Paper-Reading-Record: Cryptography Assists Privacy-Preserving Deep Learning From Academia to Industry

Yu Zheng, The Chinese University of Hong Kong

Abstract

Machine/Deep learning has gained huge attention recently in various applications such as face recognition and disease diagnoses. Both academia and industry are proposing various promising ideas and implementing practical systems from different perspectives. A natural question is how to guarantee security and privacy when applying deep learning in the real world. Privacy refers to model privacy, input privacy, and output privacy. The model owner worries about intellectual property being stolen by malicious competitors, while the data owner concerns about private data leakage. This survey summarizes standard cryptographic tools that are used in privacy-preserving learning, and presents critical works in academia and industries. In this survey, we systematize the recent findings of privacy-preserving machine/deep focusing on cryptographic protection. We categorize the protections to homomorphic encryption, secret sharing, garbled circuits, functional encryption, zero-knowledge proof, and hybrid methods. The interesting implementations from different companies are also listed to make comparisons with works from the academic world. Through these studies and explorations, we show the triangle relationship among model functionality (*e.g.*, accuracy), efficiency (*e.g.*, computation, communication), and privacy, finally proposing some exciting questions and future directions. (KEEP EXPLORING)

1 Introduction

Fueled by massive data, sufficient computational resources, and adequate models, deep learning has the ability to process complicated tasks such as medical diagnosis and self-driving. A general framework of deep learning contains two phases: a training phase constructs a model for providing a reasonable approximation extracted from the available data; and an inference phase answers the queries using the distribution as the model learned. This survey explores recent prosperous research and real-world industrial applications in the inference phase where a client expects to query a deep learning model that has been trained by (or sold to) a server. For example, a whoremonger who wishes to perform an COVID test privately provides his medical record to a hospital with an auto-diagnosing model. In this scenario, the whoremonger is willing to leak neither his medical record nor the test results. Simultaneously, the hospital needs to protect its model's intellectual property against being stolen by other bad competitors or malicious parties. We formalize this question to a simple representation: $\mathbf{y} = f(\mathbf{x})$, where $\mathbf{x}, \mathbf{y}$ are the in-/outputs to a complicated function f. Deep learning model can be regarded as a complicated function f that inputs the query $\mathbf{x}$ and outputs the results $\mathbf{y}$. Since a high-accuracy model needs to train over large amounts of valuable data and consume lots of resources (*e.g.*, electricity, GPUs), it is vital to consider privacy in various fancy applications.

Cryptography is an indispensable science for secure communication and computation in insecure environments. Deep learning nowadays has been deployed everywhere in which the number of services (*e.g.*, IoT equipment) increases crazily, thus pressing an urgent call for protecting privacy. Researchers explore various cryptographic tools to provide such protections in different scenarios

and for targeting specific goals. Homomorphic encryption (HE) is an encryption method that allows computing a function (possibly bounded) on encryption of the message, without decryption and knowledge of the private key. The server applies homomorphically operation on the encrypted model and an encrypted query, contributing to model privacy and input-query privacy. Multiparty computation (MPC) enables a set of parties to compute a function of their private inputs without revealing any information beyond the function output. MPC protocols that rely on secret sharing, garbled circuits, and/or oblivious transfer help the server and clients to collaboratively compute the model output with private inputs. Functional encryption (FE) adopted into the limited solutions of inference protections is a generalization of public-key encryption in which possessing a secret key allows one to learn a function of what the ciphertext is encrypting. Zero-knowledge proof (ZKP) achieves proving knowledge without revealing the knowledge itself, which provides another promising direction in privacy-preserving learning. This method helps a model owner to convince anybody that the model computes a correct and accurate prediction on the data sample without leaking any information about the model itself. Briefly, available cryptographic tools shooting unique privacy goals are utilized and combined in a tremendous increase amount of works day-by-day, which inspires coming out of this survey.

Another finding is that both academia and industry pay great intention to privacy-preserving deep learning. Top universities and influential companies construct the user-friendly prototypes and product-oriented platforms that are profitable and socially good. These implementations are mostly open-source. A community OpenMined [1] is built to make the world more privacy-preserving by lowering the barrier-to-entry to private AI technologies. A wide range of proposals have been proposed but leverage accuracy, efficiency, and privacy in each work. Therefore, we explore privacy-preserving deep learning from academia and industry to present a systematized understanding. In this survey, we aim to investigate the problems below:

- Introduce deep learning basics and the cryptographic implementation commonly utilized in privacy-preserving deep learning.
- Summarize recent works systematically, and present the findings consisting of cryptographic protection and unprotection.
- List real-world implementations and industrial platforms.
- Compare the works in academia and industry and show the triangle relationship among model functionality, efficiency, and privacy.
- Propose potential exciting research directions and future works.

2 Background and Preliminary

2.1 Deep Learning Basics

Deep learning [2] is a learning algorithm that utilizes multiple layers to extract higher-level features from the raw input data progressively. Learning is classified to be supervised, semi-supervised, or unsupervised, which refers to two processes, *i.e.*, training and inference .

Training. Training is to create a convergent model conditioned on the given data distribution from a random model. After initialization, a deep learning model trains over a set of data that is collected and pre-processed. The model is defined to be a parametric function $f_\theta(x)$ inputting data/feature x and a parameter vector θ. A learning algorithm analyzes training data and the model outputs to find the reasonable model parameter vector. Supervised learning adjusts the parameters

[1] https://www.openmined.org

[2] https://en.wikipedia.org/wiki/Deep_learning

by reducing the gap between model predictions $f_\theta(x)$ and the expected output indicated by the dataset. Unsupervised learning adjusts its policy to take actions that yield the highest reward, while semi-supervised leaning combines both methods.

Inference. After the training process, a model is generated to infer predictions on the inputs that are not in training. The parameter vector θ is fixed, while $f_\theta(x)$ computes model outputs on the inputs x. Each model consists of an input layer, hidden layers, and an output layer. Nodes at each layer (probably) have different activation functions that defines the output of the node if given a set of inputs. Here, we list a number of commonly used active functions: (a) Softmax is a generalization of logistic regression that can be used for multi-class classification, $f(z) = \frac{e^z}{\sum_i e^{z_i}}$; (b) Sigmoid takes the value of one of the nodes and evaluate the function, $f(z) = \frac{1}{1+\exp(-z)}$; (c) Rectified Linear takes the value of one of the nodes in the feeding layer and compute the function, (d) Maxout is defined by $f(z_1, \ldots, z_n) = \max_i(z_i)$ to generalize the rectifier. This survey focuses on privacy-preserving inference, and we need to transform the common functions in the model to a cryptographic version.

