Online-Journals.org

iJOE International Journal of Online and Biomedical Engineering

iJOE | eISSN: 2626-8493 | Vol. 19 No. 11 (2023) | OPEN ACCESS

https://doi.org/10.3991/ijoe.v19i11.40267

PAPER

Secure Data Computation Using Deep Learning and Homomorphic Encryption: A Survey

Anmar A. Al-Janabi[1,2](✉), **Sufyan T. Faraj Al-Janabi**[2], **Belal Al-Khateeb**[2]

[1]University of Technology – Iraq, Baghdad, Iraq

[2]University of Anbar, Ramadi, Iraq

anmar.a.aljanabi@uotechnology.edu.iq

ABSTRACT

Deep learning and its variant techniques have surpassed classical machine algorithms due to their high performance gaining remarkable results and are used in a broad range of applications. However, adopting deep learning models over the cloud introduces privacy and security issues for data owners and model owners, including computational inefficiency, expansion in ciphertext, error accumulation, security and usability trade-offs, and deep learning model attacks. With homomorphic encryption, computations on encrypted data can be performed without disclosing its content. This research examines the basic concepts of homomorphic encryption limitations, benefits, weaknesses, possible applications, and development tools concentrating on neural networks. Additionally, we looked at systems that integrate neural networks with homomorphic encryption in order to maintain privacy. Furthermore, we classify modifications made on neural network models and architectures that make them computable via homomorphic encryption and the effect of these changes on performance. This paper introduces a thorough review focusing on the privacy of homomorphic cryptosystems targeting neural network models and identifies existing solutions, analyzes potential weaknesses, and makes recommendations for further research.

KEYWORDS

homomorphic encryption, deep learning, privacy-preserving, convolutional neural networks, privacy-preserving deep learning, bootstrapping

1 INTRODUCTION

Algorithms for Machine Learning (ML) based on Deep Neural Networks (DNNs) have received much interest as a step forward in the advancement of Artificial Intelligence (AI). These algorithms provide impressive results and are widely used in various domains, including medical predictions, natural language processing, and financial predictions [1]. A Deep Learning (DL) algorithm is a variety of AI capable of interpreting data in the same way a human brain learns and classifies objects. By exploiting deep learning's predictive capability, we can predict the future and make

Al-Janabi, A.A., Faraj Al-Janabi, S.T., Al-Khateeb, B. (2023). Secure Data Computation Using Deep Learning and Homomorphic Encryption: A Survey. *International Journal of Online and Biomedical Engineering (iJOE)*, 19(11), pp. 53–82. https://doi.org/10.3991/ijoe.v19i11.40267

Article submitted 2023-04-08. Resubmitted 2023-05-18. Final acceptance 2023-05-23. Final version published as submitted by the authors.

decisions depending on currently accessible data that serve as training data for the DL model [2]. The deployed model will make predictions based on client data. That is how Machine Learning as a Service (MLaaS) was developed in the cloud, eliminating the requirement for clients to create their models to perform predictions. MLaaS and other services are offered by Microsoft Azure Machine Learning, Google Prediction API, and GraphLab [3]. Privacy and effective data processing are critical research topics in outsourced computing [4]. Before adopting the cloud model, the standard technique for protecting private data was encryption. It protects data during storage and transmission but not during decryption and processing. Extraction of data values requires decryption to access the raw data, which is frequently private. As a result, privacy-preserving solutions are becoming increasingly important. The underlying principle is to outsource data processing without allowing transparent access to it. Several studies have been conducted in recent years to examine the privacy protection of this sensitive data in various machine learning techniques, including linear regression, linear classifiers [5], [6], decision trees [7], and neural networks [8]. The necessity for private inference has increased due to two primary key points:

1. **Data proliferation and digitization:** As an increasing volume of sensitive data becomes digitally accessible, the need for secure methods to process this data while preserving privacy becomes paramount [9].
2. **Legal obligations:** The emergence of data-protection regulations such as the General Data Protection Regulation (GDPR) [10] and others has led to increasing legal obligations to maintain the privacy of customers' data.

Homomorphic Encryption (HE) is essential for enabling private inference. HE is a cryptographic technique that enables computations to be executed on encrypted data, producing an encrypted output corresponding to the result of operations performed on unencrypted data [11], [12]. This attribute facilitates the execution of computations on encrypted data by multiple entities without necessitating access to the unprocessed, unencrypted data. This makes it possible for multiple parties to collaborate on sensitive data, such as financial information or medical records, without revealing their respective inputs to one another. The relationship between HE and private inferencing can be observed in the following manner:

1. HE can be used as a utility within secure Multi-Party Computation (MPC) protocols [13] to facilitate computations on encrypted data, thereby facilitating private inferencing.
2. In order to maintain data privacy, HE enables computations to be offloaded to unreliable third parties (such as cloud servers). This is especially crucial when data owners lack the computational power for complicated computations yet are reluctant or legally unable to provide their raw data [14].

This review paper focuses on solutions entirely based on HE for privacy preservation and Neural Networks (NN) for ML. HE cannot be applied directly to NN. They are reconciled using a variety of advanced cryptography techniques. Among these techniques, multi-party computation enables many parties to share their inputs to obtain a unique output. Several practical constructions relying upon Garbled Circuits (GC), secret sharing, or Oblivious Transfer (OT) exist today [15]–[18]. HE is another technique, where additions or multiplications are made in the encryption using Rivest, Shamir, and Adleman (RSA) [19], ElGamal [20], or Paillier [21] cryptosystems. The potential of having Fully Homomorphic Encryption (FHE) capable of doing both additions and multiplications in any fashion in the encrypted world is

relatively recent. Since the initial implementation of FHE in 2009, several publications have been published on this subject [22]–[25]. This paper's main contributions are summarized as follows:

- Discusses and analyzes the most recent advances and related issues in HE, concentrating primarily on a combination of cryptography and neural networks.
- Provides benefits, weaknesses, and possible applications, reviewing existing development tools, languages, and libraries of HE cryptosystems.
- Covers the standard neural network structure and what modifications are required to overcome the limitations of applying the original structure to provide security and privacy.
- Provides detailed comparisons of the surveyed methods based on our defined metrics.

In summary, this study aims to serve as a starting point for academics and researchers who are new to the field by comparing various approaches and findings to preserve privacy and those interested in creating models that integrate NN with HE cryptosystems.

This paper is structured as follows: Section 2 describes classical cryptographic methods for securing DL. Implementation of DL technology with HE, limitations, and required modifications are presented in Section 3. Section 4 discusses privacy-based DL techniques. Section 5 discusses homomorphic training and evaluation. Attacks on DL models and privacy solutions are presented in Section 6. Furthermore, the most potential applications that work with HE are discussed in section 7. Section 8 presents the tools used within homomorphic schemes. Challenges are examined in Section 9. Finally, in Section 10, we present conclusions and raise issues for future research.

2 CLASSICAL CRYPTOGRAPHIC METHODS FOR SECURING DL

To ensure the privacy of private information such as financial or medical records, we need to use computation on encrypted data without disclosing the original content, which is made possible by homomorphic encryption, functional encryption, and secure multi-party computing techniques. This section demonstrates the cryptographic methods for securing deep learning to preserve privacy.

2.1 Homomorphic and functional encryption

In 1987, Rivest et al. [26] questioned the ability of an encryption system to facilitate the computation of encrypted data without the need to know secret information. HE is a term used in cryptography referring to symmetric/secret key or asymmetric/public key cryptosystem. It is capable of performing specific computations on ciphertexts. Also, the computational result is encrypted, so no decryption procedures are necessary throughout the computation. An HE system contains three primary functions: *KeyGen*, *Enc*, and *Dec*, which are responsible for generating keys, encryption, and decryption. However, it integrates an evaluation function, denoted by the term *Eval*. Assume having a collection of plaintext messages $\{m_i\}$ and their corresponding associated ciphertexts $\{c_i\}$. The evaluation function takes a public key K_{pk}, a set of ciphertexts $\{c_i\}$, and circuit f such that $Dec_{K_{sk}}(Eval_{K_{eval}}(f, c_1 \ldots\ldots, c_n)) = f(m_1, \ldots\ldots, m_n)$. Depending

on the type of computations, HE is named Partially Homomorphic Encryption (PHE) when only one type of operation is allowed; hence the name partially comes from [27]. It provides an infinite number of addition or multiplication operations on ciphertext without revealing data. For instance, the widely used RSA encryption [19] enables multiplication over ciphertext with no decryption; hence, RSA is referred to as a multiplicative HE. Another well-studied additive PHE is the Bealoh [28] and Paillier [21] cryptosystems. These encryptions have been extensively applied in a variety of applications, including electronic voting [29] and data mining. Another type of scheme known as Somewhat Homomorphic Encryption (SHE) provides homomorphic addition and multiplication operations but does not support arbitrary deep circuits. Boneh, Goh, and Nissem's BGN scheme [30] was the first to allow both operations with constant-size ciphertext and compute an unlimited number of additions but only one single multiplication.

In 2009, Gentry [14] proposed the first FHE scheme capable of calculating arbitrary functions of any circuit depth. It works in two main steps: first, it starts with a SHE and uses ciphertexts that contain a certain amount of noise to ensure security. However, for the decryption to be correct, this noise must be maintained within a certain bound, referred to as the decryption bound. In order to allow arbitrary computations on the scheme, Gentry introduces a new technique (second step) called "Bootstrapping" or "Recrypt," allowing the reduction of noise contained in a ciphertext [31]. The use of both SHE and bootstrapping results in applying an unlimited number of operations. Unfortunately, Gentry's bootstrappable SHE, which is based on Ideal Coset Problem (ICP), suffers from high computation and memory costs; hence, the scheme is inefficient for practical real-world applications. It has become the subject of optimization research, laying the groundwork for novel approaches to the performance problem. A decade later, numerous FHE systems have been designed to make it practical [32].

An FHE scheme is a public-key technique composed of four algorithms: KeyGen, Enc, Eval, and Dec [4], [14]. All operations must have a polynomial computational complexity under the security parameters λ, where:

- ***KeyGen*** accepts λ as input and outputs a key pair (public key K_{pk} and private key, K_{sk}) where K_{pk} is used to map plaintext to ciphertext, while K_{sk} does the reverse. Further, a public evaluation key K_{eval} is used during homomorphic multiplication operations.

$$\left(K_{pk}, K_{eval}, K_{sk}\right) \leftarrow KeyGen\ (\lambda)$$

- ***Enc*** takes plaintext m belongs to plaintext space M and encrypts it to cyphertext c in the ciphertext space C.

$$c \leftarrow Enc_{K_{pk}}(m)$$

- ***Eval*** accepts as input K_{eval}, a circuit f from a permitted set of circuits, and ciphertexts $C= (c_1, ..., c_t)$ that encrypt $M= (m_1, ..., m_t)$ of f; it outputs a ciphertext C', such that $DEC_{K_{Sk}}(C') = f(M)$.

$$C' \leftarrow Eval_{K_{eval}}(f, C)$$

- ***Dec*** implements the reverse process of an encryption algorithm; it accepts the inputs K_{sk} and ciphertext c and returns plaintext m, which belongs to plaintext space M.

$$m \leftarrow DEC_{K_{Sk}}(c)$$

Homomorphic encryption is the same as a conventional encryption scheme, but with an extra Eval algorithm, which allows performing operations over encrypted data. The correctness of the scheme is proven by the decryption process of the ciphertext produced by the ***Eval*** algorithm. The scheme is correct if the following statement is satisfied:

$$C' \leftarrow Eval_{K_{eval}}(f,C), then\, DEC_{K_{Sk}}(C') = f(M)$$

Also, when all circuits of the scheme are evaluated compactly, the homomorphic encryption scheme is full.

$$DEC_{K_{Sk}}\left(Eval_{K_{eval}}(f,C)\right) = f(M)$$

In 2005, Sahai and Waters [33] proposed Functional Encryption (FE). It is a public-key cryptosystem that lets users learn certain functionalities of encrypted data. It allows for secret keys with limited access, such that the holder of the key may decrypt just a small subset of the data and perform a specific function on it. Consider the functionality denoted by the notation $F: K \times X \rightarrow \{0,1\}^*$. Where K is the key space and X is the plaintext space, the function F is a deterministic function over (k,x) that returns (0,1)*. A scheme is FE for a function F over (k,x) if it can compute F(k,x) given a ciphertext of $x \in X$ and a secret key K_{sk} for $k \in K$. Both FE and HE allow for computations to be performed over encrypted data. The distinction is that the output of FE is plaintext, but the output of HE stays encrypted since HE evaluates encrypted information without decryption. Within HE frameworks, a trusted authority is unnecessary. In addition, if K_{sk} is provided, HE allows any circuit to be evaluated over the encrypted data, whereas FE allows only certain functions to be computed [1].

