applied sciences

MDPI

Article

# Privacy Preserving Classification of EEG Data Using Machine Learning and Homomorphic Encryption

**Andreea Bianca Popescu [1,2], Ioana Antonia Taca [1,3], Cosmin Ioan Nita [1,2,*], Anamaria Vizitiu [1,2], Robert Demeter [1,2], Constantin Suciu [1,2] and Lucian Mihai Itu [1,2]**

1 Advanta, Siemens SRL, 500097 Brașov, Romania; Andreea.Popescu@ctbav.ro (A.B.P.); ioana_antonia29@yahoo.com (I.A.T.); anamaria.vizitiu@siemens.com (A.V.); robert.demeter@siemens.com (R.D.); suciu.constantin@siemens.com (C.S.); lucian.itu@siemens.com (L.M.I.)
2 Department of Automation and Information Technology, Transilvania University of Brașov, 500174 Brașov, Romania
3 Department of Mathematics and Computer Science, Transilvania University of Brașov, 500091 Brașov, Romania
* Correspondence: nita.cosmin.ioan@unitbv.ro

**Abstract:** Data privacy is a major concern when accessing and processing sensitive medical data. A promising approach among privacy-preserving techniques is homomorphic encryption (HE), which allows for computations to be performed on encrypted data. Currently, HE still faces practical limitations related to high computational complexity, noise accumulation, and sole applicability the at bit or small integer values level. We propose herein an encoding method that enables typical HE schemes to operate on real-valued numbers of arbitrary precision and size. The approach is evaluated on two real-world scenarios relying on EEG signals: seizure detection and prediction of predisposition to alcoholism. A supervised machine learning-based approach is formulated, and training is performed using a direct (non-iterative) fitting method that requires a fixed and deterministic number of steps. Experiments on synthetic data of varying size and complexity are performed to determine the impact on runtime and error accumulation. The computational time for training the models increases but remains manageable, while the inference time remains in the order of milliseconds. The prediction performance of the models operating on encoded and encrypted data is comparable to that of standard models operating on plaintext data.

**Keywords:** privacy-preserving computations; homomorphic encryption; machine learning; EEG signals

check for updates

**Citation:** Popescu, A.B.; Taca, I.A.; Nita, C.I.; Vizitiu, A.; Demeter, R.; Suciu, C.; Itu, L.M. Privacy Preserving Classification of EEG Data Using Machine Learning and Homomorphic Encryption. *Appl. Sci.* **2021**, *11*, 7360. https://doi.org/10.3390/app11167360

Academic Editor: Arcangelo Castiglione

Received: 31 May 2021
Accepted: 6 August 2021
Published: 10 August 2021

## 1. Introduction

In recent years, artificial intelligence (AI) algorithms have shown great potential in several fields, including healthcare. Allowing for customized diagnosis, treatment planning and disease prevention, AI has demonstrated its ability to deliver personalized medicine [1]. However, to obtain a satisfactory performance, a significant amount of data needs to be collected, stored and processed. While achieving promising results in patient-specific medical applications, accessing sensitive data for training AI models requires proper anonymization [2]. Even at the inference phase, when an already trained AI model is employed, privacy should not be compromised. Moreover, regulations regarding personal data confidentiality (e.g., GDPR in the EU, HIPAA in the U.S.A.) emphasize the need for developing more effective privacy-preserving techniques.

Therefore, over the last few years, the development of privacy-preserving techniques that allow for machine learning-based analysis to be performed on encrypted data has received increasing interest. Techniques such as secure multiparty computation (SMPC), differential privacy and homomorphic encryption have shown great potential to be adopted in machine-learning applications, but each of them comes with limitations. Mohassel and

Zhang [3] were the first who attempted to train a neural network using SMPC for data privacy, but the usability of this method is limited due to the high computational cost. Although differential privacy is less complex from the computational point of view, it implies a trade-off between precision and privacy that can be managed through a proper noise injection mechanism. This technique was used in machine-learning applications [4–6] and has achieved promising results. Homomorphic encryption, allowing for mathematical operations on encrypted data without revealing its underlying information, represents a promising privacy-preserving method to be combined with machine-learning techniques. An early work that uses an HE cryptosystem together with neural networks was introduced in [7], requiring communication between the server and the data owner. This interaction is eliminated in CryptoNets [8], but the YASHE [9] encryption scheme does not support real numbers. To address limitations regarding model complexity, CryptoDL [10] uses low-degree polynomials to approximate nonlinear functions. On the other hand, using approximated activation functions lowers the prediction accuracy of the models. Moreover, as a result of a higher number of operations and a longer runtime, none of the above-described solutions covers the training phase of the models on encrypted data.

All these approaches prove that the integration of HE schemes in real-world AI applications is a difficult task, as there are still important limitations. For example, the BFV scheme [11], also employed in the SEAL homomorphic encryption library [12], introduces noise over plaintext values, reducing thus the number of possible consecutive operations, and does not allow for the encryption of rational numbers. To overcome this last limitation, SEAL uses a particularity of the CKKS [13] scheme that affects the computational precision. Although noise-free HE schemes also exist, most are only partially homomorphic, supporting only certain operations, e.g., Paillier [14], ElGamal [15] or AHEE [15], allowing for noise-free addition and multiplication on small integers.

While for most HE schemes, the plaintext domain is restricted to integers, medical data typically rely on rational numbers. Lacking the possibility of encrypting rational numbers hinders the utility of an HE scheme. Thus, a major challenge is the extension of HE schemes to enable computations with real-valued numbers. To achieve this, multiple encoding methods that extend existing encryption schemes have been proposed. An adaptation of the BFV scheme is introduced in [16], which allows for real number encryption, and prevents the underlying message to rapidly increase during computations. In this case, it is not the message that is encrypted, but a polynomial encoding of it, as presented in [17]. Thus, polynomial coefficients are the digits of the message $m$ in a certain base $b$. For real values, the integer part is encoded using positive powers of $b$, and the rational part uses negative powers of the base. After adding the degree of the polynomial modulus, all polynomial exponents become positive. Another approach for modifying the BFV scheme to render it usable for real values is presented in [18]. Similar to the previous approach, in [19] an encoding method based on a binary system is proposed. An encoding technique that enables the use of the Paillier cryptosystem and its homomorphic properties with rational numbers is introduced in [20], where a real value is written as a fraction of two integers. The restrictions of this method stem from certain conditions imposed on the numerator and denominator: they are bounded by a predefined domain and require an additional step of decryption and approximation with smaller values, to ensure that values remain between the imposed bounds. In [21], real numbers are expressed in a 32-bit binary representation, with sign (1 bit), exponent (8 bit) and mantissa (23 bit), which are then encrypted with HE schemes allowing for bit-wise operations. Similar work was performed by Jaschke and Armknecht [22,23] where an in-depth evaluation of multiple encoding methods was conducted, and a method for choosing the appropriate encoding scheme and parameters for specific use-cases is also described.

Another significant difficulty in performing machine learning using homomorphically encrypted data is that such models require an evaluation of non-linear activation functions and iterative gradient-based optimization methods. An option that was also considered in other similar scenarios (e.g., CryptoNets library [8]) is to approximate the non-linear activation functions with low-degree polynomials. If the approximated activation functions are used with a classical neural network, the function of the entire model can be reduced to a polynomial function containing different combinations of input features with different exponents. Under this assumption, the model-fitting process can be performed using direct analytical methods, rather than iterative gradient-based methods, which allows for model training in a fixed and deterministic number of steps. This provides control over the number of arithmetic operations performed on encrypted data, and over the choice of encryption parameters (e.g., encryption key size). Additionally, because there is no need for performing a comparison between a loss value and a threshold, this approach is suitable for being integrated with existing HE schemes.

