Machine Learning over Encrypted Data: A Survey of Fully Homomorphic Encryption for AI Systems

Dillon Davidson[1], Matthew Jackson[1], Rasheed Hussain[2], Zuobin Xiong[1], Junggab Son[1*]

[1*]Department of Computer Science, University of Nevada Las Vegas, Maryland Parkway, Las Vegas, 89154, Nevada, USA.
[2]Smart Internet Lab, University of Bristol, United Kingdom.

*Corresponding author(s). E-mail(s): junggab.son@unlv.edu;
Contributing authors: davidd11@unlv.nevada.edu; jacksm28@unlv.nevada.edu; rasheed.hussain@bristol.ac.uk; zuobin.xiong@unlv.edu;

Abstract

The increasing use of machine learning (ML) in privacy-sensitive domains such as healthcare, finance, and user-facing services has intensified the need for techniques that enable data-driven intelligence without exposing sensitive information. Fully Homomorphic Encryption (FHE) offers a cryptographically rigorous approach to privacy-preserving machine learning by allowing computations to be performed directly on encrypted data. While appealing in principle, the integration of FHE into modern ML pipelines introduces substantial challenges that fundamentally affect model design, algorithm selection, and system performance. This survey provides a comprehensive and structured review of machine learning over encrypted data using non-interactive FHE-based approaches. We examine how a wide range of machine learning models, including traditional ML algorithms, deep neural networks, and emerging transformer architectures, are adapted to operate under the arithmetic, precision, and efficiency constraints imposed by FHE. To organize a rapidly growing body of work, we introduce a unifying taxonomy that captures key design dimensions such as encryption schemes, nonlinearity handling strategies, model families, system-level optimizations, and privacy objectives. Using this framework, we synthesize existing approaches, analyze trade-offs between accuracy, efficiency, and security, and highlight recurring design patterns across the literature. Beyond summarizing existing techniques, this survey discusses practical limitations, deployment considerations, and open research challenges that shape the future of privacy-preserving machine learning

under encryption. By bridging cryptographic techniques and modern AI system design, this work aims to serve as a reference for researchers and practitioners seeking to develop secure, scalable, and trustworthy machine learning systems.

Keywords: Machine Learning, Fully Homomorphic Encryption, Privacy-Preserving Machine Learning, Encrypted Computation, AI Systems

1 Introduction

Machine learning (ML) has become a foundational technology across a wide range of application domains, leveraging large-scale data and powerful computational resources to enable increasingly accurate prediction, classification, and decision-making [1–3]. From healthcare and finance to user-facing services, ML systems play a central role in extracting value from data and supporting high-impact, mission-critical tasks [4, 5]. However, deploying ML in privacy-sensitive settings, where training and inference rely on highly confidential information [6, 7], introduces significant challenges that must be addressed to ensure secure and trustworthy adoption [8, 9]. In domains such as communications, medicine, and national security, sensitive data is often indispensable for achieving high model performance yet must remain strictly protected throughout the ML life cycle [10–12].

The use of such sensitive data exposes ML systems to a broad and evolving threat landscape. Without adequate safeguards, these systems may become vulnerable to data breaches, unauthorized disclosures, or adversarial manipulation that compromise both privacy and model integrity. This raises fundamental questions: *How can sensitive data be processed securely? How can confidentiality be preserved in the presence of untrusted infrastructure or adversarial actors?* Prior work has identified numerous attack vectors targeting ML models and pipelines, including model inversion attacks [13–16], model extraction [17], training data reconstruction [18], and adversarial example generation [19]. In parallel, ML systems remain susceptible to traditional cybersecurity threats such as data poisoning [20], man-in-the-middle attacks [21], server breaches [22], and insecure application programming interfaces [23]. These vulnerabilities can be exploited to intercept, alter, or exfiltrate sensitive data during training, deployment, or inference. As ML continues to underpin critical applications, including healthcare delivery, financial services, national security, and criminal justice, addressing this multifaceted threat landscape is essential for protecting individual privacy, ensuring system robustness, and maintaining institutional trust.

To mitigate these risks, researchers have explored a range of privacy-preserving machine learning techniques. Federated Learning (FL) distributes training across client devices and exchanges model updates rather than raw data [24]. However, such updates may still leak sensitive information, rendering FL vulnerable to gradient leakage and inversion attacks [25]. Differential Privacy (DP) introduces statistical noise during training to limit the influence of individual data records [26–28]. While DP offers strong theoretical guarantees and scalability, it often incurs non-negligible

accuracy degradation, particularly in deep learning settings. Secure Multi-Party Computation (SMPC) enables joint computation over private inputs without revealing them to participating parties, but typically requires extensive communication and synchronization, leading to significant system overhead [29–32].

In contrast, Fully Homomorphic Encryption (FHE) provides a fundamentally different paradigm for privacy-preserving ML. FHE enables arbitrary computation to be performed directly on encrypted data without requiring decryption, thereby ensuring that sensitive information remains hidden throughout the computation process. Unlike FL or SMPC, FHE operates in a non-interactive manner, eliminating the need for repeated communication between data owners and computation servers and improving scalability in outsourced settings. Modern FHE schemes, such as Cheon–Kim–Kim–Song (CKKS), offer provable cryptographic security guarantees [33], including INDistinguishability under Chosen-Plaintext Attack (IND-CPA), provided that decryption outputs remain private [34]. These strong guarantees position FHE as a compelling defense against a wide range of data leakage threats, shifting the focus from algorithm-specific mitigations to a mathematically grounded approach to secure computation.

Despite its promise, integrating FHE into practical ML systems presents substantial technical challenges. A fundamental limitation is that FHE schemes natively support only basic arithmetic operations, making them incompatible with commonly used nonlinear components such as ReLU, sigmoid, and tanh (Section 3.2). In addition, homomorphic operations are significantly more resource-intensive than their plaintext counterparts, incurring high computational and memory overhead—constraints that are particularly pronounced for deep learning models, which already demand substantial resources even without encryption.

As a result, much of the existing literature focuses on encrypted inference, which requires only forward propagation through the model. Extending FHE-based approaches to support full model training remains considerably more challenging, as it requires both forward and backward passes while managing ciphertext noise growth. Achieving practical, scalable encrypted training continues to be an open and active area of research.

This survey provides a comprehensive overview of machine learning over fully homomorphic encryption (FHE-ML). We examine the cryptographic foundations underpinning FHE, analyze key algorithmic and system-level trade-offs, and review state-of-the-art approaches for encrypted inference and training across a range of ML models. By synthesizing recent advances, practical challenges, and emerging research directions, this work aims to inform and guide researchers and practitioners seeking to develop secure, privacy-preserving machine learning systems under strong cryptographic guarantees.

1.1 Scope and Contributions

In this survey, we provide a comprehensive overview of recent research on performing ML directly over homomorphically encrypted data. Our focus is specifically on non-interactive HE-based ML approaches, which offer strong cryptographic privacy

guarantees while avoiding the communication and synchronization overhead inherent in interactive protocols. While secret sharing approaches offer robust privacy for distributed ML, they require interactive protocols; we delineate their relationship to non-interactive FHE in Section 1.2. By restricting attention to non-interactive settings, this survey concentrates on techniques that are particularly well suited for outsourced and cloud-based ML deployment scenarios.

Within this scope, we examine a broad range of machine learning models, including both traditional ML algorithms and modern deep learning architectures, and analyze how they are adapted to operate on encrypted inputs. We consider both inference and training settings and discuss how homomorphic encryption can be used to protect sensitive data throughout the ML pipeline, as well as, in some cases, preserve model confidentiality. Particular emphasis is placed on the algorithmic and system-level challenges that arise when executing ML workloads under encryption, including the limited arithmetic expressiveness of HE schemes and the resulting difficulty of implementing nonlinear components such as activation functions and normalization layers.

This survey focuses primarily on non-interactive FHE–based machine learning, including both bootstrapped and leveled settings. While other classes of homomorphic encryption such as somewhat and partially homomorphic encryption exist, they are not the focus of this work, as their limited expressiveness restricts applicability to modern machine learning workloads.

To distinguish this work from prior literature, our survey introduces the following novel contributions:

- **A Bottleneck-to-Workaround Taxonomy:** While we utilize standard dimensions such as scheme family and model type to organize the literature, the core novelty of our taxonomy is mapping universal plaintext architectural bottlenecks (e.g., nonlinearities, control flow) directly to their required FHE cryptographic workarounds (see Table 3). This frames FHE-ML as a co-design problem between the complexity of modern ML and the limitations of current FHE.
- **Coverage of Modern Architectures:** Unlike prior surveys that stop at standard deep learning models, we extend our analysis to the rapidly evolving frontier of Transformer architectures. We critically examine the specific cryptographic bottlenecks introduced by attention mechanisms and normalization layers.
- **Critical Assessment of Applicability:** Rather than merely cataloging state-of-the-art implementations, we evaluate the persistent gap between theoretical runtime and practical deployment, explicitly demarcating established inference techniques from the currently impractical domain of general encrypted training.

1.2 Relationship to Secret Sharing and Threshold FHE

While this survey focuses strictly on non-interactive Fully Homomorphic Encryption (FHE), it is essential to position these frameworks within the landscape of distributed cryptography. In particular, cryptographic primitives rooted in secret sharing are highly complementary to FHE architectures, rather than a replacement for them. Standard secret sharing schemes [35] divide sensitive data into discrete shares

distributed across multiple independent nodes. While exceptionally efficient for executing linear algebra operations, evaluating the arbitrary, nonlinear machine learning circuits over purely secret-shared data inherently requires interactive Multi-Party Computation (MPC) protocols [36], where nodes must persistently communicate during runtime. Consequently, secret sharing alone does not satisfy the non-interactive execution paradigm—where a client encrypts data, goes offline, and allows an honest-but-curious server to compute arbitrary ML functions asynchronously.

Instead, approaches in secret sharing serve as system-level primitives for managing key infrastructure, threshold decryption, multi-user access control, and hybrid privacy-preserving machine learning. For instance, threshold FHE architectures leverage secret sharing to distribute the master secret key across a federation of data silos or cloud servers [37]. This ensures that no single entity can unilaterally decrypt client ciphertexts, requiring a cryptographic consensus instead. Furthermore, long-lived, distributed trust systems utilize proactive secret sharing [38], which allows nodes to periodically refresh their secret shares over time.

In dynamic, multi-client cloud environments, secret-sharing innovations are also applied to identity and access management. Primitives such as one-time user keys for multi-user secret sharing enable flexible, fine-grained access control, allowing a central system to distribute transient computation keys without necessitating the global re-encryption of the master dataset [39, 40]. While these distributed trust mechanisms, proactive refreshes, and multi-user key management strategies are crucial for scaling FHE systems to production FHE deployments, they fundamentally focus on interactive multi-party key infrastructures and collaborative decryption phases. Since these mechanisms operate at the key management and decryption layer rather than within the ciphertext domain, they fall outside the scope of this survey.

1.3 Literature Search Methodology

To ensure a comprehensive and representative overview of the field, we employed a targeted search strategy across major academic databases IEEE Xplore, ACM Digital Library, and Google Scholar as well as prominent preprint repositories, specifically arXiv and IACR ePrint archive, to capture the rapidly evolving frontier of this research.

Our primary search queries were constructed by pairing cryptographic keywords (e.g., “fully homomorphic encryption", “FHE", “CKKS", “TFHE") with machine learning terminology (e.g., “deep learning", “neural networks" and specific architectures such as “decision trees" or “transformers").

To filter the results and maintain the scope defined in Section 1.1, we applied the following criteria during our screening process:

- **Temporal Scope**: We restricted our search to literature published or uploaded between January 2019 and January 2026. This window was chosen to capture the modern era of FHE-ML optimization, particularly following the widespread adoption and implementation of approximate arithmetic schemes (CKKS) and programmable bootstrapping (TFHE).

- **Inclusion Criteria**: Works were included if they explicitly proposed, optimized, or empirically evaluated machine learning models (training and/or inference) operating natively over homomorphically encrypted data.
- **Exclusion Criteria**: We excluded protocols that circumvent the limitations of FHE by relying on interactive Multi-Party Computation (MPC), client-side assisted computation, or legacy Partially/Somewhat Homomorphic Encryption (PHE/SHE) schemes without bootstrapped capabilities.

Candidate papers were screened by title and abstract and then categorized after full-text review by the authors using the inclusion/exclusion criteria and taxonomy defined in this survey. Categorization decisions were reviewed jointly by the author team, and disagreements were resolved through discussion until consensus was reached.

1.4 Existing Surveys

In recent years, a growing number of surveys have examined topics related to privacy-preserving machine learning (PPML) and homomorphic encryption (HE). However, most existing surveys tend to address these areas independently rather than focusing on the integrated problem of executing machine learning models directly over homomorphically encrypted data, which is the central focus of this work. As a result, the literature lacks a unified perspective on the algorithmic and system-level challenges that arise when machine learning and homomorphic encryption are combined.

Broadly, prior surveys can be grouped into two categories: those centered on PPML and those centered on HE. Surveys in the PPML category typically provide broad overviews of privacy-enhancing techniques such as Differential Privacy [41, 42], Federated Learning [43], Secure Multi-Party Computation [44], and related approaches [44–47]. In these works, homomorphic encryption is often discussed as one of several possible tools for achieving privacy, but it is rarely examined in depth. Consequently, important aspects such as the mathematical foundations of HE, the properties of modern encryption schemes, and the practical implications of deploying machine learning models over encrypted data are typically outside the scope of these surveys.

In contrast, surveys focused on homomorphic encryption provide detailed treatments of cryptographic constructions, security guarantees, and implementation techniques [48–55]. While these works offer valuable insights into the theory and practice of HE, machine learning is usually considered as one application among many rather than as a primary focus. As a result, challenges specific to privacy-preserving machine learning—such as supporting nonlinear operations, deep neural networks, and modern architectures under encryption—are not explored in a systematic, application-driven manner.

This survey seeks to bridge this gap by providing a focused and comprehensive review of homomorphic encryption in the context of machine learning. By treating machine learning over encrypted data as a unified design problem, we examine how cryptographic constraints influence model architectures, algorithm design, and system-level optimizations. A comparison between this survey and existing related surveys is summarized in Table 1.

Table 1: Related Surveys from 2019 to 2026

Focus	Paper	Year	FHE	LR	DT	SVM	NB	NN	TF
PPML	[44]	2020	✓	✗	✗	✗	✗	✓	✗
	[42]	2020	✓	✗	✗	✗	✗	✓	✗
	[43]	2020	✓	✗	✗	✗	✗	✓	✗
	[45]	2021	✓	✗	✗	✗	✗	✓	✗
	[47]	2022	✓	✗	✗	✗	✗	✓	✗
	[41]	2021	✓	✗	✓	✗	✓	✓	✗
	[53]	2023	✓	✓	✓	✗	✓	✓	✗
	[46]	2024	✓	✗	✗	✗	✗	✓	✗
FHE	[48]	2023	✓	✗	✗	✗	✗	✗	✗
	[50]	2022	✓	✗	✗	✗	✗	✗	✗
	[51]	2019	✓	✗	✗	✗	✗	✗	✗
	[54]	2023	✓	✓	✓	✓	✓	✗	✗
	[55]	2023	✓	✗	✗	✗	✗	✗	✗
Both	[52]	2020	✓	✓	✓	✗	✓	✓	✗
	[49]	2021	✓	✓	✓	✗	✗	✓	✗
	Ours	2026	✓	✓	✓	✓	✓	✓	✓

PPML: Privacy-Preserving Machine Learning, **FHE**: Fully Homomorphic Encryption, **LR**: Logistic Regression, **DT**: Decision Tree, **SVM**: Support Vector Machine, **NB**: Naive Bayes, **NN**: Neural Network, **TF**: Transformer.