2.2 Secure multi-party computation

MPC, often called Secure Multi-Party Computation (SMC), is another cryptographic method for privacy-preserving. MPC provides a solution to the challenge of collaborative computing that protects the privacy of honest/dishonest users inside a group without needing a reliable third party. Generally, an MPC configuration presumes the existence of n parties $(P_1,P_2,...,P_n)$, each with its own data $(x_1,x_2,...,x_n)$. All parties are interested in jointly computing a fixed function $f(x_1,x_2,...,x_n)$ of their inputs, where their data privacy is protected. It wasn't until Yao [18] introduced Garbled Circuit (GC) in 1986, that the idea of secure computing was explicitly articulated as secure two-party computation. All the functions in Yao's GC can be expressed in terms of a boolean circuit, and it only needs constant number of communication rounds. Another technique used by MPC is an Oblivious Transfer, where information is transferred without the recipient's knowledge. In OT protocol, a receiver participant P_R can choose i without knowledge of the sender participant P_S and receive m_i from the group of P_S messages M. However, while P_R is unaware of the other messages in M, P_S is also unaware of the one that has been selected. Secret sharing is an additional building block for secure MPC protocols. Secret sharing distributes secrets to participants. The original value may be reconstructed using a small set of secret shared values [1].

In contrast to HE and FE methods, parties in a secure MPC cooperatively calculate the function on their data via a protocol rather than a single party. During the

procedure, the confidentiality of the parties' secrets must be maintained. In secure MPC, the communication cost is high but the computational cost is low, while in the HE method, the server incurs high computational costs but barely any communication overhead. Parties exchange encrypted data over an encrypted connection to the server. The server performs an inner product calculation between the data and the first layer's weight value and returns the result to the involved parties. The parties then decode the information and calculate the non-linear transformation. The result is encrypted before being sent back to the server. This procedure is repeated until the final layer has been calculated.

3 DEEP LEARNING AND HOMOMORPHIC ENCRYPTION

Deep learning structure and functionality mimic the behavior of the human brain. It consists of interconnected nodes widely known as "*neurons*" modeled in an ordered layer's architecture (input, hidden, and output). Each neuron accepts input, performs a function depending on the layer it belongs to, and outputs the result of that function. The hidden layer is more complex than the others where the learning process happens [27]. Convolution Neural Network (CNN) differs from standard DNN in focusing on convolution and pooling layers rather than connecting all layers. CNN is extremely successful in various applications, including image recognition and classification. It has the ability to build a complex model automatically. A series of operations such as convolution filters, non-linear activation functions, batch normalization, sub-sampling, and others automatically extract features from the dataset, achieving better performance based on Stochastic Gradient Descent (SGD) algorithm on backpropagation training process [27], [34].

Preserving data privacy utilizing HE is a natural development of classical DL. The architecture of such newly developed systems suffers from high computational complexity, reduced precision, non-linear functionality, and inefficient training process. CNN is a DL variant used in privacy-preserving, which requires architectural encoding, not encryption. Approximation of non-linear functions in a conventional CNN is vital to success in developing CNN for privacy-preserving, where only low-degree polynomial functions are considered for the transformation process. The following subsections define and describe CNN architecture layers. Also, discuss what changes may needed to be done (mostly approximation) in order to function properly over encrypted data.

3.1 Convolutional layer

The primary goal of the convolutional layer of a CNN is to extract useful features from the input data. As shown in Figure 1, this is accomplished through the convolution process, which involves sliding a kernel or filter across the input image. During the convolution process, the kernel's elements systematically align with the corresponding regions of the input image. The algorithm carries out element-wise multiplication between the kernel and the input region, subsequently summing the resulting products. This calculated value represents the dot product of the filter values and their respective input elements [13]. By iteratively applying this procedure to the whole input image, the convolutional layer creates a new output matrix, known as a feature map. The feature map captures the input data's spatial information and highlights specific features corresponding to the applied filter. As a result

of this procedure, CNN can better recognize and classify images and understand complicated hierarchical patterns in the data.

In the context of HE, the challenge lies in adapting these convolutional operations to work with encrypted data without compromising the privacy of the input information. Employing HE techniques makes it possible to directly conduct the convolution operation on encrypted data to generate encrypted feature maps. These encrypted feature maps can then be used for further computations within the NNs, preserving data privacy throughout the entire process.

Thus, integrating HE with the convolutional layer in CNN aims to enable privacy-preserving feature extraction while retaining the network's capability to learn and represent complex hierarchical patterns in the data. The evaluation of this layer is straightforward since it only uses additions and multiplications. These operations are HE compatible; thus, no further adaptation is needed [34].

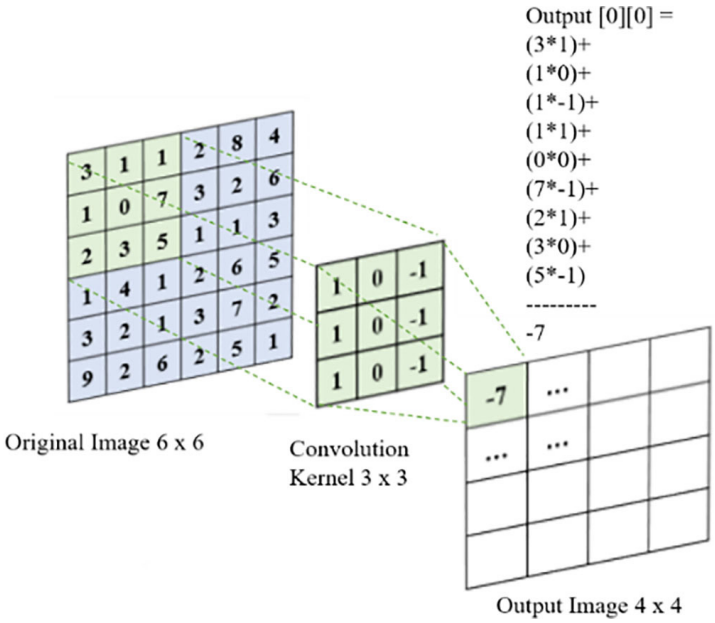

Fig. 1. Convolutional layer [34]

3.2 Activation layer

A neural network composed entirely of fully connected and convolutional layers can only classify data linearly, which is suitable for simple classification tasks. However, an activation layer was introduced to address more complex problems, as shown in Figure 2. Each neuron in the preceding layer is activated using a non-linear activation function to obtain one neuron in the current layer. Usually, we use the non-linear activation layer after each convolutional layer [35]. The activation function accepts a single number and executes a particular fixed mathematical operation on it. In practice, several activation functions do exist, including Rectified Linear Unit (ReLU) ($f(x)$ = max (0,x)), Sigmoid $\left(f(x) = \frac{1}{1 + e^{-x}} \right)$, and hyperbolic tangent (Tanh) $\left(f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \right)$. Due to the nature of these functions and their high complexity, they cannot be calculated over encrypted values. As a result, we need to find approximate alternatives for such functions, including only addition and multiplication [1], [8], [13].

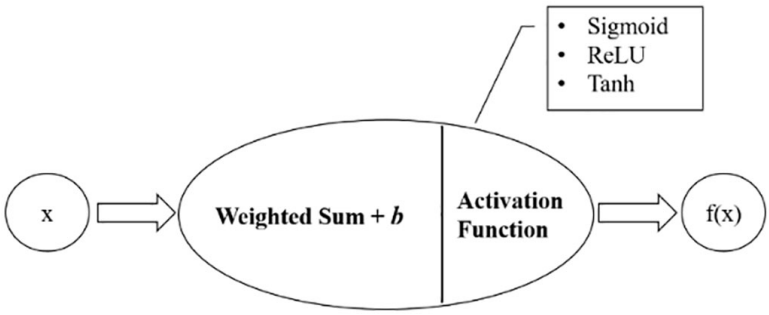

Fig. 2. Activation layer [34]

3.3 Pooling layer

Due to the large number of features extracted from raw input data by the convolution process in a convolutional layer, a down-sampling layer, called pooling layer, is required in order to reduce the size of spatial data (neurons), which reduces the computational complexity of the model, leading to faster classification process, especially in CNN [13], [35]. The pooling layer is the same as the activation layer that is non-linear. Two main functions exist in this layer: max-pooling and average-pooling. In HE, we cannot utilize the max-pooling function since the maximum value cannot be obtained over encrypted data [1]. Consequently, the average pooling function or a scaled-up version is used in HE [8]. Figure 3 depicts the pooling layer.

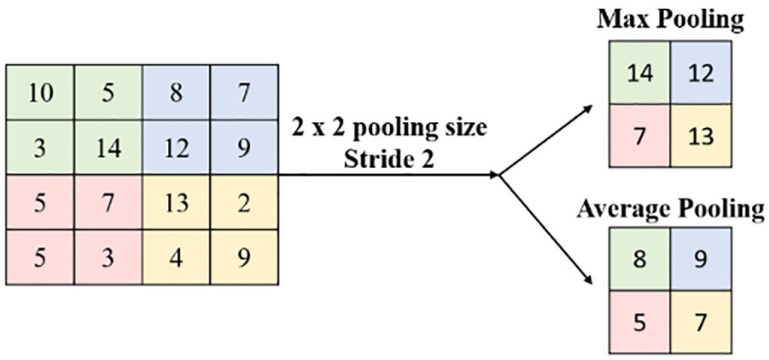

Fig. 3. Pooling layer [1]

3.4 Fully connected layer

Each neuron of the current layer is connected to every neuron in the previous layer through connections representing values called weights; hence, the name "fully connected." The operation on this layer is a dot product between the output of preceding layers' neurons and the weight associated for each neuron. Dot product operation is HE compatible. It only consists of additions and multiplications, and no effort is required to modify them for usage over encrypted data using HE [1], [36]. Figure 4 depicts a fully connected layer.

