**	**86.84**
Joint Representation	88.85	85.28	92.57	84.32

4.6 Summary

This chapter discussed the different evaluation metrics that were used in this study to measure the performance of our data and to measure how different are the anonymized and non-anonymized images. It also explained the experiments performed to establish our baseline performance with non-anonymized data.

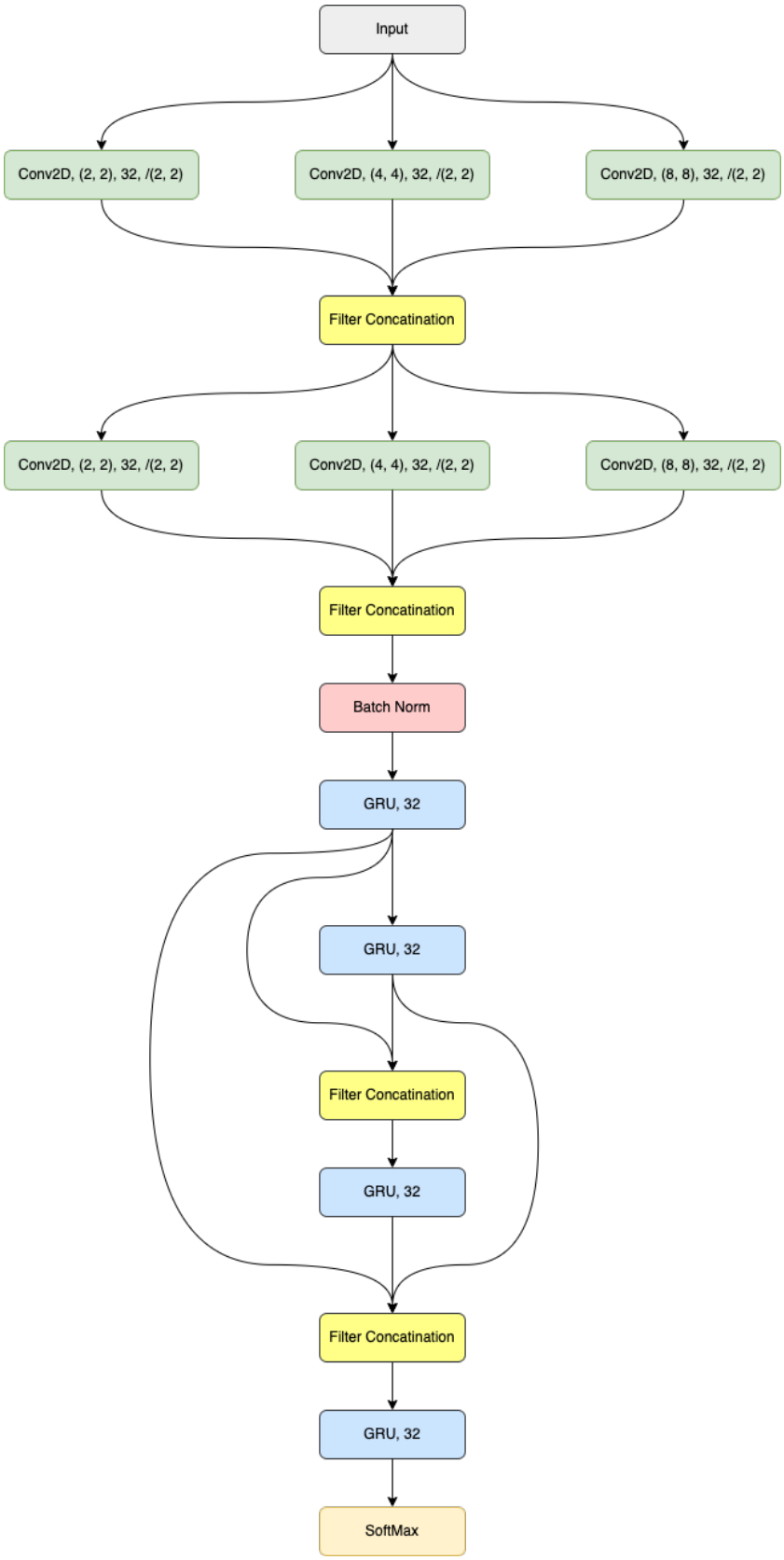

Figure 4.3: Architecture for modified ChronoNet model

Chapter 5

TRAINING MODELS WITH DIFFERENTIAL PRIVACY

5.1 Introduction

This chapter will discuss the results achieved after training ChronoNet with anonymized data. It will also draw a comparison between the inputs when they were anonymized and their originals. It then compares how different modalities performed with different values of ε.