1.5 Organization

The remainder of this paper is organized as follows. Section 2 reviews essential preliminaries on homomorphic encryption. Section 3 introduces the taxonomy that structures this survey. Sections 4 and 5 examine homomorphic-encryption-based implementations of traditional ML, deep learning, and transformer models. Section 6 discusses representative real-world applications of machine learning over encrypted data. Section 7 outlines open challenges and future research directions, and Section 8 concludes the paper.

1.6 Related Privacy Paradigms

FHE is not the only approach to privacy-preserving machine learning. A prominent alternative is federated learning (FL), in which training data remains local and only model updates are sent to a central server. This avoids centralizing raw data, but the updates themselves may leak sensitive information [56]. FHE addresses a different part of the problem: rather than keeping data local, it keeps data encrypted throughout computation, so that even the party performing the computation cannot see it. FL and FHE therefore protect different things: FL keeps raw data from leaving the client, while FHE keeps data unreadable to whoever processes it.

This distinction raises a natural question: can cryptographic protection be brought into FL itself, rather than treated as a separate alternative? Recent work suggests yes. Secret sharing, threshold HE, verifiable aggregation, and threshold signatures have been used to protect model updates, tolerate client dropout, and detect or prevent

Table 2: Notations

Notation	Definition
λ	Security parameter indicates the number of bits of security for the scheme
pk	Public key used for encryption
sk	Secret key used for decryption
evk	Evaluation key used for computations on encrypted data
m	Message (or plaintext) to be encrypted
c	Ciphertext corresponding to the message m
$\mathbf{x}$	A vector, denoted in bold lowercase
$\mathbf{X}$	A matrix, denoted in bold uppercase

malicious aggregation [57–60]. Threshold HE is the clearest example: it integrates directly into the FL aggregation step, letting FL keep its distributed training structure while limiting what the aggregator can learn from the updates it receives [58]. The two paradigms are therefore not competitors so much as answers to different questions: FL determines where training happens, while cryptographic mechanisms like threshold HE constrain what can be learned from the results.

2 The Nuts and Bolts of Homomorphic Encryption

In this section, we introduce the basics of homomorphic encryption along with the notation that will be used henceforth in this paper. We use the term fully homomorphic encryption–based machine learning (FHE-ML) to refer to machine learning models whose training or inference is performed directly on encrypted data. All logarithms are base 2 unless otherwise stated. For convenience, we note that the security parameter is denoted by λ throughout the paper. Vectors are denoted in bold lowercase (e.g., $\mathbf{x}$) and matrices are denoted in bold uppercase (e.g., $\mathbf{X}$).

2.1 Homomorphic Encryption (HE)

Homomorphic encryption was first proposed by Rivest et al. in 1978 as a way to perform operations on encrypted data without first decrypting it [61]. To begin with, let us start with some public key encryption scheme. A public key encryption scheme allows messages to be sent between two or more users without requiring some shared secret key to be established beforehand. Every user runs a key generation algorithm KeyGen to obtain a key pair: $(pk, sk) \xleftarrow{R} \mathsf{KeyGen}()$, where pk is a public key and sk is a secret key. Next, the encryption function $\mathsf{Encrypt}$ transforms a plaintext message into a ciphertext with the public key pk: $c \xleftarrow{R} \mathsf{Encrypt}(pk, m)$, where c is the ciphertext produced by encrypting the message m with the public key pk. Finally, the decryption function $\mathsf{Decrypt}$ transforms a ciphertext into the original plaintext message: $m \leftarrow \mathsf{Decrypt}(sk, c)$, where m is the plaintext message produced by decrypting the ciphertext c with the secret key sk corresponding to the public key pk.

A public key scheme is said to be homomorphic if it allows computation on ciphertexts without requiring decryption. More formally, we claim that for every valid input, the following properties hold:

$$\mathsf{Decrypt}(sk, \mathsf{Encrypt}(pk, a) \oplus \mathsf{Encrypt}(pk, b)) = a + b,$$
$$\mathsf{Decrypt}(sk, \mathsf{Encrypt}(pk, a) \otimes \mathsf{Encrypt}(pk, b)) = a \times b,$$

where $\oplus$ denotes the homomorphic addition operation and $\otimes$ denotes the homomorphic multiplication operation.

There are four types of homomorphic encryption with different levels of complexity and capability [62]. Partially homomorphic encryption is the weakest of the bunch and supports one specific operation, either addition or multiplication. The RSA scheme is an example as it supports an unbounded number of multiplications [63]. Somewhat homomorphic encryption provides a stronger property, the ability to handle a limited number of both addition and multiplication operations, Sander et al. proposed one such scheme in 1999 [64]. Leveled fully homomorphic encryption supports the evaluation of operations up to a certain depth that is defined by the security parameters. The BGV scheme introduced by Brakerski et al. was among the first practical leveled fully homomorphic scheme [65]. Fully homomorphic encryption supports the unbounded evaluation of operations over the ciphertexts without compromising security. This is due to the breakthrough development of the bootstrapping operation that was introduced by Gentry in 2009 [66].

2.2 Fully Homomorphic Encryption (FHE)

Fully homomorphic encryption (FHE) has earned the title of the "holy grail" of cryptography because it allows an unlimited number of computations to be performed over ciphertexts without sacrificing security. The first fully homomorphic encryption scheme was proposed by Gentry in 2009 [66]. Gentry showed how to construct an encryption scheme that handles the evaluation of arbitrary circuits, with the key breakthrough development of the bootstrapping operation. In non-fully homomorphic encryption schemes, every homomorphic operation increases the ciphertext's error term, imposing a built-in limit on the number of operations that can be performed before decryption fails. The bootstrapping operation refreshes the error term in the ciphertext by homomorphically decrypting and re-encrypting the ciphertext. During this process, no information is leaked and the ciphertext is "refreshed" which allows an arbitrary number of operations on a ciphertext as long as the bootstrapping operation is performed. The trade-off for this unlimited computational depth is a tremendous computational cost. Gentry's initial implementation was not computationally feasible. However, after substantial improvements to the bootstrapping operation, fully homomorphic encryption is increasingly practical [67, 68]. The bootstrapping operation is the most expensive component of any fully homomorphic encryption scheme, but is feasible to compute.

2.3 Homomorphic Encryption Schemes

In this section, we introduce the schemes used in this survey. HE schemes can be either "exact" or "approximate", referring to the correctness of the homomorphic computations. In this survey we focus on CKKS and TFHE. These are by no means the only available schemes, but we consider only them as they are the most popular and well performing of the available schemes.

The CKKS scheme supports approximate real/complex arithmetic with SIMD-style vector packing, making linear algebra operations like matrix-vector multiplications and dot products, common in machine learning, relatively efficient. The main trade offs are an approximation error and the lack of native support for non-polynomial operations. In contrast, the TFHE scheme offers exact computation and programmable bootstrapping, enabling efficient evaluation of non-polynomial operations. However, TFHE typically operates at the bit level, which makes it slower and less compact than CKKS for linear algebra workloads.

2.3.1 Cheon, Kim, Kim, and Song (CKKS)

The CKKS scheme is an approximate arithmetic FHE scheme based on the ring learning with errors problem proposed in 2016 [33]. A distinctive feature of CKKS is its ability to work with real and complex values. Initially introduced as a leveled FHE scheme, CKKS was later enhanced with bootstrapping techniques in 2018 to support FHE [69]. Its ability to handle floating point numbers and efficient computations make CKKS a strong candidate for machine learning tasks. The residue number system (RNS) version of CKKS is commonly used as it enhances the precision and efficiency [70]. This modification allows for faster operations with larger precision, which is especially beneficial for applications like machine learning.

2.3.2 Fully Homomorphic Encryption over the Torus (TFHE)

The TFHE scheme is an exact FHE scheme based on the learning with errors (LWE) and ring learning with errors problem [71–73]. Bootstrapping for TFHE results in no information loss and can execute non-polynomial functions while refreshing the ciphertext with programmable bootstrapping.

2.4 Threat Model

Modern homomorphic encryption schemes aim to achieve semantic security, formally defined as indistinguishability against chosen plaintext attacks (IND-CPA) [74].

In the context of FHE-based machine learning, we typically consider two parties: a data owner and a compute server, which we refer to as the client and the server for simplicity. The client encrypts sensitive input data using an FHE scheme and delegates computation to the server, which performs the homomorphic evaluation without access to the plaintext data. Depending on the application, the encrypted data may be input features for an inference, training data, or training labels.

The server is assumed to be honest-but-curious: it correctly follows the prescribed protocol but may attempt to infer information about the client's data from the ciphertexts it observes. Under this assumption, IND-CPA security guarantees that the

server learns no information about the plaintext inputs beyond what is revealed by the final decrypted output. We assume that the adversary does not obtain access to a decryption oracle or to intermediate decrypted values.

A malicious client, in contrast, could submit malformed ciphertexts in an attempt to extract information about the server's proprietary model or to induce incorrect computations. Mitigating such attacks typically requires additional safeguards beyond basic IND-CPA security, including input validation, circuit privacy, or verifiable computation and is not considered the goal of FHE.

While standard IND-CPA security protects data confidentiality during homomorphic computation under honest-but-curious and passive decryption assumptions, real-world deployments introduce threat vectors that fall outside this baseline:

- **Decryption-Result Feedback (IND-CPA^D Security):** Standard IND-CPA does not hold when an adversary can observe decrypted outputs or interact with a decryption oracle. In approximate encryption schemes like CKKS, the output error carries implicit information about the secret key and repeatedly observing decrypted results enables key-recovery attacks [34, 75, 76]. Securing against these decryption feedback attacks requires IND-CPA^D security, which is typically achieved via post-computation noise mechanisms such as noise flooding or differential privacy noise insertion applied to plaintexts prior to decryption.
- **Computation Integrity against Malicious Servers:** Standard FHE guarantees data confidentiality during execution but provides no guarantee of computation integrity. A malicious server could inject faulty ciphertexts or alter execution paths undetected. Countering active server adversaries requires combining FHE with verifiable computation techniques such as Zero-Knowledge Proofs (ZKPs), Message Authentication Codes (MACs), or Trusted Execution Environments (TEEs) [77].
- **Implementation Side Channels:** Even when FHE schemes provide mathematical security, physical execution traces (such as power consumption, memory access patterns, or timing variations during operations like the Number Theoretic Transform) can leak private key material or plaintext data [78].
- **Membership Inference and Training-Data Leakage:** Independent of cryptographic security, the decrypted output of an ML model can leak information about the underlying training set. Membership inference attacks can extract sensitive training data even in strict black-box settings, with no access to model weights or intermediate ciphertexts [79]. Mitigating this risk requires application-layer differential privacy (e.g., DP-SGD or output perturbation) to bound statistical leakage and is not solved by FHE.
- **Model Extraction and Model Privacy:** Repeated queries to an inference pipeline can allow an adversary to reconstruct proprietary model parameters through black-box extraction [80]. Preventing output ciphertexts from leaking details about model parameters evaluated inside the encryption circuit requires circuit privacy techniques [81].
- **Query Privacy:** Structural query patterns or memory access indices can reveal sensitive user intent even when underlying payload data remains encrypted. Protecting these access patterns requires combining FHE with Private Information Retrieval (PIR) schemes [82] or Oblivious RAM (ORAM) protocols [83].

Unless otherwise noted, the remainder of this survey adopts standard IND-CPA as the baseline security assumption, consistent with the honest-but-curious, no-decryption-oracle model described above. The threat vectors listed above are deployment-dependent extensions that a practitioner may need another layer on top of this baseline, rather than assumptions made throughout the survey itself.

3 Taxonomy and Design Space of FHE-ML

We introduce a taxonomy to organize FHE-ML systems along five orthogonal dimensions: (1) FHE scheme family, (2) nonlinearity strategy, (3) model family, (4) system-level optimization levers, and (5) privacy scope. This design space provides a unified lens for comparing systems that otherwise appear incomparable due to differing cryptographic assumptions, hardware platforms, and cost accounting practices. Table 3 synthesizes the core contribution of our taxonomy, abstracting the disparate literature into a unified map of plaintext bottlenecks and their accepted FHE solutions.

Table 3: Bottleneck-to-Solution Map for FHE-ML

Plaintext Bottleneck	**Encrypted Solution / Workaround**
Sigmoid / Tanh Activations	Low-degree polynomial approximation (e.g., Chebyshev, minimax)
Comparisons (Decision Trees)	Programmable Bootstrapping (PBS) / LUTs (TFHE) or polynomial sign approximation (CKKS)
Convolutions / Linear Layers	Multiplexed parallel packing, SIMD batching, and rotation minimization
ReLU / Max Pooling	Exact evaluation via Boolean gates / PBS (TFHE), or replacement with square/polynomial functions
Softmax	Polynomial / Gaussian kernel approximation, or attention replacement
LayerNorm	Polynomial approximation, dynamic range scaling, or complete architecture redesign

3.1 FHE Scheme Family

Choosing an FHE scheme family is one of the most consequential design decisions in FHE-ML, as it determines the supported plaintext domain, the cost model of arithmetic operations, and the feasibility of nonlinear functions. We broadly group the literature into two scheme families:

- **CKKS (approximate arithmetic):** Supports approximate computations over real/complex numbers and is therefore widely used for neural networks and other models with real-valued activations. CKKS enables efficient packed SIMD linear algebra but introduces approximation error and requires careful scale management and rescaling.

Table 4: Taxonomy of representative FHE-ML systems.

Work	FHE	Nonlinearity	Model	Key optimizations	Privacy scope
[84]	CKKS	Polynomial	Linear	Bootstrap Management	Data
[85]	TFHE	PBS	Tree/Forest	MatMul Flattening	Data
[86]	CKKS	Polynomial	CNN	Packing & Bootstrapping	Data
[87]	TFHE	PBS	CNN	QAT and Pruning	Data
[88]	CKKS	Polynomial	Transformer	Approx. Attention	Data
[89]	CKKS	Gaussian Kernel	Transformer	LoRA Adapters	Data

Table 5: Comparison of different FHE schemes.

Scheme	**Computational Efficiency**	**Numerical Precision**	**Nonlinear Op. Handling**	**Suitable ML Models**
CKKS	High SIMD throughput; costly bootstrapping	Approximate (floating point)	Polynomial approximation	DNNs, CNNs, linear/logistic regression
TFHE	Low per-gate latency; fast programmable bootstrapping	Exact (low-bit integer / Boolean)	Native, via programmable bootstrapping	Decision trees, random forests, binarized NNs

- **TFHE (Boolean / gate-level FHE):** Represents values at the bit level and supports efficient programmable bootstrapping (PBS), enabling table lookups and exact comparisons. TFHE is particularly attractive for models dominated by comparisons (e.g., decision trees) and for exact activation functions (e.g., ReLU, max pooling), but incurs high latency for large linear algebra workloads.