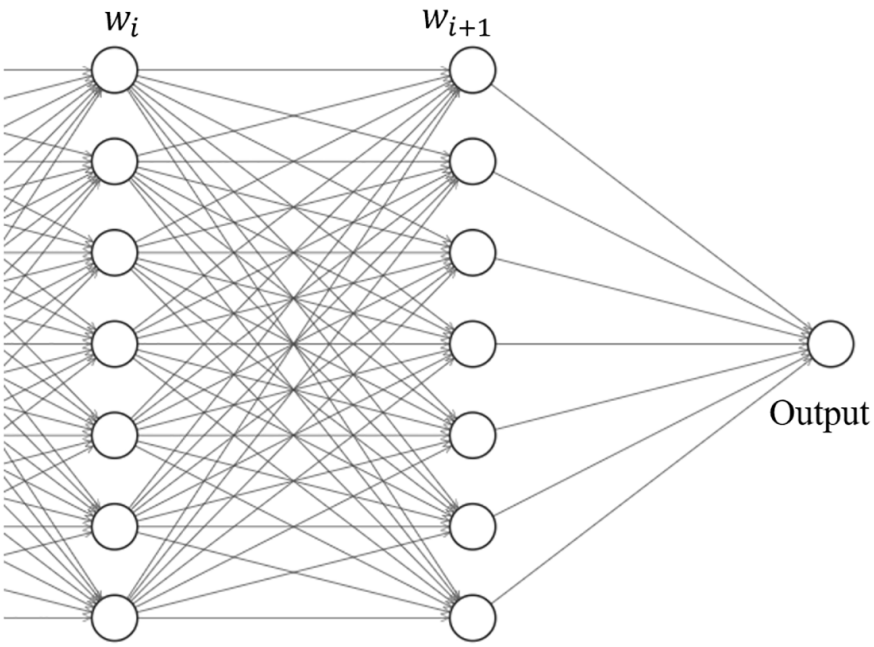

Fig. 4. Fully connected layer [35]

3.5 Dropout layer

Building the NN model contains two main phases: training and testing. When a model results in high error over testing, the model may be biased toward the training set. This problem is called overfitting during the training process. To avoid an overfitting problem, a dropout layer, as Figure 5 depicts, is used [1], [35].

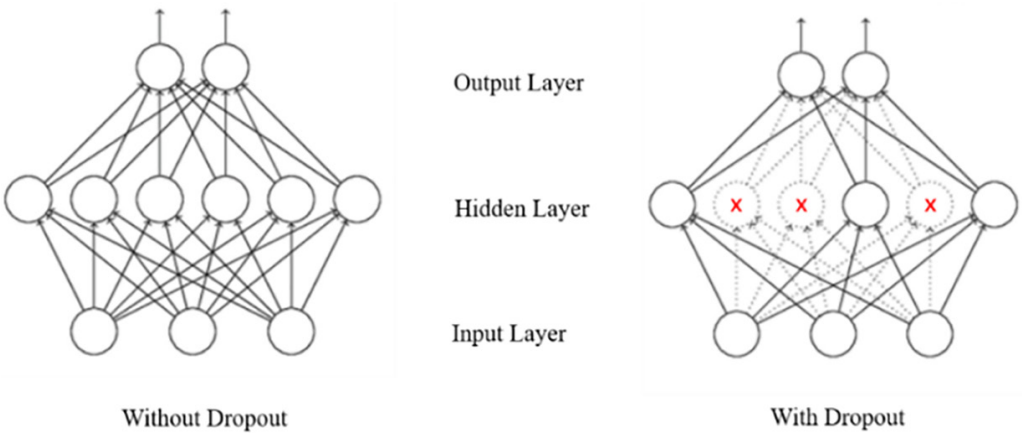

Fig. 5. Schematic diagram of dropout layer [34]

The implementation of state-of-the-art DL with HE suffers from several incompatibilities or limitations. To merge DL structure with HE approaches, modifications have to be made. The Batch Normalization (BN) layer, activation function approximation, and convolution with increased stride are the most often necessary adjustments, as shown in Figure 6.

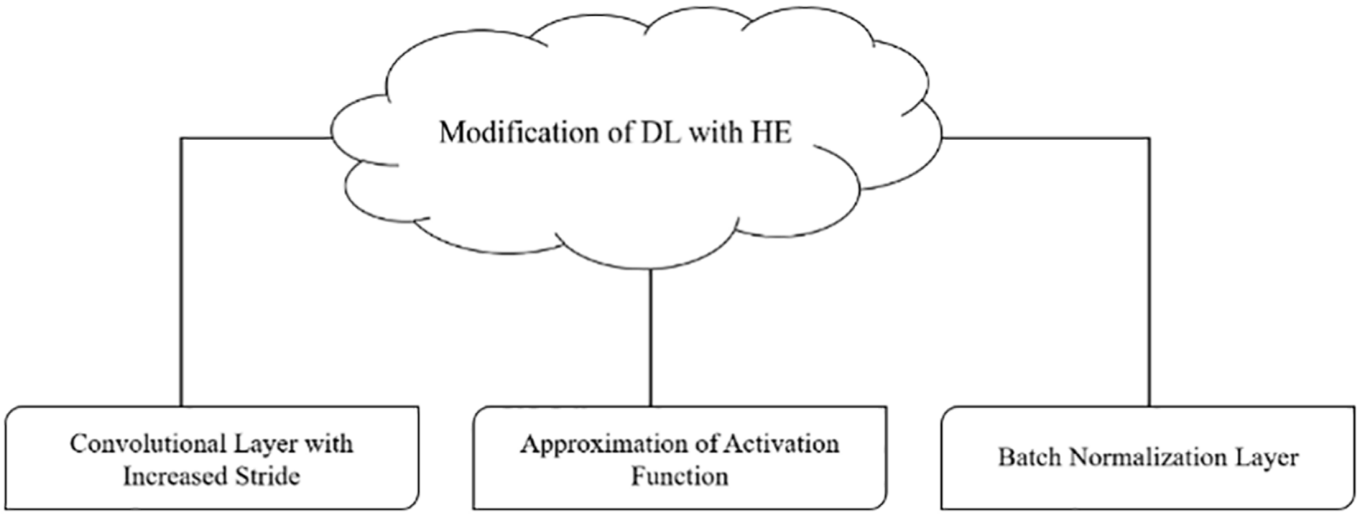

Fig. 6. Required modifications of DL with HE [1]

1. **Convolutional Layer with Increased Stride.** Liu et al. [27] proposed this architecture as a substitute for the pooling layer. The design replaced the pooling layer with a convolutional layer of an increased stride. BN is applied between Fully Connected (FC) and ReLU layers. This maintains the data's depth but reduces its dimension [37].
2. **Approximation of Activation Function.** Prior works [27], [34], [38] have used polynomial approximations to perform the activation functions. Numerical analysis, Taylor series, and approximation of polynomials relying on the activation function's derivative are all well-known approaches. The numerical analysis creates points from a ReLU activation layer and then passes them to the approximation function as inputs, While the Taylor series approximates the activation function using polynomials of different degrees [1].
3. **Batch Normalization Layer.** Ioffe and Szegedy [37] proposed a BN layer. The layer's primary goal is to speed up training by improving the NN's stability. It takes the output of an activation layer and then scales it to a number between 0 and 1. Each input is subtracted from the batch mean value and then divided by the batch mean value in the BN layer.

4 PRIVACY-BASED DEEP LEARNING METHODS

The cryptosystems presented in this section are primarily based on HE. This section is divided into subsections, including private inferencing and private training.

4.1 Private inferencing

Private inference refers to a technique for processing inferences without disclosing the underlying input data. In private inference, preserving the privacy of user input data and models is essential. Xie et al. [39] introduced BAYHENN, a novel high-level protocol as a practical interactive paradigm for secure DNN inference. The authors

claimed that their technique was the first to secure and prevent client and server privacy leakage simultaneously. They combine HE with Bayesian Neural Networks (BaNNs). Specifically, they employed HE to protect the client's raw data and BaNNs to protect the DNN weights on a cloud server.

Gilad-Bachrach et al. [40] proposed a method that converted learned NN to CryptoNets, in which a NN is applied to encrypted data that can be seen as a solution to the problem of blind, non-interactive classification. The NN employs a leveled HE scheme for inputs and homomorphically propagates data throughout the network. The weakness of CryptoNets lies in its limited performance due to computational complexity, which performs poorly on deeper NN models, where accuracy declines and error rate increases.

Chabanne et al. [41] solved the limitation of CryptoNets [40]. They designed and evaluated the first privacy-preserving classification method for DNN with a depth greater than two. They combined CryptonNets [40] with Ioffe and Szegedy's [37] BN layer and polynomial approximation for activation function. The advantage of this model was that it can be used in neural systems with a large number of non-linear levels though still offering greater accuracy, in contrast to CryptoNets, which loses accuracy as the number of non-linear layers increases.

Hesamifard et al. [34] presented CryptoDL, a newly developed technique to adopt a deep CNN with the practical limitations of HE schemes. Three main transfer functions were approximated by the authors: ReLU, Sigmoid, and Tanh. The approximation relies on the activation function's derivative to find the lowest-degree polynomial within a certain error range.

Sanyal et al. [42] introduced TAPAS, a new algorithm to accelerate computations in Binary Neural Networks (BNNs) [43], [44]. They reduced computation time, creating a DL architecture comprising fully connected, convoluted, and batch-normalized layers [37]. Also, they developed techniques such as the sparsification of NN and algorithmic tools to accelerate and parallelize computation on encrypted data through Single Instruction Multiple Data (SIMD) operations. These techniques enable highly accurate predictions on encrypted data, producing a framework that provides Encrypted Predictions as a Service (EPaaS).

Bourse et al. [45] introduced FHE-DiNN, a framework for combining FHE with a Discretized Neural Network (DiNN) in a DL environment to provide data privacy. Bootstrapping was used to achieve linearity with a weighted sum and sign activation function that ranged from –1 to 1. When comparing DiNN to standard NN, one significant distinction is that the weight, bias value, and domain of an activation function in FHE-DiNN must be discretized. Compared to Cryptonets [40], FHE-DiNN efficiently increases the speed and reduces the complexity of FHE, but at the expense of accuracy.

Jiang et al. [46] presented E2DM, a novel matrix-encoding technique and an efficient evaluation mechanism for performing fundamental matrix operations such as addition, multiplication, and transposition. The data and model were homomorphically encrypted, guaranteeing both Privacy of Client (PoC) and Privacy of Model (PoM). Also, their model fulfilled Privacy of Result (PoR), as only the client can decrypt the predicted result. E2DM delivers a smaller message size as well as reduced latency than the CryptnoNets [40] approach.

Zhou et al. [47] presented a secure binarized CNN inference and perceptron models, where inputs and weights are binarized. The models can provide privacy in a DL system composed of two components: (1) the service provider (SP) component, which trains and stores the model on unencrypted data and allows homomorphic operations and (2) the client's component, where encryption and decryption are performed.

Binarized convolution, max-pooling, and fully connected layers were used to form the model. The bit-wise operations executed operations such as addition and multiplication, speeding up CNN inference calculation.

Disabato et al. [48] proposed an innovative distributed design for Deep Learning as a Service (DLaaS) which protects customers' sensitive data while offering cloud-based DL services. The suggested architecture is executed using a client server REST-based framework for sharing encrypted data and findings among both client and server. The work adopts the Brakerski–Fan–Vercauteren (BFV) scheme [25] based on Ring Learning With Errors (RLWE) problem. A square activation function substitutes the ReLU, and the max-pooling by the average. MNIST dataset [49] and FMNIST dataset [50] were used in the analysis.

