PrivDNFIS: Privacy-preserving and Efficient Deep Neuro-Fuzzy Inference System

Hao Ren [1 2 3], **Xiao Lan** [1 2 3], **Rui Tang** [1 2 3 *], **Xingshu Chen** [1 2 3 †]

[1] School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China.
[2] Key Laboratory of Data Protection and Intelligent Management (Sichuan University), Ministry of Education, China.
[3] Cyber Science Research Institute, Sichuan University, Chengdu, China.
hao.ren, lanxiao, tangrscu, chenxsh@scu.edu.cn

Abstract

Deep Neuro-Fuzzy Inference Systems (DNFIS) seamlessly fuse neural networks with the fuzzy inference system enabling intricate decision-making and knowledge representation, while upholding a commendable degree of adaptability and interpretability. However, the challenge of privacy-preserving inference (PI) over DNFIS has remained largely uncharted, with no prior research addressing this critical issue. In this paper, we embark on an exploration of this issue. We introduce an efficient and secure PI framework for DNFIS, named PrivDNFIS, which leverages the post-quantum lattice-based homomorphic encryption to implement secure computation protocols for PI over DNFIS. Our work incorporates several non-trivial performance enhancements. Firstly, it consolidates multiple elements of input feature vectors into a single message, reducing encryption/decryption overhead. Secondly, building upon this novel encoding approach, PrivDNFIS can perform ciphertext aggregation and vector-vector inner production without necessitating time-consuming ciphertext rotation operations. Thirdly, we replace the softmax function in the DNFIS layer with a quadratic function to further enhance inference efficiency, without compromising the inference accuracy. Under the given threat model, we provide formal security proof for PrivDNFIS. In comprehensive experimental results, PrivDNFIS demonstrates an approximately 1.9 to 4.4 times reduction in end-to-end time cost compared to the benchmark.

Introduction

Background Deep Neural Networks (DNNs) (LeCun, Bengio, and Hinton 2015; He et al. 2016) excelling in areas like image recognition, natural language processing (NLP), and medical diagnosis (Jin et al. 2024). However, their complexity, involving billions of connection weights, introduces challenges in understanding and trust. This necessitates explainable AI (Talpur et al. 2023; Yeganejou, Dick, and Miller 2019) to provide human-friendly explanations and build trust. A promising solution is deep neuro-fuzzy inference systems (DNFIS) (Yeganejou, Dick, and Miller 2019; Yeganejou et al. 2023), which combine deep learning's pattern recognition with fuzzy logic's interpretability. This integration enables precise and transparent decision-making, especially in healthcare and legal contexts (Talpur et al. 2023). DNFIS leverages rule-based representations and intuitive linguistic variables for clarity, with proven high performance in fusing a DNN for feature extraction with a fuzzy system for classification (Yeganejou et al. 2023).

Despite the benefits of DNFIS, privacy concerns pose significant obstacles to its development. When users send inference requests to third-party model owners, they must share their original input, risking their privacy. In the case of DNFIS, input privacy is critical in scenarios like medical diagnosis. Users may upload sensitive health data to leverage the system's capability for handling uncertainty and imprecision in decision-making. A breach in such cases could reveal private health records, leading to severe personal data leakage. Therefore, safeguarding input privacy for model inference services like DNFIS is critical and urgently needed, as mandated by regulations like GDPR (Lu et al. 2021).