The main purpose of this study is to enable machine-learning algorithms to operate directly on encrypted data, employing homomorphic encryption. This way, original data are never distributed, as they are initially encrypted and all subsequent operations are directly performed on the encrypted data, without revealing any information about the underlying data. Thus, we are proposing a method that overcomes certain limitations of the HE schemes. Specifically, we are proposing an encoding scheme that enables HE computations to be performed on rational numbers (considering that, typically, only the addition and multiplication of integers are supported), along with a numerical optimization approach that allows for model training in a fixed and deterministic number of steps, an important property in the HE context. Once the model is trained, the goal is to perform real-time inference on new encrypted data.

To evaluate the feasibility of the proposed method for healthcare applications, we consider two use cases: epileptic seizure recognition and alcoholism predisposition detection, based on EEG recordings. Both use cases are based on publicly available data and are formulated as binary classification problems. Experiments were performed with the proposed method (on encrypted data) but also with classic machine learning methods (on plaintext data) to demonstrate that the prediction performance is similar. Since collecting EEG information for training a model or sending the EEG data to the model owner for the inference phase raises concerns about patient privacy, several studies were conducted regarding EEG signal encryption. Lin et al. [24] and Ahmad et al. [25] proposed a chaos-based EEG encryption system. Since the purpose is to secure EEG information while being stored or transmitted over an insecure channel, the encryption method does not allow for arithmetic operations. Liu et al. [26] used a feed-forward neural network where the activation function is approximated with a linear function to solve an EEG-based classification problem. The training step was performed using plaintext data, while the encryption was employed only during the inference phase. The EEG signals were encrypted using the Paillier algorithm, which only supports addition operations. To allow for the usage of decimals, an encoding method based on scaling was used. The input data was preprocessed by selecting 44 out of the 64 electrode channels, which have the highest Pearson correlation coefficient. Through a performance comparison with other studies in which other classifiers are used, the authors demonstrated that their approach represents the state of the art (90.17% accuracy). However, some differences between the non-privacy-preserving and the privacy-preserving versions of the method were reported.

The paper is structured as follows. Section 2 presents the encoding method, the algorithms, the data pre-processing, along with a description of the use cases. Section 3 details the evaluation of the method performed on synthetic data, and the results obtained for the two real use cases. The last section includes a discussion, further development ideas and conclusions.

## 2. Methods

### *2.1. Encoding Scheme*

Although integrating homomorphic encryption schemes with encoding methods addresses some of the impediments that hinder their real-world utility, there are still difficulties to be considered for obtaining an efficient encoding technique. As HE schemes are already complex, and encrypted operations require a higher computational cost than their plaintext equivalent, adding a new layer of encoding increases the complexity even more. Furthermore, expressing a value as a sequence of numbers leads to a ciphertext of growing size, as every number from the sequence is translated to its encrypted form. For example, when considering a polynomial representation, the multiplication of two encoded values results in an increased number of terms and polynomials with a higher order. As the number of operations increases, the increasing dimension is more difficult to handle, and performing computations becomes more time-consuming.

As discussed in the introduction, encoding based on a polynomial representation is a common technique used to allow for real number encryption. Our approach is to use a polynomial representation of a number, based on negative powers of 10. Since the focus lies on enabling the encryption of real numbers, we deal with the integer and the rational part separately. Let us consider $A \in R$, a real number represented as $A = a_0, a_1 a_2 \ldots a_n$. All digits of the integer part, denoted by $a_0$, are encoded in the first coefficient of the polynomial, corresponding to 10 to the power of 0. The rational part is handled differently, by multiplying each digit with a negative power of 10, according to their position in the initial number. Adding these partial products, as in Equation (1), we obtain the encoding of a real value that consists of a sequence of small integers.

$$A = a_0 10^0 + a_1 10^{-1} + a_2 10^{-2} + \ldots + a_n 10^{-n} \tag{1}$$

The $n$ value denotes the number of decimals that are taken into account in this encoded representation and can be set, depending on the required precision for a certain application. To allow for the encryption of negative numbers, our proposed approach consists of replacing each coefficient of the polynomial with a difference, as in Equation (2).

$$A = (a_0 - b_0)10^0 + (a_1 - b_1)10^{-1} + (a_2 - b_2)10^{-2} + \ldots + (a_n - b_n)10^{-n} \tag{2}$$

If $A$ is a negative number, then terms denoted by $a_0, a_1, \ldots, a_n$ are equal to zero and the underlying information is stored in the $b_0, b_1, \ldots, b_n$ terms, while preserving the sign of the number. From this point onward, referring to a polynomial coefficient designates a difference, such as $(a_i - b_i)$.

For instance, let us consider $A = -13.8701$, which can be encoded as follows:

$$-13.8701 = (0 - 13)10^0 + (0 - 8)10^{-1} + (0 - 7)10^{-2} + (0 - 1)10^{-4} \tag{3}$$

We notice in this example that null decimals are ignored, simplifying the representation and allowing for an increase in precision. If, for a certain application, we have $n = 5$, the encoding takes into account only non-zero decimals, storing in the encoded representation more information from the original number. Using this method, it is possible to encode both positive and negative real numbers, with as many decimals as required, and then encrypt each coefficient by applying an HE scheme.

The operations that can be performed on numbers encoded as previously explained are addition, subtraction and multiplication. As shown next, homomorphic properties with respect to these operations are preserved by this encoding technique since they are based on polynomial calculus.

We consider two real numbers $A_1$ and $A_2$, with their corresponding generic encoded representations as follows:

$$\begin{aligned} A_1 &= (a_0 - b_0)10^0 + (a_1 - b_1)10^{-1} + \ldots + (a_n - b_n)10^{-n} \\ A_2 &= (c_0 - d_0)10^0 + (c_1 - d_1)10^{-1} + \ldots + (c_n - d_n)10^{-n} \end{aligned} \tag{4}$$

Addition is performed by adding the corresponding coefficients as in (5), while subtraction consists of adding the opposite of the subtrahend that is computed, as in (6).

$$A_1 + A_2 = (a_0 + c_0 - (b_0 + d_0))10^0 + \ldots + (a_n + c_n - (b_n + d_n))10^{-n} \tag{5}$$

$$-A_2 = (d_0 - c_0)10^0 + (d_1 {-}_1)10^{-1} + \ldots + (d_n - c_n)10^{-n} \tag{6}$$

To compute $A_1A_2$, we multiply each coefficient of the first factor with all coefficients of the second factor and group them after the corresponding powers. This may lead to an increase in the number of coefficients. Hence, the implementation of each operation allows for selecting a different number of decimals ($n$) to be stored in the result.

As can be noticed in (1), the entire integer part of a rational number is stored in a single coefficient. Of course, the same value can be represented symmetrically by distributing individual digits between different positive powers of 10, but since most of the machine learning-based techniques make use of data normalization, all input numbers are sub-unitary. Hence, adding more coefficients to the polynomial representation only to encode a zero value leads to an avoidable computational overhead.

In combination with this encoding method, it is possible to use any HE scheme that is homomorphic for addition and multiplication, as these are the only two operations performed on ciphertexts during the encoded computations. Moreover, if a specific scheme enables scalar multiplication, the encoding method can also preserve this property, where a scalar could be, in this context, any integer that is neither encoded nor encrypted, or any encoded (but not encrypted) real value.

