information

MDPI

Article

Privacy Preservation of Data-Driven Models in Smart Grids Using Homomorphic Encryption

Dabeeruddin Syed [1,*], Shady S. Refaat [2] and Othmane Bouhali [3,4]

[1] Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843, USA
[2] Department of Electrical and Computer Engineering, Texas A&M University at Qatar, Education City, Doha 23874, Qatar; shady.khalil@qatar.tamu.edu
[3] Research Computing, Texas A&M University at Qatar, Education City, Doha 23874, Qatar; othmane.bouhali@qatar.tamu.edu
[4] Qatar Computing Research Institute, Hamad Bin Khalifa University, Doha 5825, Qatar
* Correspondence: dsyed@tamu.edu

Received: 15 May 2020; Accepted: 3 July 2020; Published: 8 July 2020

Abstract: Deep learning models have been applied for varied electrical applications in smart grids with a high degree of reliability and accuracy. The development of deep learning models requires the historical data collected from several electric utilities during the training of the models. The lack of historical data for training and testing of developed models, considering security and privacy policy restrictions, is considered one of the greatest challenges to machine learning-based techniques. The paper proposes the use of homomorphic encryption, which enables the possibility of training the deep learning and classical machine learning models whilst preserving the privacy and security of the data. The proposed methodology is tested for applications of fault identification and localization, and load forecasting in smart grids. The results for fault localization show that the classification accuracy of the proposed privacy-preserving deep learning model while using homomorphic encryption is 97–98%, which is close to 98–99% classification accuracy of the model on plain data. Additionally, for load forecasting application, the results show that RMSE using the homomorphic encryption model is 0.0352 MWh while RMSE without application of encryption in modeling is around 0.0248 MWh.

Keywords: deep learning; homomorphic encryption; fault localization; smart grids; deep neural networks

1. Introduction

The information layer embedded in the two-way communication electric smart grid has various data sources, such as Advanced Metering Infrastructure (AMI), smart electrical measurement sensors, smart meters, detectors, Phasor Measurement Units (PMU), Remote Terminal Unit (RTU), Supervisory Control, and Data Acquisition System (SCADA), etc. [1]. These sources generate huge amounts of data that satisfy the volume, velocity, variety, veracity, and value characteristics of big data [2]. The data that are generated from multiple sources are located at different nodes of an electric network. However, electric utilities demand to preserve the security and privacy of the collected and used data. The collected data represent the behavior of the customers and other network players. Those data contain sensitive information about customers, electricity consumption, trading, and operation of electricity distribution networks [3]. Those data can be utilized to develop machine learning algorithms to improve the electrical utility operation in terms of demand response, peak load shaving, fault analysis, etc.

Machine learning algorithms can be classified into two divisions; classical and deep learning techniques [4]. There are merits and demerits of each division as compared to the other. Deep

learning techniques have the merits of being more accurate, scalable with more data, no requirement of complex feature engineering, adaptability, and transferability to different domains and applications through transfer learning, etc. [5]. The classical machine learning techniques have the merits of higher performance if the size of available data is small, lower requirements of computational resources, simplicity, etc. [6]. However, all of the machine learning approaches require the data that are collected for processing from several nodes in the electric network. The data are transmitted from the electrical nodes to a unified central system to develop the data-driven models. This data transmission and collection by model developers endanger the privacy and security of the information. Hence, security approaches, such as homomorphic encryption and federated machine learning, which avoid the movement of plain data can be considered as potential solutions for securing the collected data. In recent years, many researchers have attained noteworthy progress in the field of encryption schemes.

In the past decade, the rate of generation of energy data from utilities has increased exponentially. The number of smart meters installed globally is around 660 million in 2017 with the data generated at a rate of 280 petabytes a year [7,8]. Furthermore, the number of people who have access to the data has increased multi-fold. For business benefits, each of the teams of an entity has data scientists in-house who develop machine learning models to recognize the patterns in the data and discover ways to improve business. This brings the security of the confidential information in the collected data at risk and may lead to sensitive information disclosure. It has been impartially assumed for decades that the encryption garbles up the numbers in the data and no practical mathematical operations can be executed on the encrypted data [9]. One of the earliest approaches to investigate the possibility of performing computations on the underlying data without actually having access to the unencrypted forms of data was presented by Rivest, Adleman, and Dertouzos in 1978 through their work entitled privacy homomorphisms [10]. In 1996, Ajtai et al. proposed the use of lattice-based constructions as an encryption scheme to develop a public-key cryptosystem [11]. The foundation of these works has set in motion the advancement of first working Fully Homomorphic Encryption (FHE) in 2009 [12]. The FHE scheme allows for the mathematical operations and function mapping to be employed on encrypted data.

The HE schemes have been applied with classical machine learning models initially and later implemented with deep learning models. Earlier works on encryption schemes with deep learning models presented works with activation function as high degree polynomials whose computations are slow and require huge resources. The possibility of low degree non-linear polynomials (such as square function, etc.) as activation functions in the network layers was presented in [13]. Additionally, HE schemes allow the execution of non-polynomial functions on encrypted data. Later, bootstrapping procedure and the use of sign function were introduced in later work with the aim to keep the scheme complexity linear to the depth of the neural networks [14]. These developments enable the possibility of use of HE schemes in the real-world scenarios for encryption.

In this work, the homomorphic encryption model is applied for two smart grid case studies. namely: fault identification and localization data from the simulated IEEE 68 bus system in order to develop a highly secure and accurate deep learning model for fault localization predictions, and real distribution transformer energy consumption data to develop a machine learning model for load forecasts. The main contributions of this work are classified into the following main categories:

1. Designing a secure and privacy-preserving deep neural network (DNN) model established on homomorphic encryption for smart grid applications.
2. Proposing a fault classification and localization model that has favorable accuracy when considering the fact that the model is trained on encrypted data. The encryption of the data before modeling accentuates the need for data privacy and security. The accuracy of the model trained on encrypted data is very close to the model trained on non-encrypted data.
3. Proposing a load forecasting regression model such that the accuracy of the predictions using encrypted data is close to accuracy using non-encrypted data.

The remaining of this article is divided into the following: Section 2 reviews the background work on the security and privacy pillars needed for developing secure machine learning models. Section 2 also presents the literature and the research articles in the discipline of secure deep learning models. Section 3 illustrates the proposed methodology on the homomorphic encryption-based machine learning modeling for fault classification in electrical grids and paillier-based encryption scheme for energy demand forecasting in distribution grids. The performance of the models proposed is validated in Section 4 with extensive experiments and, finally, the conclusions are presented in Section 5.

2. Background & Related Work

In this section, we first present the security and privacy foundations required for secure machine learning. Subsequently, the possible solutions to achieve the security foundations are discussed in the subsequent parts. At last, we discuss the literature review and previous works in the field of privacy-preservation of machine learning models while using homomorphic encryption.