5.2 Data Preprocessing and Feature Engineering

After receiving data from the trusted data curator this work follows the same data cleaning and pre-processing steps and then generates the images as described in Section 3.3 and Section 3.4. After generating the images they are made ε-DP by applying the Laplace mechanism to add Laplace noise to the images. The difference between the original images and the images with ε-DP after the addition of the noise in Figure 4.1 and Figure 4.2.

This chapter will now discuss the classification results of both types of data by comparing the performance of their models trained on DP data to their respective baseline accuracies in the following sections.

5.2.1 Results for Sabotage Detection Data

This study first used the sabotage detection data to evaluate our experiments. It is observed that for this dataset, Scalograms performed the best out of all other modalities with performance close to its baseline between ε values of 0.25 to 0.30 with average training accuracy of 86.07% and average testing accuracy of 81.32%. The performance of different modalities over different values of ε is Figure 6.1.

5.2.2 Results for Alpha Wave Data

To validate our findings from the sabotage dataset the same experiments are repeated with the alpha wave dataset. It is observed that for this dataset, recurrence plots were performed closest to their baseline between ε values of 0.25 to 0.30 with average training accuracy of 86.37% and average testing accuracy of 86.04%. The performance of different modalities over different values of ε is visualized in Figure 6.2.

5.3 Summary

This chapter discussed the performance of ChronoNet trained with both datasets with varying levels of ε. After carefully examining the results it is determined that ε values between 0.25 to 0.30 offer the best ε-DP performance close to non-anonymized data. This study discovered that Spectrograms and recurrence plots were the features that gave the best performance under ε-DP.

Chapter 6

RESULTS

6.1 Introduction

In this section, we discuss the classification results of both types of data, non-anonymized and anonymized, by comparing the performance of their models trained on DP data to their respective baseline accuracies. This study divided the data into training and testing data the splits were in the ratio of 70% and 30%, respectively. Five-fold cross-validation was used for the model training for 30 epochs for each fold. The models were validated using validation data within each fold.

Sections 6.1.1, 6.1.2, 6.1.3 discuss the results respective to each research question asked at the beginning of this thesis.

6.1.1 RQ 1: Comparison of ML classification performance with and without DP

RQ 1 is answered by comparing the highest accuracy achieved by ChronoNet during the cross-validation trained on data with privacy budget $\varepsilon \in [0.01, 1.0] \cup [\infty]$. Data with an infinite privacy budget has no privacy applied to it. This study started by first collecting baseline accuracies for both datasets for each feature. Table 6.1 describes the baseline scores calculated for each feature. The best baseline performance for both datasets is highlighted and can be seen that for the sabotage dataset scalograms performed the best and had the least amount of overfitting. For the alpha wave dataset RPs performed the best with the least amount of overfitting.

Table 6.1: Baseline Scores for Each Feature
Baseline Accuracies for both datasets

	Sabotage Detection Data		Alpha Wave Data	
	Training	**Testing**	**Training**	**Testing**
Raw Time Series	96.75	87.48	90.58	85.72
Scalograms	**87.23**	**85.00**	94.25	87.29
Recurrence Plot	92.99	86.64	**89.08**	**86.84**
Joint Representation	88.85	85.28	92.57	84.32

After collecting the baseline accuracies models with DP anonymized data are then trained and their performance is then recorded. Figure 6.1 and Figure 6.2 describe the performance of ChronoNet on both datasets with different features for varying levels of ε. It is observed that for the sabotage dataset, both time series and scalograms with DP were able to perform as well as non-DP data. Similarly, for the alpha wave dataset, it is observed that all the features with DP were able to perform close to the baseline.

RQ 1: This work could effectively train the ChronoNet model with image representation of data for achieving classification performance with DP close to that of non-DP data.