Related Work Preserving input privacy in DNN model inference services has been extensively studied by both AI and security communities (Liu et al. 2021). The main objective of a privacy-preserving inference (PI) scheme is to compute inference results without accessing the original input. A promising approach involves using advanced cryptographic tools like fully homomorphic encryption (FHE) (Fan and Vercauteren 2012; Viand, Jattke, and Hithnawi 2021) and secure multiparty computation (MPC) (Hastings et al. 2019; Rathee et al. 2020). These methods allow the server to evaluate inference functions on the encrypted input. FHE supports arbitrary computation on ciphertexts without decryption or interaction (Viand, Jattke, and Hithnawi 2021), but its processing of non-linear functions is computationally intensive. Therefore, state-of-the-art (SOTA) PI schemes (Rathee et al. 2020; Huang et al. 2022; Lu et al. 2025) often use FHE for linear function evaluations and MPC for non-linear functions, such as ReLU and softmax. While MPC has lower computational overhead than FHE, it incurs higher communication costs and requires multiple interaction rounds (Hastings et al. 2019). Due to the complexity of DNNs and cryptographic techniques, the existing PI scheme (Huang et al. 2022) takes over two minutes to process a single query, indicating room for efficiency improvements.

To our knowledge, PI for DNFIS has not been explored in existing work. This paper seeks to initiate this topic and en-

*The corresponding author.
†The corresponding author.

courage future research. Our proposed scheme, PrivDNFIS, involves a co-design of the DNFIS network architecture (Yeganejou et al. 2023) and cryptographic primitives (Viand, Jattke, and Hithnawi 2021) to achieve PI for DNFIS, incorporating several non-trivial performance optimizations.

Technical Challenges It is non-trivial to build a PI scheme that accommodates DNFIS (Yeganejou et al. 2023) with provable security and high efficiency.

- **Functionality.** The main challenge is scrutinizing the DNFIS network architecture and the internal structure of its neurons to specify the computational tasks. Moreover, there is no existing secure computation protocol capable of accommodating fuzzy membership functions as indicated in (Yeganejou et al. 2023). This necessitates an investigation of both DNFIS and cryptographic techniques.
- **Security.** The preservation of user input privacy necessitates a theoretical guarantee. Particularly, formally establishing its semantic security under the given threat model is a non-trivial task.
- **Efficiency.** Excising PI schemes are often criticized for their inefficiency. However, cost reduction becomes difficult when complex inference functions and provable security are essential requirements.

Our Contributions We sum the contribution of PrivDNFIS as follows.

- PrivDNFIS initializes the research on designing a PI scheme for DNFIS. A secure and efficient PI system is presented using somewhat homomorphic encryption (SHE) (Fan and Vercauteren 2012) and ciphertext extraction technique (Chen et al. 2021). PrivDNFIS is expected to motivate future efforts on this pivotal issue.
- We give a formal security proof for PrivDNFIS under the widely applied semi-honest threat model (Hastings et al. 2019). The analysis is sound and succinct.
- PrivDNFIS proposes several optimizations for secure aggregation and vector-vector inner production protocols. Specifically, the heavy ciphertext rotation operation is eliminated. Besides, we carefully tailor the last layer (softmax) of the existing DNFIS model (Yeganejou et al. 2023) to further boost the overall performance without undermining the accuracy. Compared to the benchmark, PrivDNFIS delivers roughly $1.9 \sim 4.4\times$ end-to-end time cost reduction as evidenced by the experimental results.

Preliminaries

Deep Neuro-fuzzy Inference Systems The classical Adaptive Neuro-Fuzzy Inference System (ANFIS) (Jang 1993) alters the network architecture to mimic the fuzzy inference system (Jang, Sun, and Mizutani 1997). ANFIS consists of the following five layers. ❶ **Input layer**. It receives the input features from the user. Input nodes pass the input to the next layer. ❷ **Fuzzification layer**. It fuzzifies the inputs, mapping input values to fuzzy sets. Each input node is linked to fuzzy set (i.e., membership) functions, typically triangular or Gaussian, for fuzzifying input values and yielding membership degrees that signify the input values' association with each fuzzy set. ❸ **Rule layer**. It calculates

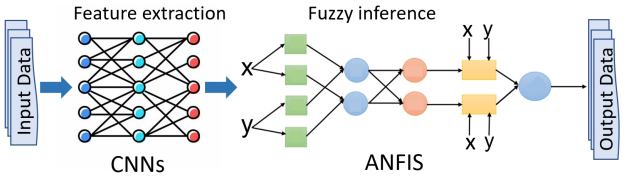

Figure 1: Diagram of Deep Neuro-Fuzzy Inference System.