The computational research community has been interested in privacy-preserving machine learning to ensure that the data remains secure during all the stages of its processing i.e., from the training stage to the predictions stage. There are four main pillars needed to achieve privacy-preserving machine learning; namely, training data privacy, model input privacy, model weights privacy, and model output privacy. They are shown in Figure 1 and summarized, as next.

- Training Data Privacy: the privacy and security layer should be in place that no malicious agent can reverse engineer the training data from the model or the output.
- Model Input Privacy: the input data cannot be obtained by any third party, including the model creator; the input data owner is assumed as different than the model creator.
- Model Weights Privacy: the model parameters and weights cannot be obtained or inferred by a malicious party.
- Model Output Privacy: the output of the model cannot be observable by any third party, except for the owner of the data.

Hence, there are multiple challenges that need to be addressed to achieve the aims of the four pillars of privacy in machine learning, which is the goal of our paper. The following subsections discuss various approaches to privacy-preserving machine learning.

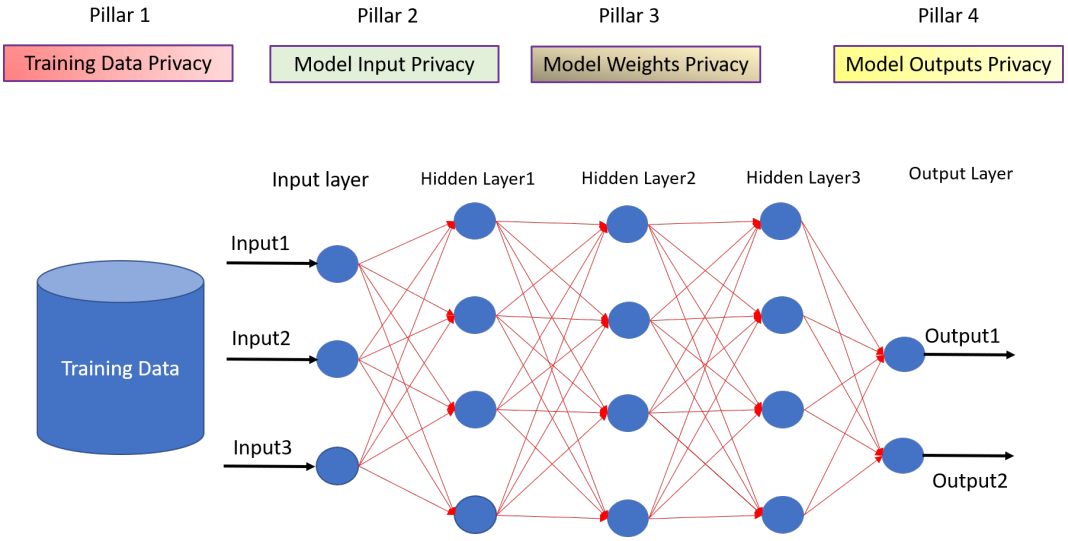

Figure 1. Four Pillars of Privacy-Preserving Machine Learning.

2.1. Training Data Privacy

Although it is difficult to reverse engineer the training data from the model parameters and model outputs, it is not impossible. In [15], the authors have indicated the leakage of information on training

data while using reverse engineering on the machine learning models. The results of their work have indicated that the generative sequence models can retain rare information from the raw data and that this memorization peaks when the test loss is set to the minimum.

There are two major proposed solutions to prevent the memorization of training data in the developed models: Differentially Private Stochastic Gradient Descent (DPSGD) [16] and Private Aggregation of Teacher Ensembles (PATE) [17]. These solutions not only provide security to the data, but also improve the generalizability of the machine learning models.

The memorization and exposure of private knowledge from training data can be reduced with the use of DPSGD. The DPSGD solution adapts differential privacy into SGD to conserve the security of training data while developing deep learning models [16].

PATE is a scalable alternative to DPSGD and it is an ensemble model. The components of the ensemble model train on the independent and identically distributed (i.i.d.) subsets of the same dataset. If most of the models in the ensemble have the same output, then it can be inferred that the models and their output do not expose any secure information from the training data and, hence, can be shared [17].

2.2. Model Input Privacy and Model Output Privacy

Input data and the outputs (predicted variable values) from the data should only be accessible to the owners of the data and protected from other parties, including the model developers. There are three solutions in the literature that have been successfully applied to preserve the model input and output privacy. Those solutions are briefly discussed below:

- Federated Learning (FL) [18]: FL is machine learning on device. It can be made secure with the use of differentially private stochastic gradient descent.
- Homomorphic Encryption (HE) [19]: homomorphic encryption makes provision for the use of non-polynomial functions on encrypted data. With this capability of homomorphic encryption, it is possible to apply classical machine learning algorithms such as linear or logistic regression, naïve bayes, random forest, etc. and deep learning models on encrypted data for training and obtain predictions [20]. However, initial works such as CryptoNets [20] have a limitation of high latency. Additionally, the cryptonets do not support the popular activation functions (Relu and Sigmoid) and the pooling functions (Max Pooling). The limitations of the use of activation functions were later overcome in [21], which approximates the continuous functions of Sigmoid, Relu, and Tanh to lower degree polynomials that are based on Chebyshev polynomials. Their results indicated that the replacement of activation function with approximated lower degree polynomials adopts neural networks to be effectively used with homomorphic encryption.
- Secure Multiparty Computation (MPC) [22]: when multiple parties are involved, they can decide on functions to calculate outputs while using their private inputs. The inputs are not revealed or exposed. The concept of secure MPC has been successfully used in generative models and machine learning algorithms to protect the data from reverse engineering.

2.3. Model Weights Privacy

The model privacy is very crucial for companies who own data to avoid having their AI applications and models easily copied or reverse engineered from inputs and outputs. Hence, model privacy and model weights' privacy are extremely crucial.

The FL, HE, and MPC solutions could also be used to enhance the model privacy. The solutions for model weights privacy include the following:

- Differentially private stochastic gradient descent
- Homomorphic Encryption

2.4. Related Work

Multiple methodologies have been utilized in the discipline of privacy-preserving machine learning [23–25]. These methodologies include, but are not limited to, differential privacy [26], federated machine learning [27], homomorphic encryption [28], multi-party computation [29], etc. In this work, we utilize the homomorphic encryption, as it achieves the four pillars of privacy-preserving machine learning for smart grid applications.

In [21], Hesamifard et al. proposed a framework, called CryptoDL, which aims to preserve the security and privacy of the training data and the classification predictions generated by the Machine Learning (ML) models especially Neural Networks (NN). The authors have applied CryptoDL to different open datasets with encouraging and accurate results. The framework accepts the training data in encrypted form, develop models, and generate classification predictions that are encrypted under the data owner's public key as well. Their experiments yield that the proposed approach of approximating activation functions to low-degree Chebyshev polynomials outperforms other HE methodologies and the developed models preserve the security of data.