6.1.2 RQ 2: Selecting ε value that can guarantee classifier accuracy close to non-anonymized data

This study first used the sabotage detection data to evaluate the results of the experiments performed. It then highlights that for this dataset, scalograms performed the best out of all other features with DP performance close to its baseline when $\varepsilon \in [0.25, 0.30]$ with average training accuracy of 86.07% and average testing accuracy of 81.32%. To validate the findings from the sabotage dataset same experiments were performed with the alpha wave dataset (eye blink dataset). This study highlights that for this dataset, RP images performed closest to their baseline when trained with

Figure 6.1: Performance of models Trained on Time Series, Scalogram, and Recurrence Plot Images with and Without DP on Sabotage Data

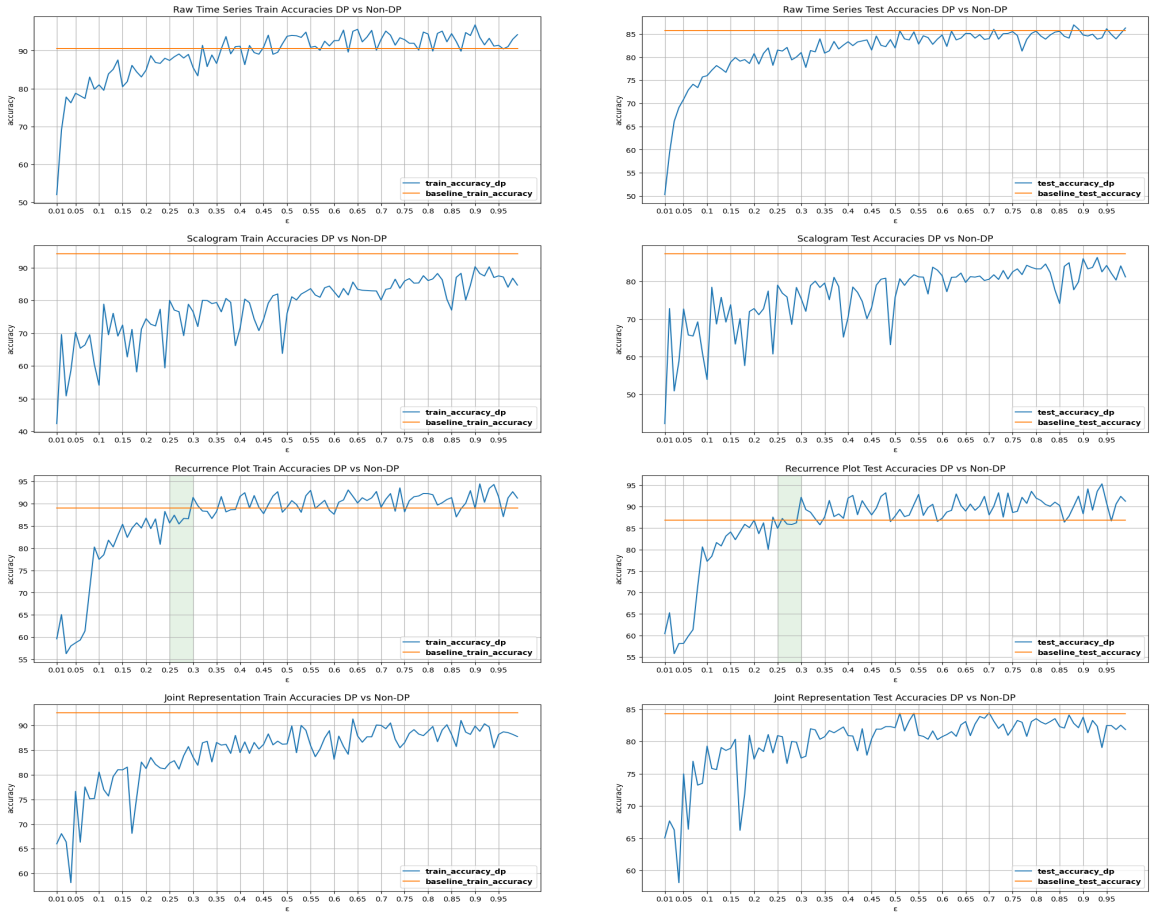

Figure 6.2: Performance of models Trained on Time Series, Scalogram and Recurrence Plot Images with and Without DP on Eye Blink Dataset