In practice, CKKS is widely favored for traditional ML models due to its efficiency for linear algebra operations, whereas TFHE is frequently used for comparison-heavy models due to its exact handling of nonlinear functions via programmable bootstrapping (Section 3.2) [41–47, 53]. For deep learning architectures, this trade-off is sharper: CKKS parallelizes convolutions and matrix multiplications via SIMD packing, while TFHE evaluates activation functions exactly. Hybrid designs combine both, evaluating linear layers in CKKS and nonlinearities in TFHE, at the cost of scheme-switching overhead.

3.2 Nonlinearity Strategy

A central obstacle in FHE-ML is that most FHE schemes natively support only addition and multiplication, whereas modern ML relies heavily on nonlinear functions (e.g., sigmoid, ReLU, max, softmax, normalization). As a result, FHE-ML systems

are largely distinguished by how they implement nonlinearities under encryption. We categorize the literature into two main strategies:

- **Polynomial approximation:** Replace nonlinear functions with low-degree polynomials (e.g., Chebyshev, minimax/Remez). This is the dominant approach in CKKS-based schemes, trading accuracy for multiplicative depth and runtime.
- **LUT-based evaluation:** Evaluate nonlinear functions via look-up tables (LUTs). In TFHE, programmable bootstrapping enables efficient lookup-table evaluation, supporting exact or high-fidelity nonlinear functions.

The nonlinearity strategy directly impacts both model accuracy and cryptographic cost. Polynomial methods avoid frequent bootstrapping but increase depth and approximation error, while LUT/PBS-based methods preserve accuracy but often increase the latency due to bootstrapping frequency.

3.3 Model Family

Different model families induce fundamentally different encrypted computation patterns. We group FHE-ML systems into the following model families:

- **Linear models:** Logistic regression and linear classifiers are primarily composed of dot products and simple nonlinearities (e.g., sigmoid), making them amenable to CKKS with polynomial approximations. The main bottleneck is the iterative optimization for training.
- **Tree-based models:** Decision trees, random forests, and gradient-boosted trees rely on comparisons and control flow, which are not natively supported in arithmetic FHE. Consequently, many works use TFHE/LUT-based comparisons or polynomial step approximations, often restructuring inference as branchless arithmetic circuits.
- **Neural networks:** These models are dominated by linear algebra (convolutions and matrix multiplications) interleaved with nonlinearities. CKKS is typically preferred due to SIMD packing and efficient linear operations, with accuracy-performance tradeoffs driven by activation approximations and bootstrapping schedules.
- **Transformers:** Transformers introduce additional challenges beyond standard neural networks due to attention mechanisms (quadratic complexity in sequence length), heavy reliance on softmax and normalization, and tensor reshaping/rotation overhead induced by packing strategies. We discuss encrypted transformer design challenges in Section 5.2.

This taxonomy dimension is useful for anticipating which cryptographic and systems bottlenecks will dominate: comparisons for trees, depth for deep networks, and rotations/packing complexity for transformers.

3.4 System-level Optimization Levers

Beyond cryptographic scheme selection, FHE-ML systems rely on a set of recurring optimization levers to reduce runtime, memory footprint, and/or accuracy loss. We highlight the most common levers used across the literature:

- **Quantization and fixed-point encoding:** Reducing precision decreases ciphertext modulus requirements, lowers multiplicative depth consumption, and can enable exact arithmetic in integer schemes. Quantization-aware training is frequently used in TFHE-based neural networks.
- **Pruning and sparsity:** Removing weights or channels reduces the number of encrypted multiplications and rotations, particularly beneficial in TFHE where bootstrapping dominates cost.
- **Packing and SIMD batching:** Efficient ciphertext packing can amortize computation across multiple samples or tensor elements. Packing strategy strongly influences rotation count and therefore performance in CKKS.
- **Bootstrapping frequency:** Some works avoid bootstrapping by limiting circuit depth (leveled FHE), while others embrace bootstrapping to enable deeper networks or exact nonlinearities. Bootstrapping policy (when, how often, and for which layers) is a major determinant of latency.
- **Hybrid scheme-switching designs:** Combining schemes can yield strong performance (e.g., CKKS for linear layers and TFHE for nonlinearities), but introduces overhead in ciphertext conversion and parameter alignment.

These levers are often interdependent: for example, more aggressive quantization can reduce bootstrapping frequency, and packing strategy can trade memory for fewer rotations.

3.5 Privacy Scope and Goals

In this survey, we categorize the FHE-ML works by their *privacy goals*, which determines what information is intended to remain hidden from involved parties. We further detail the adversarial assumptions and threat model in Section 2.4. Here, we summarize the privacy goals used throughout the taxonomy in Table 5:

- **Data Privacy:** The Server learns nothing about the Client's input features (and, in training settings, the Client's labels), beyond what may be inferred from the final output. This is the primary privacy goal addressed by nearly all FHE-ML works, including all of the works in this survey.
- **Model Privacy:** The Client learns nothing about the Server's model parameters, beyond what is revealed by the output. Model privacy is not automatically guaranteed by FHE and typically requires additional cryptographic mechanisms.
- **Query Privacy:** The Server learns nothing about the Client's query, including metadata such as whether two encrypted queries correspond to the same input or whether a particular input pattern was queried repeatedly. FHE alone does not protect query patterns; achieving this goal requires additional techniques beyond standard FHE.

While most surveyed works focus on data privacy through encrypted inference or training, model and query privacy remain rare goals and highlight areas where FHE alone is insufficient. By explicitly considering all three privacy types, we clarify both the strengths and the limitations of FHE-based approaches.

We acknowledge that Data Privacy, Model Privacy, and Query Privacy are widely accepted classification criteria in related fields such as Privacy-Preserving Federated Learning and Secure Multi-Party Computation. However, in this survey, we specifically examine how these privacy paradigms manifest uniquely under the strict constraints of non-interactive FHE, which introduces distinct system-level vulnerabilities and cryptographic overheads not present in distributed learning.

3.6 Practical System Design: Applying the Taxonomy for Scheme Selection

The five dimensions of our taxonomy (FHE scheme family, nonlinearity strategy, model family, system-level optimization levers, and privacy scope) compose naturally into a decision procedure for building a new FHE-ML pipeline. We collapse these dimensions into four questions that a practitioner can answer in sequence. Table 5 provides a complementary comparison of representative FHE schemes, illustrating how scheme-level properties such as computational efficiency, numerical precision, nonlinear operation handling, and supported ML models affect practical scheme selection.

1. **What precision does the plaintext domain require?** Real-valued, high-dimensional inputs (image or audio embeddings, CNN/Transformer activations) favor **CKKS**: approximate real/complex arithmetic using encoding, scaling, and rescaling and SIMD batching absorb the dimensionality. Discrete or categorical inputs (tabular features, exact codes) favor **TFHE**, which avoids the approximation error CKKS would otherwise accumulate.
2. **What nonlinearities does the model require?** (Table 3)
   - *Smooth, low-degree functions* (sigmoid, GELU, square): **CKKS** with Chebyshev or minimax polynomial approximation keeps multiplicative depth bounded.
   - *Comparisons or branching* (ReLU, sign, tree splits): **TFHE** via programmable bootstrapping (PBS), evaluated as a lookup table. If linear layers dominate elsewhere, a **hybrid CKKS–TFHE** design pays scheme-switching cost only at activation boundaries.
3. **Does the deployment optimize for throughput or latency?** This is a heuristic, not a strict rule: CKKS's packed SIMD vectors amortize well across large batches, making it favorable for high-throughput inference; TFHE's per-gate evaluation avoids the vector initialization and key-management overhead that penalize single-query workloads, making it favorable for low-latency settings.
4. **What security and privacy guarantees are required beyond basic data confidentiality?** Data confidentiality during evaluation is satisfied by standard FHE, provided the system operates under the assumed honest-but-curious model. Beyond this baseline, the choice of supplementary defenses depends on the target threat vectors (Section 2.4):
   - *Decryption feedback/oracle access*: Requires $\mathsf{IND\text{-}CPA}^D$-compliant schemes using noise flooding or differential privacy noise mechanisms before decryption.

- *Active/malicious servers*: Requires verifiable FHE mechanisms (ZKPs, MACs, or TEEs) to guarantee computation integrity.
- *Statistical leakage from model outputs*: Requires application-level differential privacy or output perturbation to prevent membership inference.
- *Model or query leakage*: Requires circuit privacy, output quantization, or query obfuscation.

3.7 Comparability Caveats Across FHE-ML Papers

Comparing runtimes across FHE-ML papers is challenging because experimental settings are often not standardized. In particular: (i) reported security levels vary widely (e.g., 80-bit vs 128-bit classical security), (ii) batching/SIMD packing assumptions differ and can change throughput by orders of magnitude, (iii) bootstrapping is sometimes excluded or amortized inconsistently, and (iv) hardware platforms range from single-thread CPU baselines to multi-core servers, GPUs, and custom accelerators. For these reasons, we report runtimes and accuracies as stated in the original papers and emphasize within-paper comparisons and qualitative trends.

Because of these discrepancies, direct, one-to-one performance comparisons between proposed schemes are rarely empirically sound. Consequently, the numerical values presented throughout this survey function more as illustrative indicators of magnitude rather than as robust empirical benchmarks for cross-paper comparison. We report runtimes and accuracies as stated in the original papers to provide necessary context—illustrating whether a protocol operates on the scale of milliseconds, minutes, or days—but emphasize that readers should focus on within-paper comparisons and broad qualitative trends.

To reconcile these cross-platform discrepancies and provide a transparent view of the current state-of-the-art without forcing mathematically flawed comparisons, we categorize the existing literature into discrete latency buckets. Table 6 details the state-of-the-art for encrypted inference across major model families, while Table 7 maps the performance of encrypted training to expose the persistent training-to-inference bottleneck gap.

Table 6: Encrypted Inference SOTA by Architecture

Paper	Architecture	Latency	Hardware	Scheme	Security	Bootstrapping
[85]	Decision Trees	1–10s	CPU	TFHE	128-bit	✓
[90]	ResNet-18 MNIST	1–10s	CPU	TFHE	152-bit	✗
[87]	3-Layer MNIST	10–100s	CPU	TFHE	128-bit	✓
[86]	ResNet-20 CIFAR-10	>1000s	1-Thread CPU	CKKS	128-bit	✓
[91]	D7L3 CIFAR-10	>1000s	8-Thread CPU	Hybrid	119-bit	✓
[87]	VGG-9 CIFAR-10	>1000s	CPU	TFHE	128-bit	✓

Table 7: Encrypted Training SOTA by Model Family (Highlighting the Training Gap)

Category	Model & Dataset (Paper)	Training Time / Bottleneck	Scheme
Traditional ML	Decision Tree on UCI ([92])	47 to 2,276 minutes	CKKS
	Logistic Regression on financial /MNIST ([84])	∼17 hours for large financial dataset; iterative gradient-descent bottleneck	CKKS
Neural Networks	3-Layer FCN on MNIST ([93])	>9 hours *per mini-batch* (1-Thread)	BGV
	Feedforward NN on Iris ([94])	29.8 hours (for just 150 samples)	CKKS
	TFHE-NNs on MNIST ([95])	13,140 minutes (∼219 hours)	TFHE

4 Traditional Machine Learning on Encrypted Data

Traditional ML models are often more compatible with FHE than deep learning because they can be expressed with low-depth arithmetic circuits and avoid expensive nonlinearities. In our taxonomy (Section 3), these works primarily use CKKS for approximate real arithmetic or TFHE for LUT-based comparisons. We organize this section by model family (linear models, tree models, margin-based models), and highlight the dominant nonlinearity strategies and optimization levers used in each case.

4.1 Logistic Regression

The integration of HE with logistic regression introduces a key challenge: evaluating the sigmoid function under encryption. While HE enables computation directly on encrypted data, it imposes two significant limitations—high computational overhead and support for only polynomial arithmetic operations. These constraints fundamentally alter the optimization process for logistic regression, as the sigmoid function, a non-polynomial operation, cannot be directly evaluated on encrypted data. To address this, researchers have employed polynomial approximations of the sigmoid function. In our taxonomy, FHE logistic regression is typically implemented in CKKS and relies on polynomial approximations of the sigmoid function; the main optimization levers are typically batching/packing, approximate gradient updates, and/or bootstrapping frequency control.

4.1.1 Recent Advances

Han et al. addressed the challenge of scaling encrypted training by proposing an optimized logistic regression scheme using the CKKS scheme [84]. Their method incorporates mini-batch gradient descent with Nesterov's momentum and a specific ciphertext partitioning strategy to optimize the costly bootstrapping process required for iterative training. Their scheme was stress tested on a massive financial dataset containing over 1.2 million records, achieving 80% accuracy after approximately 17

hours of training. Furthermore, tests on the MNIST dataset demonstrated 96.4% accuracy, validating the feasibility of training on high-dimensional data encrypted under FHE.

Kim et al. established a foundational framework for training logistic regression on sensitive medical data using the CKKS scheme [96]. The authors employed high-precision least-squares approximations (up to degree 7) of the sigmoid function to minimize accuracy loss. The method was evaluated on five public health datasets, including the Edinburgh Myocardial Infarction and NHANES III studies. Their results showed that the degree-7 approximation yielded negligible accuracy degradation compared to the plaintext models, though the authors highlighted that the computational overhead of homomorphic operations remains a significant constraint for larger datasets.

4.1.2 Summary and Discussion

For logistic regression, the primary computationss are dot products paired with a sigmoid nonlinearity and are well-suited to CKKS due to its efficient SIMD-packed linear algebra. As with other models (discussed later in this section), the sigmoid is replaced with a low-degree polynomial approximation. Encrypted inference is therefore efficient, but encrypted training remains costly, specifically because gradient descent requires a full homomorphic weight update for every iteration.

4.2 Decision Tree

In a standard, non-FHE decision tree, the feature vector is compared at internal nodes to guide the traversal through the tree, which leads to a final value like a class label. However, performing these comparisons under FHE is difficult as the schemes lack a comparison operator for ciphertexts. To overcome this limitation, various strategies have been proposed as described in the following state-of-the-art works. In this section we discuss the state-of-the-art homomorphic encryption decision tree schemes and their specifics. Decision trees are distinguished by their reliance on comparisons. Under CKKS, comparisons are approximated via polynomials; under TFHE, comparisons are implemented using lookup tables via programmable bootstrapping.

4.2.1 Recent Advances

Frery et al. developed a methodology for performing privacy-preserving inference on tree-based models (Decision Trees, Random Forests, XGBoost) using the TFHE scheme [85]. To overcome the decision function problem (the inability to perform control-flow operations like if/else on encrypted data), the authors employed a two-part strategy. First, they replaced standard conditional checks with table lookups implemented via programmable bootstrapping, effectively turning comparisons into mathematical look-ups. Second, they flattened the tree traversal process into a series of tensor operations (GEMM strategy), allowing the server to evaluate all potential branches simultaneously using matrix multiplication rather than sequentially selecting paths. This approach allows for high-accuracy inference (e.g., matching the accuracy

of scikit-learn models) without the client ever needing to come online during the computation. Their experiments showed near identical accuracy, f1-score, and average precision between the plaintext and encrypted tree inferences.