In [30], Shokri et al. designed a privacy system that allows multiple parties to train the deep learning models on their private data and share the knowledge from their learned models. The sharing of the models improves the overall accuracy of the final averaged model while also preserving the data privacy, because individual data is not shared between the multiple parties. Their system is based on Distributed Selective Stochastic Gradient Descent (DSSGD) and parameter exchange protocol for different parameters of the deep learning model.

In [9], the authors developed the classification models using decision trees, hyperplane decision, and naïve bayes classifiers on encrypted data while using security constraints. However, the accuracy of their classification models has not been reported in the article. The efficiency of the models is evaluated in terms of classification time and they reported that their models take a few milliseconds to a few seconds on large datasets.

A stable and significant implementation of fully homomorphic encryption was presented in [12]. The author presented a modular framework of FHE that supports the computations to a fixed depth. Subsequently, he employed a bootstrap method to enhance the framework with successful computations to a larger depth with a few constraints or assumptions. The decryption works correctly if the noise in the encrypted text is small. For example, $c_1 \rightarrow Enc(p_1)$ and $c_2 \rightarrow Enc(p_2)$ are cipher texts and these encrypted data have noise n_1, n_2 . Subsequently, during addition and multiplication operations, these noise values increase to $n_1 + n_2$ and $n_1 \cdot n_2$, respectively. Hence, if the noise is significant after the encryption of data, this methodology performs well only in shallow networks. The technique of bootstrapping is used to reduce the noise in the network after encryption as it refreshes the cipher data. Consider that $c_1 \in Enc(p_1)$ has large noise and a helper cipher data $c_k \rightarrow Enc(secretkey)$. A homomorphic evaluation of the decrypting function is performed as $Eval(Dec, Enc(c_k), c_1)$ which refreshes the cipher data, which encrypts $Dec(c_k, secretkey) = p_1$. This bootstrapping technique is used to refresh the cipher data and removes the noise in it. The cipher data are first calculated up to depth 'd', which is efficient for fully homomorphic encryption. Afterwards, bootstrapping is used to refresh the cipher data that are devoid of noise. Finally, the encryption is performed starting from depth 'd' to depth '2d', and bootstrapping is then applied at depth '2d' and so on. Dijk et al. [31] simplified the complexity of Gentry's somewhat homomorphical bootstrappable encryption scheme and applied it over the integers.

There are a few technical challenges that are associated with the privacy-preservation of data [32]. These are listed, as below:

- The data might come from multiple sources that use different secret keys for encryption. In such a case, the homomorphic encryption is not straight-forward. The solution to this will be MPC and HE.

- Additionally, developing deep learning models with homomorphic encryption may be hard when compared to developing shallow models. However, in this work, the developed deep learning models display high accuracy on testing.

There are multiple types of schemes in HE depending on the operations that they can perform. The HE encryption schemes that can perform only one type of operation are called partially homographic encryption schemes. El Gamal scheme [33] and RSA scheme [34] are partially HE schemes, and these can perform only multiplication operations homographically whereas Paillier scheme [35] can perform addition operations homographically.

MPC is similar to homomorphic encryption with an exception that the two users involved X (user with data d) and Y (developer of function f) are required to interact over multiple iterations to train the model f(d). For specific applications, MPC has proven to perform better than Homomorphic encryption. However, MPC faces the difficulties of high bandwidth requirements and network latencies. Hence, HE is more scalable for generalized smart grid applications.

In practice, there are multiple open-source implementations of homomorphic encryption schemes. These libraries are described in Table 1.

Table 1. Open-source implementations of HE.

Implementation	Description
HELib [36]	the low-level library implements HE with faster evaluation time employing optimized Brakerski-Gentry-Vaikuntanathan (BGV) scheme with bootstrapping.
cuHE [37]	highly optimized GPU-accelerated library for Homomorphic encryption.
TFHE [38]	open source gate-by-gate bootstrapping library which evaluates homomorphic encryption of binary gates, negation, and MUX gate operations and performs computation over encrypted data.
NFLlib [39]	open-source Number Theoretic Transform based Fast Lattice Library which uses low-level processor functionalities.
SEAL [40]	an extensively employed open-source library from Microsoft that supports BGV and Cheon, Kim, Kim, and Song (CKKS) encryption schemes.
$\Lambda o \lambda$ [41]	open-source Haskell library for functional lattice-based cryptography.
PALISADE [42]	open-source library for implementations of lattice-based encryption building blocks and HE scheme.
HeaAN [43]	open-source implementation of HE encryption scheme using approximate arithmetic of numbers.
Lattigo [44]	library for lattice-based cryptography and MPC, written in Golang (Go) language.

Our researched system attains significant security objectives in the context of deep learning models applied for smart grid applications. It secures the training data before it is transferred to the model developers. The model developers are enabled in order to control the learning objectives of the data-driven deep learning models. The solution allows for the application of machine learning and deep learning models along with generalization to a wide variety of electrical applications.

3. Proposed Methodology for Privacy-Preservation of Data-Driven Models

This section presents the proposed methodology for the application of deep learning over encrypted data with the homomorphic encryption method. The main objective of this methodology is to add additional security and privacy layer to deep learning models in smart grid electrical applications. As observed in Figure 2, the encryption keys are used to encrypt the data on the server-side. Only the encrypted data is communicated to the client-side or the model developers. The model is trained over the encrypted data, and then the predictions are obtained. Moreover, the predictions are encrypted which requires decryption performed at the server side to obtain the final predictions. It is important

to note that the client or model developer does not have access to the encryption keys; therefore, the client can neither decrypt the data nor decrypt the predictions. In some applications, the client is provided with a previously trained model and encrypted test data to provide encrypted predictions back to the server. In such cases, the previously trained model may have been trained on plain data on the server-side. The homomorphic encryption allows for the privacy, reliability, and security of training data, model inputs, weights, and outputs. However, the HE does not secure the data from the point of generation to the server, but only after it reaches the server. The framework assumes that the smart grid platform supports secure data acquisition from different data sources. The diverse storage services and policies should enforce privacy and security between the points of data generation and server.