Table 6.2: Average Performance of Chrononet DP With Different features where $\varepsilon \in [0.25, 0.30]$

	Sabotage dataset		Alpha Wave Dataset	
	Training Accuracy	**Testing Accuracy**	**Training Accuracy**	**Testing Accuracy**
Raw Time Series	0.8696	0.7941	0.8841	0.8086
Scalogram	**0.8607**	**0.8132**	0.7636	0.7574
Recurrence Plot	0.63	0.6019	**0.8637**	**0.8604**
Joint Representation	0.6249	0.6166	0.832	0.7963

DP data with $\varepsilon \in [0.25, 0.30]$ with average training accuracy of 86.37% and average testing accuracy of 86.04%.

The classification performance of both datasets with different features over different values of ε is visualized in Figure 6.1 and Figure 6.2. This study highlights the minimum value of ε that is between $[0.25 - -0.30]$ which gave performance close to the non-anonymized data. Table 6.2 describes the average performance of all the features when the model was trained with anonymized data where the value of ε was between $[0.25 - -0.30]$. The results of the best features are highlighted in bold for each dataset.

RQ 2: This study shows that ε between $[0.25 - -0.30]$ for image representation of time series yields classification performance with DP close to that of non-DP data.

6.1.3 Results for RQ 3: IoT Architecture capable of supporting DP and ML on images and time series data.

This study developed a robust DP-ML-IoT architecture designed to apply DP to anonymize data and perform ML tasks on the anonymized data. This architecture leverages the utmost capabilities of both the edge and the cloud. Each module in the architecture is independent of each other connected only by the data flow. Modifying and adding more modules to the architecture are made easy because of the independent and modular nature of the architecture. The proposed architecture is

also designed to keep utility costs down as cloud resources are only invoked when they are needed thus cutting down idle waiting time costs.

RQ 3: This work designed a robust IoT architecture that evaluates the DP ML model on the cloud and encapsulates our proposed method giving time series classification performance with DP close to non-DP while ensuring privacy by being irreversible due to the domain-specific image representations of time series.

Chapter 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

This work tries to address the issue of applying ε-DP for time series data where it is desired to achieve reasonable accuracy without compromising on the privacy of time series physiological data. However, it is well-known that with ε-DP there is a performance tradeoff since existing ML models trained using anonymized data with ε-DP decrease the accuracy in contrast to non-anonymized ε-DP. In this work, we present a new angle for applying ε-DP for time -series such that the ML models trained on anonymized data achieve accuracy close to non-anonymized data. We proposed to transform the time series data into 2D image representation such as the spectrogram and reaped the merit of state-of-the-art image classifiers. This paper performed an experimental study to find the optimal value for ε to anonymize physiological data, along with finding the optimal feature type for these types of datasets for training ML models. We validate our results on multiple data sets. We observed that the higher the value of ε the better our ε-DP models perform. We wanted to find out the lowest value of ε with which we can have both high data privacy and data utility and our experiments showed that ε values between 0.25 to 0.30 offer the best ε-DP performance close to non-anonymized data. We also discovered that Spectrograms and recurrence plots were the features that gave the best performance under ε-DP. It is also noteworthy that in some cases we observed that for higher values of ε our anonymized data was able to outperform non-anonymized data.

We also implemented an end-to-end IoT ML edge-cloud architecture that employs the ε privacy technique to train ML models on physiological data collected from IoT devices. Our architecture ensures the privacy of individuals while processing and analyzing the data at the edge securely and efficiently.

7.2 Future Work

Our future work includes extending the approach to conduct the same experiment with a wide variety of physiological datasets and general time series datasets. We would also like to expand our research in terms of using different models such as using pre-trained models well known for image classification such as Inception, Resnet, VGG, etc. We would also like to use a different statistical method for detecting drift the KS test requires access to both the reference data and the current data therefore we are utilizing double the amount of storage space. In the future, we would like to use a test that does not rely on reference data but instead relies on the statistical measures of the reference data, which in turn frees up a lot of space. We would also like to implement federated learning into our ε-DP-IoT-ML architecture so that data collection and model training can be distributed and we can also observe how the value of ε will change if the data and model training is distributed in the cloud.