Obla et al. [51] developed a novel approach to construct HE-friendly non-linear activation functions for deep CNN using polynomial approximations. The researchers introduced three approximation methods: Taylor expansion, best uniform approximation, and best square approximation. Additionally, they proposed novel weighted activation function approximations independent of the degree of approximation or training technique. The MNIST [49] and FMNIST [50] datasets were used to train the deep CNN model developed by Chabanne et al. [41]. Furthermore, researchers trained on CIFAR-10 [52] and employed Springenberg et al. [53] architecture, in which design involved no pooling layers.

Lee et al. [54] proposed polynomials that accurately approximated the ReLU activation function and the max-pooling function for Privacy-Preserving Machine Learning (PPML) using FHE by composing minimax approximation polynomials of small degrees. As a result, researchers may utilize the suggested approximation polynomials to substitute the ReLU and max-pooling functions in classifiers like ResNet [55], VGGNet [56], and GoogLeNet [57], and BN as core models.

Badawi et al. [58] offered an FHE-based CNN that can homomorphically classify secured images. Their work was the first to implement a GPU-accelerated Homomorphic CNN (HCNN) that operates on encrypted images using a pre-learned model. The implementation included a series of optimization techniques, including quantized NNs with low-precision training, improved FHE design and parameters selection, and GPU-accelerated implementation.

Lee et al. [59] introduced a PPML model implemented using standard ResNet-20 with Residue Number System CKKS (RNS-CKKS) [60] scheme. The model is almost the same as the state-of-the-art ResNet-20 paradigm, except that for the first time bootstrapping is incorporated. The authors claim that they were the first to implement a PPML model based on FHE and the first to apply the Softmax function in the PPML model, hence preventing model-extraction attacks. The proposed model was numerically verified achieving a 98.67% agreement ratio to the standard ResNet-20 design with unencrypted data when used over CIFAR-10 dataset [52].

4.2 Private training

The process of training DL models accurately without security is not an easy step due to all factors, including model structure, weights and biases initialization, and hypertuning parameters, that require analysis. Privacy-preserving training is almost impossible when security is applied during this phase. This section summarizes some related studies to secure training.

Mohassel and Zhang [61] suggested a protocol for privacy-preserving training and classification using secure MPC techniques based on NN. Their research employed HE as one building block of secure MPC. The data owner shares data with two servers, which perform ML approach that uses a Two-Party Computation (2PC) technique. They are particularly interested in three algorithms: linear regression, logistic regression, and NNs. Their approach comprises two stages: online and offline. They generated multiplication triplets using oblivious transfer in the offline phase and safely computed an activation function in logistic regression and NN training in the online phase.

Hesamifard et al. [62] presented a CryptoDL framework based on CryptoNets [40]. They were the first to show that it is possible and practical to train the neural model using HE and make encrypted predictions. Compared with earlier secure MPC studies, which need a high number of interactions between client and server, this method requires no contact from the server until the noise level exceeds a predefined threshold. The model was implemented using HELib [63], and the model's running time was measured for both the training and testing stages.

A few new studies have examined privacy issues during the training phase, notably for the back-propagation algorithm [64], [65]. Zhang et al. [64] also suggested utilizing the Brakerski-Gentry-Vaikuntanathan (BGV) encryption scheme to efficiently perform a high-order back-propagation algorithm for deep-computing model training in the cloud. To avoid an excessive Multiplicative Depth (MD), the new weights were given back to the parties for decryption and re-encryption after each iteration. As a result, the solution had a high communication complexity. Bu et al. [65] proposed a cloud-based back-propagation algorithm that preserves privacy. Their suggested approach offloads expensive processes to the cloud and employs a BGV scheme for protecting data privacy during the training phase.

In summary, complete datasets and features of our surveyed deep learning papers of private inferencing and training are illustrated in Tables 1 and 2, respectively. Figures 7–10 condense performance comparisons of private inferencing and training strategies based on specified metrics.

5 HOMOGRAPHIC TRAINING AND EVALUATION IN NEURAL NETWORKS

During the training process, a mapping is performed that goes from the input space to the output space based on how the weights of each neuron are modified. This mapping is developed throughout the training phase, where a network can learn from and generalize over several instances where the learning process is either supervised or unsupervised. Even for models that are not HE, training comprises computationally demanding tasks. With HE, it is already challenging to train an NN because of the large number of operations required to locate the set of weights, which minimizes loss function. The training procedure in an HE domain requires several bootstrapping runs and large encrypted messages. The computational complexity of encrypted training is several orders of magnitude higher than that in unencrypted training. Therefore, it is impractical to train classifiers that might have a large number of layers [4]. Bootstrapping or re-encrypting in the HE training phase can be controlled through using accelerators or exclusion. Most recently, hardware accelerators such as high-performance processors (GPUs and FPGAs) have become increasingly popular. These new technologies greatly

reduce runtime and can make encrypted versions equivalent to unencrypted ones. One approach to achieving exclusion is decrypting ciphertext inside a secure entity, such as a client-server. Another widely utilized exclusion approach is pre-training a neural model over unencrypted data, where weights are made public to avoid overhead.

Access to raw data is not restricted throughout the testing and training phases of NNs. Therefore, many applications may not be feasible due to legal and ethical concerns, such as medical and financial applications. Employing HE with NN can be seen as a logical extension of NN. The approach involves applying HE to the network's inputs and then homomorphically transmitting the information across the network. At the inference stage, knowing a model's MD in a priori helps in the evaluation process, where the number of operations may be estimated in advance. From an HE viewpoint, the network is analogous to a layered circuit, with levels being referred to as "layers."

Table 1. Features of our surveyed DL papers—private inferencing

Reference	Year	Methodology	ML/ DL Technique	Limitation(s)	Overcome the Limitations	Dataset(s)
Xie et al. [39]	2019	HE	DNN	• Model accuracy • Computation acceleration	Create a better algorithm to optimize BaNNs for higher accuracy and employ FPGA to accelerate computation	MNIST, Breast Cancer
Gilad-Bachrach et al. [40]	2016	HE	CNN	• Limited performance • Training: Do not cover privacy and slow over plaintext	GPUs and FPGAs to accelerate the computation. Also, finding more efficient encoding schemes to allow for smaller parameters, hence faster HE computation.	MNIST
Chabanne et al. [41]	2017	HE		• Classification accuracy relies on the approximation of activation function	Employing preprocessing techniques such as feature scaling and postprocessing techniques such as thresholding can effectively mitigate the impact of approximation.	MNIST
Hesamifard et al. [34]	2017	HE		• Do not cover privacy during training • Data and the training model can be tuned when using HE	GPU and FPGA can be used as an improvement for efficient implementation since these types of data take a very long time to train	MNIST, CIFAR-10
Sanyal et al. [42]	2018	HE		• Support only BNNs	Encrypting non-binary or real-valued neural networks can solve this limitation	Cancer, Diabetes, Faces, MNIST
Bourse et al. [45]	2018	HE		• Drop in accuracy due to discretization procedure	Train a DiNN instead of discretizing an already-trained model to improve classification accuracy	MNIST

(Continued)

Table 1. Features of our surveyed DL papers—private inferencing *(Continued)*

Reference	Year	Methodology	ML/DL Technique	Limitation(s)	Overcome the Limitations	Dataset(s)
Jiang et al. [46]	2018	HE		• Simple matrix operations	Extending the advanced matrix computation will be promising future work	MNIST
Disabato et al. [48]	2020	HE			–	MNIST, FMNIST
Obla et al. [51]	2020	HE		• Classification accuracy	Softplus can increase network performance and classification accuracy. Changing the network architecture or adjusting hyperparameters can assist.	MNIST, FMNIST, CIFAR-10
Badawi et al. [58]	2021	HE			–	MNIST, CIFAR-10
Zhou et al. [47]	2020	HE	Binarized CNN	• FC layer takes 51.2% of the time in the overall model	Train the model with binary value and creates a method to replace batch normalization under FHE	MNIST, Breast Cancer
Lee et al. [54]	2021	HE	CNN— (ResNet, VGGNet, GoogLeNet)		–	CIFAR-10 ImageNet
Lee et al. [59]	2022	HE	CNN— (ResNet-20)	• Running time: 4 hours • Security level: 98-bits • Model training: only once	Implementation with various accelerators can be realized using GPU, FPGA, or ASIC. Changing parameters of the RNS-CKKS scheme to raise security level.	CIFAR-10

Table 2. Features of our surveyed DL papers—private training

Reference	Year	Methodology	ML/DL Technique	Limitation(s)	Overcome the limitations	Dataset(s)
Mohassel and Zhang [61]	2017	MPC	NN	–	–	MNIST
Hesamifard et al. [62]	2018	HE	CNN	• Do not approximate non-continued activation functions • Susceptible to attacks	Using piecewise polynomial approximations. Also, exploiting deferential privacy, or zero-knowledge proofs for SMPC to increase security.	MNIST, CIFAR-10
Zhang et al. [64]	2016	HE	DCM (Deep Computation Model)	• High communication complexity	–	STL-10, NUS-WIDE, PeMS, DLeMP
Bu et al. [65]	2015	HE		–	–	STL-10, CIFAR-10