As explained above, the encoding step transforms the input rational number in a sequence of integers representing coefficients of a polynomial function. In the following experiments, each of these coefficients is encrypted using the HE scheme proposed in [27]. The resulting ciphertext is also a sequence of integers. This scheme allows for homomorphic addition and multiplication, and it also supports scalar multiplication. The encoding method does not affect the homomorphic properties of the ciphertext. Specifically, in [27], the encryption algorithm is described by Equation (7):

$$E_p(x) = smod((x + sign(x) \cdot rand() \cdot p), n) \tag{7}$$

where $x$ is the message (the coefficient) to be encrypted, $p$ is a large prime number, and $n$ is the product between $p$ and another large prime number. The *smod* operation is described in Equation (8), *mod* is the common modulo operation, *rand* yields a random positive integer, and the *sign* function provides the sign of $x$.

$$smod(m, p) = \begin{cases} mod(m, p), & m \geqslant 0. \\ -(mod(|m|, p)), & m < 0. \end{cases} \tag{8}$$

An example of a ciphertext obtained by encrypting the coefficients from the Equation (3) with this HE scheme, using a 32-bit key (for a simpler visualization), is represented in Equation (9).

$$\begin{aligned} -13.8701 = {} & (19214182057 - 24703948372)10^0 + (8234649453 - 19214182065)10^{-1} \\ & + (13724415755 - 8234649460)10^{-2} + (5489766302 - 10979532605)10^{-4} \end{aligned} \tag{9}$$

For the experiments using encoded and encrypted data from the two real use-cases, the encoding precision parameter is set equal to 35, and a 256-bit encryption key is used.

## 2.2. Optimization Formulation

Numerical optimization is typically performed using gradient-based iterative methods that converge in an unknown number of steps. Unfortunately, such formulations are difficult to apply on homomorphically encrypted data because they require an unknown (and typically large) number of operations, there are often nonlinear functions involved, and they require a comparison operation for evaluating the convergence. As most HE schemes are homomorphic only for addition and multiplication, they do not allow for division and do not support nonlinear function evaluation. The comparison between encrypted values is another operation that cannot be performed [28].

To bypass these requirements, we are proposing a solution inspired by previous work, where non-linear functions are approximated with low degree polynomials. Specifically, when performing such approximations, the resulting model turns into a polynomial function. If the loss function to be minimized is also analytically differentiable, the entire optimization problem is convex and can be solved analytically.

Let $X$ be a $N$ by $M$ matrix containing the input data where each row $\mathbf{x}_i$ is a data sample and $\mathbf{y}$ is a column vector of size $N$ containing the output values.

Following the idea of encoding or encrypting the numbers into a polynomial form as suggested in [9,12,18], we assume that the model function to be optimized can be reduced to a multivariate polynomial function as in Equation (10):

$$P(\mathbf{x}) = \sum_{i=0}^{N_t} a_i p_i(\mathbf{x}), \tag{10}$$

where $p_i(\mathbf{x})$ is a monomial term combining components of $\mathbf{x}$ at different powers, e.g., $x_1x_2, x_1^2x_2, x_1^2x_2^3$ etc., $\mathbf{a} = [a_1, a_2, \ldots, a_{N_t}]$ are the model parameters, and $N_t$ is the number of terms. As discussed in [3], to determine the model parameters, a cost function must be defined and optimized. Thus, we assume that the objective is to minimize the sum of squares, described by Equation (11):

$$C(\mathbf{a}) = \sum_{i=0}^{N} (P(\mathbf{x}_i) - y_i)^2, \tag{11}$$

With respect to the inputs $x_i$, C($\mathbf{a}$) is a highly nonlinear function. However, with respect to the parameters, $\mathbf{a}$ is a quadratic function where the minimum can be computed directly by solving the normal equation of the over-determined linear system $P\mathbf{a} = y$. The already derived, widely known and used Equation (12) is employed for this purpose:

$$\mathbf{a} = \left(P^T P\right)^{-1} P^T \mathbf{y}, \tag{12}$$

where $P$ is a $N$ by the $N_t$ matrix containing the numerical values obtained when evaluating each polynomial feature $p_i(\mathbf{x}), i \in [1, N_t]$ for each data sample $\mathbf{x}_i, i \in [1, N]$.

Although all the steps above require only algebraic operations, since the proposed encoding scheme does not allow for the division of two numbers, we cannot completely compute $\left(P^T P\right)^{-1}$ in the encoded–encrypted form. Therefore, we only compute the dot products $P^T P$ and $P^T \mathbf{y}$ in the encrypted form and then perform the remaining operations after decryption. This is a notable limitation that prevents the model fitting process to be performed completely on encrypted data. However, typically, the number of data samples $N$ is much larger than the number of model parameters $N_t$; therefore, the dot product $P^T P$ results in a small matrix $N_t$ by $N_t$, while $P^T \mathbf{y}$ results in a vector of size $N_t$. Since the input data are significantly regressed by these operations, decrypting does not expose the original data.

Referring to overfitting issues, they typically appear when the model is too complex. In our case, if the number of polynomial terms is not too large, the model does not attempt to learn or explain the random error or noise present in the data. Furthermore, overfitting can be addressed by using larger training datasets: the more samples are used for training, the more difficult it is for the model to learn the error or noise. Thus, according to [29], the number of samples should be at least 10–15 for each term in the model to avoid overfitting. In our use cases, this condition is met in all experiments, as the number of samples exceeds the required threshold (e.g., 9200 training samples for 80 features in the seizure detection use case).

For those experiments where EEG signals are used in the unencrypted form, we also applied SVM methods, as they previously delivered state-of-the-art results in studies employing EEG information [30,31]. Specifically, we used nuSVR from the scikit-learn library [32] on plaintext data and compared the results with the performance obtained by applying polynomial regression on both plaintext and encrypted data. We performed a grid search for the parameters of the SVM classifier ($nu$, which controls the number of support vectors, and the penalty parameter $C$) to determine their optimal values. The parameters obtained are reported in the Results section.

The matrix multiplication operation was parallelized to reduced the computational time (using the Python multiprocessing package [33]). The elements of the resulting matrix were computed by distributing the corresponding rows and columns of the input matrices across multiple processes. All experiments were run on a machine equipped with an Intel Core i7 CPU running at 4.2 GHz and 32 GB RAM.

### *2.3. Use Case 1—Seizure Detection*

The objective in this use case is to determine if an epileptic seizure activity occurred during an EEG recording. An input sample contains a sequence of real numbers representing an EEG signal, which can stem from surface recordings from healthy individuals (with their eyes closed or open) or intracranial recordings from epilepsy patients (during the seizure-free interval or epileptic seizures) from different brain areas. A 128-channel amplifier system was used to record all EEG signals [34].

The original dataset [34] contains 500 files, each of them corresponding to 23.6 s of EEG recording of one individual, at a sampling rate of 173.61 Hz. Each file represents a single-channel EEG segment that was selected from continuous multichannel EEG recordings. Each time sequence contains 4097 EEG values equally sampled in time. We used a restructured version of this dataset from [35], where every recording was split into 23 chunks corresponding to 1 s of recording, consisting of 178 data points each. The resulting dataset contains 11,500 samples. The models were evaluated using five-fold cross-validation: the dataset was divided into five folds, each with 2300 samples, which led to 9200 training samples and 2300 testing samples for each experiment. A ground truth label is associated with each EEG recording: 1—epileptic seizure activity, 2—tumor area, 3—healthy brain area of a patient with an identified tumor, 4—the patient had their eyes closed during recording, and 5—the patient had their eyes open during recording.