BIBLIOGRAPHY

[1] 5 types of brain waves frequencies: Gamma, beta, alpha, theta, Delta. https://research.kudelskisecurity.com/2020/03/11/differential-privacy-a-comparison-of-libraries/.

[2] What is the cloud? — Cloud definition. https://www.cloudflare.com/learning/cloud/what-is-the-cloud/, Aug 2020.

[3] What is machine learning? A definition. https://www.expert.ai/blog/machine-learning-definition/, March 2022.

[4] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[5] P. A. Abhang, B. W. Gawali, and S. C. Mehrotra. Chapter 2 - Technological Basics of EEG Recording and Operation of Apparatus. In P. A. Abhang, B. W. Gawali, and S. C. Mehrotra, editors, *Introduction to EEG- and Speech-Based Emotion Recognition*, pages 19–50. Academic Press, 2016.

[6] M. Adnan, S. Kalra, J. C. Cresswell, G. W. Taylor, and H. R. Tizhoosh. Federated learning and differential privacy for medical image analysis. *Scientific reports*, 12(1):1953, 2022.

[7] Z. Ahmad, A. Tabassum, L. Guan, and N. M. Khan. ECG Heartbeat Classification Using Multimodal Fusion. *IEEE Access*, 9:100615–100626, 2021.

[8] N. Ahmadi, M. Nilashi, B. Minaei-Bidgoli, M. Farooque, S. Samad, N. O. Aljehane, W. A. Zogaan, and H. Ahmadi. Eye State Identification Utilizing EEG Signals: A Combined Method Using Self-Organizing Map and Deep Belief Network. *Scientific Programming*, 2022, 2022.

[9] A. Al Wazrah and S. Alhumoud. Sentiment analysis using stacked gated recurrent unit for arabic tweets. *IEEE Access*, 9:137176–137187, 2021.

[10] A. A. AlArfaj and H. A. H. Mahmoud. A Deep Learning Model for EEG-Based Lie Detection Test Using Spatial and Temporal Aspects. *CMC-COMPUTERS MATERIALS & CONTINUA*, 73(3):5655–5669, 2022.

[11] M. Ali. Understanding Data Drift and model drift: Drift Detection in Python. https://www.datacamp.com/tutorial/understanding-data-drift-model-drift, Jan 2023.

[12] N. Amiet. Differential Privacy: A comparison of libraries. https://research.kudelskisecurity.com/2020/03/11/differential-privacy-a-comparison-of-libraries/, March 2020.

[13] J. A. S. Aranda, L. P. S. Dias, J. L. V. Barbosa, J. V. de Carvalho, J. E. d. R. Tavares, and M. C. Tavares. Collection and analysis of physiological data in smart environments: a systematic mapping. *Journal of Ambient Intelligence and Humanized Computing*, 11:2883–2897, 2020.

[14] T. O. Ayodele. Types of machine learning algorithms. *New advances in machine learning*, 3:19–48, 2010.

[15] E. Bagdasaryan, O. Poursaeed, and V. Shmatikov. Differential privacy has disparate impact on model accuracy. *Advances in neural information processing systems*, 32, 2019.

[16] T. Ball, M. Kern, I. Mutschler, A. Aertsen, and A. Schulze-Bonhage. Signal quality of simultaneously recorded invasive and non-invasive EEG. *Neuroimage*, 46(3):708–716, 2009.

[17] V. W. Berger and Y. Zhou. Kolmogorov-Smirnov Test: Overview. *Wiley statsref: Statistics reference online*, 2014.

[18] J. Brownlee. Difference between classification and regression in machine learning. https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/, May 2019.

[19] A. J. Casson, S. Smith, J. S. Duncan, and E. Rodriguez-Villegas. Wearable EEG: what is it, why is it needed and what does it entail? In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5867–5870, 2008.

[20] G. Cattan, P. L. C. Rodrigues, and M. Congedo. EEG Alpha Waves dataset. https://doi.org/10.5281/zenodo.2348892, Dec 2018.