Akavia et al. proposed a privacy-preserving decision tree training and classification scheme using the CKKS scheme with the Microsoft SEAL library [92]. The authors scheme is non-interactive, with the only communication being the client sending their encrypted data to be classified. By utilizing a low degree approximation for the step functions using the least squares method, the authors solved the ever-troublesome problem of comparison over encrypted data. By approximating the step function with a polynomial, comparisons are now possible over encrypted data. Building off of this step function approximation, the rest of the decision tree is straightforward. The authors tested their scheme on four datasets from the UCI Repository and compared it with a plaintext implementation. Their scheme achieved very similar accuracy to the original, plaintext accuracy. The execution time of the training ranges from 47 minutes to 2,276 minutes based on the size and complexity of the dataset.

Shin et al. introduced their Homomorphic Binary Decision Tree and Homomorphic Random Forests, a fully non-interactive framework for training and inferencing decision trees and random forests on encrypted data using the CKKS scheme [97]. To address the decision function problem (the inability to perform direct comparisons in FHE), the authors adopted different strategies for training and inference. During training, they replaced exact comparisons with an approximate sign function ($\mathsf{ApproxSign}$) based on polynomials, allowing them to compare Modified Gini Impurity values and select optimal splits without decryption. For inference, they avoided calculating comparisons entirely by using one-hot encoded inputs; this converts the "decision" at each node into an arithmetic operation (multiplication and addition) where the valid paths sum to zero, achieving an $O(1)$ multiplicative depth. Their method proved to be at least 3.7 times faster in inference than prior work by Akavia et al., while maintaining accuracy comparable to plaintext models [92].

4.2.2 Summary and Discussion

For Decision Trees and Random Forests, the fundamental computational bottlenecks are the comparison operations and control-flow branching, neither of which are natively supported in arithmetic FHE. To solve this, researchers either employ TFHE architectures that naturally handle comparisons via programmable bootstrapping look-up tables, or they utilize CKKS with a polynomial step/sign approximation. The essential design pattern across both schemes is "branchless evaluation," where the sequential traversal of a tree is flattened into tensor operations so that all branches are evaluated simultaneously. Encrypted tree inference retains accuracy nearly identical to plaintext, though runtime remains heavily dependent on tree depth and batching strategy.

4.3 Support Vector Machine

We explore how homomorphic encryption transforms the training and classification processes of the Support Vector Machine [98].

4.3.1 Recent Advances

Huang et al. proposed a privacy-preserving client-server architecture designed to improve the efficiency of SVM classification [99]. In their model, the server holds a pre-trained SVM model and performs inference on encrypted data uploaded by the client, ensuring that the server never accesses the raw input and the client gains no knowledge of the model parameters. The authors implemented their scheme using the CKKS scheme and leveraged Single Instruction Multiple Data (SIMD) packing to optimize the linear operations. The key contribution of their work was the replacement of the computationally expensive sign function with an efficient polynomial approximation, which significantly reduced the computational overhead. Experimental results on the CIFAR-10 dataset demonstrated that this optimization reduced classification time from hundreds of seconds to tens of seconds and encryption/decryption latency to mere milliseconds.

Park et al. introduced a method for privacy-preserving fair learning using FHE to mitigate bias in machine learning models while protecting data privacy [100]. To address the high computational costs typically associated with FHE, the authors proposed a hybrid encryption strategy where only sensitive attributes (like gender, race, etc.) are encrypted during the training process. Utilizing the CKKS scheme for its native handling of real numbers, they developed a protocol that avoids computationally intensive bootstrapping operations. Their method was evaluated on multiple real-world datasets, including the Adult Income, COMPAS, and German Credit datasets. Their results indicated that this approach can achieve a superior fairness-accuracy trade-off compared to existing privacy-preserving algorithms and effectively reduced discrimination in the model outputs without incurring prohibitive computational costs.

4.3.2 Summary and Discussion

For Support Vector Machines, the primary operations involve linear evaluations followed by a sign function to determine the classification margin. As with other linear models discussed in this section, CKKS is favored for its SIMD-optimized linear algebra, and the sign function is replaced with a polynomial approximation, reducing classification runtime from hundreds of seconds to tens of seconds. Some recent frameworks also explore hybrid encryption strategies—encrypting only sensitive attributes under CKKS—to avoid bootstrapping entirely, yielding strong fairness-accuracy trade-offs during training without prohibitive cost.

4.4 Naive Bayes

We explore how homomorphic encryption transforms the training and classification processes of Naive Bayes classifiers. Only one paper met the criteria for this survey discussed in Section 1.3.

4.4.1 Recent Advances

Han et al. proposed the first fully homomorphic Naive Bayes classifier that operates entirely on encrypted data for both training and classification with the CKKS

scheme [101]. Standard Naive Bayes requires computing natural logarithms of probabilities (a non-polynomial operation difficult for FHE) so the authors developed a novel method to approximate $\ln(x)$ specifically for probability values within the CKKS domain. This innovation eliminates the need for any intermediate decryption or trusted third parties. The resulting scheme demonstrated a dramatic performance improvement, with training times in tens of seconds and inference times in just a few seconds. Overall, the authors reported a 28–56× speedup compared to previous state-of-the-art secure Naive Bayes implementations.

4.4.2 Summary and Discussion

For Naive Bayes classifiers, the dominant bottleneck is the natural logarithm used to compute probabilities, a non-polynomial operation incompatible with native FHE. Recent CKKS-based methods approximate $\ln(x)$ directly within the domain of probability values. Unlike more complex models, this approximation makes both encrypted inference and encrypted training highly feasible, with modern implementations achieving training in tens of seconds and inference in a few seconds—without any intermediate decryption.

4.5 Lessons Learned from Traditional ML on Encrypted Data

Our review of traditional machine learning models highlights the critical trade-offs between privacy, accuracy, and performance. Notably, encrypted inference typically achieves accuracy comparable to plaintext counterparts, often within a 1–2% margin. However, this comes at a significant computational cost, with inferences requiring orders of magnitude more time than the plaintext equivalents. Despite this, inference times generally remain tractable, ranging from a few seconds to several minutes depending on model and dataset size.

Simpler models such as logistic regression, decision trees, Naive Bayes, and Support Vector Machines are the most studied due to their limited computational complexity. However, performance varies significantly based on the underlying operations:

- **Iteration Bottleneck:** Logistic regression, despite its conceptual simplicity, incurs high training costs under FHE. This is primarily due to its reliance on iterative gradient descent, which becomes prohibitively expensive when every update involves homomorphic operations on encrypted weights.
- **Comparison Barrier:** Decision trees exhibit slower performance because they rely on branching and comparisons, which are not natively supported in FHE. Implementing these requires deep arithmetic circuits or look up table-based strategies that further increase computational latency.
- **Arithmetic Advantage:** Naive Bayes and SVMs fare better under FHE. Their reliance on basic arithmetic operations (additions and multiplications) rather than complex control flow or deep iterative loops makes them highly amenable to encrypted computation for both training and inference.

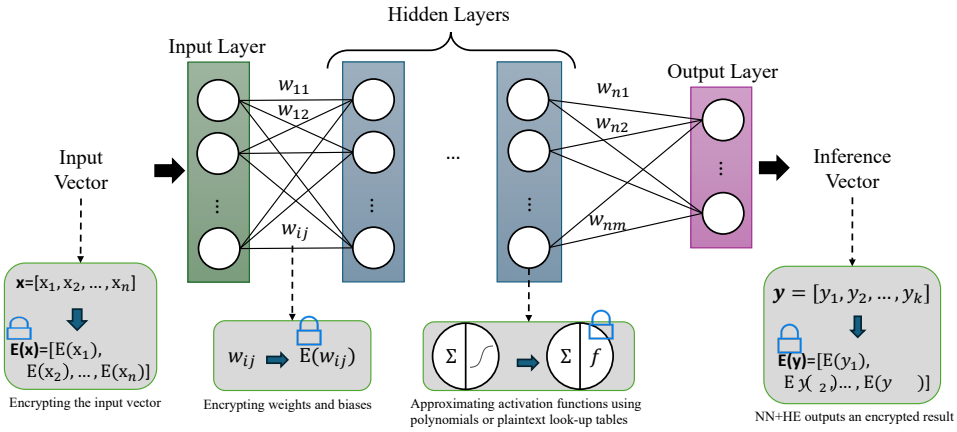

Figure 1: The architecture of a neural network performing inference with fully homomorphic encryption. The gray boxes depict the modified network layers to be compatible with fully homomorphic encryption. $E()$ denotes an HE function that transforms plaintext inputs into ciphertexts, to ensure data privacy through the inference process. The blue lock represents the encrypted inputs and outputs, like in the case of the approximate activation function. The plaintext input vector $\mathbf{x}$ and network weights w_{ij} are encrypted by the encryption function E. The subsequent operations of perceptrons, including summations $\sum$ and approximated activation functions f are executed on this encrypted data within the hidden layers, and ultimately produces an encrypted output vector $E(\mathbf{y})$.

5 Deep Learning on Encrypted Data

This section explores deep learning algorithms adapted for data encrypted with FHE, enabling training and/or inference directly on encrypted data. We begin with a brief overview of each algorithm, followed by implementation details of the corresponding homomorphic encryption schemes. Where available, total classification times are reported; however, the specific components included, such as encryption and decryption, may vary across studies. In most cases, encryption and decryption contribute minimally to the overall runtime, so their omission is unlikely to significantly affect comparative evaluations.

5.1 Neural Networks with Encrypted Data

Activation functions are the central obstacle to applying FHE to neural networks (Section 3.2): while addition and multiplication under HE are sufficient to construct a network's basic structure, popular activation functions are nonlinear and not natively supported. Polynomial approximation is the more commonly adopted strategy, as restricting models to HE-compatible activations directly often degrades accuracy unacceptably. LUT-based evaluation is typically faster but introduces storage overhead and the challenge of designing tables that cover the full input range; this was

first explored by Crawford et al. [102], with more recent tools such as AutoFHE [103] automating table design and optimization for convolutional networks.

An illustration of a neural network operating over encrypted data is provided in Figure 1. In our taxonomy, neural network inference papers differ primarily by their nonlinearity strategy: (i) polynomial activations under CKKS, (ii) LUT/PBS-based activations under TFHE, or (iii) hybrid CKKS–TFHE designs.

5.1.1 Neural Networks with Homomorphic Encryption

This section explores how homomorphic encryption transforms the training and classification processes of neural networks. To illustrate these changes, we present a representative approach that addresses the core challenges of operating over encrypted data. For consistency and clarity, we follow the notation and terminology introduced in Section 2.

5.1.2 Privacy-Preserving Inference

Here we discuss the state-of-the-art schemes and implementations for neural network inference over encrypted data.

Lee et al. pushed the limits of deep network inference in RNS-CKKS by directly tackling the massive bootstrapping overhead inherent in strided convolutions [86]. Their approach centers on a multiplexed parallel convolution algorithm, which repurposes multiplexed packing originally intended for pooling to densely pack tensors from multiple channels into a single, larger ciphertext. This packing strategy, combined with a novel convolution algorithm that processes multiple output channels simultaneously, reduced the number of expensive rotation operations by 62%. Crucially, the authors identified a "catastrophic divergence" phenomenon in deeper networks, where the errors from polynomial ReLU approximations accumulate to destroy accuracy. They resolved this by introducing their novel imaginary-removing bootstrapping procedure, enabling the first successful FHE implementation of ResNet models up to 110 layers. This resulted in drastic efficiency gains: their ResNet-20 implementation on CIFAR-10 achieved a runtime of 2,271 seconds on a single thread, effectively outperforming the 10,602 seconds required by the prior state-of-the-art running on 64 threads. While outperforming prior benchmarks, it is critical to contextualize this runtime within real-world deployment constraints. A latency of nearly 38 minutes to classify a single, low-resolution 32×32 pixel image highlights that even state-of-the-art CKKS remains unsuitable for real-time or user-facing applications, restricting its use entirely to offline, asynchronous batch processing.

Stoian et al. developed a methodology for building deep neural networks that strictly adhere to TFHE constraints to ensure deterministic and noise-free inferences [87]. By recognizing that the efficiency of TFHE's programmable bootstrapping depends heavily on minimizing accumulator bit-widths, the authors eschewed standard training in favor of quantization-aware training combined with unstructured pruning. This forced the model to learn weights that naturally respect the cryptographic bounds of the accumulator. Because the resulting computation is exact, the encrypted models guarantee identical accuracy to their plaintext counterparts. However, this exactness comes with a significant latency trade-off for complex architectures. For example,

while a 3-layer MNIST network completed inference in 31 seconds, a larger VGG-9 architecture on CIFAR-10 extended this latency to 18,000 seconds. However, ensuring deterministic TFHE execution comes at a severe computational and structural cost. The strict requirement to bound accumulator bit-widths necessitates aggressive quantization and L^1-norm unstructured pruning during training. While this successfully fits the model into TFHE's cryptographic constraints, this extreme reduction in model capacity poses a significant challenge for scaling to larger, more complex datasets as the representational power of the network is heavily bottlenecked.

Liu et al. addressed the precision bottlenecks inherent in CKKS-TFHE hybrid architectures [91]. While hybrid schemes combine efficient linear CKKS operations with precise TFHE nonlinearities, the authors identified that the "scale-down" process required for programmable bootstrapping introduces severe quantization errors when neural network activations exceed the limited message domain of the Look-Up Table (LUT). To mitigate this problem, they introduced a LUT-aware fine-tuning protocol. This method empirically estimates the dynamic range of activation inputs using plaintext data and injects a linear mapping function into the convolutional weights, ensuring that intermediate values align perfectly with the optimal $[-B, B)$ domain of the cryptographic LUT. By folding this alignment into the linear layers, the protocol incurs no additional inference cost while recovering significant accuracy; for a D7L3 model on CIFAR-10, the accuracy was restored from a degraded 47.5% to a competitive 80%, matching the plaintext baseline. Despite the high accuracy, the hybrid overhead remained substantial, with inference taking approximately 43 minutes (2,558 seconds) on an 8-thread CPU. While this LUT-aware fine-tuning effectively restores accuracy, it introduces a significant assumption regarding data availability and threat modeling. The protocol fundamentally relies on analyzing the feature maps of plaintext training data to estimate the dynamic range of intermediate activations. This introduces a vulnerability to data distribution shifts: if the distribution of a client's real-world encrypted queries deviates from the plaintext calibration data, the resulting intermediate values will fall outside the optimized Look-Up Table domain. This would immediately reintroduce the severe quantization errors the method was designed to solve. Furthermore, an inference time of 2,558 seconds for a D7L3 model underscores that hybrid overheads remain a substantial barrier to deployment despite the significant improvements.