We have formulated a binary classification problem by considering that all recordings labeled as 2, 3, 4 or 5 indicate the absence of an epileptic seizure (ground truth label 0), while ground truth label 1 indicates the presence of an epileptic seizure. The resulting dataset is imbalanced since only 20% of the recordings have a class label of 1 (epileptic seizure recorded). As our experiments employ a linear model, the class imbalance is handled by setting the threshold used to discretize the output, closer to 0. Hence, all samples with an output value greater than 0.1 are classified as class 1. Input data are normalized in the range $[-1, 1]$.

Because in the use case described above all the features of an input sample represent a time sequence, the original signal of each entry was downsampled through interpolation to reduce the complexity of the problem. We used one-dimensional linear interpolation (numpy.interp [36]). Normalized cross-correlation (NCC) was computed between the initial and the resampled signal to evaluate the impact of the interpolation operation. Thus, a trade-off was performed between the model complexity and loss of information, and we chose to use the resampled signal with the minimum number of data points and with an NCC value greater than 0.95. This condition led to 40 data points for the downsampled signal (Figure 1).

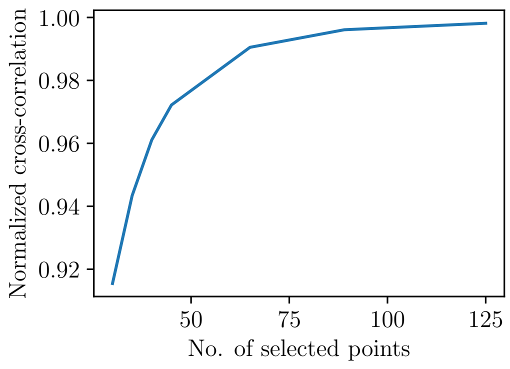

**Figure 1.** NCC values for different versions of downsampled EEG signals.

Considering the restructured version of the dataset, where 178 points correspond to 1 s of the recording, and the downsampled signal (40 points for 1 s), the frequency was reduced from 178 Hz to 40 Hz. A comparison between the original EEG signal and some downsampled versions is displayed in Figure 2.

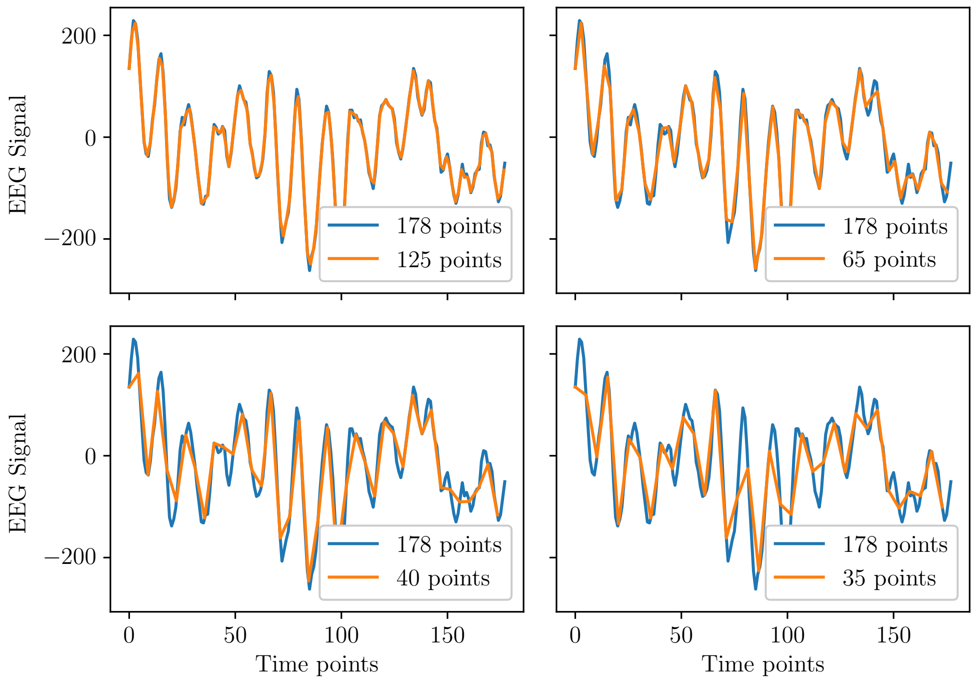

**Figure 2.** Comparison between the downsampled and the original EEG signals.

### 2.4. Use Case 2—Predisposition to Alcoholism

Another use case where EEG signals provide valuable information is the analysis of the predisposition to alcoholism, based on the differences between the frequency bands of alcoholic and non-alcoholic individuals [37]. We used a dataset created by Henri Begleiter (Neurodynamics Laboratory, State University of New York Health Center, Brooklyn), made available at [35]. The data were collected during a study evaluating the correlation between EEG signals and genetic predisposition to alcoholism. The signals were measured by 64 electrodes scanned at 256 Hz (3.9 ms epoch) for 1 s. The sensors were placed on the subjects' scalps according to the International 10–20 standard. The subjects were exposed to one stimulus ($S_1$) or two stimuli ($S_1$ and $S_2$) represented by pictures from the Snodgrass and Vanderwart picture set [38]. The two stimuli were shown in a matched condition ($S_1$ was identical to $S_2$) or in a non-matched condition ($S_1$ differed from $S_2$).

The original dataset is split into two folders—train and test—each of them containing 10 runs for 10 alcoholic and 10 nonalcoholic subjects. The folders consist of trails files with the following columns: trial number, sensor position, sample number, sensor value, subject identifier, matching, channel number, name, and time. In a single trial run, a patient provides 256 samples for each of the 64 electrodes. The pre-processed training dataset contains 153,600 samples of 64 features corresponding to the 64 electrodes; the corresponding ground truth labels are 0 if the subject is non-alcoholic and 1 if the subject is alcoholic. The pre-processed testing dataset also contains 153,600 samples of 64 features and the ground truth label. The input data are normalized in the range $[-1, 1]$.

## 3. Results

### *3.1. Synthetic Data*

To test and evaluate the applicability and limitations of the proposed method, we performed multiple experiments on synthetic, randomly generated data, which were then encrypted with the scheme proposed in [27], as described in Section 2.1. All the experiments were simultaneously performed on plaintext (unencrypted) data and on the encrypted equivalent to ensure that the same result was obtained. The experiments consisted of performing a multivariate linear regression with different data sizes. Specifically, we generated a matrix $X$ containing random data samples in the $[0, 1]$ interval, and a corresponding random output vector $y$. We then computed the regression coefficients as described in Section 2.2.

We analyzed the following metrics as a function of data size: run time, the magnitude of the resulting values, and the number of arithmetic operations performed. The magnitudes of the values and the number of operations are of crucial importance in the HE context. More specifically, the chosen HE scheme (and the vast majority of other HE schemes) only operates on a fixed range of integer values. Performing operations on encrypted (and therefore unknown) values can result in values becoming too large, thus exiting the valid range. Therefore, it is imperative that the magnitude of values encountered throughout the computations can be determined beforehand. Clearly, this cannot be generally determined. However, in this synthetic setup, where the magnitude range of the input values and the total number of operations is known, such an analysis can be performed.