Hands-on framework. (1) TensorFlow [3] is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. (2) PyTorch [4] is an open source machine learning framework that accelerates the path from research prototyping to production deployment.

2.2 Cryptographic Tools for Private Computation

In the lectures of CSCI5440, we have learned the notions of DDH, LWE, HE, SS, MPC, and so on. Based on these pieces of knowledge, we summarize the common tools in privacy-preserving deep learning and their open-source code repositories in the following.

2.2.1 CKKS

CKKS [CKKS17, CHK+18, CCS19] is a leveled homomorphic encryption scheme (*i.e.*, evaluate a circuit of fixed depth) for approximate arithmetic, supporting addition and multiplication over encryptions. The main idea is to add an encryption noise and compute the most significant bits of a resulting message after homomorphic operations. The decryption structure is, $\langle c, sk\rangle = m + e \mod q$, resulting in that the decryption is an approximate value of the original message. This can be achieved by the hardness assumptions such as LWE, ring-LWE, and NTRU problems. Notably, the size of a plaintext need to be small for guaranteeing that the homomorphic ciphertexts are still under the modulus.

Let $q_l = p^l \cdot q_0$ for $1 < l \leq L$, where $p > 0, q_0$ are respectively a base and a modulus. The integer p is for scaling in approximate computation. For a security parameter λ, a parameter of cyclotomic polynomial is set to, $M = M(\lambda, q_L)$. For a level $0 \leq l \leq L$, a ciphertext of level l is a vector in $\mathcal{R}_{q_L}^k$ for a fixed integer k, where a polynomial ring is $\mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$. CKKS includes five algorithms ($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, \mathsf{Mult}$) with constants B_{clean} and $B_{\mathsf{mult}}(l)$ for noise estimation.

- $\mathsf{KeyGen}(1^\lambda)$. Generate a secret value sk, a public information pk for encryption, and a evaluation key evk.
- $\mathsf{Enc}_{pk}(m)$. For a given polynomial $m \in \mathcal{R}$, output a ciphertext $c \in \mathcal{R}_{q_L}^k$. An encryption c of m will satisfy $\langle c, sk\rangle = m + e \pmod{q_L}$ for some small e. The constant B_{clean} denotes an

[3] https://www.tensorflow.org
[4] https://pytorch.org

encryption bound, i.e., error polynomial of a fresh ciphertext satisfies $\|e\|_\infty^{\mathsf{can}} \leq B_{\mathsf{clean}}$ with an overwhelming probability.

- $\mathsf{Dec}_{sk}(c)$. For a ciphertext c at level l, output a polynomial $m' \leftarrow \langle c, sk\rangle (\mathrm{mod}\ q_l)$ for the secret key sk.
- $\mathsf{Add}(c_1, c_2)$. For given encrypts of m_1 and m_2, output an encryption of $m_1 + m_2$. An error of output ciphertext is bounded by sum of two errors in input ciphertexts.
- $\mathsf{Mult}_{evk}(c_1, c_2)$. For a pair of ciphertexts (c_1, c_2), output a ciphertext $c_{\mathsf{mult}} \in \mathcal{R}_{q_L}^k$ which satisfies $\langle c_{\mathsf{mult}}, sk\rangle = \langle c_1, sk\rangle \cdot \langle c_2, sk\rangle + e_{\mathsf{mult}}$ (mod q_l) for some polynomial $e_{\mathsf{mult}} \in \mathcal{R}$ with $\|e_{\mathsf{mult}}\|_\infty^{\mathsf{can}} \leq B(l)$.

[CHK+18] extends above CKKS to a fully homomorphic encryption by refreshing low-level ciphertexts via Gentry's bootstrapping procedure. This methodology refreshes ciphertexts and make it bootstrappable for the evaluating an arbitrary circuit. As for the bootstrapping procedure, [CHK+18] employs ciphertext packing and combines it with evaluation strategy (recryption over the packed ciphertexts). Ciphertext packing allows encrypting multiple messages in a single ciphertext and performing computation parallelly, which is achieved by identifying a cyclotomic polynomial with real coefficients to a vector of complex numbers. [CCS19] speeds up the bootstrapping by two orders of magnitude, *i.e.*, from about 1s to 0.01s. It adopts a level-collapsing technique [HS14] for evaluating DFT-like linear transforms on a ciphertext and utilizes Chebyshev approximation for approximating the scaled sine function.

Code repository. (1) Microsoft Seal [5] is an open-source (MIT licensed) homomorphic encryption library developed by the Cryptography and Privacy Research group at Microsoft. Microsoft SEAL is written in modern standard C++ and is easy to compile and run in many different environments. It allows additions and multiplications to be performed on encrypted integers or real numbers. The CKKS scheme in Microsoft SEAL allows additions and multiplications on encrypted real or complex numbers, but yields only approximate results. In applications such as summing up encrypted real numbers, evaluating machine learning models on encrypted data, or computing distances of encrypted locations, CKKS is going to be by far the best choice. (2) HElib [6] is an open-source (Apache License v2.0) software library that implements homomorphic encryption (HE). It implements BGV and CKKS along with some optimizations to make homomorphic evaluation run faster. (3)HEAAN [7] is software library that implements HE that supports fixed point arithmetic and approximate operations between rational numbers. (3) Chet [DSC+19] (Did not find the code) is an optimizing compiler for fully-homomorphic neural-network inference.

2.2.2 ABY Series

ABY2.0 [PSSY20] (Arithmetic, Boolean, and Yao) achieves scalar production, matrix multiplication, comparison, maxpool, and equality testing. The 2PC for any functionality runs the following steps: 1) share all inputs via the sharing protocols; 2) gate by gate evaluation; 3) output the reconstruction via reconstruction protocol. Here, we first introduce Arithmetic, Boolean, Yao, and then the conversions between each two.