Lou and Jiang proposed SHE (Shift-accumulation-based Homomorphic Encryption), a framework designed to eliminate the reliance on polynomial approximations that plagues the standard leveled-HE architectures [90]. Leveraging TFHE, the authors replaced expensive homomorphic multiplications with logarithmic quantization, converting model weights into powers of 2 to enable efficient homomorphic bit-shifts. To further optimize the circuit, they designed a mixed-bitwidth accumulator tree that dynamically adjusts the adder precision at each level, minimizing the bootstrap overhead. Crucially, SHE implements exact ReLU and Max Pooling operations using TFHE Boolean gates. Unlike polynomial approximations, which consume the multiplicative depth budget and degrade accuracy in deep networks, these Boolean gates incur negligible depth costs. This allowed SHE to support significantly deeper architectures than previous works, such as ResNet-18 and LSTMs. Their empirical results

on MNIST demonstrated a 76–94% reduction in latency compared to prior art, achieving 9.3s per inference while maintaining state-of-the-art accuracy of 99.54%. Despite this low inference latency on MNIST, this performance must be carefully contextualized. While avoiding polynomial approximation preserves network depth and accuracy, a latency of nearly ten seconds to classify a single, low-resolution 28×28 grayscale image indicates that the underlying Boolean operations remain computationally restrictive. Consequently, while SHE represents a significant algorithmic optimization, it still falls far short of the real-time, high-throughput requirements necessary for modern, high-resolution computer vision deployments.

5.1.3 Privacy-Preserving Training

We now begin our discussion of state-of-the-art schemes and implementations for training neural networks over encrypted data.

Nandakumar et al. established a baseline for non-interactive neural network training using stochastic gradient descent with the BGV scheme [93]. To mitigate the immense computational complexity of backpropagation, the authors employed several optimizations: simplifying network architecture, using 8-bit signed integer representations for inputs, and implementing smart ciphertext packing to facilitate parallelization. They utilized a 3-layer fully connected network and replaced complex activations with a sigmoid function implemented via homomorphic table lookups. While training a full 784-input network proved computationally prohibitive, a reduced 64-input version (NN2) achieved 96% accuracy on MNIST. Their results highlighted the "horrendous" execution time of FHE training, noting that a single mini-batch took over 9 hours on a single thread, though multi-threading with 30 threads reduced this to 40 minutes.

Mihara et al. proposed a scheme for training a neural network on encrypted data using the CKKS scheme [94]. The authors introduced a novel diagonal packing method for the weight matrix, designed to minimize computational overhead during matrix transposition operations required for backpropagation. By leveraging this packing method, they significantly reduced the total number of rotations and multiplications needed, achieving a threefold reduction in computation time compared to traditional row packing methods. For their experiments, the authors trained a simple feedforward neural network with one hidden layer on the Iris dataset, a standard classification dataset with 150 samples across three classes. The network consisted of an input layer with four nodes, a hidden layer with ten nodes, and an output layer with three nodes, using square activation functions to maintain compatibility with CKKS. The training was performed over 400 epochs, with ciphertext training achieving a final accuracy of 98.47%, slightly surpassing the plaintext training accuracy of 98.05%. The authors reported a total training time of approximately 29.8 hours when using batch sizes of one, with further reductions achieved via OpenMP-based parallelization. They highlighted that their method significantly reduced the multiplication circuit depth and computational overhead and while the scheme demonstrated practicality for smaller datasets and networks, the authors noted the challenges of scaling to more complex architectures and larger datasets due to the inherent constraints of homomorphic encryption.

Colombo et al. designed TFHE-NNs, a family of TFHE-compliant neural networks for non-interactive training on encrypted data [95]. Instead of backpropagation, the authors used direct feedback alignment, due to the need for integer values in the entire architecture. Also, they proposed a cross validation procedure that selects the TFHE-NN that guarantees the highest validation accuracy. The trained model had identical validation and testing accuracies for both the plaintext and encrypted datasets, but the plaintext model trained in 0.05 minutes while the encrypted model trained in 13,140.0 minutes. However, they were able to reduce this encrypting training time to 6,570.0 minutes with parallel training at the cost of slightly reduced accuracy of about 86.5%.

Collectively, these metrics demonstrate that while encrypted inference is nearing viability for specific targeted applications, non-interactive encrypted training currently faces profound scalability barriers. The extreme computational overhead observed across early implementations indicates that general-purpose encrypted training requires further paradigm shifts before practical production deployment is feasible. Current demonstrations on trivially small datasets (such as the 150-sample Iris dataset or cropped MNIST images) requiring tens to thousands of hours of computing power suggest that direct deep learning training under FHE remains highly prohibitive. Without significant architectural paradigm shifts, full-scale encrypted training presents a severe challenge for near-term viability [93].

5.1.4 Summary and Discussion

For neural networks and CNNs, the central bottlenecks arise from heavy linear algebra workloads interleaved with nonlinear activation functions. CKKS is strongly preferred for processing convolutions and matrix multiplications due to its highly efficient SIMD packing capabilities, though it suffers from approximation errors that accumulate when using polynomial replacements for activations. Alternatively, TFHE provides exact, high-fidelity nonlinearities (like ReLU) using Look-Up Tables (LUTs) and Boolean gates, but it incurs massive latency penalties and high computational overhead for linear workloads. To balance these trade-offs, researchers have developed CKKS-TFHE hybrid designs that compute linear layers in CKKS and activations in TFHE; however, these architectures introduce new challenges, such as the quantization errors and severe scale-down overhead required during scheme conversion.

5.2 Transformers over Encrypted Data

The transformer architecture was first proposed by researchers at Google in 2017 [104]. Since then, transformers have found applications in various domains such as natural language processing, computer vision, and multimodal learning, to name a few. An example is provided in Figure 2.

5.2.1 Privacy-Preserving Inference

Zimerman et al. presented the first practical FHE–based transformer inference using polynomial approximations under the CKKS scheme [88]. This work marked a

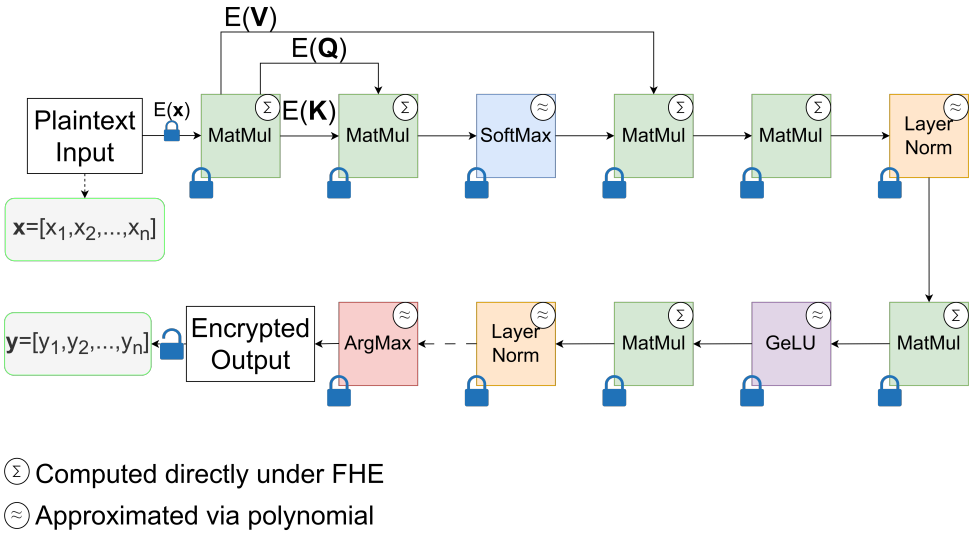

Figure 2: The architecture of a transformer performing an inference with FHE. The light green boxes depict the a compatible matrix multiplication protocol (MatMul) with fully homomorphic encryption. $E()$ denotes an encryption function that transforms plaintext inputs into ciphertext outputs to ensure data privacy throughout the inference process. The blue lock represents the encrypted inputs and outputs. The plaintext input $\mathbf{x}$, query matrix ($\mathbf{Q}$), key matrix ($\mathbf{K}$), and value matrix ($\mathbf{V}$) are encrypted by the encryption function E. The subsequent operations of softmax (SoftMax), layer normalization (LayerNorm), Gaussian error linear unit (GeLU), and the arguments of the maxima (ArgMax) are approximated functions executed on the encrypted data within the hidden layers, and ultimately produces an encrypted output $E(\mathbf{y})$.

key milestone in privacy-preserving machine learning by demonstrating that transformer models can achieve competitive performance on both vision and NLP tasks under FHE. The main challenge was the incompatibility of standard transformer components with homomorphic encryption, as naive polynomial substitutions cause numerical instability [105, 106]. The authors addressed this problem by replacing the non-polynomial operations with HE-friendly alternatives, including a pointwise activation-based attention mechanism and polynomial activation functions. Layer normalization was adapted using task-specific strategies for vision and language models. The resulting models were evaluated on CIFAR-100, Tiny-ImageNet, and Wikitext-103, achieving accuracy and perplexity close to plaintext baselines. For example, a 12-layer polynomial NLP model achieved a perplexity of 18.91 compared to 16.89 in plaintext. These results demonstrate that carefully designing polynomial transformers prove the practicality of FHE-based inferences. However, this approach introduces a critical structural compromise. By replacing the standard softmax-based attention mechanism with pointwise activations, the architecture deviates significantly from true transformer dynamics. While sufficient for smaller benchmarks like Wikitext-103, it

remains speculative whether this approximation can scale to the complex contextual mapping required by modern Large Language Models (LLMs) without catastrophic degradation in reasoning capabilities.

Zhang et al. proposed NEXUS, a non-interactive protocol for secure transformer inference that eliminates the multi-round client-server interaction required by prior approaches [107]. In NEXUS, the client sends a single encrypted query and the server returns an encrypted prediction, with all noise management performed server-side. This design is highly relevant for transformers, whose inference workloads are dominated by large matrix multiplications and expensive ciphertext rotations, and whose nonlinear components (normalization and output selection) are difficult to implement efficiently under FHE. To improve the throughput, NEXUS introduces SIMD slot folding and ciphertext compression techniques tailored to transformer-specific linear algebra. The system also includes an efficient protocol for output selection (Argmax) designed to scale to the large vocabularies typical of LLMs. In the authors' evaluation on BERT-base and Llama-3-8B, NEXUS reports up to a 372× reduction in bandwidth compared to interactive baselines such as BOLT, and achieves a reported 37.3 seconds end-to-end inference time for BERT when leveraging GPU acceleration. NEXUS's results prove the ability to adapt existing transformers to the bounds of FHE while remaining computationally feasible. Despite the impressive latency reductions, a critical evaluation of NEXUS reveals a heavy reliance on high-end GPU acceleration to achieve their reported 37.3 second inference time. This highlights a persistent limitation in current state-of-the-art FHE-ML: algorithmic optimizations alone are often insufficient for practicality. The requirement for enterprise-grade hardware clusters shifts the deployment bottleneck from pure latency to severe economic costs, which poses a barrier for widespread cloud adoption.

Rho et al. introduced an encryption-friendly transformer architecture tailored for secure personalized fine-tuning and inference under homomorphic encryption using the CKKS scheme [89]. A key challenge they address is that conventional full-model fine-tuning requires updating all model parameters, which forces weight matrices into ciphertext form and leads to costly ciphertext-ciphertext multiplications. To avoid this, the authors adopt Low-Rank Adaptation (LoRA), freezing the pre-trained model weights in plaintext and training only a small set of low-rank adapter matrices [108]. This design allows most linear operations to remain as efficient plaintext-ciphertext multiplications, substantially reducing computational overhead. To further limit circuit depth, the standard softmax attention mechanism is replaced with a Gaussian kernel. Unlike softmax, which relies on division and exponentiation over an unbounded input range, the Gaussian kernel removes the division step and confines the exponential approximation to the numerically stable region where ($x \leq 0$). Nonlinear functions are approximated using a combination of Remez polynomial approximation and Newton's method, balancing accuracy and efficiency under encryption. The proposed architecture was evaluated on the GLUE benchmark and showed substantial performance gains over full-model encrypted baselines, achieving a 6.94 times speedup during fine-tuning and a 2.33 times speedup during inference. For example, on the RTE task, training time per epoch dropped from 19.5 hours to 4.8 hours, while accuracy remained close to that of plaintext models (with up to 99% accuracy retention

on most tasks). These results suggest that privacy-preserving personalization may be feasible for selected tasks and open-model settings, although the approach depends on plaintext base weights and does not resolve end-to-end encrypted inference for proprietary LLMs. While LoRA effectively mitigates the ciphertext-ciphertext multiplication overhead, it introduces a strict boundary on the applicable threat model. Freezing the pre-trained weights in plaintext implicitly assumes that the base model is publicly available. In commercial scenarios involving proprietary foundation models, exposing the base weights in plaintext violates intellectual property constraints. Therefore, while highly efficient for personalized fine-tuning of open models, this approach does not resolve the broader challenge of end-to-end encrypted inference for proprietary LLMs.

5.2.2 Summary and Discussion

For transformer architectures, the dominant cryptographic bottlenecks are the quadratic complexity of attention mechanisms, the reliance on nonlinear softmax and normalization layers, and the tensor reshaping/rotation overhead induced by ciphertext packing. Because standard transformer components cause severe numerical instability when naively substituted with polynomials, CKKS-based strategies frequently alter the semantics of the model, substituting operations with HE-friendly alternatives like pointwise activation-based attention. Furthermore, to achieve feasible runtimes during fine-tuning, system-level optimizations heavily rely on Low-Rank Adaptation (LoRA) strategies to keep the vast majority of base weights in plaintext. Within the stated search scope of current FHE-ML literature, TFHE-native exact transformers remain largely unexplored, with polynomial approximations dominating the field.

5.3 Absence of TFHE Transformer Implementations

As outlined in our literature search methodology (Section 1.3), our systematic review of works published between January 2019 and January 2026 yielded no *non-interactive* transformer inference schemes that operate purely under TFHE. While TFHE has been successfully applied to decision trees and neural networks, the literature—within the bounds of our search—has yet to produce a fully TFHE-native transformer implementation that meets practical latency or depth constraints. To the best of our knowledge at the time of writing, this remains an open area of research.

The absence of TFHE-native transformers likely reflects a fundamental mismatch between TFHE's strengths and the computational structure of transformers:

- **SIMD throughput vs. PBS flexibility:** Transformers are dominated by large matrix multiplications and attention score computation. CKKS can amortize these operations via packed SIMD arithmetic, whereas TFHE's programmable bootstrapping is typically applied to individual (or small batches of) values, making transformer-scale linear algebra significantly less efficient.
- **High-precision nonlinearities:** Key transformer components such as softmax and normalization require nontrivial nonlinear evaluation at moderate precision. While TFHE supports LUT/PBS-based nonlinear evaluation, achieving transformer-level

accuracy may require multi-precision techniques and frequent bootstrapping, which can be prohibitively expensive.

- **Tensor packing and data movement:** Efficient transformer execution under FHE depends critically on packing strategies and data movement (e.g., reductions and reshaping in attention). CKKS supports efficient slot rotations for packed tensors, whereas TFHE lacks an equally efficient analogue for high-dimensional packed tensor operations, increasing the cost of attention and related primitives.

Despite this absence, future work could explore hybrid designs, improved packing techniques, or alternative attention mechanisms that reduce depth and bootstrapping requirements, potentially enabling TFHE transformer implementations.