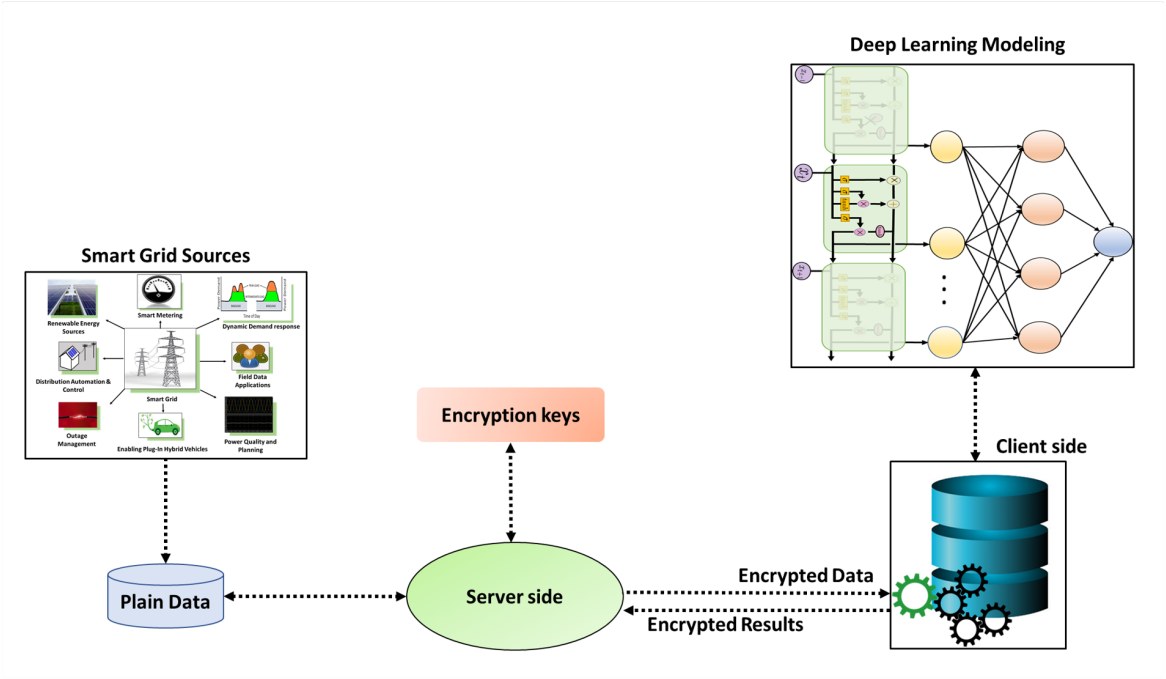

Figure 2. Homomorphic encryption-based deep learning modeling for smart grid applications.

In the following sections, the main steps of the proposed methodology are described.

3.1. Data Description

Two datasets have been collected for evaluating the performance of machine learning models employing homomorphic encryption. One of the datasets is the simulation data for fault localization in a power system network and another dataset is the time-series load demand data at the distribution grid level (transformers). The following section provides the description of the two datasets and the pre-processing steps employed on the datasets.

3.1.1. Dataset 1

The data utilized are the fault localization simulation data from the IEEE 68 bus system, which consist of 68 buses, 16 machines, 16 generators, and 20 transformers [45,46]. Figure 3 shows the reduced-order 68-bus system test simulation model. The simulation data is PMU measurements acquired for pre and during fault conditions for a subset of the grid buses in the system. The extracted features that characterize the location and occurrence of the faults are determined to be the bus voltage variations before and during the faults. The final feature set that is given as input to the classifier is given by the imaginary part of the following Equation (1).

$$\Psi = Y^0 \Delta V \tag{1}$$

where Y^0 denotes the admittance matrix of the bus system before the faults and ΔV represents the difference in the bus voltage before and during the faults.

The label of the data indicates the line number at which the fault occurs and, hence, there are 86 classes. 75% of the data are utilized as a training data set and 25% of the data are utilized as a validation data set.

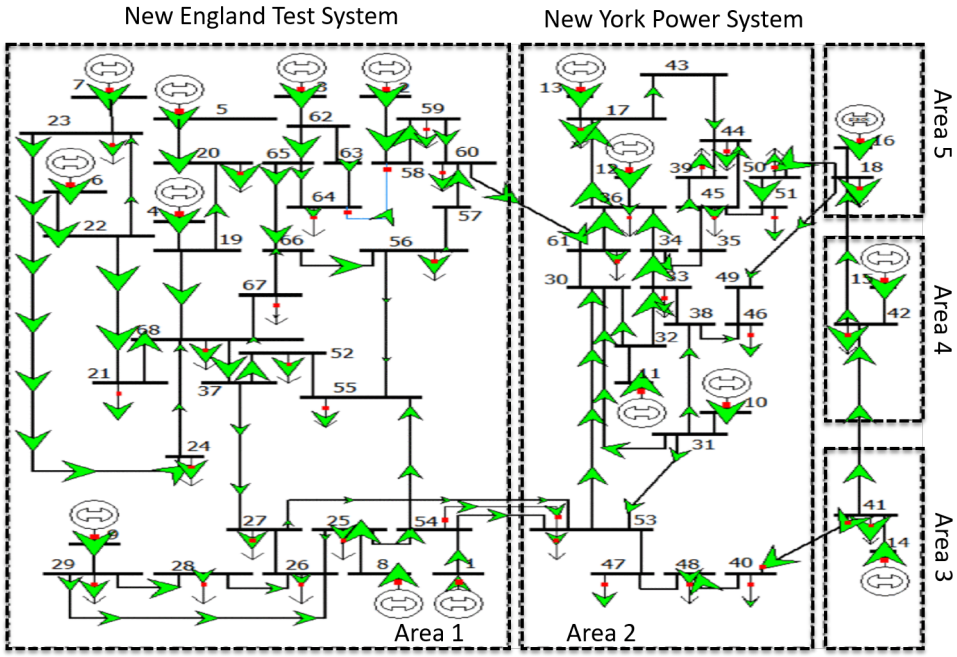

Figure 3. IEEE 68-bus system test simulation model.

3.1.2. Dataset 2

The second case study is on the load forecasting at the distribution level. The data acquired are the time series hourly load demand data at the distribution transformers level for a real power grid between January 2018 to December 2018. After data acquisition, the only features were time stamp, transformer id, season, and hourly load demand values. In the pre-processing steps, the weather data is extracted and added as features to this dataset using a freemium Application Programming Interface (API) called Darksky [47]. The extracted features include maximum temperature, minimum temperature, cloud cover, dew point, humidity, precipitation intensity, pressure, Ultraviolet rays (UV) index, visibility, wind gust, wind speed, etc. This extraction introduces missing values that are filled using an average of forward and backward fill for numerical features [48]. Additionally, the 24 lag hour values are fed back to the dataset as additional features.

3.2. Homomorphic Encryption

The fully homomorphic encryption or simply homomorphic encryption is a category of encryption methodology that differs from other classical methods in a way that enables the computations to be executed on the encrypted data without having a requirement to access the secret encryption key [49]. The output of such computations is also encrypted, and it can be decrypted with the help of a secret key that the data owner possesses. A function $\mho$: $\mathbb{R}_1 \longrightarrow \mathbb{R}_2$ is said to be additive and multiplicative homomorphic, if for every $r_1, r_2 \in \mathbb{R}_1$, it implies that $\mho(r_1 + r_2) = \mho(r_1) \oplus \mho(r_2)$ and $\mho(r_1 \cdot r_2) = \mho(r_1) \otimes \mho(r_2)$, respectively, where $\oplus$ and $\otimes$ are the operations in $\mathbb{R}_2$ [50].