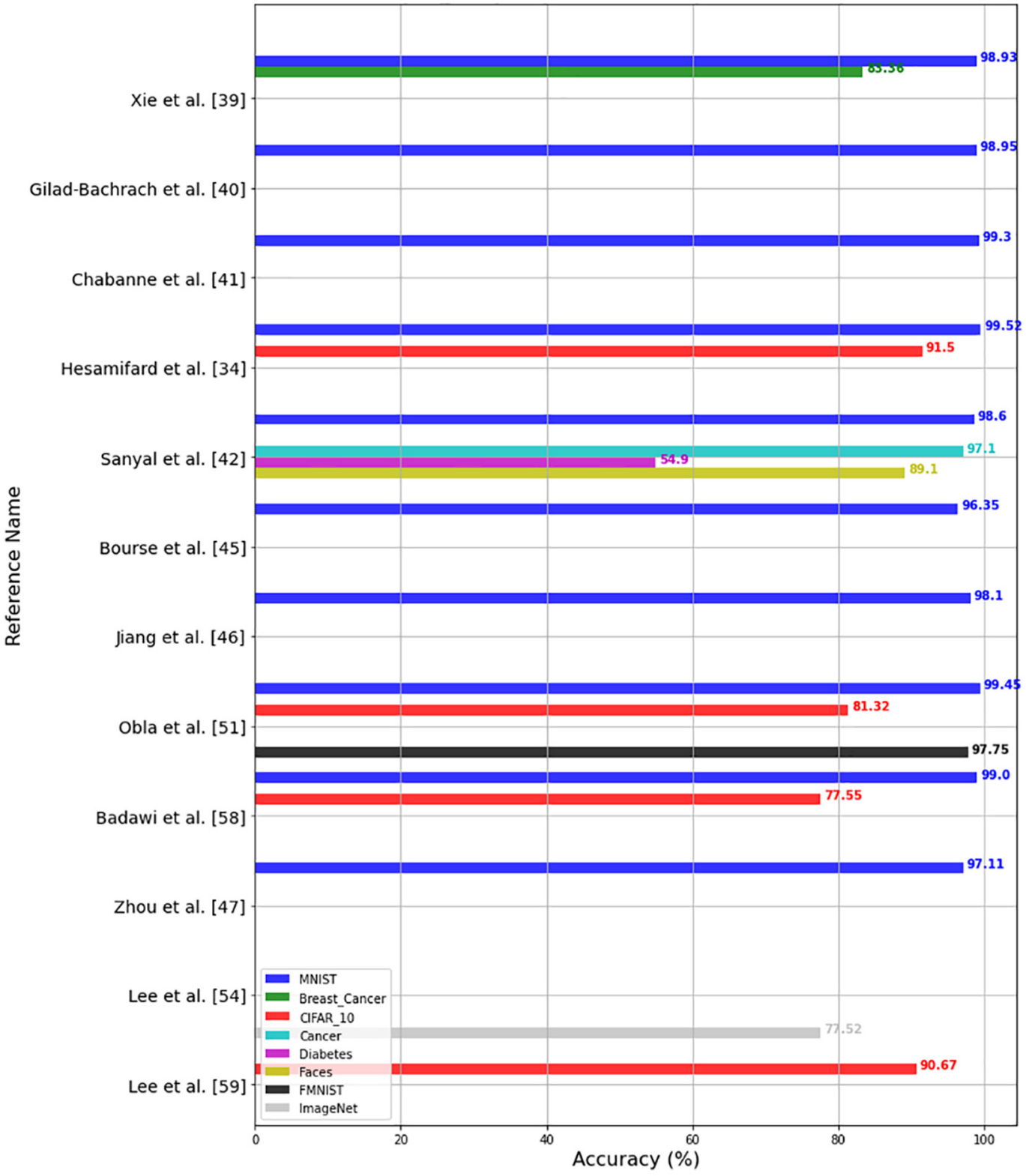

Fig. 7. The accuracy of research studies for various datasets—private inference

Figure 7 depicts the accuracy of research studies utilizing private inference techniques for various datasets. The x-axis indicates the accuracy percentage, while the y-axis depicts the various research studies. The colored bars reflect different datasets, such as MNIST, CIFAR-10, Breast Cancer, and others. The graph depicts the varying degrees of accuracy obtained by various studies performed on each dataset. The study by Xie et al. [39] obtained the highest possible accuracy on the MNIST dataset, whereas Study Lee et al. [59] did well on the CIFAR-10 dataset. These results demonstrate the significance of evaluating the performance of private inference approaches across multiple datasets.

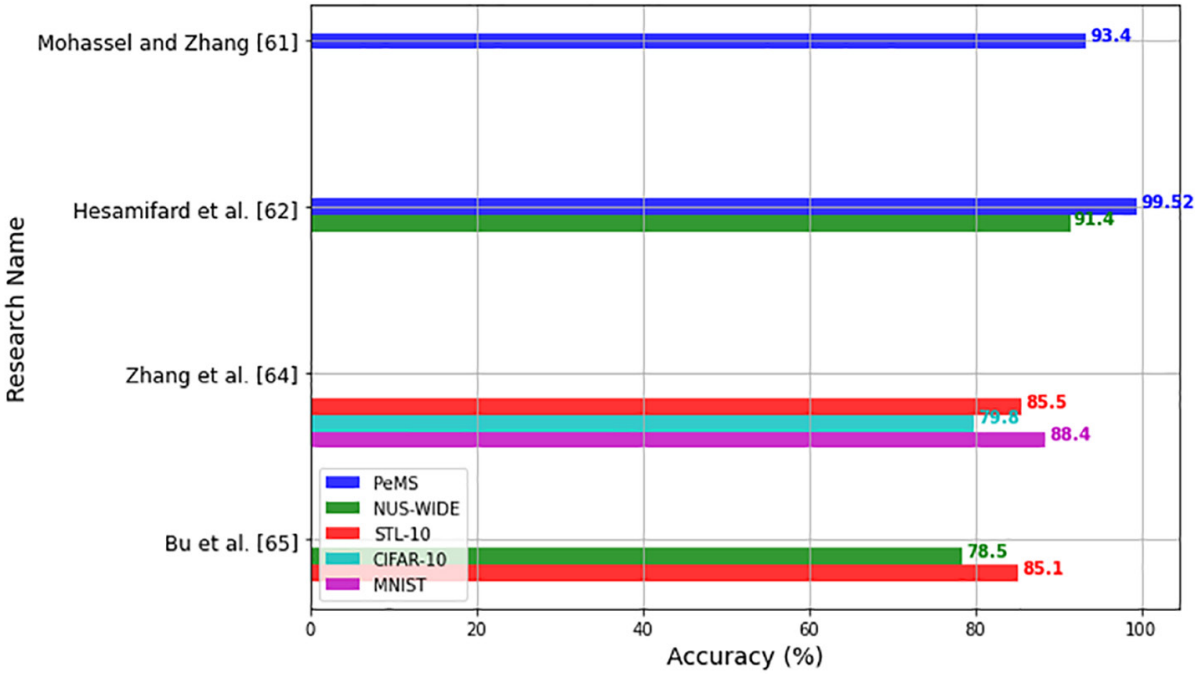

Fig. 8. The accuracy of research studies for various datasets—private training

Figure 8 visualizes the accuracy of different studies across multiple datasets: MNIST, CIFAR-10, STL-10, NUS-WIDE, and PeMS. It simplifies performance comparison and helps identify the best-performing models. The x-axis indicates the accuracy percentage, while the y-axis depicts the various research studies.

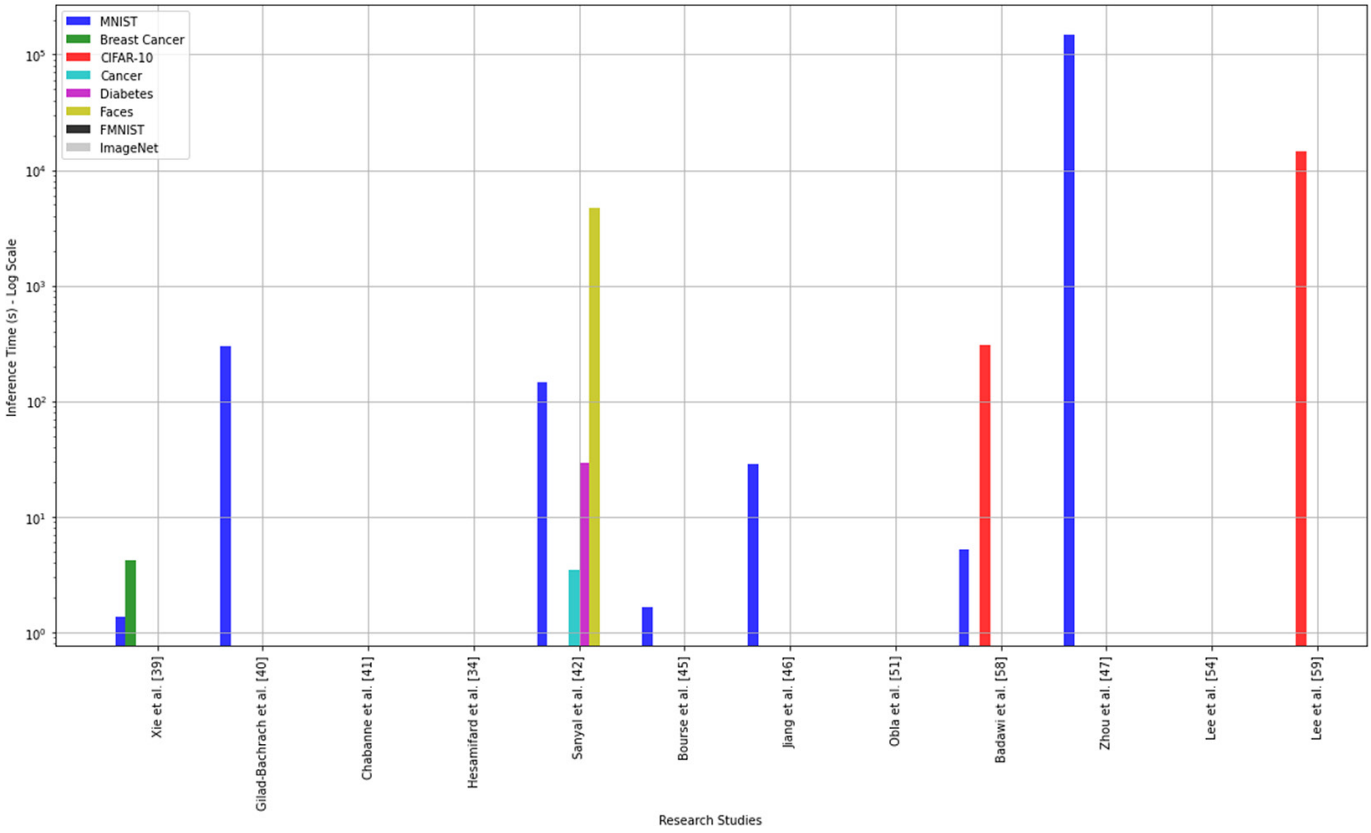

Fig. 9. The inference time of research studies for various datasets—private inference

Figure 9 depicts the inference times of private inference for various research studies utilizing various datasets. It regulates time metrics by displaying them on a logarithmic scale to reflect potentially large value differences (seconds, minutes, hours). This visualization facilitates cross-study and cross-dataset performance comparison. The x-axis indicates the various research studies, while the y-axis indicates the inference time in the log scale. Figure 10 is similar to Figure 9, but it shows the inference time when models are both trained and classified over encrypted data.

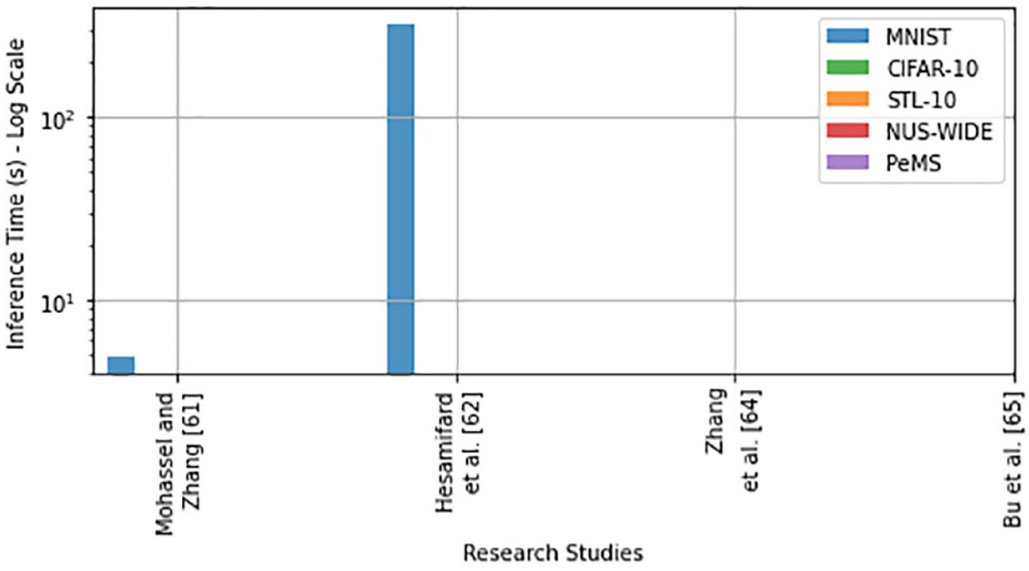

Fig. 10. The inference time of research studies for various datasets—private training

The cryptosystem implements an encryption algorithm with a predetermined noise budget without bootstrapping techniques. In other terms, a leveled HE scheme is sufficient for the inference stage since the quantity of noise allowed by the encrypted message is known and the maximal degree of the polynomial functions on the encrypted data is fixed. Since deep learning requires many hidden layers in a network, it requires a scale-invariant FHE technique. Because of this, bootstrapping or any other recrypt function must manage a significant amount of noise [4].