To study the impact of the number of features, we fixed the number of samples (500 samples) and varied the number of features between 2 and 102. To study the influence of the number of samples, we varied it from 15 to 19,915 while maintaining the number of features fixed (two features). For consistency reasons, during this analysis, the precision parameter (i.e., number of terms in the polynomial representation) was set to 20, and a 256-bit encryption key was considered.

Figure 3 displays the values of the chosen metrics as a function of the data size. An increasing number of features has an exponential impact on both the number of operations and on the ciphertext size, while an increasing number of samples appears to affect these metrics linearly. The same applies when analyzing the computational time. A similar conclusion can also be drawn for the maximum magnitude of the polynomial coefficients obtained as a result. Although the computational time is significantly increased, the proposed approach is able to handle data sizes of real-world proportions.

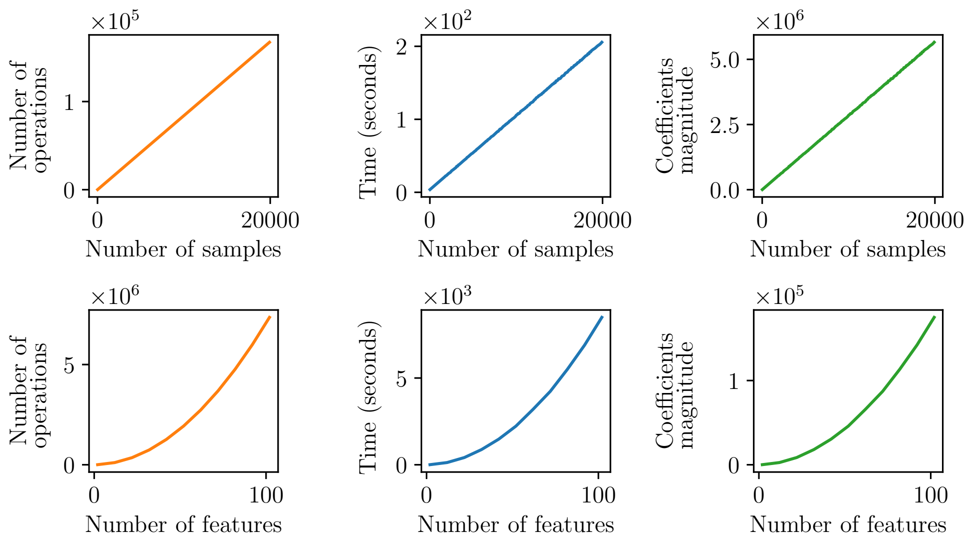

**Figure 3.** Impact of input data size on the total number of arithmetic operations (**left**), total runtime (**mid**) and maximum value (**right**). The total number of data samples was varied while keeping the number of features fixed (**top**) and the number of input features was varied while keeping the number of samples fixed (**bottom**).

To analyze the prediction error, we computed the mean absolute error and the root-mean-square error for all experiments, where the error is defined as the difference between the outputs obtained, using plaintext and ciphertext values. The error was close to zero (in double precision). On the other hand, as this encoding method allows for a choice of the precision parameter, a large value of the error might indicate that this parameter is too small for a certain application. A randomly generated dataset with 1000 samples and 5 features was used for all experiments synthesized in Figure 4. To determine the dependence between the number of terms in the polynomial encoding (the precision parameter), the numerical error and the execution time, respectively, the same experiment was run multiple times, varying the precision parameter from 1 to 30. Figure 4 displays the trade-off between the accuracy and computational time: an increase in the number of terms used for the encoding leads to a significant decrease in error, but with a cost in terms of the computational time.

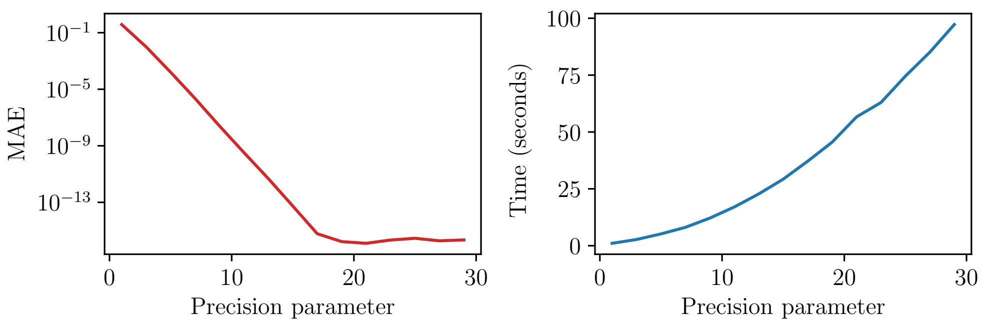

**Figure 4.** Dependence between the number of terms in the polynomial encoding, the numerical error and the execution time, respectively.

### 3.2. Use Case 1—Seizure Detection

We ran multiple experiments for the seizure detection use case, relying on different input data configurations:

- Using all 178 initial data points in each sample as input.
- Using all 178 initial data points in each sample and the 178 corresponding quadratic terms (356 features overall).

- Using a downsampled version of each sample. The number of data points was gradually reduced from 178, and the mean NCC over the entire dataset was computed for each downsampled version. We selected the downsampled version with the smallest number of data points for which the average NCC was still larger than 0.95. The selected downsampled version had 40 features (instead of the initial 178) and the corresponding quadratic terms were added, leading to a total of 80 features per sample. Figure 1 displays the variation of the mean NCC as a function of the number of samples. Figure 2 displays, for one signal, the original sample and the resampled sample for different numbers of data points.

For all three input data configurations, the experiments were first conducted using the plaintext data: both a multivariate polynomial regression and an SVM model were evaluated, using five-fold cross-validation. The optimal values of the SVM parameters were determined by performing a grid search. For this use case, the best results were obtained for $\nu = 0.3$ and $C = 1$. Thus, by setting $\nu = 0.3$, we considered that, at most, 30% of the training samples are allowed to be wrongly classified and at least 30% of them act as support vectors.

Table 1 displays the classification metrics aggregated over all five folds: accuracy, sensitivity, specificity, positive predictive value, negative predicted value. For each of the five folds, the metrics were computed only on the testing subset. While the SVM model is largely independent of the input data configuration, the performance of the polynomial regression model increases significantly both when adding the quadratic terms and when reducing the number of data points per sample (increase in performance stems mainly from an increase in sensitivity). After analyzing the performance obtained using SVM, before and after downsampling the signal, we conclude that this pre-processing step does not influence the classification process. For the last input data configuration (80 features) the polynomial regression model was also trained on ciphertext values, considering the encoding precision parameter equal to 35 and a 256-bit encryption key. The results were identical down to double precision when compared to the plaintext experiment. When choosing the precision parameter for this use case, the synthetic date results were taken into account. As the quadratic resampled dataset has a larger number of features (80 compared to 5) and training samples (9200 compared to 1000), to ensure no loss in performance, the precision parameter was chosen to be equal to 35, which is larger than the values used in the synthetic data experiments displayed in Figure 4.

**Table 1.** Performance comparison when using different input data configurations in the EEG-based seizure detection use case.

| | **Initial Data 178 Features** | | **Quadratic Data 356 Features** | | **Resampled Data 80 Features** | | |
|---|---|---|---|---|---|---|---|
| | **PR** | **SVM** | **PR** | **SVM** | **PR Plaintext** | **PR Ciphertext** | **SVM** |
| Accuracy | 80.59 | 95.78 | 89.22 | 95.86 | 92.26 | 92.26 | 95.85 |
| Sensitivity | 34.25 | 98.82 | 84.40 | 98.60 | 86.56 | 86.56 | 98.70 |
| Specificity | 92.17 | 95.02 | 90.43 | 95.17 | 93.68 | 93.68 | 95.14 |
| PPV | 52.46 | 83.23 | 68.81 | 83.62 | 77.44 | 77.44 | 83.54 |
| NPV | 84.86 | 99.69 | 95.86 | 99.63 | 95.54 | 95.54 | 99.65 |

In terms of computational time, a comparison between the plaintext and ciphertext polynomial regression experiments is displayed in Table 2. For both training and inference, the time increases for the encrypted version. However, the inference time remains in the order of milliseconds. For the parallelized version, four processes were employed.