A homomorphic encryption technique has a supplemental algorithm property, called *Eval*, which can be executed or computed over encrypted data. Any party can run *Eval* function on the encrypted data without requiring access to the private key with which data are initially encrypted.

That is, the ciphertext need not be decrypted to allow computations on it in the evaluation function, and this maintains the privacy of the underlying information in the encrypted data. Figure 4 illustrates the different mapping functions and computations that are involved in Homomorphic Encryption Evaluation.

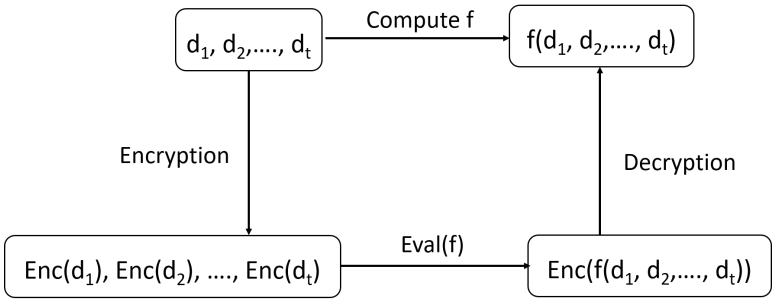

Figure 4. Homomorphic Encryption Evaluation.

The homomorphic evaluation of a deep neural network with discretized inputs and weights involves two steps mainly. These steps are enumerated, as following:

1. Computation of multisum between the encrypted inputs that are given to the neurons and the discretized weights at the respective neurons: the calculation of multisum uses homomorphic addition as basic operation.
2. Extraction of the sign of the output at each neuron.

To make the neural network scalable in terms of the number of layers, the bootstrapping operation is performed at every neuron in the layers to effectively encrypt the sign of the output and use it in further computations to the next layers in the network.

For privacy-preserving machine learning, we use the orthogonal matrix transformations-based homomorphic encryption in the classification case study. The secret key of the proposed matrix transformations-based HE scheme is an invertible matrix (U1) of size m, where m is the number of records in training data. Additionally, Paillier cryptosystem is utilized for the case study of regression.

Encryption of the Data

The data are encrypted using HE algorithm based on matrix transformations. The aim of the encryption of the data is to generate highly accurate deep learning or machine learning models and use the encrypted data for training or testing, so that the model developers do not have access to the plain data all of the time.

In the first case study for fault localization, two sets of simulations have been conducted. In the first set, the DNN model is developed when the training data is plain, and the model developer has access to the plain data. Here, there are no privacy-preserving techniques involved. In the second set of experiments, the first stage encrypts the fault identification and localization data using homomorphic encryption on the source side. The data provided to the model developers are the encrypted version of data. Accordingly, no information from the data can be leaked. The model developers develop the DNN model while using the encrypted data and return the encrypted predictions to the source side. The data owner then uses the keys to decrypt the encrypted predictions to obtain the final predictions. These predictions are used to calculate the accuracy of the developed models using ground truth values.

Algorithms 1–4 present the encryption and decryption steps that are used on the server-side in the fault localization case study.

Algorithm 1 Encryption of Train Data

function ENCRYPTIONTRAIN($(X)_{mxn}$, $(y)_{mx1}$)
  Choose $(Q_1)_{mxm}$ as orthogonal matrix.
  if $n > 1$ **then**
    Choose orthogonal Q_2 of size n
  else
    Choose orthogonal Q_2 of size 1
  end if
  $(X)_{encrypted} = Q_1^{\intercal} \cdot X \cdot Q_2$
  $(y)_{encrypted} = Q_1^{\intercal} \cdot y$
  return $(X)_{encrypted}, (y)_{encrypted}, Q_1, Q_2$
end function

Algorithm 2 Decryption of Train Data

function DECRYPTIONTRAIN(X, y, Q_1, Q_2)
  Choose $(Q_1)_{mxm}$ as orthogonal matrix.
  $X_{decrypted} = Q_1^{\intercal} \cdot X \cdot {Q_2}^{-1}$
  $y_{decrypted} = Q_1^{\intercal} \cdot y$
  return $X_{decrypted}, y_{decrypted}$
end function

Algorithm 3 Encryption of Test Data

function ENCRYPTIONTEST(X, Q_2)
  Choose $(Q_1)_{mxm}$ as orthogonal matrix.
  if $noOfRows(X) > 1$ **then**
    Choose orthogonal Q_3 of size noOfRows(X)
  else
    Choose orthogonal Q_2 of size 1
  end if
  $(X)_{encrypted} = Q_3 \cdot X \cdot {Q_2}^{-1}$
  return $(X)_{encrypted}, Q_3$
end function

Algorithm 4 Decryption of Test Data

function DECRYPTIONTEST($y_{encrypted}$, Q_3)
  $y_{decrypted} = {Q_3}^{-1} \cdot y_{encrypted}$
  return $y_{decrypted}$
end function

In the second case study for load demand forecasting, a machine learning model is developed while using the plain data where the model developers have initial access to data. The knowledge gained i.e., model and its parameters can be transferred to points where load demand predictions are required. At these points, access to the plain data is not required and it is encrypted during the training phase. In this work, a paillier encryption scheme [35] is employed and public, and private key pair is generated. The test data is encrypted using a public key and the predictions are generated using the model on encrypted testing data. The load forecast values are encrypted and these can only be decrypted with the help of private key which is available only to the data owner.

The homomorphic encryption can secure a multitude of electrical applications in smart grids by providing the following advantages:

1. Securing data stored in cloud server.
2. Enabling data analytics in regulated electric utilities.
3. HE protects the systems against eavesdropping attacks after the data has reached the server. The HE potentially renders any data leaked through eavesdropping attacks or man-in-the-middle attacks indecipherable to attackers.
4. HE can protect the data against unauthorized sharing.

4. Results

This section presents the efficacy of the proposed homomorphic encryption-based deep learning models for fault localization and detection. Additionally, the results of another case study of Paillier scheme for load demand predictions are presented.

The experiments have been conducted utilizing Python 3.7 for encryption and decryption schemes. The execution was run on a computer with an Intel Core i7-7700HQ CPU @ 2.80 GHz with eight logical processors, four physical cores, and 16 GB of RAM.

4.1. Case Study 1

The machine learning model utilized to train on plain and encrypted data is the deep neural network (DNN). It comprises of three categories of layers—an input layer, an output layer, and hidden layers. The inherent non-linear characteristics of the layers enable the DNNs structure to recognize and generalize the underlying patterns and information in the data.