the "firing strength" for each fuzzy rule, typically obtained as the product of all antecedent membership functions. ❹ **Hidden layer**. It computes the output for each rule, and its output results from multiplying the rule's activation by the fuzzification layer's output. The output values will be aggregated using node weights (trainable parameters), typically through a linear combination. ❺ **Output layer**. This layer computes and outputs the class probabilities. As shown in Fig. 1, DNFIS uses CNNs for feature extraction that produce the inputs for ANFIS. This framework follows the paradigm of the sequential deep neuro-fuzzy system (DNFS) (Talpur et al. 2023). It enjoys the benefits provided by both CNNs and ANFIS and is easy to deploy in practice.

Somewhat Homomorphic Encryption (SHE) SHE (Viand, Jattke, and Hithnawi 2021) is rooted in the Learning With Errors (LWE) problem (Gentry, Sahai, and Waters 2013) and its ring variant (RLWE) (Fan and Vercauteren 2012). They share common public parameters denoted as $\mathsf{HE.pp} = N, p, q, \sigma$, where p and q are integers with $q \gg p > 0$, and σ represents the standard deviation of error sampling. In the RLWE scheme, plaintext messages are polynomials in $R_{N,p}$. This scheme is comprised of three essential algorithms, namely $\mathsf{R.KeyGen}(\mathsf{HE.pp})$ for public/private key $(\mathrm{pk}_\mathrm{R}, \mathrm{sk}_\mathrm{R})$ generation, $\mathsf{R.Enc}(\mathrm{pk}_\mathrm{R}, \widehat{m})$ for encryption of messages $\widehat{m} \in R_{N,p}$ into ciphertexts $\mathsf{CT} \in R^2_{N,q}$, and $\mathsf{R.Dec}(\mathrm{sk}_\mathrm{R}, \mathsf{CT})$ for recovering message $\widehat{m}$. In contrast, the LWE scheme operates with plaintexts in $\mathbb{Z}_p$ and ciphertexts in $\mathbb{Z}_q^{N+1}$. Its structure mirrors that of the RLWE scheme, encapsulated in $\mathsf{L.KeyGen}(\cdot), \mathsf{L.Enc}(\cdot), \mathsf{L.Dec}(\cdot)$. SHE shall support the following homomorphic evaluations.

- $\mathsf{CtPtAdd}(\mathsf{CT}_{\widehat{u}}, \widehat{v}) \to \mathsf{CT}$. It takes the ciphertext $\mathsf{CT}_{\widehat{u}}$ of the plaintext message $\widehat{u}$ and another plaintext message $\widehat{v}$ as the inputs. It returns CT, that is a ciphertext of $\widehat{u} + \widehat{v}$.
- $\mathsf{CtCtAdd}(\mathsf{CT}_{\widehat{u}}, \mathsf{CT}_{\widehat{v}}) \to \mathsf{CT}$. It takes the ciphertext $\mathsf{CT}_{\widehat{u}}$, $\mathsf{CT}_{\widehat{v}}$ of the plaintetext messages $\widehat{u}, \widehat{v}$, respectively as the inputs. It returns CT, that is a ciphertext of $\widehat{u} + \widehat{v}$.
- $\mathsf{CtPtMul}(\mathsf{CT}_{\widehat{u}}, \widehat{v}) \to \mathsf{CT}$. It takes the ciphertext $\mathsf{CT}_{\widehat{u}}$ of the plaintext message $\widehat{u}$ and another plaintext message $\widehat{v}$ as the inputs. It returns CT i.e. a ciphertext of $\widehat{u} \times \widehat{v}$.
- $\mathsf{CtCtMul}(\mathsf{CT}_{\widehat{u}}, \mathsf{CT}_{\widehat{v}}) \to \mathsf{CT}$. It takes the ciphertext $\mathsf{CT}_{\widehat{u}}$ of $\widehat{u}$ and another ciphertext of $\widehat{v}$ as the inputs. It returns CT i.e. a ciphertext of $\widehat{u} \times \widehat{v}$.
- $\mathsf{Extract}(\mathsf{CT}, i)$. For a given message $\widehat{m}$ and its RLWE ciphertext CT, this function can extract the i-th coefficient of $\widehat{m}$ from CT, and transfer it to a LWE format ciphertext. The decryption key can be computed by a key switch algorithm (Chen et al. 2021).