**Table 2.** Training and inference runtime comparison between plaintext and ciphertext implementation versions for the seizure detection use case.

| | **Plaintext** | **Ciphertext** | **Ciphertext Parallelized** |
|---|---|---|---|
| Training runtime/fold | 0.0119 s | 23.60 h | 7.05 h |
| Inference runtime/sample | $1.73 \times 10^{-6}$ s | 0.012 s | |

Considering the scenario where the model is used only on ciphertext data, without any previous analysis regarding its performance, we chose the precision parameter to be large enough (35 powers) to ensure zero loss in accuracy. This led indeed to the same results as those obtained on plaintext data, but the runtime was significantly larger. To determine the influence of choosing a lower precision parameter, we ran multiple experiments and analyzed the computational time and performance loss. As displayed in Table 3, the runtime can be drastically improved by choosing a smaller number of terms for the encoding step, but after a certain threshold is exceeded (21 powers for this use case), the performance decreases.

**Table 3.** Influence of the precision parameter on the loss of accuracy and on the computational time for the EEG-based seizure detection use case.

| | **Precision Parameter (Number of Terms)** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **35** | **32** | **29** | **26** | **23** | **21** | **20** |
| Loss in accuracy | 0 | 0 | 0 | 0 | 0 | 0 | 1.66 |
| Runtime/fold (hours) | 7.05 | 6.02 | 4.69 | 3.89 | 2.97 | 2.41 | 2.16 |

### *3.3. Use Case 2—Predisposition to Alcoholism*

The evaluation of the second use case was performed on the pre-processed testing dataset. When designing the experiment, we started from the fact that the dataset consists of 153,600 samples of 64 features corresponding to the 64 electrodes, and the ground truth labels. Moreover, we also took into account that the computational time required for an experiment on the ciphertext data increases considerably once the number of samples becomes larger than 10,000. Thus, as the purpose of this paper is not to find a state-of-the-art model, a balanced subset of 10,000 samples from the training dataset was considered for our experiments.

We have designed the following plaintext experiments:

- $E_1$— The machine learning models were trained on the original dataset (153,600 samples and 64 features).
- $E_2$— The machine learning models were trained on 10,000 balanced samples. The number of features was unchanged: 64. This is the baseline experiment to be conducted also on the ciphertext data ($E_1$ cannot be run on ciphertext since the number of samples is too large).
- $E_3$— The machine learning models were trained on the entire dataset (153,600 samples) and all polynomial combinations of features with degrees less than or equal to 2 were considered, leading to a total of 2145 features. The idea is to determine whether increasing the model complexity leads to better performance. Due to the large number of features, it is not feasible to run this experiment on ciphertext data.
- $E_4$— The machine learning models were trained on 10,000 balanced samples with 128 features (the original 64 and their quadratic terms). Considering the computational overhead introduced by the encoding and encryption steps, $E_4$ represents a suitable trade-off solution.

For the plaintext experiments, we also used the SVM approach. The parameters of the SVM classifier that lead to the best performance were obtained through a grid search ($\nu = 0.5$ and $C = 1$). In Table 4, we display the classification results for samples representing individual time points, recorded by the 64 electrodes at a given moment. All reported metrics were computed using only the testing dataset. The columns represent the results of the experiments described above, where the input was normalized in the range $[-1, 1]$. Because in $E_4$, only some of the polynomial combinations of features are used (the original and quadratic features) and the training is performed on only the 10,000 samples subset from $E_2$, the accuracy is increased when compared to $E_1$ (original dataset), and the computational cost is significantly reduced compared to $E_3$. While adding the quadratic terms improves the performance of the polynomial regression algorithm, the SVM algorithm leads to similar performance for $E_2$ and $E_4$.

For the results in Table 5, a voting scheme is applied to determine a single prediction per patient (alcoholic or non-alcoholic): the outputs of all samples corresponding to a patient (which form a time series) are used as input to the voting scheme, where each sample has the same weight.

In Table 6, we display the runtime for both inference and training. As in the first use case, the runtime increases when processing encrypted data, but it still remains reasonably small for the inference step. For the same reasons stated in Section 3.2, a similar analysis regarding the influence of the precision parameter was performed also in this use case. The same decreasing tendency in the runtime can be noticed in Table 7, but the minimum number of terms required for zero loss in performance is smaller than in the previous use case. Although the number of operations is larger, the initial precision of the training data is also reflected in the encoding precision.

**Table 4.** Performance comparison when using different input data configurations in the EEG-based predisposition to alcoholism detection use case.

| | $E_1$ (153,600, 64) | | $E_2$ (10,000, 64) | | $E_3$ (153,600, 2145) | | $E_4$ (10,000, 128) | | |
|---|---|---|---|---|---|---|---|---|---|
| | **PR** | **SVM** | **PR** | **SVM** | **PR** | **SVM** | **PR Plain Text** | **PR Ciphertext** | **SVM** |
| Accuracy | 57.34 | 87.46 | 57.27 | 79.33 | 83.92 | 84.27 | 68.52 | 68.52 | 79.08 |
| Sensitivity | 57.26 | 89.06 | 61.20 | 80.15 | 83.81 | 83.97 | 68.45 | 68.45 | 79.22 |
| Specificity | 57.43 | 85.86 | 53.35 | 78.51 | 84.03 | 84.57 | 68.59 | 68.59 | 78.94 |
| PPV | 57.35 | 86.30 | 56.74 | 78.86 | 84.00 | 84.48 | 68.55 | 68.55 | 79.00 |
| NPV | 58.85 | 88.70 | 57.90 | 79.82 | 83.85 | 84.07 | 68.49 | 68.49 | 79.16 |

**Table 5.** Performance comparison when using different input data configurations in the EEG-based predisposition to alcoholism detection use case after applying the voting scheme.

| | $E_1$ (153,600, 64) | | $E_2$ (10,000, 64) | | $E_3$ (153,600, 2145) | | $E_4$ (10,000, 128) | | |
|---|---|---|---|---|---|---|---|---|---|
| | **PR** | **SVM** | **PR** | **SVM** | **PR** | **SVM** | **PR Plain Text** | **PR Ciphertext** | **SVM** |
| Accuracy | 59.66 | 96.66 | 60.16 | 94.16 | 95.50 | 93.66 | 79.83 | 79.83 | 96.16 |
| Sensitivity | 63.00 | 98.00 | 65.33 | 95.66 | 97.33 | 95.33 | 77.66 | 77.66 | 97.66 |
| Specificity | 56.33 | 95.33 | 55.00 | 92.66 | 93.66 | 92.00 | 82.00 | 82.00 | 94.66 |
| PPV | 50.06 | 95.45 | 59.21 | 92.88 | 93.89 | 92.25 | 81.18 | 81.18 | 94.82 |
| NPV | 60.35 | 97.94 | 61.33 | 95.53 | 97.23 | 95.17 | 78.59 | 78.59 | 97.59 |