[21] M. A. P. Chamikara, P. Bertok, I. Khalil, D. Liu, and S. Camtepe. Privacy preserving face recognition utilizing differential privacy. *Computers & Security*, 97:101951, 2020.

[22] M. Chaudhary. An Efficient Machine Learning Software Architecture for Internet of Things. https://yorkspace.library.yorku.ca/xmlui/handle/10315/38477, Jul 2021.

[23] M. Chaudhary, M. S. Adams, S. Mukhopadhyay, M. Litoiu, and L. E. Sergio. Sabotage detection using DL models on EEG data from a cognitive-motor integration task. *Frontiers in human neuroscience*, 15:662875, 2021.

[24] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[25] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[26] A. Craik, Y. He, and J. L. Contreras-Vidal. Deep learning for electroencephalogram (EEG) classification tasks: a review. *Journal of neural engineering*, 16(3):031001, 2019.

[27] S. P. Dash. The impact of IoT in healthcare: global technological change & the roadmap to a networked architecture in India. *Journal of the Indian Institute of Science*, 100(4):773–785, 2020.

[28] S. De, L. Berrada, J. Hayes, S. L. Smith, and B. Balle. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650*, 2022.

[29] E. Debie, N. Moustafa, and M. T. Whitty. A Privacy-Preserving Generative Adversarial Network Method for Securing EEG Brain Signals. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[30] M. Devarajan, V. Subramaniyaswamy, V. Vijayakumar, and L. Ravi. Fog-assisted personalized healthcare-support system for remote patients with diabetes. *Journal of Ambient Intelligence and Humanized Computing*, 10:3747–3760, 2019.

[31] C. Dwork. Differential Privacy. In *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33*, pages 1–12. Springer, 2006.

[32] C. Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation: 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008. Proceedings 5*, pages 1–19. Springer, 2008.

[33] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[34] C. Dwork and G. N. Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.

[35] D. R. Edla, S. Dodia, A. Bablani, and V. Kuppili. An efficient deep learning paradigm for deceit identification test on EEG signals. *ACM Transactions on Management Information Systems (TMIS)*, 12(3):1–20, 2021.

[36] K. El Emam, S. Rodgers, and B. Malin. Anonymising and sharing individual patient data. *BMJ*, 350, 2015.

[37] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk. Generating private recommendations efficiently using homomorphic encryption and data packing. *IEEE transactions on information forensics and security*, 7(3):1053–1066, 2012.

[38] J. Faouzi and H. Janati. pyts: A Python Package for Time Series Classification. *Journal of Machine Learning Research*, 21(46):1–6, 2020.

[39] G. R. Garcia, G. Michau, M. Ducoffe, J. S. Gupta, and O. Fink. Temporal signals to images: Monitoring the condition of industrial assets with deep

learning image processing algorithms. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 236(4):617–627, 2022.

[40] P. Goyal, M. Schtern, S. Mukhopadhyay, and M. Litoiu. Towards a Differential Privacy Machine Learning Edge-Cloud Architecture - An Experimental Study. In *Proceedings of the 32nd Annual International Conference on Computer Science and Software Engineering*, CASCON '22, page 175–180, USA, 2022. IBM Corp.

[41] P. Goyal, M. Schtern, S. Mukhopadhyay, and M. Litoiu. Towards a Differential Privacy Machine Learning Edge-Cloud Architecture-An Experimental Study. In *Proceedings of the 32nd Annual International Conference on Computer Science and Software Engineering*, pages 175–180, 2022.

[42] A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, and M. S. Hämäläinen. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7(267):1–13, 2013.

[43] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

[44] R. Hanumantharaju, K. Shreenath, B. Sowmya, and K. Srinivasa. Fog based smart healthcare: a machine learning paradigms for IoT sector. *Multimedia Tools and Applications*, 81(26):37299–37318, 2022.

[45] Harvard University Privacy Tools Project. Differential Privacy. https://privacytools.seas.harvard.edu/differential-privacy.

[46] J. J. Hathaliya and S. Tanwar. An exhaustive survey on security and privacy issues in Healthcare 4.0. *Computer Communications*, 153:311–335, 2020.