5.4 Lessons Learned from Deep Learning on Encrypted Data

Our analysis of state-of-the-art deep learning implementations under FHE reveals several critical insights into the current trade-offs between cryptographic security and model performance:

- **Incompatibility of Standard Architectures:** Porting "off-the-shelf" deep learning models to FHE-ML is generally ineffective because standard nonlinearities like ReLU or Softmax are not natively supported by FHE schemes. Successful implementations require "cryptography-aware" designs, such as using quantization-aware training to manage bit-widths or replacing standard activations with low-degree polynomials.
- **Scalability Gap in Training:** While encrypted inference has reached near-practical latencies for some tasks, training remains a massive bottleneck. Encrypted training is estimated to be four to five orders of magnitude slower than plaintext training. Current successes in training are limited to smaller networks or reduced datasets, such as cropping MNIST images to smaller pixels to make the parameter count manageable.
- **Scheme-Specific Performance Profiles:** The choice of FHE scheme dictates the model's strengths. CKKS is highly efficient for the large-scale linear algebra required by Transformers and CNNs but introduces approximation errors that can accumulate and cause "catastrophic divergence" in deep networks. Conversely, TFHE provides exact, deterministic results using programmable bootstrapping for nonlinearities, but incurs a significant latency penalty for complex architectures like VGG-9.
- **Necessity of Structural Optimization:** Performance gains in deep FHE-ML often come from system-level tricks rather than just faster hardware. Techniques such as multiplexed parallel convolutions to reduce rotations, diagonal packing for matrix transposition, and Low-Rank Adaptation (LoRA) to keep most weights in plaintext during fine-tuning have proven essential for achieving feasible runtimes.
- **Accuracy Retention vs. Computational Cost:** Deep FHE-ML can achieve accuracies nearly identical to plaintext counterparts (often within 1%), but the cost is a massive increase in energy and time. For instance, an exact TFHE-based MNIST inference can take 31 seconds compared to milliseconds in plaintext, representing

a trade-off currently acceptable only in high-stakes privacy domains like healthcare or finance.

6 Applications of Machine Learning on Encrypted Data

Machine learning (ML) on encrypted data using fully homomorphic encryption (FHE) is increasingly being explored for real-world applications where privacy is essential, enabling secure computations without exposing the sensitive content.

6.1 Machine Learning as a Service

Cloud computing, particularly Machine Learning as a Service (MLaaS), presents a natural application for machine learning over encrypted data. With the rapid expansion of cloud infrastructure over the past decade, MLaaS has emerged as an attractive alternative to the costly investment in GPUs, servers, and system maintenance. It offers users a flexible, scalable, and cost-effective solution for training machine learning models and serving inference requests. However, a critical challenge in this paradigm is the lack of trust in the cloud provider. The cloud server is typically considered a semi-honest adversary—one that follows the protocol correctly but may attempt to infer sensitive information from the data it processes.

To address privacy concerns, FHE offers a compelling solution. With FHE, users can encrypt their training data, model parameters, inference inputs, and results, allowing the cloud server to operate entirely on encrypted data. This can provide strong data privacy in a non-interactive fashion. Model privacy and query privacy, however, require additional mechanisms beyond standard FHE, as discussed in Section 3.5. Unlike techniques such as secure multi-party computation or functional encryption, FHE-ML requires minimal communication between the user and the server.

In a typical workflow, the user encrypts and uploads data and computation to the server. The user can then submit encrypted inputs for inference and receive encrypted predictions in return. Only the user, with access to the secret key, can decrypt the results, so the cloud server does not learn the plaintext input data or outputs under the stated threat model. Protecting model privacy and query-pattern privacy requires additional safeguards.

While FHE-based MLaaS offers strong privacy guarantees, it comes with high computational and energy costs [109, 110], making the cost-privacy trade-off a central concern. Recent work by Lin et al. [111] addresses this challenge by proposing a latency-sensitive framework for privacy-preserving neural network inference using HE. Despite the current limitations, homomorphic encryption remains one of the most effective tools for enabling secure MLaaS and continues to be the focus of active research and optimization.

6.2 Healthcare

Hospitals and research institutions can greatly benefit from machine learning with homomorphically encrypted data techniques when analyzing sensitive health data, such

as patient records or genomic information. Strict privacy regulations such as HIPAA govern the handling of medical data [52]. Homomorphic encryption (HE) enables machine learning over encrypted data, offering compliance with these regulations while allowing for secure data sharing and collaborative research.

By operating on encrypted data, HE adds a robust layer of security that meets the stringent privacy standards required for handling medical information [96, 99, 112, 113]. For instance, a pre-trained machine learning model designed to predict disease susceptibility can execute secure inference directly over encrypted patient data, ensuring both privacy and utility. Furthermore, while full-scale encrypted training remains a long-term aspiration (especially for deep learning models as discussed in Section 5), future scaling could theoretically allow a machine learning model to be trained directly on encrypted patient data and provide stronger protection for sensitive patient data under the stated threat model. This capability enables hospitals or research institutions to safely outsource training to third-party cloud providers.

6.3 Finance

The financial sector faces significant challenges in balancing privacy, security, and the need to analyze sensitive user data effectively. FHE-ML offers a promising solution by enabling secure processing of financial data without compromising confidentiality. For instance, machine learning models for fraud detection can analyze encrypted transaction data to identify anomalies or suspicious activity, all while maintaining customer privacy. This capability is especially valuable in anti-money laundering initiatives, where collaboration between institutions is essential but often limited by strict privacy regulations.

FHE enables such collaboration by allowing encrypted databases from multiple financial entities to be securely combined and analyzed without disclosing private customer information [114]. This approach supports compliance with rigorous data protection laws while enhancing the utility of machine learning for fraud detection. Similarly, recent works emphasize the importance of privacy preservation in credit analysis [84, 115]. They demonstrate that homomorphic encryption can protect sensitive credit data during cloud-based computations.

7 Future Research Directions

Implications for AI Research. The study of machine learning over fully homomorphically encrypted data highlights a set of constraints that fundamentally reshape the design, training, and deployment of AI models. Unlike conventional ML settings, FHE restricts available operations, limits numerical precision, and introduces significant computational overhead, forcing researchers to reconsider common assumptions about model complexity, activation functions, and training dynamics. These constraints encourage the development of models that are inherently more structured, numerically stable, and amenable to low-precision computation.

From an AI systems perspective, FHE-based learning emphasizes the importance of cryptography-aware model design, where architectural choices, optimization strategies, and training procedures are co-designed with the underlying execution substrate.

Techniques such as polynomial activation approximations, quantization-aware training, model compression, and architectural simplification—originally motivated by encrypted computation—also align with broader trends in efficient and robust AI. As a result, insights gained from FHE-ML research may inform the design of models that are more efficient, interpretable, and deployable even outside encrypted settings.

More broadly, FHE-based machine learning provides a concrete testbed for studying trustworthy and privacy-preserving AI under strong adversarial assumptions. By enabling computation on sensitive data without revealing inputs to the computing platform, FHE challenges the traditional trade-off between data utility and confidentiality. Continued progress in this area has the potential to influence how future AI systems are built for regulated and high-stakes domains, where privacy, security, and accountability are first-class design requirements rather than afterthoughts.

7.1 Hybrid and Interactive Deployments

As discussed, the computational overhead of pure FHE remains a formidable barrier. Consequently, much of the current real-world momentum in privacy-preserving machine learning relies on interactive protocols to distribute this burden. This includes cross-silo federated learning, secure aggregation, and hybrid cryptographic systems that combine FHE with Multi-Party Computation (MPC) [116–118].

While these interactive paradigms offer a practical stopgap for current deployments, they inherently shift the system bottleneck from computation to communication. Evaluating scalability in interactive settings introduces qualitative differences, as performance becomes heavily dependent on client-side hardware, synchronization overhead, and unpredictable network bandwidth. Therefore, while hybrid systems dominate the immediate deployment landscapes, we argue that the non-interactive paradigm surveyed in this work provides the most robust architectural model for secure, asynchronous cloud-based AI outsourcing, as it removes the persistent client-side availability requirements inherent in multi-party configurations. Advancing non-interactive FHE-ML ensures that clients can asynchronously transmit encrypted data and retrieve results without remaining online, ultimately providing a more robust architecture for cloud-based AI.

7.2 FHE-ML Software and System Ecosystem

The FHE-ML ecosystem has evolved from isolated cryptographic libraries into a tiered deployment stack. At the foundational layer, several mature, open-source libraries provide the core cryptographic primitives. Microsoft SEAL [119] and Lattigo [120] are widely adopted for the leveled schemes (BFV, BGV, CKKS), with Lattigo supporting multiparty variations. OpenFHE [121] offers a comprehensive suite covering both leveled and fully bootstrapped schemes (including FHEW and TFHE), while TFHE-rs [122] provides specialized APIs for programmable bootstrapping and exact Boolean and integer operations.

Above this cryptographic foundation, a layer of higher-level frameworks and compilers has emerged to bridge the gap with applied machine learning. Concrete ML [123] offers a scikit-learn-compatible interface that abstracts FHE implementation details

for practitioners, handling quantization and scheme selection automatically. Google's HEIR [124], built on the MLIR compiler infrastructure, serves as a general-purpose FHE compiler toolchain targeting multiple schemes, backends, and hardware accelerators. Together, these tiers represent a maturing pipeline from raw cryptographic operations to deployable privacy-preserving ML applications.

7.3 Reproducibility

A critical secondary challenge within the FHE-ML literature is the widespread lack of standardized reproducibility. While the broader machine learning community has heavily adopted open-source code mandates and standardized benchmarks (e.g., MLPerf), the FHE-ML domain remains fragmented [125].

Currently, evaluating the reproducibility of proposed schemes reveals three consistent gaps:

- **Code Availability:** While ecosystem-level frameworks like Concrete ML and HEIR are open-source, many bespoke algorithmic improvements proposed in the literature do not release their underlying implementation code [123, 124]. This omission prevents independent verification of latency and accuracy claims.
- **Parameter Transparency:** The sensitivity of FHE performance to cryptographic parameters cannot be overstated. Reproducing a result requires exact knowledge of the polynomial ring degree, ciphertext modulus, and target security level (λ). Papers that report runtimes without explicitly disclosing these parameters cannot be rigorously reproduced or audited for security flaws.
- **Lack of Standardized Benchmarks:** The literature heavily relies on datasets like MNIST and CIFAR-10 for evaluation. However, differences in data preprocessing, feature extraction (e.g., performing initial layers in plaintext before encrypting), and quantization mean that even when two papers evaluate the same dataset, their baselines are rarely identical.

For FHE-ML to mature into a highly deployable technology, the field must embrace rigorous reproducibility standards. While early initiatives like the FHE Benchmarking Suite[1] represent a promising step toward standardized evaluation, broader adoption is required. Future works should be expected to provide open-source reference implementations, exact cryptographic parameter checklists, and evaluations against these emerging fully encrypted baselines.

7.4 Optimization

The primary barrier to the widespread adoption of homomorphic encryption for privacy-preserving machine learning is its significant computational overhead. FHE-ML workloads incur profound computational and memory latency, which compounds significantly when scaled to deep learning architectures. Although the past decade has seen substantial progress in reducing execution times, one major challenge remains unresolved: training. Training neural networks over encrypted data is currently impractical due to the immense computational demands. While numerous schemes exist

[1] https://fhe-benchmarking.github.io/

for encrypted inference, they rely on pre-trained models. If an organization requires both training and privacy for deep learning, the computational cost poses a substantial barrier to adoption. For homomorphic encryption to achieve large-scale, real-world deployment in machine learning applications, this issue must be addressed.

Fortunately, there are promising avenues for improving existing schemes and implementations. The first involves algorithmic improvements, which refer to enhancements in the underlying homomorphic encryption techniques. These may include reducing ciphertext growth, minimizing error terms, decreasing the number of bootstrapping operations required by FHE schemes, or optimizing ciphertext packing to accelerate computations.

The second category involves implementation improvements, which focus on optimizing the software and hardware used to execute homomorphic operations. For instance, parallel programming techniques have shown strong potential for reducing execution times. Additionally, hardware acceleration, whether it be custom hardware or existing GPUs used for deep learning, could significantly cut computational costs and make encrypted training more feasible.

7.5 Usability

Implementing FHE-ML is a nontrivial task. Once a specific encryption scheme and library have been selected, the ML algorithm itself often must be implemented manually. This process is complicated by the inherent limitations in FHE (such as the lack of data-dependent branching) which makes many conventional programming techniques infeasible. Another major challenge lies in selecting appropriate security parameters. Poor parameter selection can easily result in either insecure implementations—due to parameters that are too small—or impractical ones—due to excessively large parameters that significantly degrade performance. While most FHE libraries provide default security settings, developers must still carefully evaluate whether these defaults offer an adequate level of security and efficiency for their use case.

Encouragingly, recent years have seen the emergence of homomorphic encryption compilers, which significantly lower the barrier to adoption. These compilers take programs written in more user-friendly languages and automatically convert them into homomorphic versions. They handle critical tasks such as parameter selection, data formatting, and bootstrapping operations, thereby simplifying development. These FHE tools and compilers can be evaluated along three key dimensions: usability (how easy the tool is to use), expressiveness (the degree of control it offers to developers), and performance (the efficiency of the generated homomorphic code) [126]. While performance is undeniably important for making FHE practical, usability is equally crucial. If FHE tools are too complex for developers without a background in cryptography, their adoption will remain limited to a small pool of experts.

To promote the broader adoption of privacy-preserving machine learning, FHE tools must be designed with the end user in mind. This includes minimizing the cryptographic expertise required to build secure, functional applications. One promising research direction is the development of libraries or frameworks that provide pre-built, homomorphically encrypted implementations of commonly used machine

learning algorithms. Such tools would reduce the need for manual implementation and debugging, saving developers time and effort.

7.6 Quantum Compatibility

Quantum computers are of particular interest to cryptographers due to their potential to break widely used public-key encryption schemes exponentially faster than classical computers. As demonstrated by Peter Shor in the 1994, a sufficiently powerful quantum computer can solve the discrete logarithm and integer factorization problems in polynomial time [127, 128]. This implies that cryptographic systems such as RSA, Diffie-Hellman, and elliptic-curve cryptography would become insecure in a post-quantum world. In anticipation of this so-called "Q Day," substantial research is underway in the emerging field of post-quantum cryptography to design and rigorously evaluate alternative cryptographic schemes.

While there has been considerable progress in developing fully homomorphic encryption (FHE) schemes tailored to quantum computations, relatively little attention has been paid to assessing the resilience of current FHE schemes against quantum attacks. This presents an important gap in the literature. Additionally, another promising research direction lies in exploring whether quantum computing could be leveraged to accelerate FHE computations themselves. If quantum speedups can be applied to FHE operations, this could significantly enhance the practicality of homomorphic encryption in large-scale applications.

8 Conclusion

Fully Homomorphic Encryption (FHE) enables computations to be performed directly on ciphertexts, producing encrypted results that, once decrypted, match the outcome of operations as if they were performed on the plaintext. While the idea of performing operations on encrypted data was first proposed in 1978, it was not until Gentry's breakthrough in 2009 that the first FHE scheme was realized. Since then, research in FHE has accelerated rapidly, transforming the idea of operating on encrypted data into a practical reality. This technology holds tremendous promise for privacy-preserving machine learning, enabling sensitive data to remain encrypted throughout the entire computation process. We introduced our taxonomy for categorizing FHE-ML works in Section 3. Today, a wide variety of machine learning algorithms can be adapted for use with FHE, as discussed in Sections 4 and 5. FHE-ML shows strong potential for privacy-critical domains, and early systems demonstrate feasibility for selected workloads, as explored in Section 6. While FHE has made significant strides in its nearly four-decade evolution, substantial challenges remain. As discussed in Section 7, continued progress in performance, usability, and integration with machine learning frameworks is essential to fully realize the vision of complete privacy for data in use.