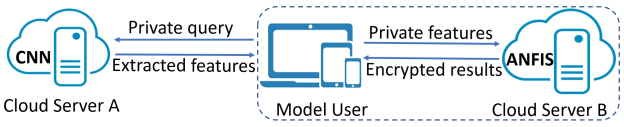

Figure 2: System model of PrivDNFIS.

Problem Statement

System Model As shown in Fig. 2, the system of PrivDNFIS consists of three entities, that are the cloud server A ($\mathcal{CS}_A$), cloud server B ($\mathcal{CS}_B$), and model user ($\mathcal{MU}$). In the following, we elaborate on the role of each entity.

- $\mathcal{CS}_A$. It receives the private query request from the $\mathcal{MU}$ to extract the features using the CNN model. This task is considered as the prelude of PrivDNFIS.
- $\mathcal{MU}$. It could be any individual or organization in practice, that needs to issue inference service requests in a privacy-preserving manner. It first obtains the encrypted extracted features from $\mathcal{CS}_A$ and interacts with $\mathcal{CS}_B$ to acquire the encrypted results over ANFIS.
- $\mathcal{CS}_B$. It receives the private encrypted features from $\mathcal{MU}$ and computes the inference over the ciphertext domain. At last, it returns the final results to $\mathcal{MU}$. In PrivDNFIS, we put main efforts into designing secure computing protocols for $\mathcal{CS}_B$ over the modified ANFIS model.

Threat Model $\mathcal{MU}$ is considered to be fully trusted. It will strictly follow the designed secure protocols. In practice, $\mathcal{MU}$ has no motivation to manipulate the queries at the price of receiving inaccurate inference results. Cloud service providers follow pre-defined secure protocols and return the correct results for economic rewards. However, the cloud service providers may be interested in the content of the received data and peek at the users' privacy (Lu et al. 2021) for economic interests. This threat model is practical and prevalently applied by existing schemes (Lu et al. 2021). Prior works dubbed it as "semi-honest" (Lindell 2017). Under this model, $\mathcal{CS}_A$ and $\mathcal{CS}_B$ will faithfully execute the given private computing protocols but attempt to peek into user data. We acknowledge that it is vital to develop malicious user detection mechanisms. Due to space limitations, we will study this issue in future work.

Proposed Scheme

We now present PrivDNFIS layer by layer, detailing how the secure computation tasks are evaluated. We commit to co-design on ANFIS (Jang 1993) and SHE acceleration to boost the efficiency with mild accuracy loss.

Input Layer

The input layer takes the encrypted input from $\mathcal{MU}$ and feeds it to the next layer. However, this is non-trivial to generate the inputs that meet the following requirements. First, the original content of the features should be strictly concealed from the server $\mathcal{CS}_B$ with semantic security. Second, the private input should support the processing of the consecutive layers. Third, the message packing method needs to be compatible with the homomorphic evaluation functions. It is non-trivial to achieve these three goals simultaneously because they demand strong privacy preservation, rich functionality, and high efficiency in one scheme, that is the design goal of PrivDNFIS.