6 DL MODEL ATTACKS AND PRIVACY-PRESERVING MODEL AS A SOLUTION

Deep learning privacy-related model attacks aim to let the adversary obtain private knowledge that was not supposed to be publicly available. Preserving privacy in deep learning frameworks has three main security goals. The first is to preserve the client's private information by preventing the server from accessing it during the training phase. The second prevents the server from directly obtaining model input in the classification stage. The third only applies when the client has delegated prediction to a server. Attacks on DL models that privacy-preserving approaches can mitigate are membership inference, model inversion, and model-extraction attacks [1].

Membership inference aims to determine if a piece of input data was used as part of the training dataset—in other words, if sample data is used to generate some aggregation of data. A model, including its parameters, may be considered an aggregate of the training data within the context of DL. It is one of the most common types of attack, initially described by Shokri et al. [66], and it violates the first security goal mentioned above. Extraction of some attributes of sensitive training data, or even recovery of training data, are examples of more advanced forms of membership inference attacks. Security against the membership inference attack may be simplified to the security of the underlying cryptosystems in Privacy-Preserving Deep Learning (PPDL) models. Model parameters are encrypted, so an adversarial server cannot read them in plaintext unless the model is available to the public. The privacy parameter affects the model accuracy and membership inference attack performance for deep private models. The model inversion attack described by Fredrikson et al. [67] is a type of prediction-phase attack that violates the second security purpose of PPDL by targeting the models themselves. Model inversion attacks attempt to identify the sensitive features of input data by starting with the non-sensitive features and working backward through the prediction outcomes produced by the model. There has been relatively limited research on the trade-off between the accuracy of the model and the effectiveness of the attack for deep private-based models. Last but not least, model-extraction attacks [68], commonly referred to as model-stealing attacks, target the third security objective of PPDL. The goal of model-extraction attacks is to build a model that is functionally equal to a given black-box (target) model. If an attacker is successful in a model-extraction attack, they gain access to a white-box model. If the owner of the model sells access to it, the attacker can exploit it directly. The generated model can then be used as a "stepping stone" for other attacks with white-box models.

7 APPLICATIONS OF HOMOMORPHIC ENCRYPTION

This section explores several applications of various flavors of HE. Some of them require PHE, SHE, or FHE. In general, an FHE scheme can compute anything over encrypted data, while PHE and SHE schemes are more restricted. Theoretical and

practical applications exist in several fields of cryptography. Some of the most significant applications are as follows:

7.1 Outsourcing storage and computation

One of the major applications in HE is outsourcing storage and computation of data. It can be seen as a delegation of computation of a function to a server that provides computation without disclosing sensitive information [69]. Consider a small company that wants to outsource computations to the cloud provider, and the cloud may be malicious or subjected to malfunctions. For the cloud to process data, it must have access to the company's sensitive information. Hence, the company does not trust the result of computations and requires proof that the computations were done correctly and were more efficient than the company itself. HE delivers an elegant solution to this problem. The company encrypts data then sends it over to the cloud provider (server) to process it in its encrypted form. After computations are done, the server sends back the encrypted result to the company to decrypt it [70], [71].

7.2 Private information retrieval

Another direct use of HE is the ability to conduct personal queries on a database or search engine. The most straightforward analogy is Private Information Retrieval (PIR), in which a server hosts a huge database, such as the US patent database, and a client wishes to obtain a single record from the database with no knowledge by the server of the requested record. The user can use HE to encrypt the index of the record he wishes to retrieve. The server performs evaluation of a function $f_{db}(i) = db[i]$ on the encrypted index, returning encrypted result to the client. The client will decrypt the result using his own private key and gets the plaintext record. Utilizing HE to the index of a single required record and retrieving it in an encrypted format, PIR becomes quicker, more secure, and private while sustaining confidentiality, integrity, and availability. [29], [71].

7.3 Zero-knowledge proofs

This fundamental cryptographic protocol serves as a theoretical application of homomorphic cryptosystems. Zero-knowledge proofs are employed to establish the presence of knowledge of certain secret information. The user wishes for his private information, such as a password, to remain secret and secure during the protocol's operation. Zero-knowledge proofs ensure that the protocol sends just the information intended and no (zero) additional information [72]. It is noteworthy that Gentry's seminal work shows that HE can be utilized to construct Non-Interactive Zero-Knowledge (NIZK) proofs of small size [14].

7.4 Healthcare

Health care systems operate in an environment where sensitive data must be secured from exposure while being available as input to computations needed for daily operations. For some applications in the healthcare industry, HE can assist in

balancing risk and potential value in information exchange. Two such applications are billing and report generating. In both circumstances, analysts require access to personal medical data to do computations on their content. By allowing such computation without disclosing such records in plain view, breaches may be avoided without compromising mission-critical applications. HE allows a breach-proof solution for such applications in a clinic environment. For example, an analyst searches current medical records for data such as prescription statistics and medical encounters given by the clinic. A potentially insecure server maintains an encrypted collection of relevant data, including individual medical records protected under privacy and policies. HE enables queries to be computed upon encrypted data and delivers an encrypted result to the analyst. The analyst then decrypts the answer, including the relevant report or invoice query results. Since the data corpus stays encrypted, both at rest and during computation, adversaries gain no knowledge of the data or the results of these queries [73]. Many other applications concerning HE exist, such as Data Mining (DM), forensic image recognition, signatures, secret sharing schemes, election schemes, watermarking schemes, financial privacy, and many other applications.

8 HOMOMORPHIC ENCRYPTION TOOLS

This section exhibits the most widely used crypto libraries with HE implementations in real-world applications. High-quality implementations should boost theoretical research. In recent years, corporate and scientific groups have released several open-source libraries.

Simple Encrypted Arithmetic Library (SEAL) [74] is the most widely used accessible tool developed by Microsoft® that supports BFV and CKKS crypto schemes. The tool is written in C++ and actively developed for additional languages such as Python and JavaScript. It is capable of compressing data in order to obtain considerable memory footprint reductions. SEAL is used by [40], [58], [75]–[78].

Homomorphic Encryption Library (HElib) [63] is a C++ based FHE library built on the BGV cryptographic system that also supports the CKKS scheme. It concentrates on the effective usage of ciphertext-packing optimization. A downside of HElib is that it has a limited capability for bootstrapping. It is used by [34], [41], [79], [80].

Fast Fully Homomorphic Encryption (TFHE) over Tours [23] is a library implemented in a Gentry, Sahai, and Waters (GSW) ring variant with a torus format. Developed under C/C++, the library supports a highly rapid gate-by-gate bootstrapping mechanism; it places no restrictions on the number of gates. It is used by [81], [82].

Homomorphic Encryption for Arithmetic of Approximate Numbers (HEAAN) [83] is a library that implements the CKKS bootstrapping and CKKS crypto scheme. It is written in C++ and offers capabilities for fixed-point computation. HEAAN is used by [46], [78].

PALISADE [84] is a standalone HE library written in C++ funded by DARPA defense contractors and supports BGV, Fast Fully Homomorphic Encryption Library over the Torus (THEW), and many other schemes. The library provides an extension for multiparty. This library achieves high performance by utilizing RNS algorithms [4].

CUDA Homomorphic Encryption (cuHE) [85] is a library that uses GPU to accelerate evaluations with homomorphic schemes. It is implemented in C++ for parallel-platform CUDA. To deal with large polynomial operands, arithmetic functions use the Chinese Remainder Theorem (CRT), the Number Theoretic Transform (NTT), and Barrett reduction [4].

Along with the libraries mentioned above, there are many other libraries, such as FHEW [86], FV-NFLlib [87], lattigo [88], concrete [89], and nuFHE [90]. Last but not

least, GPU-accelerated libraries can further improve already-fast TFHE bootstrapping speeds at about two orders of magnitude. Still, they are significantly more costly and less common in commercial data centers [91]. Table 3 shows FHE libraries and includes a list of supported languages, schemes, features, and accessibility considerations.

Table 3. Overview of existing FHE libraries [91]

Name	Language	Supported Schemes			Features		Accessibility
		BFV	CKKS	GSW	Bootstrap	Levels	Documentation
SEAL [74]	C++, .NET	•	•			•	•
HElib [63]	C++	•	•		•	•	•
TFHE [23]	C++			•	•		•
HEAAN [83]	C++		•		•	•	
PALISADE [84]	C++	•	•	•	•	•	•
FHEW [86]	C++			•	•		
FV-NFLlib [87]	C++	•				•	
Lattigo [88]	Go	•	•			•	
Concrete [89]	Rust			•	•		•
cuHE [85]	C++, Python			•	•		
nuFHE [90]	C++, Python			•	•		•

9 CHALLENGES AND WEAKNESSES

This section analyzes the weaknesses of HE based on the surveyed papers. While HE standards and implementations contribute to the development of Deep Learning with Homomorphic Encryption (DL-HE), still several key challenges remain critical, including the following [4]:

1. **Computational Overhead:** DL-HE has a substantial overhead as contrasted with its plaintext counterpart, causing it to be unusable for various fields of applications. For non-HE models, the training phase is a computationally intensive process. It becomes increasingly more difficult with HE, even with the advancement of technologies. A current trend is to avoid the training process, favoring pre-trained modeling approaches to strike the right balance of complexity and accuracy.
2. **Parallelization:** A strategy for reducing computational overhead is to use well-known and novel parallelization technologies. DL-HE methods are amenable to modification for usage with high-performance computation and distributed designs. Multi-core modern devices such as GPU, FPGA, or tailored ASIC chips provide a more efficient and friendlier environment. Using batching and parallelization of many bootstrapping operations together helps improve the model's overall efficiency.
3. **Polynomial Approximation:** Since DL requires operations that HE does not enable, it is critical to build cryptographically appropriate replacement functions for use with encrypted data. The activation function is a critical component of a successful DL-HE model. It ensures the accuracy and computational efficiency of

the model. Additionally, the activation function significantly influences the network's convergence speed. Also, its derivative, sometimes referred to as *gradient*, is important during the learning process. Several strategies tackle the restrictions of non-linear operations by polynomial approximation with cryptographically interoperable polynomial pattern. These methods should balance between complexity and accuracy [92]. Practically, an inadequate approximation function results in a model performing poorly and taking a long time to process. Furthermore, it produces large coded messages, which consumes more memory. The main issue is to create an estimate of an activation function for a low-degree polynomial with the least amount of error and the greatest degree of precision possible.

4. **Leveled HE schemes:** Another important aim is to build schemes without bootstrapping, which support NN evaluation of pre-computed depths. Such homomorphic schemes improve the performance by reducing the complexity produced by the bootstrapping mechanism. However, this strategy restricts DL implementation. At the same time, it is efficient for the pre-computed DL model.
5. **Automatization:** Developing HE solutions demands manual customization and a high level of competence in a multitude of disciplines, including scheme refinements, parameter setting, and low-level programming. The inadequate setup might result in low performance, encryption vulnerability, and unrecoverable data. The implementation should be simple for beginners and highly customizable for advanced users.

Among other challenges that can be seen as open problems are Binary Neural Network (BNN), where a blind non-interactive DL-HE model can be achieved, and developing a common framework as a standard framework that simplifies the adoption of libraries, algorithms, and measures.