Declarations

- Funding: This work was supported in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea

government (MSIT) (No. RS-2024-00431388, the Global Research Support Program in the Digital Field program). This work was also supported in part by the National Science Foundation under Grant 2429960 and Grant 2434899.

- Conflict of interest/Competing interests: The authors declare that they have no competing interests.
- Ethics approval and consent to participate: This article does not contain any studies involving human participants or animals performed by any of the authors.
- Consent for publication: All authors have read and approved the final manuscript.
- Data availability: Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.
- Materials availability: Not applicable.
- Code availability: Code sharing is not applicable to this article.
- Author contribution: Dillon Davidson: Literature review, Writing – original draft, Visualization. Matthew Jackson: Literature review, Writing – original draft. Rasheed Hussain: Conceptualization, Writing – review & editing. Zuobin Xiong: Writing – review & editing. Junggab Son: Conceptualization, Writing – review & editing, Supervision.

References

[1] Zhu, M., Wang, J., Yang, X., Zhang, Y., Zhang, L., Ren, H., Wu, B., Ye, L.: A review of the application of machine learning in water quality evaluation. Eco-Environment and Health **1**(2), 107–116 (2022) https://doi.org/10.1016/j.eehl.2022.06.001

[2] Halbouni, A., Gunawan, T.S., Habaebi, M.H., Halbouni, M., Kartiwi, M., Ahmad, R.: Machine learning and deep learning approaches for cybersecurity: A review. IEEE Access **10**, 19572–19585 (2022) https://doi.org/10.1109/ACCESS.2022.3151248

[3] Munir, H., Vogel, B., Jacobsson, A.: Artificial intelligence and machine learning approaches in digital education: A systematic revision. Information **13**(4) (2022) https://doi.org/10.3390/info13040203

[4] An, Q., Rahman, S., Zhou, J., Kang, J.J.: A comprehensive review on machine learning in healthcare industry: Classification, restrictions, opportunities and challenges. Sensors **23**(9) (2023) https://doi.org/10.3390/s23094178

[5] Nazareth, N., Ramana Reddy, Y.V.: Financial applications of machine learning: A literature review. Expert Systems with Applications **219**, 119640 (2023) https://doi.org/10.1016/j.eswa.2023.119640

[6] Zhou, I., Tofigh, F., Piccardi, M., Abolhasan, M., Franklin, D., Lipman, J.: Secure multi-party computation for machine learning: A survey. IEEE Access **12**, 53881–53899 (2024) https://doi.org/10.1109/ACCESS.2024.3388992

[7] Sagar, S., Keke, C.: Confidential machine learning on untrusted platforms: a survey. Cybersecurity **4**(1), 30 (2021) https://doi.org/10.1186/s42400-021-00092-8

[8] Song, L., Mittal, P.: Systematic evaluation of privacy risks of machine learning models. In: 30th USENIX Security Symposium (USENIX Security 21), pp. 2615–2632. USENIX Association, ??? (2021). https://www.usenix.org/conference/usenixsecurity21/presentation/song

[9] Toreini, E., Aitken, M., Coopamootoo, K., Elliott, K., Zelaya, C.G., Moorsel, A.: The relationship between trust in ai and trustworthy machine learning technologies. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. FAT* '20, pp. 272–283. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3351095.3372834

[10] Williamson, S.M., Prybutok, V.: Balancing privacy and progress: A review of privacy challenges, systemic oversight, and patient perceptions in ai-driven healthcare. Applied Sciences **14**(2) (2024) https://doi.org/10.3390/app14020675

[11] Plan, S.: The national artificial intelligence research and development strategic plan. National Science and Technology Council, Networking and Information Technology Research and Development Subcommittee (2016)

[12] Hoadley, D.S., Lucas, N.J.: Artificial intelligence and national security. Congressional Research Service Washington, DC (2018)

[13] Zhang, Y., Jia, R., Pei, H., Wang, W., Li, B., Song, D.: The secret revealer: Generative model-inversion attacks against deep neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)

[14] Dibbo, S.V.: Sok: Model inversion attack landscape: Taxonomy, challenges, and future roadmap. In: 2023 IEEE 36th Computer Security Foundations Symposium (CSF), pp. 439–456 (2023). https://doi.org/10.1109/CSF57540.2023.00027

[15] He, Z., Zhang, T., Lee, R.B.: Model inversion attacks against collaborative inference. In: Proceedings of the 35th Annual Computer Security Applications Conference. ACSAC '19, pp. 148–162. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3359789.3359824 . https://doi.org/10.1145/3359789.3359824

[16] Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 3–18 (2017). https://doi.org/10.1109/SP.2017.41

[17] Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing

machine learning models via prediction APIs. In: 25th USENIX Security Symposium (USENIX Security 16), pp. 601–618. USENIX Association, Austin, TX (2016). https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer

[18] Balle, B., Cherubin, G., Hayes, J.: Reconstructing training data with informed adversaries. In: 2022 IEEE Symposium on Security and Privacy (SP), pp. 1138–1156 (2022). https://doi.org/10.1109/SP46214.2022.9833677

[19] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and Harnessing Adversarial Examples (2015). https://arxiv.org/abs/1412.6572

[20] Goldblum, M., Tsipras, D., Xie, C., Chen, X., Schwarzschild, A., Song, D., Mądry, A., Li, B., Goldstein, T.: Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(2), 1563–1580 (2023) https://doi.org/10.1109/TPAMI.2022.3162397

[21] Wang, D., Li, C., Wen, S., Nepal, S., Xiang, Y.: Man-in-the-middle attacks against machine learning classifiers via malicious generative models. IEEE Transactions on Dependable and Secure Computing **18**(5), 2074–2087 (2021) https://doi.org/10.1109/TDSC.2020.3021008

[22] Barona, R., Anita, E.A.M.: A survey on data breach challenges in cloud computing security: Issues and threats. In: 2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT), pp. 1–8 (2017). https://doi.org/10.1109/ICCPCT.2017.8074287

[23] Zhu, R., Chen, G., Shen, W., Xie, X., Chang, R.: My Model is Malware to You: Transforming AI Models into Malware by Abusing TensorFlow APIs . In: 2025 IEEE Symposium on Security and Privacy (SP), pp. 486–503. IEEE Computer Society, Los Alamitos, CA, USA (2025). https://doi.org/10.1109/SP61157.2025.00012

[24] Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine **37**(3), 50–60 (2020) https://doi.org/10.1109/MSP.2020.2975749

[25] Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., Qi, H.: Beyond inferring class representatives: User-level privacy leakage from federated learning. In: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, pp. 2512–2520 (2019). https://doi.org/10.1109/INFOCOM.2019.8737416

[26] Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) Automata, Languages and Programming, pp. 1–12. Springer, Berlin, Heidelberg (2006)

[27] 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. CCS '16, pp. 308–318. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2976749.2978318

[28] Chaudhuri, K., Monteleoni, C., Sarwate, A.D.: Differentially private empirical risk minimization. Journal of Machine Learning Research **12**(3) (2011)

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

[30] 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. AISec'19, pp. 1–11. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3338501.3357370 . https://doi.org/10.1145/3338501.3357370

[31] Ben-David, A., Nisan, N., Pinkas, B.: Fairplaymp: a system for secure multi-party computation. In: Proceedings of the 15th ACM Conference on Computer and Communications Security. CCS '08, pp. 257–266. Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1455770.1455804

[32] Volgushev, N., Schwarzkopf, M., Getchell, B., Varia, M., Lapets, A., Bestavros, A.: Conclave: secure multi-party computation on big data. In: Proceedings of the Fourteenth EuroSys Conference 2019. EuroSys '19. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3302424.3303982

[33] Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017, pp. 409–437. Springer, Cham (2017)

[34] Li, B., Micciancio, D.: On the security of homomorphic encryption on approximate numbers. In: Canteaut, A., Standaert, F.-X. (eds.) Advances in Cryptology – EUROCRYPT 2021, pp. 648–677. Springer, Cham (2021)

[35] Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979) https://doi.org/10.1145/359168.359176

[36] Goldreich, O.: Secure multi-party computation. Manuscript. Preliminary version **78**(110), 1–108 (1998)

[37] Boneh, D., Gennaro, R., Goldfeder, S., Jain, A., Kim, S., Rasmussen, P.M.R.,

Sahai, A.: Threshold cryptosystems from threshold fully homomorphic encryption. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018, pp. 565–596. Springer, Cham (2018)

[38] Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: How to cope with perpetual leakage. In: Coppersmith, D. (ed.) Advances in Cryptology — CRYPT0' 95, pp. 339–352. Springer, Berlin, Heidelberg (1995)

[39] Caniglia, A., Franchini, F., Galantucci, S., Pirlo, G., Semeraro, G.: A multiple user cryptography approach using a one-time user key model and a (1, n) threshold polynomial secret sharing. Cryptography **10**(2) (2026) https://doi.org/10.3390/cryptography10020026

[40] Galantucci, S., Impedovo, D., Pirlo, G.: One time user key: A user-based secret sharing xor-ed model for multiple user cryptography in distributed systems. IEEE Access **9**, 148521–148534 (2021) https://doi.org/10.1109/ACCESS.2021.3124637

[41] Liu, B., Ding, M., Shaham, S., Rahayu, W., Farokhi, F., Lin, Z.: When machine learning meets privacy: A survey and outlook. ACM Comput. Surv. **54**(2) (2021) https://doi.org/10.1145/3436755

[42] Boulemtafes, A., Derhab, A., Challal, Y.: A review of privacy-preserving techniques for deep learning. Neurocomputing **384**, 21–45 (2020) https://doi.org/10.1016/j.neucom.2019.11.041

[43] Kaissis, G.A., Makowski, M.R., Rückert, D., Braren, R.F.: Secure, privacy-preserving and federated machine learning in medical imaging. Nature Machine Intelligence **2**(6), 305–311 (2020)

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

[45] Xu, R., Baracaldo, N., Joshi, J.: Privacy-Preserving Machine Learning: Methods, Challenges and Directions (2021). https://arxiv.org/abs/2108.04417

[46] Tran, A.-T., Luong, T.-D., Huynh, V.-N.: A comprehensive survey and taxonomy on privacy-preserving deep learning. Neurocomputing **576**, 127345 (2024) https://doi.org/10.1016/j.neucom.2024.127345

[47] Podschwadt, R., Takabi, D., Hu, P., Rafiei, M.H., Cai, Z.: A survey of deep learning architectures for privacy-preserving machine learning with fully homomorphic encryption. IEEE Access **10**, 117477–117500 (2022) https://doi.org/10.1109/ACCESS.2022.3219049

[48] Doan, T.V.T., Messai, M.-L., Gavin, G., Darmont, J.: A survey on implementations of homomorphic encryption schemes. J. Supercomput. **79**(13), 15098–15139 (2023) https://doi.org/10.1007/s11227-023-05233-z

[49] Pulido-Gaytan, L.B., Tchernykh, A., Cortés-Mendoza, J.M., Babenko, M., Radchenko, G.: A survey on privacy-preserving machine learning with fully homomorphic encryption. In: Nesmachnow, S., Castro, H., Tchernykh, A. (eds.) High Performance Computing, pp. 115–129. Springer, Cham (2021)

[50] Munjal, K., Bhatia, R.: A systematic review of homomorphic encryption and its contributions in healthcare industry. Complex & Intelligent Systems **9**(4), 3759–3786 (2023)

[51] Alloghani, M., M. Alani, M., Al-Jumeily, D., Baker, T., Mustafina, J., Hussain, A., J. Aljaaf, A.: A systematic review on the status and progress of homomorphic encryption technologies. Journal of Information Security and Applications **48**, 102362 (2019) https://doi.org/10.1016/j.jisa.2019.102362

[52] Wood, A., Najarian, K., Kahrobaei, D.: Homomorphic encryption for machine learning in medicine and bioinformatics. ACM Comput. Surv. **53**(4) (2020) https://doi.org/10.1145/3394658

[53] Pulido-Gaytan, B., Tchernykh, A., Cortés-Mendoza, J.M., Babenko, M., Radchenko, G., Avetisyan, A., Drozdov, A.Y.: Privacy-preserving neural networks with homomorphic encryption: Challenges and opportunities. Peer-to-Peer Networking and Applications **14**(3), 1666–1691 (2021)

[54] Marcolla, C., Sucasas, V., Manzano, M., Bassoli, R., Fitzek, F.H.P., Aaraj, N.: Survey on fully homomorphic encryption, theory, and applications. Proceedings of the IEEE **110**(10), 1572–1609 (2022) https://doi.org/10.1109/JPROC.2022.3205665

[55] Alharbi, A., Zamzami, H., Samkri, E.: Survey on homomorphic encryption and address of new trend. International Journal of Advanced Computer Science and Applications **11** (2020)

[56] Lyu, L., Yu, H., Ma, X., Chen, C., Sun, L., Zhao, J., Yang, Q., Yu, P.S.: Privacy and robustness in federated learning: Attacks and defenses. IEEE Transactions on Neural Networks and Learning Systems **35**(7), 8726–8746 (2024) https://doi.org/10.1109/TNNLS.2022.3216981

[57] Xia, Y., Liu, Y., Dong, S., Li, M., Guo, C.: Svca: Secure and verifiable chained aggregation for privacy-preserving federated learning. IEEE Internet of Things Journal **11**(10), 18351–18365 (2024) https://doi.org/10.1109/JIOT.2024.3363712

[58] Xia, Y., Liu, Y., Chen, J., Liang, Y., Khan, F., Alturki, R., Wang, X.: Teva:

Training-efficient and verifiable aggregation for federated learning for consumer electronics in industry 5.0. IEEE Transactions on Consumer Electronics **71**(2), 4248–4264 (2025) https://doi.org/10.1109/TCE.2024.3517739

[59] Chen, J., Yan, H., Liu, Z., Zhang, M., Xiong, H., Yu, S.: When federated learning meets privacy-preserving computation. ACM Comput. Surv. **56**(12) (2024) https://doi.org/10.1145/3679013

[60] Chen, J., Wang, Z., Srivastava, G., Alghamdi, T.A., Khan, F., Kumari, S., Xiong, H.: Industrial blockchain threshold signatures in federated learning for unified space-air-ground-sea model training. Journal of Industrial Information Integration **39**, 100593 (2024) https://doi.org/10.1016/j.jii.2024.100593

[61] Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Foundations of Secure Computations (1978)

[62] Acar, A., Aksu, H., Uluagac, A.S., Conti, M.: A survey on homomorphic encryption schemes: Theory and implementation. ACM Comput. Surv. **51**(4) (2018) https://doi.org/10.1145/3214303

[63] Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978) https://doi.org/10.1145/359340.359342

[64] Sander, T., Young, A., Yung, M.: Non-interactive cryptocomputing for nc/-sup 1/. In: 40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039), pp. 554–566 (1999). https://doi.org/10.1109/SFFCS.1999.814630

[65] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory **6**(3) (2014) https://doi.org/10.1145/2633600

[66] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. STOC '09, pp. 169–178. Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1536414.1536440

[67] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016, pp. 3–33. Springer, Berlin, Heidelberg (2016)