The size of the input layer of DNN is 64 and the size of the output layer is 86. The number of neurons in each hidden layer is 70 neurons. The activation function utilized in the DNN model trained on plain and encrypted data is hyperbolic tangent (tanh) activation. The input space is scaled to the range of $(-1, 1)$ and the tanh activation function ranges between the same values. It is also maintained that the weights have discrete values over the domain of $\mathbb{Z}$, as real-valued weights render the use of homomorphic encryption incompatible. A dropout percentage of 20% is utilized for the model trained on plain data. A dropout percentage of 50% is utilized for the model trained on encrypted data.

Table 2. Evaluation of homomorphic encryption deep learning model for classification problem (fault localization).

Model Name	DNN	HE + DNN
Accuracy on the validation dataset	98.32	97.71
Accuracy on the test dataset	99.98	98.59
Execution time (s)	20.61	23.64
The computational complexity of activation function	O(1)	O(1)

As shown in Table 2, the performance of the deep learning model on encrypted data is close to the performance of the model on plain data. The validation accuracy of the DNN model on plain data of fault identification and localization is 98.32%. However, when the DNN model is trained on the encrypted data, the validation accuracy is 97.71%. Additionally, a separate fault simulation data is used as a test data set. Additionally, on the test data set, the accuracy of DNN model trained on plain data is 99.98%, whereas the accuracy of the model trained on encrypted test data is 98.59%. Figure 5 shows the graphical representation of the results. Homomorphic encryption-based DNN takes more time, but it effectively protects the data and generates a comparative accurate model.

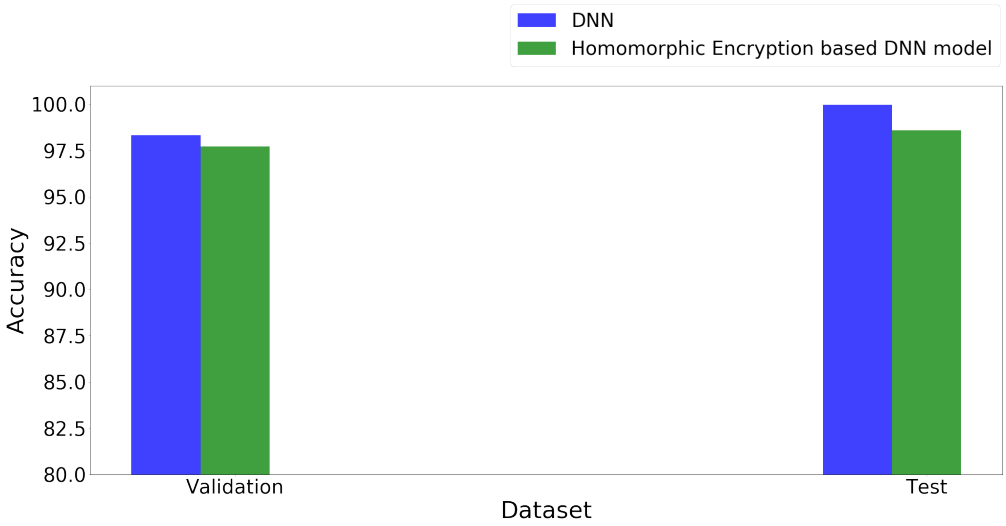

Figure 5. Accuracy-wise performance of the proposed methodology.

4.2. Case Study 2 (a)

The classical machine learning algorithm of linear regression is applied over the Paillier scheme for the case study on load demand predictions. Mini-batch Gradient descent optimizer is used with learning rate 0.01 and training epochs as 1000, as the data are big and the computational memory is limited. Training of the model is performed on unencrypted data which is then transferred to the encrypted testing data to make encrypted predictions. Only the server-side private key can be used to decrypt the predictions. However, a public key can be used to encrypt any data. 80% of data is used for training and 20% of data is used as a validation dataset.

Figure 6 depicts the training history in terms of mean square error against the number of epochs of training. The smoothness in the graph indicates that the cost function of the machine learning model keeps on decreasing over the set number of epochs and the model approaches to convergence with minimum prediction error.

Figure 6. The mean square error as training progresses.

Figure 7 plots the predicted values of load demand in testing data set after decryption against the true values of load demand. A point on the straight line through origin would indicate that there is no error in the predicted value and it is the same as the actual load demand value. Thus, the homomorphic encryption-based modeling was thus evaluated for the regression problem of load demand forecasting. The results of the evaluation are displayed in Table 3. The results indicate that the machine learning

model without encryption has a coefficient of variation (CV) of 7% and the CV is around 10% when encryption is employed.

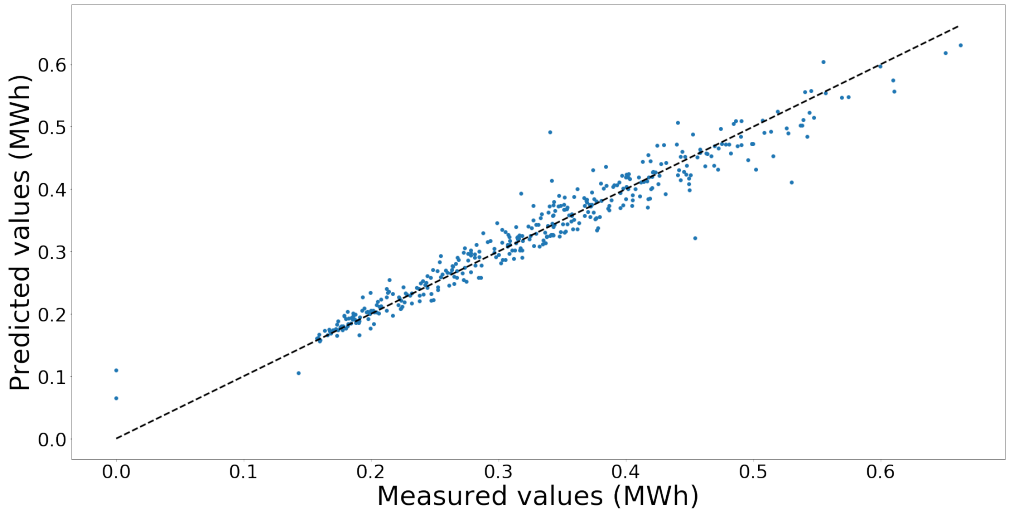

Figure 7. Predicted and actual values of load demand.

Table 3. Evaluation of homomorphic encryption for regression problem (load forecasting).