10 CONCLUSION

The deployment and implementation of deep learning techniques in a cloud environment make homomorphic cryptosystems paramount in solving security and privacy concerns. In this study, we discussed and analyzed the standard neural network structure and what modifications are required to overcome the limitations of applying the original structure to provide security and privacy. The work also addresses a trade-off between accuracy and complexity of computations associated with the replacement activity of non-linearity of activation functions to linear approximations for compatibility purposes. Several studies cover the fundamentals of homomorphic encryption, whereas others introduce novel techniques and frameworks as a building block for future considerations. The studies also examined the most recent advances in homomorphic encryption systems, concentrating primarily on a combination of cryptography and neural networks, highlighting potentials and limitations. We have covered privacy-preserving deep model's security objectives, as well as attack types and their potential solutions. The paper also reviews existing development tools, research trends, and relevant application fields. An open issue is to reduce the computational load by efficiently distributing it between client and server to acquire the most acceptable performance, which requires future investigation. Furthermore, federated learning as a foundation for PPDL implementation is a promising area of study. Future private deep learning model research should combine federated learning with conventional PPDL to address privacy concerns.

11 REFERENCES

[1] H. C. Tanuwidjaja, R. Choi, S. Baek, and K. Kim, "Privacy-preserving deep learning on machine learning as a service—a comprehensive survey," *IEEE Access*, vol. 8, pp. 167425–167447, 2020, https://doi.org/10.1109/ACCESS.2020.3023084

[2] E. Hesamifard, H. Takabi, M. Ghasemi, and C. Jones, "Privacy-preserving machine learning in cloud," *CCSW 2017 – Proc. 2017 Cloud Comput. Secur. Work. co-located with CCS 2017*, pp. 39–43, 2017, https://doi.org/10.1145/3140649.3140655

[3] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, and M. Naehrig, "Crypto-Nets: Neural Networks over Encrypted Data," pp. 1–9, 2014, [Online]. Available: http://arxiv.org/abs/1412.6181

[4] B. Pulido-Gaytan *et al.*, "Privacy-preserving neural networks with Homomorphic encryption: Challenges and opportunities," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 3, pp. 1666–1691, 2021, https://doi.org/10.1007/s12083-021-01076-8

[5] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," *Cryptol. ePrint Arch.*, pp. 1–34, 2015, https://doi.org/10.14722/ndss.2015.23241

[6] D. Demmler, T. Schneider, and M. Zohner, "ABY—A Framework for Efficient Mixed-Protocol Secure Two-Party Computation," pp. 8–11, February 2015, https://doi.org/10.14722/ndss.2015.23113

[7] L. J. M. Aslett, P. M. Esperança, and C. C. Holmes, "Encrypted statistical machine learning: New privacy preserving methods," pp. 1–39, 2015, [Online]. Available: http://arxiv.org/abs/1508.06845

[8] E. Hesamifard, H. Takabi, and M. Ghasemi, "Deep neural networks classification over encrypted data," *CODASPY 2019 – Proc. 9th ACM Conf. Data Appl. Secur. Priv.*, pp. 97–108, 2019, https://doi.org/10.1145/3292006.3300044

[9] N. Jovanovic, M. Fischer, S. Steffen, and M. Vechev, "Private and reliable neural network inference," *Proc. ACM Conf. Comput. Commun. Secur.*, pp. 1663–1677, 2022, https://doi.org/10.1145/3548606.3560709

[10] European Parliament and of the Council, "The EU General Data Protection Regulation (GDPR)," *Off. J. Eur. Communities*, vol. OJ L 119/1, no. April 2016, pp. 1–88, 2016, [Online]. Available: http://data.europa.eu/eli/reg/2016/679/oj

[11] S. A. B. Salman, S. Al-Janabi, and A. M. Sagheer, "Valid blockchain-based e-voting using elliptic curve and homomorphic encryption," *Int. J. Interact. Mob. Technol.*, vol. 16, no. 20, pp. 79–97, 2022, https://doi.org/10.3991/ijim.v16i20.33173

[12] S. Prakancharoen, "Database secure manipulation based on paillier's homomorphic encryption (DSM-PHE)," *Int. J. Interact. Mob. Technol.*, vol. 13, no. 12, pp. 136–151, 2019, https://doi.org/10.3991/ijim.v13i12.11396

[13] S. V. Monir Azraoui, Muhammad Bahram, Beyza Bozdemir, Sébastien Canard, Eleonora Ciceri, Orhan Ermis, Ramy Masalha, Marco Mosconi, Melek Önen, Marie Paindavoine, Boris Rozenberg, and Bastien Vialla, "SoK: Cryptography for Neural Networks," 2021, https://doi.org/10.1007/978-3-030-42504-3_5

[14] Craig Gentry, "A fully homomorphic encryption scheme," Stanford University, 2009.

[15] D. Beaver, "Efficient multiparty protocols using circuit randomization," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 576, no. 814, pp. 420–432, 1992, https://doi.org/10.1007/3-540-46766-1_34

[16] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via MiniONN transformations," *Proc. ACM Conf. Comput. Commun. Secur.*, pp. 619–631, 2017, https://doi.org/10.1145/3133956.3134056

[17] A. C. Yao, "Protocols for secure computations," in *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, 1982, pp. 160–164, https://doi.org/10.1109/SFCS.1982.38

[18] A. C.-C. C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (SFCS 1986)*, 1986, no. 1, pp. 162–167, https://doi.org/10.1109/SFCS.1986.25

[19] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978, https://doi.org/10.1145/359340.359342

[20] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, 1985, https://doi.org/10.1109/TIT.1985.1057074

[21] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology – 'EUROCRYPT '99*, vol. 1592, Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238, https://doi.org/10.1007/3-540-48910-X_16

[22] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference on – ITCS '12*, 2012, vol. 6, no. 3, pp. 309–325, https://doi.org/10.1145/2090236.2090262

[23] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," in *ASIACRYPT 2016: Advances in Cryptology – ASIACRYPT 2016*, 2016, vol. 10031, pp. 3–33, https://doi.org/10.1007/978-3-662-53887-6

[24] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE," in *ASIACRYPT 2017: Advances in Cryptology – ASIACRYPT 2017*, 2017, vol. 10624, pp. 377–408, https://doi.org/10.1007/978-3-319-70694-8_14

[25] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *Proc. 15th Int. Conf. Pract. Theory Public Key Cryptogr.*, 2012, pp. 1–16, [Online]. Available: https://eprint.iacr.org/2012/144

[26] L. A. R. L. Rivest and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Found. of Secure Comput.*, vol. 4, no. 11, pp. 169–180, 1978, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.500.3989&rep=rep1&type=pdf

[27] W. Liu, F. Pan, X. A. Wang, Y. Cao, and D. Tang, "Privacy-preserving all convolutional net based on homomorphic encryption," in *International Conference on Network-Based Information Systems*, 2018, pp. 752–762, https://doi.org/10.1007/978-3-319-98530-5_66

[28] V. Migliore *et al.*, "A high-speed accelerator for Homomorphic Encryption using the Karatsuba algorithm," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 5s, pp. 1–17, 2017, https://doi.org/10.1145/3126558

[29] M. Alloghani *et al.*, "A systematic review on the status and progress of homomorphic encryption technologies," *J. Inf. Secur. Appl.*, vol. 48, p. 102362, 2019, https://doi.org/10.1016/j.jisa.2019.102362

[30] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Lecture Notes in Computer Science*, 2005, vol. 3378, pp. 325–341, https://doi.org/10.1007/978-3-540-30576-7_18

[31] C. Bonte, "Optimising privacy-preserving computations," KU Leuven, 2021. [Online]. Available: https://www.esat.kuleuven.be/cosic/publications/thesis-415.pdf

[32] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Manual for using homomorphic encryption for bioinformatics," *Proc. IEEE*, vol. 105, no. 3, pp. 552–567, 2017, https://doi.org/10.1109/JPROC.2016.2622218

[33] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 6597, no. subaward 641, pp. 253–273, https://doi.org/10.1007/978-3-642-19571-6_16

[34] E. Hesamifard, H. Takabi, and M. Ghasemi, "CryptoDL: Deep Neural Networks over Encrypted Data," pp. 1–21, 2017, [Online]. Available: http://arxiv.org/abs/1711.05189

[35] H. C. Tanuwidjaja, R. Choi, and K. Kim, "A survey on deep learning techniques for privacy-preserving," in *Second International Conference, ML4CS 2019*, 2019, pp. 29–46, https://doi.org/10.1007/978-3-030-30619-9_4

[36] R. Podschwadt, D. Takabi, and P. Hu, "SoK: Privacy-preserving Deep Learning with Homomorphic Encryption," *arXiv Prepr. arXiv2112.12855*, 2021, [Online]. Available: http://arxiv.org/abs/2112.12855

[37] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015, vol. 37, pp. 448–456, [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.pdf

[38] Q. Zhang, L. T. Yang, A. Castiglione, Z. Chen, and P. Li, "Secure weighted possibilistic c-means algorithm on cloud for clustering big data," *Inf. Sci. (Ny).*, vol. 479, pp. 515–525, 2018, https://doi.org/10.1016/j.ins.2018.02.013

[39] P. Xie, B. Wu, and G. Sun, "BAYHENN: Combining Bayesian Deep Learning and Homomorphic Encryption for Secure DNN Inference," *arXiv Prepr. arXiv1906.00639*, 2019, https://doi.org/10.24963/ijcai.2019/671

[40] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *33rd International Conference on Machine Learning, ICML 2016*, 2016, vol. 1, pp. 342–351, https://doi.org/10.1109/JPROC.2016.2622218

[41] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and Emmanuel Prouff, "Privacy-preserving classification on deep neural network," *Med. Today*, vol. 4, no. 7, pp. 59–64, 2017.

[42] A. Sanyal, M. J. Kusner, A. Gascón, and V. Kanade, "TAPAS: Tricks to Accelerate (encrypted) Prediction As a Service," in *35th Int. Conf. Mach. Learn. ICML 2018*, vol. 10, pp. 7145–7154, Jun. 2018, [Online]. Available: http://arxiv.org/abs/1806.03461

[43] M. Kim and P. Smaragdis, "Bitwise Neural Networks," *CoRR*, vol. abs/1601.0, Jan. 2016, [Online]. Available: http://arxiv.org/abs/1601.06071

[44] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2018, [Online]. Available: http://arxiv.org/abs/1609.07061

[45] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Advances in Cryptology – CRYPTO 2018*, 2018, pp. 483–512, https://doi.org/10.1007/978-3-319-96878-0_17

[46] X. Jiang, M. Kim, K. Lauter, and Yongsoo Song, "Secure outsourced matrix computation and application to neural networks," in *CCS '18: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1209–1222, https://doi.org/10.1145/3243734.3243837

[47] J. Zhou, J. Li, E. Panaousis, and K. Liang, "Deep binarized convolutional neural network inferences over encrypted data," in *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2020, pp. 160–167, https://doi.org/10.1109/CSCloud-EdgeCom49738.2020.00035

[48] S. Disabato, A. Falcetta, A. Mongelluzzo, and M. Roveri, "A privacy-preserving distributed architecture for deep-learning-as-a-service," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8, https://doi.org/10.1109/IJCNN48605.2020.9207619

[49] Y. LeCun, C. Cortes, and C. J. C. Burges, "The MNIST database of handwritten digits," 1998, http://yann.lecun.com/exdb/mnist/

[50] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv Prepr. arXiv1708.07747*, pp. 1–6, 2017, [Online]. Available: http://arxiv.org/abs/1708.07747

[51] S. Obla, X. Gong, A. Aloufi, P. Hu, and D. Takabi, "Effective activation functions for homomorphic evaluation of deep neural networks," *IEEE Access*, vol. 8, pp. 153098–153112, 2020, https://doi.org/10.1109/ACCESS.2020.3017436

[52] Alex Krizhevsky and Geoffrey Hinton, "Learning Multiple Layers of Features from Tiny Images," 2009.