Instead of using traditional Chinese Remainder Theorem (CRT) packing methods like SV (Smart and Vercauteren 2014), PrivDNFIS directly packs messages into the coefficients of the plaintext polynomial. For example, given a feature vector $\mathbf{v} = [1, 2, 3, 4]$, the vector is encoded as the polynomial $\widehat{v} = 1x^0 + 2x^1 + 3x^2 + 4x^3$. Each element of $\mathbf{v}$ is mapped into the polynomial coefficients, with unused coefficients set to 0. This method can also support Single Instruction Multiple Data (SIMD) (Viand, Jattke, and Hithnawi 2021) homomorphic evaluations. Let $\widehat{u}, \widehat{v} \in R_{N,p}$, then we have the addition $\widehat{a} = \widehat{u} + \widehat{v}$ over $R_{N,p}$ is defined as $\widehat{a}[i] = \widehat{u}[i] + \widehat{v}[i]$. The product $\widehat{p}[i] = \widehat{u} \times \widehat{v}$ is calculated as the following equation.

$$\widehat{p}[i] = \sum_{0 \le j \le i} \widehat{u}[j]\widehat{v}[i-j] - \sum_{i < j < N} \widehat{u}[j]\widehat{v}[N+j-i] \bmod p. \tag{1}$$

The correctness of Equation 1 is based on the property $x^N \equiv -1 \bmod x^N + 1$. Given RLWE ciphertexts $\mathsf{CT}_{\widehat{u}}$ and $\mathsf{CT}_{\widehat{v}}$, invoking $\mathsf{CtPtAdd}(\mathsf{CT}_{\widehat{u}}, \widehat{v})$ or $\mathsf{CtCtAdd}(\mathsf{CT}_{\widehat{u}}, \mathsf{CT}_{\widehat{v}})$ returns a ciphertext of $\widehat{u}+\widehat{v}$, supporting element-wise addition homomorphically (SIMD). Note that, invoking $\mathsf{CtPtMul}(\mathsf{CT}_{\widehat{u}}, \widehat{v})$ results in a ciphertext of $\widehat{p}$, not element-wise multiplication, meaning SIMD isn't supported.

Let $\mathbf{d}$ be the feature vector, $\mathcal{MU}$ first maps the elements into a polynomial using the above method. We write the polynomial as $\widehat{d} \in R_{N,p}$. Then $\mathcal{MU}$ obtains the ciphertext of $\widehat{d}$ as $\mathsf{CT}_{\widehat{d}} \leftarrow \mathsf{R.Enc}(\mathrm{pk}_\mathrm{R}, \widehat{d}) \in R^2_{N,q}$. In practice, the time cost of ciphertext-ciphertext multiplication homomorphic evaluation can be $100\times$ than $\mathsf{CtCtAdd}(\cdot)$, $\mathsf{CtPtAdd}(\cdot)$, and $\mathsf{CtPtMul}(\cdot)$ (Mughees and Ren 2023). Thus, if we can refrain from this operation, the entire performance will be significantly improved. To achieve this, $\mathcal{MU}$ calculates $\mathbf{d}^*[i] = (\mathbf{d}[i])^2, i \in [|\mathbf{d}|]$. Then it is encoded as a polynomial $\widehat{d^*}$. At last, $\mathcal{MU}$ encrypts $\widehat{d^*}$ as $\mathsf{CT}_{\widehat{d^*}} \leftarrow \mathsf{R.Enc}(\mathrm{pk}_\mathrm{R}, \widehat{d^*})$. Thus far, $\mathcal{MU}$ has finished the preparation of the private input, that is a pair of RLWE SHE ciphertexts $\{\mathsf{CT}_{\widehat{d}}, \mathsf{CT}_{\widehat{d^*}}\}$. It will be uploaded to $\mathcal{CS}_B$ as the output of the input layer.

Fuzzification Layer

The fuzzification layer converts input data into fuzzy linguistic variables. It does this by linking each input variable to fuzzy sets (rules) defined by membership functions, which determine how strongly an input value corresponds to linguistic terms like "low," "medium," or "high." For a feature vector with n dimensions and l fuzzy rules, $n \cdot l$ membership functions are needed. In PrivDNFIS, Gaussian membership functions, commonly used in fuzzy systems (Bai et al. 2021; Zhang et al. 2021), are employed for each input and rule. These functions have two trainable parameters: the mean and the standard deviation.