(1) *2PC for Arithmetic.* Consider two parties P_0, P_1 with private values a, b, respectively. The goal is to compute $\mathsf{a} \cdot \mathsf{b}$ privately, by modifying Beavers' circuit [Bea91] on input gate to compute on output gate. In details, P_0 and P_1 first generate multiplication triple $(\delta_\mathsf{a}, \delta_\mathsf{b}, \delta_\mathsf{ab})$ with $\delta_\mathsf{ab} = \delta_\mathsf{a}\delta_\mathsf{b} = \delta_\mathsf{c}$ offline (use CANS19 [RSS19], Eurocrypt15 [ALSZ15]). During online phase, P_0 and

[5] https://github.com/microsoft/SEAL
[6] https://github.com/homenc/HElib
[7] https://github.com/snucrypto/HEAAN

P_1 exchange the shares of $\Delta_{\mathsf{a}} = (\mathsf{a} + \delta_{\mathsf{a}})$ and $\Delta_{\mathsf{b}}(\mathsf{b} + \delta_{\mathsf{b}})$ to enable public reconstruction of Δ_{a} and Δ_{b}. Given scalars η_1, η_2 for linear operations, parties locally compute $\langle \mathsf{Ans} \rangle = \eta_1 \cdot \langle a \rangle + \eta_2 \cdot \langle b \rangle$, and locally set, $\Delta_{\mathsf{Ans}} = \eta_1 \cdot \Delta_{\mathsf{a}} + \eta_2 \cdot \Delta_{\mathsf{b}}, [\delta_{\mathsf{Ans}}]_i = \eta_1 \cdot [\delta_{\mathsf{a}}]_i + \eta_2 \cdot [\delta_{\mathsf{b}}]_i$ (2) *2PC for Boolean.* All protocols are identically same as "2PC for Arithmetic", but are defined over $\mathbb{Z}_{2^1}$ ($\mathbb{Z}_{2^k}$ in arithmetic). The addition is replaced by XORs ($\oplus$), and multiplication is by ANDs. (3) *2PC for Yao.* (Same as ABY) Use free XOR (4) *Conversation.* $\{\mathbf{A}, \mathbf{B}, \mathbf{Y}\}$ denote the Arithmetic, Boolean, and Yao. **Y2B.** XOR the last bit of key $\mathsf{K}_u^0, \mathsf{K}_u^1$. $\mathsf{K}_u^0, \mathsf{K}_u^1$ represent keys for a wire u, where $\mathsf{K}_u^1 = \mathsf{K}_u^0 \oplus R$ and R is a global string with $R[0] = 1$. **B2Y.** To convert a boolean u, P_i locally set $u_i = (1-i)\Delta_u \oplus [\delta_u]_i$ to keep $u = u_0 + u_1$. **A2Y.** To convert a boolean v, P_i locally set $v_i = (1-i)\Delta_v - [\delta_v]_i$ to keep $u = u_0 + u_1$. Then, A's side acts as a garbler, and Y's side acts as an evaluator. **Y2A.** To convert a Yao's v from P_0 to P_1, P_0 acts as a garbler, and shares arithmetic r and Yaos's r to P_1, where r is random number. P_1 acts as an evaluator and gets Yao's values $(v + r)$. Then P_1 decodes (both defined over $\mathbb{Z}_{2^k}$) Yao's value to arithmetic value $v + r$, and locally compute $v = v + r - r$ by arithmetic r. **A2B.** First "A2Y", then "Y2B". **B2A.** i) Use "Y2A" (*i.e.*, $\mathbb{Z}_{2^k}$ with $k = 1$). ii) For a boolean v (regarded as a bit), the boolean operation $v = \Delta_v \oplus \delta_v$ can be represented by arithmetic operation $v = \Delta_v + \delta_v - 2\Delta_v\delta_v$. P_i locally computes $[v]_i = i \cdot \Delta_v + (1 - 2\Delta_v) \cdot [\delta]_i$ and does the arithmetic share.

Code repository. (1) ABY [8] library allows to pre-compute almost all cryptographic operations and provides efficient conversions between secure computation schemes based on pre-computed oblivious transfer extensions with OT extension library. This code is provided as an experimental implementation for testing purposes and should not be used in a productive environment. (2) ABY3 library [9] provides the semi-honest implementation of ABY3 [MR18] and SS-Database [MRR20]. ABY3 is for switching back and forth among 3PC which is of independent interest, while SS-Database supports performing SQL-style joins and private-set-intersection atop multiparty computation. Blaze [PS20] (Do not find the code) improves the truncation method of ABY3 that consumes a round complexity. (3) Chameleon [RWT+18] (Do not find the code) is a secure computation framework built on ABY. (4) Do not find the code of ABY2.0.

2.2.3 SPDZ Variants

SPDZ [DPSZ12] is a multiparty computation protocol secure against an active adversary corrupting up to $n-1$ of n players. This protocol based on a somewhat homomorphic cryptosystem computes arithmetic circuits over any finite field $\mathbb{F}_{p^k}$. This protocol has a preprocessing phase and an online phase, where the preprocessing phase is independent of the function to be computed and of the inputs. The online phase has total computational complexity linear in n, the number of layers. The complexity of preprocessing phase is dominated by the public-key operations, *i.e.*, $O(n^2/s)$, where s is a parameter that increases with the security parameter of the cryptosystem.

[DPSZ12] integrates the classic circuit randomization [Bea91] (for parallel preprocessing), an SIMD technique [SV14] (no need bootstrapping), an SHE cryptosystem [BV11], and zero-knowledge proof [BDOZ11] (ZKF, for online verification). In the preprocessing phase, applying SIMD approach into the SHE cryptosystem is for handling many values in parallel in a single ciphertext. Then, a distributed decryption procedure is only against passive attacks, where an active attacker may add a known error term to the decryption. These wrong decryptions result in incorrect shares of secret values, which are solved by checking involving MACs (via ZKF) in the online phase.

Practical-SPDZ [DKL+13] improves the underlying SPDZ protocol above in the aspect of supporting with associated security models and optimizing the implementation layer. SPDZ-2 [KSS13]

[8] https://github.com/encryptogroup/ABY
[9] https://github.com/ladnir/aby3

builds a runtime environment automatically determines the minimum number of rounds needed for a computation, thus minimizing the overall cost of the computation. MASCOT [KOS16] reduces multiplication cost in preprocessing phase by using simple consistency checking and privacy amplification techniques via oblivious transfer. SPD$\mathbb{Z}_{2^k}$ [CDE+18] changes definition [DPSZ12] from field $\mathbb{F}_{p^k}$ to ring $\mathbb{Z}_{2^k}$, which are useful for simplifying implementations and applications. Overdrive [KPR18] replaces cut-and-choose proof with simpler Schnorr-like protocol, which guarantees security against active malicious parties and improves the efficiency compared with SPDZ and MASCOT. MP-SPDZ [Kel20] is a fork of SPDZ-2, which extends SPDZ-2 to 34 MPC protocol variants. [Kel20] covers all commonly used security models (honest/dishonest majority and semi-honest/malicious corruption) as well as computation of binary and arithmetic circuits (the latter modulo primes and powers of two). The underlying primitives employed include secret sharing, oblivious transfer, homomorphic encryption, and garbled circuits.