Model Name	Validation Dataset		Test Dataset	
	RMSE (MWh)	Coefficient of Variation (CV) (%)	RMSE (MWh)	CV (%)
LR	0.0248	7.49	0.0250	7.55
HE + LR	0.0352	10.63	0.0374	11.30

4.3. Case Study 2 (b)

In this set of experiments, a similar methodology, as in Section 4.2, is applied on a different hourly load forecasting scenario with the aim to compare the results of our methodology to existing work. The public data set is provided by PJM Interconnection LLC and consists of hourly energy consumption data recorded by various distribution utilities in the Eastern Interconnection grid in the United States. We utilized the data from the EIC grid for area AECO between the periods of 1 March 2019 and 19 June 2020. The time-series data contains 8713 records. In the feature engineering step, twenty-four lag hour values of energy consumption are added as features to the data set. 90% of the data are employed for training and 10% of data are used as a testing data set.

Table 4 presents the results of our experiments. As observed in the Table 4, the first forecasting experiment is performed without any encryption on the data and this is a baseline to relatively compare the results of encryption-based forecasting.

Table 4. HE for load forecasting problem—PJM dataset.

Model Name	Testing Dataset	
	RMSE (MWh)	Mean Absolute Percentage Error (MAPE) (%)
LR	28.18	2.007
HE+LR	31.65	6.15

As per the results in Table 4, when the encryption-based methodology is applied, MAPE is obtained as 6.15%. For the same data set, without any encryption involved, the load forecasting MAPE is 2.007%. As the security of data is increased using encryption, the forecasting accuracy is definitely decreased. However, we still obtain high accuracy with a high degree of privacy. The decrease in

accuracy, after our proposed methodology is applied, is around 4.15% as against the 8.69% decrease in accuracy, as reported by previous work [51] in the same load forecasting scenario.

5. Conclusions

In this paper, a privacy-preserving deep learning model that is based on homomorphic encryption is presented for classification and a classical machine learning model based on the Paillier scheme for homomorphic encryption is presented for regression. The proposed model can achieve machine learning modeling in the encrypted domain. The simulations of the model on different datasets indicate that the performance on encrypted data is as accurate as modeling on plain data. Application of the proposed methodology in the smart grids data processing is promising and implies a wide range of real-life electrical utility applications. Hence, data privacy can be maintained by the use of encryption and the machine learning models can be trained on encrypted data and still achieve the same accuracy and training. The main contribution of this work is a data-driven method that benefits from the accuracy of classical and deep machine learning algorithms with the lucidity of employing homomorphic encryption. The amalgamation of algorithms from cryptography, machine learning, and electrical engineering has demonstrated a methodology that achieves the accuracy and security of electrical data while maintaining high throughput for computations. However, there is a scope of improvement in terms of computation time using Graphics Processing Units (GPUs). Additionally, more effective encoding paradigms can be employed that require fewer parameters and obtain accelerated computations.

Author Contributions: D.S., Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing—original draft; S.S.R., Funding acquisition, Project administration, Resources, Supervision, Validation, Writing—review & editing; O.B., Project administration, Resources, Supervision, Validation, Writing—review & editing. All authors have read and agreed to the published version of the manuscript.

Funding: This publication was made possible by NPRP grant [NPRP10-0101-170082] from the Qatar National Research Fund (a member of Qatar Foundation), the co-funding by IBERDROLA QSTP LLC and sponsorship by Texas A & M Energy Institute Fellowship.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DL	Deep Learning
DNN	Deep Neural Network
DPSGD	Differentially Private Stochastic Gradient Descent
DNN	Deep Neural Networks
FHE	Fully Homomorphic Encryption
FML	Federated Machine Learning
GPU	Graphics Processing Unit
HE	Homomorphic Encryption
LSTM	Long Short Term Memory
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MPC	Secure Multiparty Computation
NFLlib	Number Theoretic Transform based Fast Lattice Library
NN	Neural Networks
PATE	Private Aggregation of Teacher Ensembles
RMSE	Root Mean Square Error
UV	Ultraviolet

References

1. Tuballa, M.L.; Abundo, M.L. A review of the development of Smart Grid technologies. *Renew. Sustain. Energy Rev.* **2016**, *59*, 710–725.
2. Zhang, Y.; Huang, T.; Bompard, E.F. Big data analytics in smart grids: A review. *Energy Inform.* **2018**, *1*, 8.
3. Efthymiou, C.; Kalogridis, G. Smart grid privacy via anonymization of smart metering data. In Proceedings of the 2010 First IEEE International Conference on Smart Grid Communications, Gaithersburg, MD, USA, 4–6 October 2010; pp. 238–243.
4. Raschka, S.; Mirjalili, V. *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow 2*; Packt Publishing Ltd.: Birmingham, UK, 2019.
5. Li, H. Deep learning for natural language processing: Advantages and challenges. *Natl. Sci. Rev.* **2018**, *5*, 22–24.
6. Gjoreski, H.; Bizjak, J.; Gjoreski, M.; Gams, M. Comparing deep and classical machine learning methods for human activity recognition using wrist accelerometer. In Proceedings of the IJCAI 2016 Workshop on Deep Learning for Artificial Intelligence, New York, NY, USA, 9–15 July 2016; Volume 10.
7. Global Smart Meter Total to Double by 2024 with Asia in the Lead. 2020. Available online: https://www.woodmac.com/news/editorial/global-smart-meter-total-h1-2019/ (accessed on 3 July 2020).
8. Consulting, C. Big Data BlackOut: Are Utilities Powering Up Their Data Analytics? Available online: https://www.capgemini.com/wp-content/uploads/2017/07/big_data_blackout-_are_utilities_powering_up_their_data_analytics.pdf (accessed on 3 July 2020).
9. Bost, R.; Popa, R.A.; Tu, S.; Goldwasser, S. Machine learning classification over encrypted data. In Proceedings of the Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 8–11 February 2015; Volume 4324, p. 4325.
10. Rivest, R.L.; Adleman, L.; Dertouzos, M.L. On data banks and privacy homomorphisms. *Found. Secur. Comput.* **1978**, *4*, 169–180.
11. Ajtai, M.; Dwork, C. A public-key cryptosystem with worst-case/average-case equivalence. In Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, El Paso, TX, USA, 4–6 May 1997; pp. 284–293.
12. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June, 2009; pp. 169–178.
13. 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.
14. Bourse, F.; Minelli, M.; Minihold, M.; Paillier, P. Fast homomorphic evaluation of deep discretized neural networks. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 483–512.
15. Carlini, N.; Liu, C.; Erlingsson, Ú.; Kos, J.; Song, D. The secret sharer: Evaluating and testing unintended memorization in neural networks. In Proceedings of the 28th {USENIX} Security Symposium ({USENIX} Security 19), Santa Clara, CA, USA, 14–16 Augustt, 2019; pp. 267–284.
16. Rajkumar, A.; Agarwal, S. A differentially private stochastic gradient descent algorithm for multiparty classification. In Proceedings of the Artificial Intelligence and Statistics, La Palma, Spain, 21–23 April 2012; pp. 933–941.
17. Papernot, N.; Song, S.; Mironov, I.; Raghunathan, A.; Talwar, K.; Erlingsson, Ú. Scalable private learning with pate. *arXiv* **2018**, arXiv:1802.08908.
18. Smith, V.; Chiang, C.K.; Sanjabi, M.; Talwalkar, A.S. Federated multi-task learning. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 4424–4434.
19. Takabi, H.; Hesamifard, E.; Ghasemi, M. Privacy preserving multi-party machine learning with homomorphic encryption. In Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain, 5–10 December 2016.
20. Gilad-Bachrach, R.; Finley, T.W.; Bilenko, M.; Xie, P. Neural Networks for Encrypted Data. U.S. Patent 9,946,970, 17 April 2018.