For each input value $\mathbf{d}[i], i \in [n]$, all $j \in [l]$, then the Gaussian membership functions can be computed as:

$$\omega_{i,j} \leftarrow \mathsf{M}_{i,j}(\mathbf{d}[i]) = \exp(-\frac{(\mathbf{d}[i]-\mu_{i,j})^2}{2\sigma_{i,j}^2}). \quad (2)$$

The parameter $\mu_{i,j}$ is the mean and $\sigma_{i,j}$ is the standard deviation. A straightforward method to evaluate Equation 2 over ciphertexts $\{\mathsf{CT}_{\widehat{d}}, \mathsf{CT}_{\widehat{d^*}}\}$ is using a polynomial to simulate the Gaussian function. However, this method will not only compromise accuracy but also place a substantial computational burden on $\mathcal{CS}_B$. Thus, we compute the natural logarithm instead of the Gaussian function (Yeganejou et al. 2023). Then we are relieved from handling exponential computation over ciphertext. In addition, the logarithm offers better numerical stability in the tails. As shown in the Equation 3, the problem is transformed into computing a quadratic polynomial.

$$\log\omega_{i,j} = -\frac{1}{2\sigma_{i,j}^2}(\mathbf{d}[i])^2 - \frac{1}{2}(\frac{\mu_{i,j}}{\sigma_{i,j}})^2 + \frac{\mu_{i,j}}{\sigma_{i,j}}\mathbf{d}[i]. \quad (3)$$

Then $\mathcal{CS}_B$ maps all the constant terms into polynomials. Note that, all the input values are packed and encrypted as one ciphertext. Thus, for any j-th rule, all the n parameters (e.g., $-1/2\sigma_{i,j}^2$, $\mu_{i,j}/\sigma_{i,j}$) should be packed using the same method. For simplicity, we omit it as a default operation. ❶ Then the first term can be computed as $\mathsf{CT}^1_{\cdot,j} \leftarrow \mathsf{CtPtMul}(\mathsf{CT}_{\widehat{d^*}}, -1/2\sigma^2_{\cdot,j})$. ❷ The computation of the second term is trivial as it involves no homomorphic evaluation. ❸ The third term can be computed as $\mathsf{CT}^3_{\cdot,j} \leftarrow \mathsf{CtPtMul}(\mathsf{CT}_{\widehat{d}}, \mu_{\cdot,j}/\sigma_{\cdot,j})$. ❹ Then, $\mathcal{CS}_B$ merge the three terms one by one. It calculates $\mathsf{CT}^{1,2}_{\cdot,j} \leftarrow \mathsf{CtPtAdd}(\mathsf{CT}^1_{\cdot,j}, -\mu^2_{\cdot,j}/2\sigma^2_{\cdot,j})$. ❺ At last, $\mathcal{CS}_B$ obtains a ciphertext of $\log\omega_{\cdot,j}$ by $\mathsf{CT}_{\log\omega_{\cdot,j}} \leftarrow \mathsf{CtCtAdd}(\mathsf{CT}^1_{\cdot,j}, \mathsf{CT}^{1,2}_{\cdot,j})$. For all rules $j \in [l]$, $\mathcal{CS}_B$ repeats the above calculations and passes $\mathsf{CT}_{\log\omega_{\cdot,j}}$ to the rule layer.

Rule Layer

This layer is committed to evaluating the firing strength (Talpur et al. 2023; Bai et al. 2021) for each fuzzy rule. The existing schemes usually compute the product of all antecedent membership functions like $\omega^{\times}_{\cdot,j} = \prod_{i=1}^{n}\omega_{i,j}$. This needs ciphertext-ciphertext multiplication evaluation over the n ciphertext slots within the same packed and encrypted input vector. Unfortunately, it seems out of reach to achieve this using SHE schemes (Viand, Jattke, and Hithnawi 2021). To conquer this problem, PrivDNFIS turns to compute the summation of the logarithm of the memberships. In specific, we have $\omega^{+}_{\cdot,j} = \sum_{i=1}^{n}\log\omega_{i,j}$.