Code repository. (1) Awesome-MPC [10] collects almost all open-source software designed for MPC. (2) MP-SPDZ [11] is a software to benchmark various MPC protocols such as SPDZ, SPD$\mathbb{Z}_{2^k}$, MASCOT, Overdrive, etc. The breadth of implemented protocols coupled with an accessible high-level interface makes it suitable to benchmark the cost of computation in various security models for researchers both with and without a background in secure computation. (3) SCALE-MAMBA [12] provides an integrated framework for computation in prime order fields with both dishonest and honest majority. The SCALE system consists of three main sub-systems, *i.e.*, An offline phase, an online phase, and a compiler, where online and offline phases are fully integrated. The compiler takes a program written in a special language MAMBA, and then turns it into byte-code which can be executed by the SCALE system.

3 Critical Works with Cryptographic Protection

In recent years, the number of works in privacy-preserving prediction with deep learning increases *crazily* in both the security community and the computer vision community. Privacy-preserving prediction refers to input privacy, model privacy, and output privacy. Considering that a query is sent to a classification model, the query is unwilling to be exposed to the model. In contrast, the model owner expects to keep the model intelligence property. It's not wisdom that we summarize all up-to-date works in this survey due to the paper explosion. We try to analyze the critical works and classify existing research, eventually presenting our findings.

3.1 Challenge on Encrypted Learning from Cryptography Limitations

Transforming a deep learning model to a cryptographic version is not trivial and straight-forward. Deep learning models contains float model parameters, non-polynomial functions, while cryptography primitives are always defined over integer field. This part investigates transforming from pure deep learning to encrypted learning and, meantime, reaching the required model accuracy.

Data conversion. When constructing MPC protocols via HE, GC, and SS, we have three types of computations, *i.e.*, boolean, arithmetic, and Yao's circuit. Both boolean and Yao's circuit are defined over $\{0, 1\}$, but the arithmetic circuit is defined over an integer field $\mathbb{Z}$. One problem is to convert an integer to a bit string (*e.g.*, modular 2), and conversely, convert it back. In deep learning, except for a number of models (*e.g.*, Bayesian neural network), most of the models have rational parameters. Another question is to convert a rational number to an integer, *e.g.*, $\mathbb{Z}_{2^l}$,

[10] https://github.com/rdragos/awesome-mpc
[11] https://github.com/data61/MP-SPDZ
[12] https://homes.esat.kuleuven.be/~nsmart/SCALE/, https://github.com/KULeuven-COSIC/SCALE-MAMBA

where l is the bit length. A rational number can be represented by two integers, one for decimal part and the other for integer part. Another question is how to guarantee the accuracy of digits before/after the dot. For example, $0.5 + 0.5$ is equal to 1.0. We may lose the carry if trivially splitting 0.5 into 0 and 5 since $0 + 0 = 0$. Lots of works based on the primitives above contribute to optimizing circuit depth, communication rounds, computation, and overhead.

Function representation. Practical HE schemes are limited to (bounded) additions and multiplications, while deep neural networks performs computation on non-linear functions. Non-linear/polynomial functions can not be represented simply by combining additions and multiplications. This intrinsically makes HE hard to support computations of deep learning directly and efficiently but requires some tricks for modification. Linear function or convolution layer are combinations of addition and multiplication, which can be translated to an encrypted version via HE operations. The computational complexity is linear or polynomial to the number of operations. HE-based works optimize the heavy cryptographic computation and reduce the encryption size from the perspectives of math (*e.g.*, use simple functions for approximating complex/non-polynomial functions), programming (*e.g.*, simplify the compiler/circuit), and bootstrapping. Generally, HE-based solutions are suitable for computing small multiplicative multiplications with large input size (*e.g.*, convolutional layer).

SS shares the secret (*e.g.*, parameters of the function) to multiple participants to collaboratively reveal sensitive information after enough shares are collected. SS-based MPC has less computation for protecting the function, but incurs some communication load. It shares the model parameters, database, input query, and secret key for fully HE to multiple clients and/or servers. Many works preprocess computations offline as much as possible for leveraging the online communication. For example, ABY3 [MR18] reduces to only one-round online communication under the assumption of malicious adversary. GC computes non-linear functions such as logistic regression but demands an interactive protocol between non-linear and linear circuits. A feasible solution is to approximate/replace the non-polynomial functions by polynomial function. There are some methods of approximating non-polynomial function such as numerical approximation, Taylor expansion, and Chebyshev polynomials. Yet, an accurate approximation of non-polynomial function always has high degrees, resulting in lots of multiplication operations.

3.2 Potential Response Built atop Cryptography

Various cryptographic primitives play different roles in privacy-preserving deep learning. Here, we briefly summarize the attributes of each class of works below.

HE-based works. Models protected by HE need to change the structures and reshape them to LHE-friendly activation functions such as the square function and polynomial functions to reduce the computational needs of LHE as possible. The LHE-friendly activation functions may lower the accuracy of original models. Homomorphic encryption performs better than garbled circuits when the computation with small multiplicative depth and large input size, *e.g.*, convolution operation and matrix multiplication.

GC-based works. GC supports arbitrary functions and requires only a constant round of interactions which is independent of the computation depth. However, it requires a high communication cost and overhead for multiplication, *i.e.*, quadratic to input operands. The non-linear computations (unsupported directly) like ReLU or maxpool functions need be reformatted as linear-size circuits for using garbled circuits. GMW needs less communication than GC but requires many rounds of interactions between the two parties.

SS-based works. SS-based schemes are (computationally) relatively cheap for multiplication but require a strong assumption of several (non-colluding) computing servers. The number of

interactions between server and client (*i.e.*, round complexity) is linear with the computation depth. This brings difficulty in the scalability of more complex, deeper models with higher accuracy.