**Table 6.** Training and inference runtime comparison between plaintext and ciphertext versions in the EEG-based predisposition to alcoholism detection use case.

| | **Plaintext** | **Ciphertext** | **Ciphertext Parallelized** |
|---|---|---|---|
| Training | 0.0286 s | 59.69 h | 32.28 h |
| Inference | $3.46 \times 10^{-6}$ s | 0.018 s | |

**Table 7.** Influence of the precision parameter on the loss of accuracy and on the computational time in the EEG-based predisposition to alcoholism detection use case.

| | **Precision Parameter (Number of Terms)** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **35** | **32** | **29** | **26** | **23** | **20** | **17** | **14** |
| Loss in accuracy | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.41 |
| Runtime (hours) | 32.28 | 26.53 | 22.17 | 17.91 | 13.02 | 10.90 | 8.16 | 5.75 |

## 4. Discussion and Conclusions

We proposed a method for performing privacy-preserving machine learning-based predictions on homomorphically encrypted medical data. Although there are many proposed HE methods and significant research has been performed on this topic, such methods are still relatively far from being usable in real-world scenarios. Besides existing limitations, such as computational complexity and noise accumulation, typical HE methods are only capable of performing addition and (at most) multiplication on integer values. This significantly limits their usability, especially in the medical field.

A typical usage scenario is when sensitive patient data need to be externalized to a cloud environment for developing machine learning-based models for various prediction tasks. To this extent, we applied the proposed methodology on two real-world use cases involving patient data with different privacy requirements: the prediction of predisposition to alcoholism and the detection of seizures, both using EEG signals. Although the computations on encrypted data are significantly impaired, the prediction performance is maintained at a level comparable to that of standard methods operating on unencrypted data. To achieve this, we proposed a set of methods targeting the inherent limitations in HE computations.

First, we proposed an encoding scheme that allows HE schemes to be applied to real-valued numbers. The encoding scheme relies on transforming real-valued numbers in a sequence of integer values, and for algebraic operations to be performed on the encoded form, using only addition and multiplication. Encoded values can then be encrypted, using existing HE schemes that support addition and multiplication, resulting in a method capable of performing arbitrary algebraic computations on homomorphically encrypted real values.

Another significant difficulty in HE computations is that the numerical values must remain in a predefined (and typically small) interval; exceeding this interval no longer allows for decryption and the information is lost. The initial values that are encrypted can be forced (e.g., through normalization) to be in a certain range; however, values derived from operations on input data can exceed the limit and become unusable. This is a major difficulty, especially in machine learning-based applications, as algorithms in most cases must perform an unknown number of iterations (e.g., training steps, and epochs). To this extent, we employed a polynomial model together with a direct (non-iterative) fitting method such that the required number of operations can be determined a priori. Knowing the input data range, along with the exact number of operations to be performed, allows for a choice of the right encryption scheme along with its parameters for a particular task, resulting in a guaranteed correct outcome.

As for the limitations of the proposed scheme, it is difficult to perform the division of two encoded values. Although several possibilities were explored, this still remains a difficult task to achieve within the current approach. One possibility would be to introduce an additional encoding layer by rewriting a real number as a fraction, and then encoding the numerator and denominator as in (1), but the computational overhead would increase drastically. Furthermore, these supplementary operations do not guarantee a correct result because of the increasing noise that some HE schemes are based on. Following the idea of computing division by multiplying the numerator with the inverse of the denominator, another question that emerges is how can the inverse of an encoded and encrypted number be calculated. One solution might be the Newton–Raphson iterative method for solving an equation. The equation that one would try to solve is $1/X - den = 0$, which provides the inverse of the denominator. The correctness of the result is dependent on the initial approximation. As the underlying information is secret, it is not possible to evaluate which initialization would guarantee convergence to the optimum value in a few iterations. On the other hand, because a comparison cannot be performed between encrypted numbers, a standard/fixed number of iterations needs to be performed. This approach is prone to errors that might occur because of an insufficient number of iterations, or because the plaintext domain is exceeded during the computations.

In this use case, each data point is considered an individual feature (no temporal meaning is considered). Furthermore, although the model achieves good prediction performance, a baseline EEG analysis might be invalidated by downsampling the signal to 40 Hz. As mentioned in [34], a band-pass filter (that allows for frequencies between 0.53 and 40 Hz) was used to filter the signal after data acquisition. According to the Nyquists' theorem and considering the highest frequency of the EEG (40 Hz), to obtain a valid signal, the sampling rate should be at least 80 Hz. The overall performance for the five folds in this experiment (160 features: 80 original features and their squares) on the plaintext data lies between the experiments with 356 and 80 features, respectively. The time required for training on ciphertext data was approximated to be 27 h per fold if 256-bit encryption, 23 encoding terms and 2 processes in the parallelized version were considered.

The computational overhead introduced through encoding and encryption represents another limitation of the method. As experiments have shown, a larger precision parameter and a larger encryption key lead to a higher computational time for the training phase. Regarding the precision parameter, we indicate that the number of terms required to ensure no loss in accuracy cannot be determined precisely based on the dataset dimensionality, as it also depends on the initial precision of the numbers contained within a certain dataset. The total number of operations performed on encoded and encrypted data is $N_{op} = 2sf(f+1)$, where $s$ is the number of training samples and $f$ is the number of features. The computational time can also be predicted according to the number of samples, the number of features and the number of encoding terms $n$ by computing $T = T_{ad}\frac{N_{op}}{2}n + T_{mul}\frac{N_{op}}{2}n^2$, where $T_{ad}$ and $T_{mul}$ denote the average time in seconds required for an addition or a multiplication operation between two encrypted coefficients of the polynomial representation (1). The value of $T$ is also expressed in seconds.

The proposed method can be successfully employed in real-world scenarios. It allows for homomorphic addition, subtraction and multiplication on real numbers, while only requiring an encryption scheme capable of performing the addition and multiplication of small integers. Distributing the information between multiple ciphertext values (polynomial coefficients) allows for the encoding of real numbers of arbitrary sizes, but at the same time, a trade-off between the computational accuracy and the computational time is made.

**Author Contributions:** Conceptualization, A.B.P. and C.I.N.; methodology, A.B.P., I.A.T., A.V., R.D., C.S. and L.M.I.; software, A.B.P. and I.A.T. ; validation, C.I.N., A.V. and L.M.I.; formal analysis, C.I.N.; investigation, A.B.P., I.A.T., C.I.N., A.V. and L.M.I.; resources, L.M.I.; data curation, A.B.P. and I.A.T.; writing—original draft preparation, A.B.P. and I.A.T.; writing—review and editing, C.I.N., A.V., R.D., C.S. and L.M.I.; visualization, A.B.P.; supervision, R.D., C.S. and L.M.I.; project administration, L.M.I.; funding acquisition, L.M.I. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by a grant of the Romanian Ministry of Education and Research, CCCDI—UEFISCDI, project number PN-III-P2-2.1-PED-2019-2415, within PNCDI III. This work was also supported by a grant of the Ministry of Education and Research, CNCS/CCCDI—UEFISCDI, project number PN-III-P3-3.6-H2020-2020-0145/2021. This project received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 875351.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Publicly available datasets were analyzed in this study. Data can be accessed at: https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition (accessed on 21 March 2021) and https://www.kaggle.com/chaditya95/epileptic-seizures-dataset (accessed on 21 March 2021).