[68] Jung, W., Kim, S., Ahn, J.H., Cheon, J.H., Lee, Y.: Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with gpus. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(4), 114–148 (2021) https://doi.org/10.46586/tches.v2021.i4.114-148

[69] Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018, pp. 360–384. Springer, Cham (2018)

[70] Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: A full rns variant of approximate homomorphic encryption. In: Selected Areas in Cryptography–SAC 2018: 25th International Conference, Calgary, AB, Canada, August 15–17, 2018, Revised Selected Papers 25, pp. 347–368 (2019). Springer

[71] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Tfhe: fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (2020)

[72] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing. STOC '05, pp. 84–93. Association for Computing Machinery, New York, NY, USA (2005). https://doi.org/10.1145/1060590.1060603

[73] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010, pp. 1–23. Springer, Berlin, Heidelberg (2010)

[74] Goldwasser, S., Micali, S.: Probabilistic encryption & how to play mental poker keeping secret all partial information. In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing. STOC '82, pp. 365–377. Association for Computing Machinery, New York, NY, USA (1982). https://doi.org/10.1145/800070.802212 . https://doi.org/10.1145/800070.802212

[75] Guo, Q., Nabokov, D., Suvanto, E., Johansson, T.: Key recovery attacks on approximate homomorphic encryption with Non-Worst-Case noise flooding countermeasures. In: 33rd USENIX Security Symposium (USENIX Security 24), pp. 7447–7461. USENIX Association, Philadelphia, PA (2024). https://www.usenix.org/conference/usenixsecurity24/presentation/guo-qian

[76] Cheon, J.H., Choe, H., Passelègue, A., Stehlé, D., Suvanto, E.: Attacks against the ind-cpad security of exact fhe schemes. In: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security. CCS '24, pp. 2505–2519. Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3658644.3690341 . https://doi.org/10.1145/3658644.3690341

[77] Viand, A., Knabenhans, C., Hithnawi, A.: Verifiable Fully Homomorphic Encryption (2023). https://arxiv.org/abs/2301.07041

[78] Aydin, F., Aysu, A.: Leaking secrets in homomorphic encryption with side-channel attacks. Journal of Cryptographic Engineering **14**(2), 241–251 (2024)

[79] Hu, R., Shang, Y.-M., Luo, W., Tao, Y., Zhang, X.: When reasoning leaks membership: Membership inference attack on black-box large reasoning models. In: Proceedings of the ACM Web Conference 2026. WWW '26, pp. 3323–3334. Association for Computing Machinery, New York, NY, USA (2026). https://doi.org/10.1145/3774904.3792588 . https://doi.org/10.1145/3774904.3792588

[80] Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction APIs. In: 25th USENIX Security Symposium (USENIX Security 16), pp. 601–618. USENIX Association, Austin, TX (2016). https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer

[81] Bourse, F., Del Pino, R., Minelli, M., Wee, H.: Fhe circuit privacy almost for free. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016, pp. 62–89. Springer, Berlin, Heidelberg (2016)

[82] Angel, S., Chen, H., Laine, K., Setty, S.: Pir with compressed queries and amortized query processing. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 962–979 (2018). https://doi.org/10.1109/SP.2018.00062

[83] Stefanov, E., Dijk, M.V., Shi, E., Chan, T.-H.H., Fletcher, C., Ren, L., Yu, X., Devadas, S.: Path oram: An extremely simple oblivious ram protocol. J. ACM **65**(4) (2018) https://doi.org/10.1145/3177872

[84] Han, K., Hong, S., Cheon, J.H., Park, D.: Logistic regression on homomorphic encrypted data at scale. Proceedings of the AAAI Conference on Artificial Intelligence **33**(01), 9466–9471 (2019) https://doi.org/10.1609/aaai.v33i01.33019466

[85] Frery, J., Stoian, A., Bredehoft, R., Montero, L., Kherfallah, C., Chevallier-Mames, B., Meyre, A.: Privacy-Preserving Tree-Based Inference with Fully Homomorphic Encryption. Cryptology ePrint Archive, Paper 2023/258 (2023). https://eprint.iacr.org/2023/258

[86] Lee, E., Lee, J.-W., Lee, J., Kim, Y.-S., Kim, Y., No, J.-S., Choi, W.: Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (eds.) Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162, pp. 12403–12422. PMLR, ??? (2022). https://proceedings.mlr.press/v162/lee22e.html

[87] Stoian, A., Frery, J., Bredehoft, R., Montero, L., Kherfallah, C., Chevallier-Mames, B.: Deep neural networks for encrypted inference with tfhe. In: Dolev, S., Gudes, E., Paillier, P. (eds.) Cyber Security, Cryptology, and Machine Learning, pp. 493–500. Springer, Cham (2023)

[88] Zimerman, I., Baruch, M., Drucker, N., Ezov, G., Soceanu, O., Wolf, L.: Converting Transformers to Polynomial Form for Secure Inference Over Homomorphic Encryption (2023). https://arxiv.org/abs/2311.08610

[89] Rho, D., Kim, T., Park, M., Kim, J.W., Chae, H., Cheon, J.H., Ryu, E.K.: Encryption-Friendly LLM Architecture (2024). https://arxiv.org/abs/2410.02486

[90] Lou, Q., Jiang, L.: She: A fast and accurate deep neural network for encrypted data. In: Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc., ??? (2019)

[91] Liu, T.-L., Ku, Y.-T., Ho, M.-C., Liu, F.-H., Chang, M.-C., Hsu, C.-F., Chen, W.-C., Hung, S.-H.: An efficient ckks-fhew/tfhe hybrid encrypted inference framework. In: Katsikas, S., Abie, H., Ranise, S., Verderame, L., Cambiaso, E., Ugarelli, R., Praça, I., Li, W., Meng, W., Furnell, S., Katt, B., Pirbhulal, S., Shukla, A., Ianni, M., Dalla Preda, M., Choo, K.-K.R., Pupo Correia, M., Abhishta, A., Sileno, G., Alishahi, M., Kalutarage, H., Yanai, N. (eds.) Computer Security. ESORICS 2023 International Workshops, pp. 535–551. Springer, Cham (2024)

[92] Akavia, A., Leibovich, M., Resheff, Y.S., Ron, R., Shahar, M., Vald, M.: Privacy-preserving decision trees training and prediction. ACM Trans. Priv. Secur. **25**(3) (2022) https://doi.org/10.1145/3517197

[93] Nandakumar, K., Ratha, N., Pankanti, S., Halevi, S.: Towards deep neural network training on encrypted data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2019)

[94] Mihara, K., Yamaguchi, R., Mitsuishi, M., Maruyama, Y.: Neural Network Training With Homomorphic Encryption (2020). https://arxiv.org/abs/2012.13552

[95] Montero, L., Frery, J., Kherfallah, C., Bredehoft, R., Stoian, A.: Neural Network Training on Encrypted Data with TFHE (2024). https://arxiv.org/abs/2401.16136

[96] Kim, M., Song, Y., Wang, S., Xia, Y., Jiang, X.: Secure logistic regression based on homomorphic encryption: Design and evaluation. JMIR Med Inform **6**(2), 19 (2018) https://doi.org/10.2196/medinform.8805

[97] Shin, H., Choi, J., Lee, D., Kim, K., Lee, Y.: Fully homomorphic training and inference on binary decision tree and random forest. In: Garcia-Alfaro, J., Kozik, R., Choraś, M., Katsikas, S. (eds.) Computer Security – ESORICS 2024, pp. 217–237. Springer, Cham (2024)

[98] Cortes, C.: Support-vector networks. Machine Learning (1995)

[99] Huang, H., Wang, Y., Zong, H.: Support vector machine classification over encrypted data. Applied Intelligence **52**(6), 5938–5948 (2022)

[100] Park, S., Byun, J., Lee, J.: Privacy-preserving fair learning of support vector machine with homomorphic encryption. In: Proceedings of the ACM Web Conference 2022. WWW '22, pp. 3572–3583. Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3485447.3512252 . https://doi.org/10.1145/3485447.3512252

[101] Han, B., Shin, H., Kim, Y., Choi, J., Lee, Y.: Heaan-nb: Non-interactive privacy-preserving naive bayes using ckks for secure outsourced cloud computing. IEEE Access **12**, 110762–110780 (2024) https://doi.org/10.1109/ACCESS.2024.3438161

[102] Crawford, J.L.H., Gentry, C., Halevi, S., Platt, D., Shoup, V.: Doing real work with fhe: The case of logistic regression. In: Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography. WAHC '18, pp. 1–12. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3267973.3267974 . https://doi.org/10.1145/3267973.3267974

[103] Ao, W., Boddeti, V.N.: {AutoFHE}: Automated adaption of {CNNs} for efficient evaluation over {FHE}. In: 33rd USENIX Security Symposium (USENIX Security 24), pp. 2173–2190 (2024)

[104] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention Is All You Need (2023). https://arxiv.org/abs/1706.03762

[105] Zhou, J., Qian, H., Lu, X., Duan, Z., Huang, H., Shao, Z.: Polynomial activation neural networks: Modeling, stability analysis and coverage bp-training. Neurocomputing **359**, 227–240 (2019) https://doi.org/10.1016/j.neucom.2019.06.004

[106] Goyal, M., Goyal, R., Lall, B.: Improved polynomial neural networks with normalised activations. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2020). https://doi.org/10.1109/IJCNN48605.2020.9207535

[107] Zhang, J., Yang, X., He, L., Chen, K., Lu, W., Wang, Y., Hou, X., Liu, J., Ren, K., Yang, X.: Secure transformer inference made non-interactive. In: 32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025. The Internet Society, ??? (2025). https://www.ndss-symposium.org/ndss-paper/secure-transformer-inference-made-non-interactive/

[108] Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen,

W.: LoRA: Low-Rank Adaptation of Large Language Models (2021). https://arxiv.org/abs/2106.09685

[109] Zheng, M., Ju, L., Jiang, L.: Cofhe: Software and hardware co-design for fe-based machine learning as a service. Frontiers in Electronics **3** (2023) https://doi.org/10.3389/felec.2022.1091369

[110] Petrean, D.-E., Potolea, R.: Random forest evaluation using multi-key homomorphic encryption and lookup tables. International journal of information security **23**(3), 2023–2041 (2024)

[111] Lin, Y., Zhang, T., Mao, Y., Zhong, S.: Crossnet: A low-latency mlaas framework for privacy-preserving neural network inference on resource-limited devices. IEEE Transactions on Dependable and Secure Computing **22**(2), 1265–1280 (2025) https://doi.org/10.1109/TDSC.2024.3431590

[112] Morshed, T., Alhadidi, D., Mohammed, N.: Parallel linear regression on encrypted data. In: 2018 16th Annual Conference on Privacy, Security and Trust (PST), pp. 1–5 (2018). https://doi.org/10.1109/PST.2018.8514158

[113] Kahrobaei, D., Wood, A., Najarian, K.: Homomorphic encryption for machine learning in medicine and bioinformatics. Association for Computing Machinery (2020)

[114] Effendi, F., Chattopadhyay, A.: Privacy-preserving graph-based machine learning with fully homomorphic encryption for collaborative anti-money laundering. In: Knechtel, J., Chatterjee, U., Forte, D. (eds.) Security, Privacy, and Applied Cryptography Engineering, pp. 80–105. Springer, Cham (2025)

[115] Divakar Allavarpu, V., Naresh, V.S., Krishna Mohan, A.: Privacy-preserving credit risk analysis based on homomorphic encryption aware logistic regression in the cloud. Security and Privacy **7**(3), 372 (2024)

[116] Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., Liu, Y.: Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In: 2020 USENIX Annual Technical Conference (USENIX ATC 20), pp. 493–506. USENIX Association, Virtual (2020). https://www.usenix.org/conference/atc20/presentation/zhang-chengliang

[117] Mansouri, M., Önen, M., Ben Jaballah, W., Conti, M.: SoK: Secure Aggregation based on cryptographic schemes for Federated Learning. In: PETS 2023, 23rd Privacy Enhancing Technologies Symposium, vol. 2023. Lausanne, Switzerland, pp. 140–157 (2023). https://doi.org/10.56553/popets-2023-0009 . IACR. https://hal.science/hal-04282321

[118] Das, D.: Secure cloud computing algorithm using homomorphic encryption

and multi-party computation. In: 2018 International Conference on Information Networking (ICOIN), pp. 391–396 (2018). https://doi.org/10.1109/ICOIN.2018.8343147

[119] Microsoft SEAL (release 4.1). https://github.com/Microsoft/SEAL. Microsoft Research, Redmond, WA. (2023)

[120] Mouchet, C.V., Bossuat, J.-P., Troncoso-Pastoriza, J.R., Hubaux, J.-P.: Lattigo: A multiparty homomorphic encryption library in go, pp. 64–70 (2020). https://doi.org/10.25835/0072999 . https://infoscience.epfl.ch/handle/20.500.14299/193451

[121] Al Badawi, A., Bates, J., Bergamaschi, F., Cousins, D.B., Erabelli, S., Genise, N., Halevi, S., Hunt, H., Kim, A., Lee, Y., Liu, Z., Micciancio, D., Quah, I., Polyakov, Y., R.V., S., Rohloff, K., Saylor, J., Suponitsky, D., Triplett, M., Vaikuntanathan, V., Zucca, V.: Openfhe: Open-source fully homomorphic encryption library. In: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography. WAHC'22, pp. 53–63. Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3560827.3563379 . https://doi.org/10.1145/3560827.3563379

[122] Zama: TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data. https://github.com/zama-ai/tfhe-rs (2022)

[123] Zama: Concrete ML: a Privacy-Preserving Machine Learning Library using Fully Homomorphic Encryption for Data Scientists. https://github.com/zama-ai/concrete-ml (2022)

[124] Ali, A., Choi, J., Gipson, B., Gorantala, S., Kun, J., Legiest, W., Lim, L., Viand, A., Demissie, M.Z., Zheng, H.: HEIR: A Universal Compiler for Homomorphic Encryption (2025). https://arxiv.org/abs/2508.11095

[125] Reddi, V.J., Cheng, C., Kanter, D., Mattson, P., Schmuelling, G., Wu, C.-J., Anderson, B., Breughe, M., Charlebois, M., Chou, W., Chukka, R., Coleman, C., Davis, S., Deng, P., Diamos, G., Duke, J., Fick, D., Gardner, J.S., Hubara, I., Idgunji, S., Jablin, T.B., Jiao, J., John, T.S., Kanwar, P., Lee, D., Liao, J., Lokhmotov, A., Massa, F., Meng, P., Micikevicius, P., Osborne, C., Pekhimenko, G., Rajan, A.T.R., Sequeira, D., Sirasao, A., Sun, F., Tang, H., Thomson, M., Wei, F., Wu, E., Xu, L., Yamada, K., Yu, B., Yuan, G., Zhong, A., Zhang, P., Zhou, Y.: MLPerf Inference Benchmark (2019)

[126] Viand, A., Jattke, P., Hithnawi, A.: 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

[127] Shor, P.W.: Polynomial-time algorithms for prime factorization

and discrete logarithms on a quantum computer. SIAM Review **41**(2), 303–332 (1999) https://doi.org/10.1137/S0036144598347011 https://doi.org/10.1137/S0036144598347011

[128] Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124–134 (1994). https://doi.org/10.1109/SFCS.1994.365700