Before introducing our solution, we first show off an interesting observation as follows. Assume that we have two polynomials $\widehat{p}_1 = 1+2x+3x^2+4x^3, \widehat{p}_2 = 1+x+x^2+x^3$. According to the Equation 1, $\widehat{p}_1 \times \widehat{p}_2$ equals: $1+(2+1)x+(3+2+1)x^2+(\mathbf{4}+\mathbf{3}+\mathbf{2}+\mathbf{1})x^3+(4+3+2)x^4+(4+3)x^5+4x^6$. We can find that the coefficient of the fourth term (x^3) equals the summation of all the coefficients of $\widehat{p}_1$. By extension, if we can extract the ciphertext of a certain coefficient over the ciphertext domain, we can compute the sum of the coefficients of any polynomial. In PrivDNFIS, $\mathcal{CS}_B$ generates a polynomial $\widehat{e} = 1+x+x^2+,...,+x^{n-1}$ and then invokes $\mathsf{CT}'_{\cdot,j} \leftarrow \mathsf{CtPtMul}(\mathsf{CT}_{\log\omega_{\cdot,j}}, \widehat{e})$ for all $j \in [l]$. At last, $\mathcal{CS}_B$ extracts LWE ciphertexts of n-th coeffient of $\mathsf{CT}'_{\cdot,j}$. This can be achieved using an existing ciphertext extraction function (Chen et al. 2021). In specific, $\mathcal{CS}_B$ generates LWE ciphertexts for all $j \in [l]$ as $\mathsf{LCT}_{\omega^+_{\cdot,j}} \leftarrow \mathsf{Extract}(\mathsf{CT}'_{\cdot,j}, n)$ and $\mathsf{LCT}_{\omega^+_{\cdot,j}}$ are then feed to the hidden layer.

Hidden Layer

In the hidden layer, activation is computed by taking a linear combination of input values $\mathbf{d}$ and multiplying it by the normalized firing strength of the corresponding rule. To simplify this, it is adjusted to use the sum of the logarithm of the firing strength and the linear combination of inputs, without applying the logarithm to the inputs themselves. This approach creates fuzzy regions (clusters) while keeping the deep convolutional backend fully trainable, avoiding gradient explosion issues. Let the $n \times l$ matrix $\mathbf{M}$ represent activation weights, and $\mathbf{b}$ be the bias vector for all rules. The activation for the j-th rule is then computed as follows.

$$\rho_j = \omega^{+}_{\cdot,j} + (\sum_{i=1}^{n}\mathbf{M}[i][j]\mathbf{d}[i] + \mathbf{b}[j]). \quad (4)$$

The task of $\mathcal{CS}_B$ is to evaluate the above equation privately. Intuitively, the main challenge of computing Equation 4 is designing a privacy-preserving vector inner protocol for $\sum_{i=1}^{n}\mathbf{M}[i][j]\cdot\mathbf{d}[i]$. Specifically, the problem is computing the ciphertext-plaintext vector inner product in a privacy-preserving manner. A widely applied method is packing vectors using CRT based scheme (Smart and Vercauteren 2014), and then evaluating the element-wise multiplication by simply invoking $\mathsf{CtPtMul}(\cdot)$. For instance, given two vectors $\mathbf{v}_1 = [a_1, a_2, a_3, a_4], \mathbf{v}_2 = [b_1, b_2, b_3, b_4]$, one can obtain the ciphertext of the vector $\mathbf{v}'_0 = [a_1b_1, a_2b_2, a_3b_3, a_4b_4]$ using the above method. Afterward, one needs to rotate the ciphertext of $\mathbf{v}'$ for one slot to acquire the ciphertext of vector $\mathbf{v}'_1 = [a_2b_2, a_3b_3, a_4b_4, a_1b_1]$. By evaluating the homomorphic addition of $\mathbf{v}'_0$ and $\mathbf{v}'_1$, we can have the ciphertext of vector $\mathbf{v}'_2 = [a_1b_1+a_2b_2, a_2b_2+a_3b_3, a_3b_3+a_4b_4, a_4b_4+a_1b_1]$. Then it rotates the vector $\mathbf{v}'_2$ for two ciphertext slots and obtain a ciphertext of vector $\mathbf{v}'_3 = [a_3b_3+a_4b_4, a_4b_4+a_1b_1, a_1b_1+a_2b_2, a_2b_2+a_3b_3]$. Lastly, one-time addition homomorphic evaluation on ciphertexts of $\mathbf{v}'_2, \mathbf{v}'_3$ will outputs a ciphertext of vector $\mathbf{v}'_4 = [a_1b_1+a_2b_2+a_3b_3+a_4b_4, ...]$. Apparently, the first element of $\mathbf{v}'_4$ is the inner product of $\mathbf{v}_1$ and $\mathbf{v}_2$. This method introduces prohibitive computational costs. The ciphertext rotation operation is extremely time-consuming, costing roughly $30\times$ more than modular exponentiation (Huang et al. 2022; Ren et al. 2024).