[47] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[48] A. Jain, H. Patel, L. Nagalapatti, N. Gupta, S. Mehta, S. Guttula, S. Mujumdar, S. Afzal, R. Sharma Mittal, and V. Munigala. Overview and Importance of Data Quality for Machine Learning Tasks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '20, page 3561–3562, New York, NY, USA, 2020. Association for Computing Machinery.

[49] J. P. Kelly, B. L. Golden, and A. A. Assad. Cell suppression: Disclosure protection for sensitive tabular data. *Networks*, 22(4):397–417, 1992.

[50] D. C. Klonoff. Fog computing and edge computing architectures for processing data from diabetes devices connected to the medical internet of things. *Journal of diabetes science and technology*, 11(4):647–652, 2017.

[51] E. Knorr and G. Gruman. What cloud computing really means. *InfoWorld*, 7(20-20):1–17, 2008.

[52] M. KOKLU and K. SABANCI. The classification of eye state by using kNN and MLP classification models according to the EEG signals. *International Journal of Intelligent Systems and Applications in Engineering*, 3(4):127–130, 2015.

[53] Z. Koudelková and M. Strmiska. Introduction to the identification of brain waves based on their frequency. In *MATEC Web of Conferences*. EDP Sciences, 2018.

[54] J. Lee and C. Clifton. How much is enough? choosing ε for differential privacy. In *Information Security: 14th International Conference, ISC 2011, Xi'an, China, October 26-29, 2011. Proceedings 14*, pages 325–340. Springer, 2011.

[55] E. Luo, M. Z. A. Bhuiyan, G. Wang, M. A. Rahman, J. Wu, and M. Atiquzzaman. Privacyprotector: Privacy-protected patient data collection in IoT-based healthcare systems. *IEEE Communications Magazine*, 56(2):163–168, 2018.

[56] N. Marwan, J. Webber, Charles L., E. E. N. Macau, and R. L. Viana. Introduction to focus issue: Recurrence quantification analysis for understanding complex systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(8), 08 2018. 085601.

[57] B. M. Mathunjwa, Y.-T. Lin, C.-H. Lin, M. F. Abbod, and J.-S. Shieh. ECG arrhythmia classification by using a recurrence plot and convolutional neural network. *Biomedical Signal Processing and Control*, 64:102262, 2021.

[58] D. Mercier, A. Lucieri, M. Munir, A. Dengel, and S. Ahmed. Evaluating privacy-preserving machine learning in critical infrastructures: A case study on time-series classification. *IEEE Transactions on Industrial Informatics*, 18(11):7834–7842, 2021.

[59] I. J. Mohammed and L. E. George. A Survey for Lie Detection Methodology Using EEG Signal Processing. *Journal of Al-Qadisiyah for computer science and mathematics*, 14(1):Page–42, 2022.

[60] J. Montoya-Martínez, J. Vanthornhout, A. Bertrand, and T. Francart. Effect of number and placement of EEG electrodes on measurement of neural tracking of speech. *Plos one*, 16(2):e0246769, 2021.

[61] J. Muradeli. ssqueezepy. *GitHub. Note: https://github.com/OverLordGoldDragon/ssqueezepy/*, 2020.

[62] M. Naldi and G. D'Acquisto. Differential privacy: An estimation theory-based method for choosing epsilon. *arXiv preprint arXiv:1510.00917*, 2015.

[63] J. Near, D. Darais, and K. Boeckl. Differential privacy for privacy-preserving data analysis: An introduction to our blog series. https://www.nist.gov/blogs/cybersecurity-insights/differential-privacy-privacy-preserving-data-analysis-introduction-our, Aug 2020.

[64] R. B. Pachori. *Time-frequency analysis techniques and their applications*. CRC Press, 2023.

[65] M. Paul, L. Maglaras, M. A. Ferrag, and I. AlMomani. Digitization of healthcare sector: A study on privacy and security concerns. *ICT Express*, 2023.

[66] P. Porwik and B. M. Dadzie. Detection of data drift in a two-dimensional stream using the Kolmogorov-Smirnov test. *Procedia Computer Science*, 207:168–175, 2022.

[67] ProjectPro. How to Build an End to End Machine Learning Pipeline? https://www.projectpro.io/article/machine-learning-pipeline-architecture/567, 26 Apr 2023.