**Conflicts of Interest:** Andreea Bianca Popescu and Ioana Antonia Taca received scholarships from Siemens SRL, Brasov, Romania. Cosmin Ioan Nita, Anamaria Vizitiu, Robert Demeter, Constantin Suciu, and Lucian Mihai Itu are employees of Siemens SRL, Brasov, Romania. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## References

1. Miotto, R.; Wang, F.; Wang, S.; Jiang, X.; Dudley, J.T. Deep learning for healthcare: Review, opportunities and challenges. *Briefings Bioinform.* **2018**, *19*, 1236–1246. [CrossRef] [PubMed]
2. Shokri, R.; Shmatikov, V. Privacy-preserving deep learning. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 1310–1321.
3. Mohassel, P.; Zhang, Y. Secureml: A system for scalable privacy-preserving machine learning. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 19–38.
4. McMahan, H.B.; Ramage, D.; Talwar, K.; Zhang, L. Learning differentially private recurrent language models. *arXiv* **2017**, arxiv:1710.06963.
5. Phan, N.; Wang, Y.; Wu, X.; Dou, D. Differential privacy preservation for deep auto-encoders: an application of human behavior prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
6. Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H.B.; Mironov, I.; Talwar, K.; Zhang, L. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 308–318.
7. Orlandi, C.; Piva, A.; Barni, M. Oblivious neural network computing via homomorphic encryption. *EURASIP J. Inf. Secur.* **2007**, *2007*, 1–11. [CrossRef]
8. Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.; Naehrig, M.; Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016; pp. 201–210.
9. Bos, J.W.; Lauter, K.; Loftus, J.; Naehrig, M. Improved security for a ring-based fully homomorphic encryption scheme. In Proceedings of the IMA International Conference on Cryptography and Coding, Oxford, UK, 17–19 December 2013; pp. 45–64.
10. Hesamifard, E.; Takabi, H.; Ghasemi, M. Cryptodl: Deep neural networks over encrypted data. *arXiv* **2017**, arxiv:1711.05189.
11. Fan, J.; Vercauteren, F. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.* **2012**, *2012*, 144.
12. *Microsoft SEAL (Release 3.6)*; Microsoft Research: Redmond, WA, USA, 2020. Available online: https://github.com/Microsoft/SEAL (accessed on 14 April 2021).
13. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; pp. 409–437.
14. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International conference on the theory and applications of cryptographic techniques, Prague, Czech Republic, 2–6 May 1999; pp. 223–238.
15. ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory* **1985**, *31*, 469–472. [CrossRef]
16. Chen, H.; Laine, K.; Player, R.; Xia, Y. High-precision arithmetic in homomorphic encryption. In Proceedings of the Cryptographers' Track at the RSA Conference, San Francisco, CA, USA, 16–20 April 2018; pp. 116–136.
17. Chen, H.; Laine, K.; Player, R. Simple encrypted arithmetic library-SEAL v2. 1. In Proceedings of the International Conference on Financial Cryptography and Data Security, Sliema, Malta, 3–7 April 2017; pp. 3–18.
18. Arita, S.; Nakasato, S. Fully homomorphic encryption for point numbers. In Proceedings of the International Conference on Information Security and Cryptology, Beijing, China, 4–6 November 2016; pp. 253–270.
19. Dowlin, N.; Gilad-Bachrach, R.; Laine, K.; Lauter, K.; Naehrig, M.; Wernsing, J. Manual for using homomorphic encryption for bioinformatics. *Proc. IEEE* **2017**, *105*, 552–567. [CrossRef]
20. Fouque, P.A.; Stern, J.; Wackers, G.J. Cryptocomputing with rationals. In Proceedings of the International Conference on Financial Cryptography, Southampton, Bermuda, 11–14 March 2002; pp. 136–146.
21. Moon, S.; Lee, Y. An efficient encrypted floating-point representation using HEAAN and TFHE. *Secur. Commun. Netw.* **2020**, *2020*, 1250295. [CrossRef]
22. Jäschke, A.; Armknecht, F. (Finite) field work: Choosing the best encoding of numbers for FHE computation. In Proceedings of the International Conference on Cryptology and Network Security, Hong Kong, China, 30 November–2 December 2017; pp. 482–492.

23. Jäschke, A.; Armknecht, F. Accelerating homomorphic computations on rational numbers. In Proceedings of the International Conference on Applied Cryptography and Network Security, London, UK, 19–22 June 2016; pp. 405–423.
24. Lin, C.F.; Shih, S.H.; Zhu, J.D. Chaos based encryption system for encrypting electroencephalogram signals. *J. Med. Syst.* **2014**, *38*, 1–10. [CrossRef] [PubMed]
25. Ahmad, M.; Farooq, O.; Datta, S.; Sohail, S.S.; Vyas, A.L.; Mulvaney, D. Chaos-based encryption of biomedical EEG signals using random quantization technique. In Proceedings of the 2011 4th International Conference on Biomedical Engineering and Informatics (BMEI), Shanghai, China, 15–17 October 2011; Volume 3, pp. 1471–1475.
26. Liu, Y.; Huang, H.; Xiao, F.; Malekian, R.; Wang, W. Classification and recognition of encrypted EEG data based on neural network. *J. Inf. Secur. Appl.* **2020**, *54*, 102567. [CrossRef]
27. Xiang, G.; Chen, X.; Zhu, P.; Ma, J. A method of homomorphic encryption. *Wuhan Univ. J. Nat. Sci.* **2006**, *11*, 181–184.
28. Sathya, S.S.; Vepakomma, P.; Raskar, R.; Ramachandra, R.; Bhattacharya, S. A review of homomorphic encryption libraries for secure computation. *arXiv* **2018**, arxiv:1812.02428.
29. Babyak, M.A. What you see may not be what you get: A brief, nontechnical introduction to overfitting in regression-type models. *Psychosom. Med.* **2004**, *66*, 411–421. [PubMed]
30. Alomari, M.H.; AbuBaker, A.; Turani, A.; Baniyounes, A.M.; Manasreh, A. EEG mouse: A machine learning-based brain computer interface. *Int. J. Adv. Comput. Sci. Appl.* **2014**, *5*, 193–198.
31. Shenoy, H.V.; Vinod, A.P.; Guan, C. Shrinkage estimator based regularization for EEG motor imagery classification. In Proceedings of the 2015 10th International Conference on Information, Communications and Signal Processing (ICICS), Singapore, 2–4 December 2015; pp. 1–5.
32. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
33. Van Rossum, G.; Drake, F.L. *Python 3 Reference Manual*; CreateSpace: Scotts Valley, CA, USA, 2009.
34. Andrzejak, R.G.; Lehnertz, K.; Mormann, F.; Rieke, C.; David, P.; Elger, C.E. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Phys. Rev. E* **2001**, *64*, 061907. [CrossRef] [PubMed]
35. Dua, D.; Graff, C. UCI Machine Learning Repository. 2017. Available online: http://archive.ics.uci.edu/ml (accessed on 21 March 2021).
36. Bressert, E. *SciPy and NumPy: An Overview for Developers*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2012.
37. Sun, Y.; Ye, N.; Xu, X. EEG analysis of alcoholics and controls based on feature extraction. In Proceedings of the 2006 8th International Conference on Signal Processing, Guilin, China, 16–20 November 2006; Volume 1.
38. Snodgrass, J.G.; Vanderwart, M. A standardized set of 260 pictures: Norms for name agreement, image agreement, familiarity, and visual complexity. *J. Exp. Psychol. Hum. Learn. Mem.* **1980**, *6*, 174. [CrossRef]