FE-based works. FE enables the data owner to determine what to compute but limits the function to inner-product in existing privacy-preserving works. The first work was proposed in 2019, and we would spend more time exploring this stream in the future.

ZK-based works. This is another promising direction that guarantees the integrity of returning results for the clients and validates machine learning models (*e.g.*, decision tree) of the owner. State-of-arts are at the scenario of shallow machine learning models or in the early stages of developing ZK deep learning. The reasons for hindering deeper models are expensive communication and time-consuming proof generation.

3.3 Recent Works on Privacy-Preserving Learning

We highlight the main works in privacy-preserving learning. Many works combine multiple cryptographic tools for different considerations.

CryptoNets [13] [GDL+16] converts plaintext neural networks to cryptonets that can be applicable to encrypted data via leveled HE. Let $c = [q/t]m + e + hs$ be the encrypted message m and w be the weight vector. Encryption of $(m + w)$ is by multiplying w by $[q/t]$ and adding that to c, which results in $[q/t](m + w) + e + hs$. For multiplication, $cw = [q/t]mw + e' + hs'$ is computed for a sparse w. The major concern lies in preventing the coefficients of plaintext polynomials from overflowing t. Selecting a large t may solve it, but enlarges noise rapidly in the ciphertext and decreases (with fixed q) upper bound of noise. [GDL+16] utilizes SIMD operation to parallelly encode large numbers and merge them later.

MultiFheNN [ZDH20] Data owners jointly compute the key generation algorithm (of an FHE) using a generic MPC protocol to achieve multi-key HE.

CryptoDL [HTG17, HTG19, HTGJ17, HTGW18] approximates derivation of the activation functions (*i.e.*, ReLU, Sigmoid, and Tanh) with low degree polynomials which is later encrypted by leveled HE. The (leveled) homomorphically encrypted network performs classification over encrypted data.

MiniONN [14] [LJLA17] transforms a neural network to an oblivious one for privacy-preserving predictions, which supports linear transformations, popular activation functions and pooling operations via additive secret sharing and garbled circuits. Nonlinear functions (*e.g.*, sigmoid and tanh) are approximated by polynomial splines. The rough idea is to additively share each (server and client) of the input and output values for every layer of a neural network.

Gazelle [JVC18] builds a protocol to protect model and input during the inference phase. The idea is to combine homomorphic encryption (*i.e.*, BFV) and traditional two-party computation (*i.e.*, Yao's garble circuit). Homomorphic linear algebra kernels via packed additively HE supports SIMD additions for speeding up operations. [JVC18] designs an encryption switching protocols which convert between homomorphic and garbled circuit encoding for building interfaces. Switching protocol alternates the use of packed additively HE and garbled circuits to evaluate the targeted neural network. The client encrypts its inputs by SIMD linear homomorphic encryption scheme and sends them to the server, who later performs homomorphic operation on the first layer (either convolution or fully connected). The results are packed into a ciphertext that contains inputs to the non-linear (ReLU) layer which is evaluated by garbled circuits.

XONN [RSC+19] solves the problem of privacy-preserving inference on binary (*i.e.*, ± 1) NN (first integer layer for inputting data and the following binary layers) via GC protocol. XONN

[13] https://github.com/microsoft/CryptoNets
[14] https://github.com/SSGAalto/minionn

replaces the costly matrix-multiplication operations by XNOR operations [KS08] that are free in GC, and requires only a constant round of interactions for any number of layers in the model.

SecureNN [15] [WGC19] uses ABY3 for private model inference under semi-honest and malicious adversaries. It computes non-linear functions (*e.g.*, ReLU, pooling) by computing deviation since the deviation of ReLU(x) is 1 if $x \geq 0$ and 0 otherwise. This approach maintains the origin model accuracy compared with approximation.

Garbled NN [16] [BCM+19] utilizes pure GC techniques to protect NN for constant-round complexity. It develops a library for garbling neural networks, containing the first implementation of the BMR garbling scheme on NN.

DeepSecure [RRK18] proposes a scalable privacy-preserving inference by Yao's gable circuit and a pro-processing approach, which involves four steps: (i) The client garbles Boolean circuit of the DL architecture. (ii) The client sends garbled tables to the server along with her input wire labels. Both client and cloud obliviously transfer the wire labels associated with cloud server's inputs. (iii) The cloud server evaluates the garbled circuit and computes the corresponding encrypted data inference. (iv) The result is sent back to the client who later decrypts it under garbled keys to reveal the inference label. The pre-processing approach is by: (i) transformation of input data to an ensemble of lower-dimensional sub-spaces, and (ii) avoiding the execution of (inactive) neurons by leveraging the sparsity structure inherent in models.

CrypTFlow [17] [KRC+20] is a system that converts TensorFlow inference code into Secure Multi-party Computation (MPC) protocols at the push of a button. It has three components: 1) Athos is a compiler that compiles TensorFlow inference code to secure computation protocols; 2) Porthos is an improved semi-honest 3-party secure computation protocol (tolerating one corruption) that builds upon SecureNN [WGC19]; 3) Aramis compiles MPC protocols secure against semi-honest adversaries into MPC protocols that are secure against malicious adversaries, by leveraging secure hardware.

CrypTFlow2 [18] [RRK+20] is a cryptographic framework for secure inference over realistic Deep Neural Networks (DNNs) using secure 2-party computation. Using CrypTFlow2, the authors present the first secure inference over ImageNet-scale DNNs like ResNet50 and DenseNet121.

TF-Encrypted [19] [DMD+18] is an open source library built on TensorFlow and SPDZ for private prediction.

nGraph-HE [20] [BCCW19, BLCW19] is a deep learning compiler that enables data scientists to deploy models with popular frameworks such as TensorFlow and PyTorch. nGraph- HE2 extends nGraph-HE to enable privacy-preserving inference on standard, pre-trained models using their native activation functions and number fields (supporting real numbers). [BCCW19] presents CKKS-specific optimizations, speeding up runtime in scalar encoding and packing CKKS plaintext into complex numbers for fast multiplication and addition.

MP2ML [BCD+20] integrates nGraph-HE and the secure two-party computation framework ABY to overcome the limitations of leaking the intermediate feature maps to the data owner. nGraph-HE computes linear layers using the CKKS. The non-polynomial activation functions are evaluated by data owner who obtains the intermediate feature maps, which results in leaking the feature maps to the data owner. MP2ML overcomes this problem by adding independent random numbers for each layer via ABY.