[68] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio. Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2):92–102, 2018.

[69] T. K. Reddy and L. Behera. Online Eye state recognition from EEG data using Deep architectures. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 000712–000717, 2016.

[70] S. Roy, I. Kiral-Kornek, and S. Harrer. Chrononet: a deep recurrent neural network for abnormal EEG identification. In *Artificial Intelligence in Medicine: 17th Conference on Artificial Intelligence in Medicine, AIME 2019, Poznan, Poland, June 26–29, 2019, Proceedings 17*, pages 47–56. Springer, 2019.

[71] S. Sanei and J. A. Chambers. *EEG signal processing*. John Wiley & Sons, 2013.

[72] A. R. Shahid and S. Talukder. A study of differentially private machine learning in healthcare. In *2021 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 1–6. IEEE, 2021.

[73] A. Shankar, H. K. Khaing, S. Dandapat, and S. Barma. Analysis of epileptic seizures based on EEG using recurrence plot images and deep learning. *Biomedical Signal Processing and Control*, 69:102854, 2021.

[74] F. Tadel, D. Pantazis, E. Bock, and S. Baillet. Tutorial 24: Time-frequency. https://neuroimage.usc.edu/brainstorm/Tutorials/TimeFrequency, Jan 2022.

[75] S. Tayebi Arasteh, A. Ziller, C. Kuhl, M. Makowski, S. Nebelung, R. Braren, D. Rueckert, D. Truhn, and G. Kaissis. Private, fair and accurate: Training large-scale, privacy-preserving AI models in radiology. *arXiv e-prints*, pages arXiv–2302, 2023.

[76] P. I. Terrill, S. J. Wilson, S. Suresh, D. M. Cooper, and C. Dakin. Attractor structure discriminates sleep states: recurrence plot analysis applied to

infant breathing patterns. *IEEE transactions on biomedical engineering*, 57(5):1108–1116, 2010.

[77] M. Z. Uddin. A wearable sensor-based activity prediction system to facilitate edge computing in smart healthcare system. *Journal of Parallel and Distributed Computing*, 123:46–53, 2019.

[78] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[79] S. Vijayarani and A. Tamilarasi. An efficient masking technique for sensitive data protection. In *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, pages 1245–1249. IEEE, 2011.

[80] T. Wang, Y. Mei, W. Jia, X. Zheng, G. Wang, and M. Xie. Edge-based differential privacy computing for sensor–cloud systems. *Journal of Parallel and Distributed computing*, 136:75–85, 2020.

[81] K. Xia, W. Duch, Y. Sun, K. Xu, W. Fang, H. Luo, Y. Zhang, D. Sang, X. Xu, F.-Y. Wang, and D. Wu. Privacy-Preserving Brain-Computer Interfaces: A Systematic Review. *IEEE Transactions on Computational Social Systems*, pages 1–13, 2022.

[82] X. Xiao, G. Wang, and J. Gehrke. Differential Privacy via Wavelet Transforms. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1200–1214, 2011.

[83] Z. Xu, B. Yu, and F. Wang. Artificial intelligence/machine learning solutions for mobile and wearable devices. *Digital Health: Mobile and Wearable Devices for Participatory Health Applications*, 55, 2020.

[84] W. Xue, Y. Shen, C. Luo, W. Xu, W. Hu, and A. Seneviratne. A differential privacy-based classification system for edge computing in IoT. *Computer Communications*, 182:117–128, 2022.

[85] Y.-g. Zhang, J. Tang, Z.-y. He, J. Tan, and C. Li. A novel displacement prediction method using gated recurrent unit model with time series analysis in the Erdaohe landslide. *Natural Hazards*, 105:783–813, 2021.

[86] A. Ziller, D. Usynin, M. Knolle, K. Hammernik, D. Rueckert, and G. Kaissis. Complex-valued deep learning with differential privacy. *arXiv preprint arXiv:2110.03478*, 2021.

[87] Z. Zuo, M. Watson, D. Budgen, R. Hall, C. Kennelly, and N. Al Moubayed. Data anonymization for pervasive health care: Systematic literature mapping study. *JMIR Medical Informatics*, 9(10):e29871, 2021.