21. Hesamifard, E.; Takabi, H.; Ghasemi, M.; Wright, R.N. Privacy-preserving machine learning as a service. *Proc. Priv. Enhancing Technol.* **2018**, *2018*, 123–142.
22. Miyajima, H.; Shigei, N.; Miyajima, H.; Miyanishi, Y.; Kitagami, S.; Shiratori, N. New privacy preserving back propagation learning for secure multiparty computation. *IAENG Int. J. Comput. Sci.* **2016**, *43*, 270–276.
23. Kim, H.; Kim, S.H.; Hwang, J.Y.; Seo, C. Efficient privacy-preserving machine learning for blockchain network. *IEEE Access* **2019**, *7*, 136481–136495.
24. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol. (TIST)* **2019**, *10*, 1–19.
25. Al-Rubaie, M.; Chang, J.M. Privacy-preserving machine learning: Threats and solutions. *IEEE Secur. Priv.* **2019**, *17*, 49–58.
26. 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.
27. Truex, S.; Baracaldo, N.; Anwar, A.; Steinke, T.; Ludwig, H.; Zhang, R.; Zhou, Y. A hybrid approach to privacy-preserving federated learning. In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, Orlando, FL, USA, 13 Novermber 2019; pp. 1–11.
28. Kuri, S.; Hayashi, T.; Omori, T.; Ozawa, S.; Aono, Y.; Wang, L.; Moriai, S. Privacy preserving extreme learning machine using additively homomorphic encryption. In Proceedings of the 2017 IEEE symposium series on computational intelligence (SSCI), Honolulu, HI, USA, 27 November–1 December 2017; pp. 1–8.
29. Ma, X.; Zhang, F.; Chen, X.; Shen, J. Privacy preserving multi-party computation delegation for deep learning in cloud computing. *Inf. Sci.* **2018**, *459*, 103–116.
30. 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.
31. Van Dijk, M.; Gentry, C.; Halevi, S.; Vaikuntanathan, V. Fully homomorphic encryption over the integers. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, France, 30 May–3 June 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 24–43.
32. Brenner, M.; Dai, W.; Halevi, S.; Han, K.; Jalali, A.; Kim, M.; Laine, K.; Malozemoff, A.; Paillier, P.; Polyakov, Y.; et al. *A Standard API for RLWE-based Homomorphic Encryption*; Technical Report; HomomorphicEncryption.org: Redmond, WA, USA, 2017.
33. Tsiounis, Y.; Yung, M. On the security of ElGamal based encryption. In *International Workshop on Public Key Cryptography*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 117–134.
34. Patel, N.; Oza, P.; Agrawal, S. Homomorphic Cryptography and Its Applications in Various Domains. In *International Conference on Innovative Computing and Communications*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 269–278.
35. El Makkaoui, K.; Ezzati, A.; Beni-Hssane, A. Securely adapt a Paillier encryption scheme to protect the data confidentiality in the cloud environment. In Proceedings of the International Conference on Big Data and Advanced Wireless Technologies, Blagoevgrad, Bulgaria, 10–11 November 2016; pp. 1–3.
36. Halevi, S.; Shoup, V. Algorithms in helib. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 554–571.
37. Dai, W.; Sunar, B. cuHE: A homomorphic encryption accelerator library. In Proceedings of the International Conference on Cryptography and Information Security in the Balkans, Koper, Slovenia, 3–4 September 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 169–186.
38. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. TFHE: Fast Fully Homomorphic Encryption Library, August 2016. Available online: https://tfhe.github.io/tfhe/ (accessed on 15 March 2012).
39. Aguilar-Melchor, C.; Barrier, J.; Guelton, S.; Guinet, A.; Killijian, M.O.; Lepoint, T. NFLlib: NTT-based fast lattice library. In Proceedings of the Cryptographers' Track at the RSA Conference, San Francisco, CA, USA, 29 February–4 March 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 341–356.
40. Laine, K.; Player, R. *Simple Encrypted Arithmetic Library-Seal (v2. 0)*; Technical Report; Microsoft Research, Redmond, WA, USA, 2016.
41. Crockett, E.; Peikert, C. $\Lambda o \lambda$: Functional Lattice Cryptography. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 993–1005.

42. Polyakov, Y.; Rohloff, K.; Ryan, G.W. *PALISADE Lattice Cryptography Library User Manual*; Technical Report; Cybersecurity Research Center, New Jersey Institute ofTechnology (NJIT): Newark, NJ, USA, 2017.
43. Takagi, T.; Peyrin, T. Advances in Cryptology–ASIACRYPT 2017. In Proceedings of the 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; Springer: Berlin/Heidelberg, Germany, 2017; Volume 10625.
44. Lattigo 1.3.1. 2020. Available online: http://github.com/ldsec/lattigo (accessed on 3 July 2020).
45. Zainab, A.; Refaat, S.S.; Syed, D.; Ghrayeb, A.; Abu-Rub, H. Faulted Line Identification and Localization in Power System using Machine Learning Techniques. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 2975–2981.
46. Li, W.; Deka, D.; Chertkov, M.; Wang, M. Real-Time Faulted Line Localization and PMU Placement in Power Systems Through Convolutional Neural Networks. *IEEE Trans. Power Syst.* **2019**, *34*, 4640–4651.
47. Dark Sky API—Weather Conditions. Available online: https://darksky.net/dev/ (accessed on 3 July 2020).
48. Syed, D.; Refaat, S.S.; Abu-Rub, H.; Bouhali, O.; Zainab, A.; Xie, L. Averaging Ensembles Model for Forecasting of Short-term Load in Smart Grids. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 2931–2938.
49. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory (TOCT)* **2014**, *6*, 1–36.
50. Domingo-Ferrer, J. A provably secure additive and multiplicative privacy homomorphism. In *International Conference on Information Security*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 471–483.
51. Soykan, E.U.; Bilgin, Z.; Ersoy, M.A.; Tomur, E. Differentially Private Deep Learning for Load Forecasting on Smart Grid. In Proceedings of the 2019 IEEE Globecom Workshops (GC Wkshps), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.