[15] https://github.com/snwagh/securenn-public; Following work: https://github.com/snwagh/falcon-public
[16] https://github.com/GaloisInc/swanky
[17] https://github.com/mpc-msri/EzPC
[18] https://github.com/mpc-msri/EzPC
[19] https://github.com/tf-encrypted/tf-encrypted
[20] https://github.com/IntelAI/he-transformer

CryptoNN [XJL19] protects inference by functional encryption [ABCP15]. A trusted authority generates a master secret key *msk* and public key *mpk*, and distributes *mpk* to the server and clients. The client first needs to pre-process the training data as required and then encrypt all the pre-processed training data using the public key, *mpk*. The server trains over encrypted data and decrypts the (encrypted) parameters to update its model. For each training iteration, the server receives the function derived key *skf* from the authority for decrypting a specific function (*e.g.*, dot-product, element-wise subtract) between the input layer and the first hidden layer. The server then uses the decryption to compute the rest of hidden layers.

QuadFEML [21] [RSG+19] proposes a new functional encryption scheme to compute quadratic functions that exist in the model. The data owner provides a decryption key to allow the server to learn a specific function evaluation of some encrypted data. Another problem is that functional encryption schemes are not against identifying a set of selected sensitive features via the classification results. Performing adversarial training on the first layers is to avoid a malicious client recovering specific sensitive features (during the inference) according to the plaintext output.

Delphi [22] [MLS+20]designs a hybrid cryptographic protocol that automatically generates neural network architecture configurations on navigating the performance-accuracy trade-offs of the hybrid protocol.

Misc. k-ishNN [SFR20] is a scale classifier that relaxes kNN to allow $\kappa \approx k$ nearest neighbors with some probability. SQNN [BEDK19] combines QNN [JKC+18] and MP-SPDZ to use quantized model parameters for better accuracy. LoLa [BGE19] change the message/input representation to achieve low latency privacy-preserving Inference. AriaNN [RPB20] is a low-interaction privacy-preserving deep Learning via function secret sharing.

3.4 Remarks on Cryptographic Unprotection

Cryptography is not a panacea. Cryptographic protection [PMSW18] guarantees security and privacy, strictly relying on the security model. In the scenario of inference, the client who may be honest-but-curious or malicious sends a query to the server who may be honest-but-curious or malicious, too. Informally, privacy of clients refers that the server knows "nothing" of the plaintext query or returning results while protecting the server's model means not "inferring" the model parameters by I/Os of the model. "Nothing" contrasts with the allowing (reasonable) leakage, and "inferring" may (or may not) consider advanced attacks. What to protect should be defined strictly with the consideration of adversary's ability, privacy goal, etc. On one side, different cryptography primitives own distinct security models; On the other side, more and more attack techniques have been proposed to infer data or break model accuracy, thus resulting in that only cryptographic protection is not enough. Security models on the sides of both client and server are against the eavesdroppers.

Key generation. Many schemes in cryptography assume a trusted party for generating and/or distributing keys. Actually, this is a very strong assumption in the real-world, which should be removed as possible.

Attack. Deep learning captures features from data, which stimulates various active attacks both in white-box and black-box settings. The inevitable problem is that the client can know the input-output pairs after querying a server. These pairs can be utilized to perform black-box attacks by training a shadow model on the client's side. This may be solved by an adversarial training technique that reduces the leakage of specific sensitive features; however, cryptography can do nothing.

[21] https://github.com/LaRiffle/collateral-learning
[22] https://github.com/mc2-project/delphi

Stateless model. Existing works assume that a trained model exists. The problem is formalized to how to transform a well-trained model to a cryptographic version. The server does not consider the security of model update, *i.e.*, stateless model. Updating models require interactive protocols, which may leaks more sensitive information of model and data, leading to a stronger security guarantee.

4 Industrial Platform and Product

Works from the industry focus on different (more profit-motivated) points compared with that in academia. The academic researchers may start a topic from personal interests, but influential companies pay great effort to build user-friendly platforms and create profitable products. Business environments are more complicated and crafty since companies rush to snatch profits and occupy market share. This section shares the industrial works starting from analyzing the real-world concerns below.

4.1 Real-World Concern

We are able to assume an ideal world in our paper while the engineers in a company consider more practical settings.

Stronger security. It is reasonable for a research paper to assume an honest-but-curious party. Yet, there may exist more active adversaries to steal the private information of a company. An industrial product needs a more robust security model that is against various attacks.

More efficiency. An industrial platform may contain vast amounts of data. For example, a database of medical records may be in the size of billions. Communication overhead is extremely enormous for all the data. Besides, industrial deep learning models are probably large and complex, so that they have the ability of featuring details. Real-time usage on computation-limiting devices (*e.g.*, IoT equipment) needs to be more communication-efficient.

Specific data. A company may be required to train a specific model by inputting uncommon data. For example, Google works with multiple hospitals to train a disease diagnosis model. The hospitals provide rare disease information that cannot be obtained in public. This requires a company to optimize the model by considering the particular data type.

Various applications. Industrial works face with various applications such as financial, medical treatment, business, etc. Different applications have different focal points, *i.e.*, what sensitive information is. Tedious details need to be dealt with here and there.

More profitable. The economic cost is another perspective. A team leader in the company considers money investment for a project and potential profits in the future. He takes the speed of entering the market of products into consideration and balances the cost and product performance. People get used to a specific product while others similar products may be difficult to be adopted by users.

4.2 Existing Works from Industry

This part lists the existing platforms and research which can be utilized for future works.

Google. [23] AI-platform [24] can be deployed to perform privacy-preserving deep learning.

[23] https://research.google/pubs/?area=security-privacy-and-abuse-prevention

[24] https://cloud.google.com/ai-platform

Microsoft. [25] Many projects such as CryptoNets [GDL+16], EzPC, CrypTFlow [KRC+20], CrypTFlow2 [RRK+20], Falcon [WTB+21] have been conducted.

Facebook. [26] Facebook AI creates CrypTen [27] that is a new framework built on PyTorch to facilitate research in secure and privacy-preserving machine learning.

Intel. nGraph-HE2 [BCCW19, BLCW19] is from Intel.

Cape. [28] This team worked at large enterprises like Salesforce, Cisco, IBM, and DataDog, as well as startups like GoInstant, Snips, DigitalOcean, Zenly and OmniSci.