[53] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv Prepr. arXiv1412.6806*, pp. 1–14, 2015, https://doi.org/10.48550/arXiv.1412.6806

[54] J. Lee, E. Lee, J.-W. Lee, Y. Kim, Y.-S. Kim, and J.-S. No, "Precise Approximation of Convolutional Neural Networks for Homomorphically Encrypted Data," *arXiv Prepr. arXiv2105.10879*, 2021, https://doi.org/10.48550/arXiv.2105.10879

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2006, pp. 770–778.

[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd Int. Conf. Learn. Represent. ICLR 2015 – Conf. Track Proc.*, pp. 1–14, 2015, [Online]. Available: http://arxiv.org/abs/1409.1556

[57] C. Szegedy *et al.*, "Going deeper with convolutions Christian," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9, https://doi.org/10.1109/CVPR.2015.7298594

[58] A. Al Badawi *et al.*, "Towards the AlexNet moment for homomorphic encryption: HCNN, the first homomorphic CNN on encrypted data with GPUs," *IEEE Trans. Emerg. Top. Comput.*, vol. 9, no. 3, pp. 1330–1343, 2021, https://doi.org/10.1109/TETC.2020.3014636

[59] J.-W. Lee *et al.*, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," *IEEE Access*, vol. 10, pp. 30039–30054, 2022, https://doi.org/10.1109/ACCESS.2022.3159694

[60] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A Full RNS Variant of Approximate Homomorphic Encryption," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, J. Cid and M. J. Jacobson, Ed., vol. 11349, Springer International Publishing, 2019, pp. 347–368, https://doi.org/10.1007/978-3-030-10970-7_16

[61] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 19–38, https://doi.org/10.1109/SP.2017.12

[62] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, "Privacy preserving deep computation model on cloud for big data feature learning," *Proc. Priv. Enhancing Technol.*, vol. 2018, no. 3, pp. 123–142, 2018, https://doi.org/10.1515/popets-2018-0024

[63] S. Halevi and V. Shoup, "Design and implementation of a homomorphic-encryption library," *IBM Res.*, vol. 6, pp. 8–36, 2013, [Online]. Available: https://people.csail.mit.edu/shaih/pubs/he-library.pdf

[64] Q. Zhang, L. T. Yang, and Z. Chen, "Privacy preserving deep computation model on cloud for big data feature learning," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1351–1362, 2016, https://doi.org/10.1109/TC.2015.2470255

[65] F. Bu, Y. Ma, Z. Chen, and H. Xu, "Privacy preserving back-propagation based on BGV on cloud," in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, 2015, pp. 1791–1795. https://doi.org/10.1109/HPCC-CSS-ICESS.2015.323

[66] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," *Proc. – IEEE Symp. Secur. Priv.*, pp. 3–18, 2017, https://doi.org/10.1109/SP.2017.41

[67] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Oct. 2015, pp. 1322–1333. https://doi.org/10.1145/2810103.2813677

[68] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," *Proc. 25th USENIX Secur. Symp.*, vol. abs/1609.0, no. Ml, pp. 601–618, 2016, [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer

[69] T. Dugan and X. Zou, "Privacy-preserving evaluation techniques and their application in genetic tests," *Smart Heal.*, vol. 1–2, pp. 2–17, 2017, https://doi.org/10.1016/j.smhl.2017.03.003

[70] F. Armknecht *et al.*, "A guide to fully homomorphic encryption," vol. 1192, pp. 1–35, 2015, [Online]. Available: https://ia.cr/2015/1192

[71] S. Halevi, "Homomorphic encryption," in *Tutorials on the Foundations of Cryptography*, 1st ed., Yehuda Lindell, Ed. Springer, Cham, pp. 219–276, 2017, https://doi.org/10.1007/978-3-319-57048-8_5

[72] J. Sen, "Homomorphic encryption: Theory & Application," *Theory Pract. Cryptogr. Netw. Secur. Protoc. Technol.*, vol. 31, 2013, https://doi.org/10.5772/56687

[73] D. Archer *et al.*, "Applications of homomorphic encryption," in *Crypto Standardization Workshop, Microsoft Research*, p. 14, 2017.

[74] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library – SEAL v2.1.," in *International Conference on Financial Cryptography and Data Security*, 2017, pp. 3–18, https://doi.org/10.1007/978-3-319-70278-0_1

[75] M. Bakshi and M. Last, "CryptoRNN – Privacy-preserving recurrent neural networks using homomorphic encryption," in *International Symposium on Cyber Security Cryptography and Machine Learning*, 2020, vol. 12161 pp. 245–253, https://doi.org/10.1007/978-3-030-49785-9_16

[76] E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, and L. Fei-fei, "Faster CryptoNets: Leveraging sparsity for real-world encrypted inference," 2018, [Online]. Available: https://arxiv.org/abs/1811.09953

[77] A. A. L. Badawi, L. Hoang, C. F. Mun, K. Mi, and M. I. Aung, "PrivFT: Private and fast text classification with homomorphic encryption," vol. 8, pp. 226544–226556, 2020, https://doi.org/10.1109/ACCESS.2020.3045465

[78] R. Dathathri *et al.*, "CHET: An optimizing compiler for fully-homomorphic neural-network inferencing," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2019, pp. 142–156, https://doi.org/10.1145/3314221.3314628

[79] R. Podschwadt and D. Takabi, "Classification of encrypted word embeddings using recurrent neural networks," in *PrivateNLP@ WSDM*, pp. 27–31, 2020.

[80] K. Nandakumar, N. Ratha, S. Pankanti, and S. Halevi, "Towards deep neural network training on encrypted data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 40–48, https://doi.org/10.1109/CVPRW.2019.00011

[81] A. Kawachi, K. Tanaka, and K. Xagawa, "Multi-bit cryptosystems based on lattice problems," in *International Workshop on Public Key Cryptography*, pp. 315–329, 2007, https://doi.org/10.1007/978-3-540-71677-8_21

[82] T. Sander, A. Young, and Moti Yung, "Non–Interactive CryptoComputing for NC 1," in *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, 1999, pp. 554–566, https://doi.org/10.1109/SFFCS.1999.814630

[83] J. H. C. B, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security.*, 2017, pp. 409–437, https://doi.org/10.1007/978-3-319-70694-8_15

[84] Yuriy Polyakov, Kurt Rohloff, Gerard W. Ryan, and Dave Cousins, "Palisade lattice cryptography library user manual (v1.11.6)," 2022, https://palisade-crypto.org/community/

[85] W. Dai and B. Sunar, "cuHE: A homomorphic encryption accelerator library," in *BalkanCryptSec 2015: Cryptography and Information Security in the Balkans*, pp. 169–186, 2016, https://doi.org/10.1007/978-3-319-29172-7_11

[86] Léo Ducas and D. Micciancio, "FHEW: Bootstrapping homomorphic encryption in less than a second," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques EUROCRYPT 2015: Advances in Cryptology*, 2015, pp. 617–640, https://doi.org/10.1007/978-3-662-46800-5

[87] CryptoExperts, "FV-NFLlib," 2016, https://github.com/CryptoExperts/FV-NFLlib (accessed Feb. 23, 2022).

[88] C. Mouchet, J.-P. Bossuat, J. Troncoso-Pastoriza, J. Hubaux, and Jean-Pierre Hubaux, "Lattigo: A multiparty homomorphic encryption library in go," in *WAHC 2020–8th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, vol. 15, 2020, [Online]. Available: https://homomorphicencryption.org/wp-content/uploads/2020/12/wahc20_demo_christian.pdf

[89] I. Chillotti, M. Joye, D. Ligier, J.-B. Orfila, and S. Tap, "CONCRETE: Concrete operates on ciphertexts rapidly by extending TfhE," in *WAHC 2020–8th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, vol. 15, 2020, [Online]. Available: https://homomorphicencryption.org/wp-content/uploads/2020/12/wahc20_demo_damien.pdf

[90] NuCypher, "nufhe," 2019, https://github.com/nucypher/nufhe (accessed Feb. 23, 2022).

[91] A. Viand, P. Jattke, and Anwar Hithnawi, "SoK: Fully homomorphic encryption compilers," in *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1092–1108, 2021, https://doi.org/10.1109/SP40001.2021.00068

[92] J. M. Cortés-Mendoza *et al.*, "Privacy-Preserving Logistic Regression as a Cloud Service based on Residue Number System," in *Russian Supercomputing Days: RuSCDays 2020: Supercomputing*, pp. 598–610, 2020, https://doi.org/10.1007/978-3-030-64616-5_51

12 AUTHORS

Anmar A. Al-Janabi is a Ph.D. student in the Computer Science and Information Technology Department, University of Anbar. He currently works as a faculty member and lecturer at the Computer Sciences Department, University of Technology—Iraq, Baghdad, Iraq. He earned his B.Sc. from the University of Baghdad, Iraq, in 2003 and his M.Sc. in Computer Science from Al-Balqa' Applied University, Jordan, in

2010. His current research interests include information security, data privacy, and deep learning (email: anmar.a.aljanabi@uotechnology.edu.iq).

Sufyan T. Faraj Al-Janabi received his B.Sc., M.Sc., and Ph.D. in Electronic and Communications Engineering from Al-Nahrain University in Baghdad in 1992, 1995, and 1999, respectively. He became a faculty member in the Computer Engineering Dept. at the University of Baghdad in 1999, and served as the department head in 2001. He was also the Dean of College of Information Technology at Al-Nahrain University from May 2004 to June 2006, and the Dean of College of Computer Science and Information Technology at University of Anbar, Ramadi from July 2006 to May 2010. His research interests include internet protocols, information security, and quantum cryptography. He has received several awards including the 1st Award for the Best Research Paper in Information Security from the Association of Arab Universities in Jordan in 2003, and fellowships from ISEP in 2009 and Fulbright in 2010, both in the USA. He is a member of ACM, ASEE, IACR, and IEEE (email: sufyan.aljanabi@uoanbar.edu.iq).

Belal Al-Khateeb received the B.Sc. (first class) degree in computer science from Al-Nahrain University, Baghdad, Iraq, in 2000, and the M.Sc. degree in computer science from Al-Nahrain University, Baghdad, Iraq, in 2003, and the Ph.D. degree from the School of Computer Science, University of Nottingham, Nottingham, U.K., in 2011. He is currently a professor at the College of Computer Science and Information Technology, University of Anbar. He has published over 82 refereed journal and conference papers. His current research interests include deep learning, particularly in health care, evolutionary and adaptive learning particularly in computer games, expert systems, and heuristics and meta/hyper-heuristics. He has a particular interest in computer games programming. Prof. Al-Khateeb is a reviewer of twenty-three international journals (including many Clarivate journals) and thirty conferences (email: belal-alkhateeb@uoanbar.edu.iq).