Ant Group. TF-encrypted [29] is a framework for encrypted machine learning in TensorFlow, which is built by Ant Group and Cape.

Huakong Tsingjiao. [30] PrivPy [LX19] is a framework for privacy-preserving collaborative data mining by integrating ABY and SPDZ, providing a practical implementation in python.

RISE. [31] A research group from UC Berkeley builds MC2 [32], which is adopted by IBM, ERICSSON, Ant Group, and Microsoft.

Openmined. [33] This community consists of open-source repositories on encrypted computation.

5 Triangle Relationship

5.1 Privacy

Privacy refers to model privacy on the server and data (both input and output) privacy on the client. Cryptographic protection provides privacy in the way of hiding the plaintext information of the model and data from other parties. Provable security is guaranteed but vulnerable to advanced attacks, *e.g.*, inferring data via input-output pairs. Under the assumption that the model is global optimal and model architecture is fixed, we get an almost similar model conditioned on the same data. This property can be utilized to steal model parameters.

5.2 Functionality

Functionality is a general notion of evaluative metrics, which refers to model accuracy, false negative, the ability to learn a new task, etc. A model deployed on the industrial platform is additionally required to evaluate the ability to earn money. Functionality has nothing with security or privacy but summarizes the performance of a pure model in the plaintext. To make a plaintext model to be secure, several works modifies the origin model to adapt cryptographic schemes, resulting in sacrificing a bit of accuracy.

5.3 Efficiency

Efficiency evaluates the cryptographic protocols on privacy-preserving deep learning, such as communication and computation. Yet, most cryptographic protections are still heavy for computations

[25] https://www.microsoft.com/en-us/research/research-area/security-privacy-cryptography
[26] https://ai.facebook.com
[27] https://crypten.ai/;https://github.com/facebookresearch/crypten
[28] https://capeprivacy.com
[29] https://tf-encrypted.readthedocs.io/en/latest/
[30] https://www.tsingj.com/
[31] https://rise.cs.berkeley.edu/
[32] https://github.com/mc2-project/mc2
[33] https://github.com/OpenMined

in deep learning, especially for the non-polynomial functions. Deep learning needs a large set of high dimensional inputs such as images and videos, resulting in inflexibility of direct usage of cryptography.

5.4 Call Extra Assistance

There are multiple existing tools for enhancing privacy-preserving deep learning. From the prospective of privacy, we can consider differential privacy, federated learning, and trusted execution environment. Model distillation and quantization may be utilized to improve functionality and efficiency. Taking attacks into consideration, we may integrate adversarial learning for generating a robust model.

Differential privacy [DR14] adds noise on the model gradients and training data for privacy-preserving deep learning.

Federated learning provides a way of training over multiple data sources and meantime protects the data privacy. Amounts of work combine federated learning and differential privacy for protecting private data during model training.

Trusted execution environment [CSFP20, ZZ16] is utilized to generate keys and perform decryption (or acting as a trusted party) for improve the efficiency.

Model distillation and pruning compresses the large deep learning models to a smaller one while maintaining functionality. This method gives another view on what to protect instead of the model parameters.

Quantization both from prospective of learning and cryptography (or Math) enables to connect deep learning and cryptography (or other privacy enhancing techniques) better.

Adversarial training [TB19] is typically tailored to defense the attacks via adversarial samples that may hurt model or steal data.

6 Conclusion

This survey explores the existing works in privacy-preserving and presents some interesting findings. We find that this topic is exciting to conduct research, finally giving some simple and superficial [34] views.

References

[ABCP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 733–751. Springer, 2015.

[ALSZ15] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*

[34] The views are just from a clumsy Ph.D. student who is not experienced enough to raise something deep.

- *34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 673–701. Springer, 2015.

[BCCW19] Fabian Boemer, Anamaria Costache, Rosario Cammarota, and Casimir Wierzynski. ngraph-he2: A high-throughput framework for neural network inference on encrypted data. In Michael Brenner, Tancrède Lepoint, and Kurt Rohloff, editors, *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography, WAHC@CCS 2019, London, UK, November 11-15, 2019*, pages 45–56. ACM, 2019.

[BCD+20] Fabian Boemer, Rosario Cammarota, Daniel Demmler, Thomas Schneider, and Hossein Yalame. MP2ML: a mixed-protocol machine learning framework for private inference. In Melanie Volkamer and Christian Wressnegger, editors, *ARES 2020: The 15th International Conference on Availability, Reliability and Security, Virtual Event, Ireland, August 25-28, 2020*, pages 14:1–14:10. ACM, 2020.

[BCM+19] Marshall Ball, Brent Carmer, Tal Malkin, Mike Rosulek, and Nichole Schimanski. Garbled neural networks are practical, 2019.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 169–188. Springer, 2011.

[Bea91] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 1991.

[BEDK19] Assi Barak, Daniel Escudero, Anders P. K. Dalskov, and Marcel Keller. Secure evaluation of quantized neural networks. *IACR Cryptol. ePrint Arch.*, 2019:131, 2019.

[BGE19] Alon Brutzkus, Ran Gilad-Bachrach, and Oren Elisha. Low latency privacy preserving inference. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 812–821. PMLR, 2019.

[BLCW19] Fabian Boemer, Yixing Lao, Rosario Cammarota, and Casimir Wierzynski. ngraph-he: a graph compiler for deep learning on homomorphically encrypted data. In Francesca Palumbo, Michela Becchi, Martin Schulz, and Kento Sato, editors, *Proceedings of the 16th ACM International Conference on Computing Frontiers, CF 2019, Alghero, Italy, April 30 - May 2, 2019*, pages 3–13. ACM, 2019.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.

[CCS19] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Improved bootstrapping for approximate homomorphic encryption. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 34–54. Springer, 2019.

[CDE+18] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. Spd$\mathbb{F}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 769–798. Springer, 2018.

[CHK+18] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 360–384. Springer, 2018.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437. Springer, 2017.

[CSFP20] David Cerdeira, Nuno Santos, Pedro Fonseca, and Sandro Pinto. Sok: Understanding the prevailing security vulnerabilities in trustzone-assisted TEE systems. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 1416–1432. IEEE, 2020.

[DKL+13] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, volume 8134 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.

[DMD+18] Morten Dahl, Jason Mancuso, Yann Dupis, Ben Decoste, Morgan Giraud, Ian Livingstone, Justin Patriquin, and Gavin Uhma. Private machine learning in tensorflow using secure computation. *CoRR*, abs/1810.08130, 2018.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[DSC+19] Roshan Dathathri, Olli Saarikivi, Hao Chen, Kim Laine, Kristin E. Lauter, Saeed Maleki, Madanlal Musuvathi, and Todd Mytkowicz. CHET: an optimizing compiler for fully-homomorphic neural-network inferencing. In Kathryn S. McKinley and Kathleen Fisher, editors, *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019*, pages 142–156. ACM, 2019.

[GDL+16] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 201–210. JMLR.org, 2016.

[HS14] Shai Halevi and Victor Shoup. Algorithms in helib. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2014.

[HTG17] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data, 2017.

[HTG19] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Deep neural networks classification over encrypted data. In Gail-Joon Ahn, Bhavani M. Thuraisingham, Murat Kantarcioglu, and Ram Krishnan, editors, *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy, CODASPY 2019, Richardson, TX, USA, March 25-27, 2019*, pages 97–108. ACM, 2019.

[HTGJ17] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Catherine Jones. Privacy-preserving machine learning in cloud. In Bhavani M. Thuraisingham, Ghassan Karame, and Angelos Stavrou, editors, *Proceedings of the 9th Cloud Computing Security Workshop, CCSW@CCS 2017, Dallas, TX, USA, November 3, 2017*, pages 39–43. ACM, 2017.

[HTGW18] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N. Wright. Privacy-preserving machine learning as a service. *Proc. Priv. Enhancing Technol.*, 2018(3):123–142, 2018.

[JKC+18] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2704–2713. IEEE Computer Society, 2018.

[JVC18] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 1651–1669. USENIX Association, 2018.

[Kel20] Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1575–1590. ACM, 2020.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 830–842. ACM, 2016.

[KPR18] Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ great again. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 158–189. Springer, 2018.

[KRC+20] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow: Secure tensorflow inference. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 336–353. IEEE, 2020.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 486–498. Springer, 2008.

[KSS13] Marcel Keller, Peter Scholl, and Nigel P. Smart. An architecture for practical actively secure MPC with dishonest majority. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 549–560. ACM, 2013.

[LJLA17] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. Oblivious neural network predictions via minionn transformations. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 619–631. ACM, 2017.

[LX19] Yi Li and Wei Xu. Privpy: General and scalable privacy-preserving data mining. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 1299–1307. ACM, 2019.

[MLS+20] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. Delphi: A cryptographic inference service for neural networks. In Srdjan Capkun and Franziska Roesner, editors, *29th USENIX Security Symposium,*

USENIX Security 2020, August 12-14, 2020, pages 2505–2522. USENIX Association, 2020.

[MR18] Payman Mohassel and Peter Rindal. Aby^3: A mixed protocol framework for machine learning. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 35–52. ACM, 2018.

[MRR20] Payman Mohassel, Peter Rindal, and Mike Rosulek. Fast database joins and PSI for secret shared data. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1271–1287. ACM, 2020.

[PMSW18] Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha, and Michael P. Wellman. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, pages 399–414. IEEE, 2018.

[PS20] Arpita Patra and Ajith Suresh. BLAZE: blazing fast privacy-preserving machine learning. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

[PSSY20] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. ABY2.0: improved mixed-protocol secure two-party computation. *IACR Cryptol. ePrint Arch.*, 2020:1225, 2020.

[RPB20] Theo Ryffel, David Pointcheval, and Francis Bach. ARIANN: low-interaction privacy-preserving deep learning via function secret sharing. *CoRR*, abs/2006.04593, 2020.

[RRK18] Bita Darvish Rouhani, M. Sadegh Riazi, and Farinaz Koushanfar. Deepsecure: scalable provably-secure deep learning. In *Proceedings of the 55th Annual Design Automation Conference, DAC 2018, San Francisco, CA, USA, June 24-29, 2018*, pages 2:1–2:6. ACM, 2018.

[RRK+20] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow2: Practical 2-party secure inference. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 325–342. ACM, 2020.

[RSC+19] M. Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin E. Lauter, and Farinaz Koushanfar. XONN: xnor-based oblivious deep neural network inference. In Nadia Heninger and Patrick Traynor, editors, *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 1501–1518. USENIX Association, 2019.

[RSG+19] Theo Ryffel, Edouard Dufour Sans, Romain Gay, Francis Bach, and David Pointcheval. Partially encrypted machine learning using functional encryption. *CoRR*, abs/1905.10214, 2019.

[RSS19] Deevashwer Rathee, Thomas Schneider, and K. K. Shukla. Improved multiplication triple generation over rings via rlwe-based AHE. In Yi Mu, Robert H. Deng, and Xinyi Huang, editors, *Cryptology and Network Security - 18th International Conference, CANS 2019, Fuzhou, China, October 25-27, 2019, Proceedings*, volume 11829 of *Lecture Notes in Computer Science*, pages 347–359. Springer, 2019.

[RWT+18] M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, pages 707–721. ACM, 2018.

[SFR20] Hayim Shaul, Dan Feldman, and Daniela Rus. Secure k-ish nearest neighbors classifier. *Proc. Priv. Enhancing Technol.*, 2020(3):42–61, 2020.

[SV14] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptogr.*, 71(1):57–81, 2014.

[TB19] Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5858–5868, 2019.

[WGC19] Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: 3-party secure computation for neural network training. *Proc. Priv. Enhancing Technol.*, 2019(3):26–49, 2019.

[WTB+21] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. In *Privacy Enhancing Technologies Symposium (PETS)*, June 2021.

[XJL19] Runhua Xu, James B. D. Joshi, and Chao Li. Cryptonn: Training neural networks over encrypted data. In *39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019, Dallas, TX, USA, July 7-10, 2019*, pages 1199–1209. IEEE, 2019.

[ZDH20] Ruiyu Zhu, Changchang Ding, and Yan Huang. Practical MPC+FHE with applications in secure multi-partyneural network evaluation. *IACR Cryptol. ePrint Arch.*, 2020:550, 2020.

[ZZ16] Fengwei Zhang and Hongwei Zhang. Sok: A study of using hardware-assisted isolated execution environments for security. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016, HASP@ICSA 2016, Seoul, Republic of Korea, June 18, 2016*, pages 3:1–3:8. ACM, 2016.