PrivDNFIS proposes to use a novel vector encoding method to compute the inner product without rotation. Here we give a toy example. Given two integer vectors $\mathbf{v}_1 = [a_1, a_2, a_3, a_4], \mathbf{v}_2 = [b_1, b_2, b_3, b_4]$, the $\mathbf{v}_1$ is encoded in *reverse order* onto the coefficients of a polynomial $\widehat{v}_1 = a_4+a_3x+a_2x^2+a_1x^3$, and $\mathbf{v}_2$ is encoded in order onto the

coefficients of a polynomial $\widehat{v}_2 = b_1 + b_2x + b_3x^2 + b_4x^3$. $\widehat{v}_1 \times \widehat{v}_2$ is computed as follows.

$$\begin{aligned}\widehat{v}_1 \times \widehat{v}_2 = a_4b_1 &+ (a_4b_2 + a_3b_1)x \\ &+ (a_4b_3 + a_3b_2 + a_2b_1)x^2 \\ &+ (a_4b_4 + a_3b_3 + a_2b_2 + a_1b_1)x^3 \\ &+ (a_3b_4 + a_2b_3 + a_1b_2)x^4 \\ &+ (a_2b_4 + a_1b_3)x^5 + a_1b_4x^6.\end{aligned} \tag{5}$$

The coefficient of the term (marked blue) equals the inner product. By leveraging this finding and the coefficient extraction function $\mathsf{Extract}(\cdot)$, $\mathcal{CS}_B$ can privately compute the vector inner product efficiently without any decryption or rotation operation (Viand, Jattke, and Hithnawi 2021).

Algorithm 1: Privately compute the hidden layer on $\mathcal{CS}_B$

Input: The LWE ciphertexts $\mathsf{LCT}_{\omega^+_{\cdot,j}}$, the private input values $\mathsf{CT}_{\widehat{d}}$. The weight matrix $\mathbf{M}$ and the bias vector $\mathbf{b}$.
Output: The LWE ciphertexts LCT_{ρ_j} of ρ_j for all $j \in [l]$.
for $j \in [l]$ **do**
  for $i \in [n]$ **do**
    $\mathbf{m}[i] \leftarrow \mathbf{M}[i][j]$.
  end for
  $\widehat{m} \leftarrow \pi(\mathbf{m})$; $\mathsf{CT}_\alpha \leftarrow \mathsf{CtPtMul}(\mathsf{CT}_{\widehat{d}}, \widehat{m})$.
  $\mathsf{LCT}_\alpha \leftarrow \mathsf{Extract}(\mathsf{CT}_\alpha, n)$.
  $\mathsf{LCT}_\beta \leftarrow \mathsf{CtPtAdd}(\mathsf{LCT}_\alpha, \mathbf{b}[j])$.
  $\mathsf{LCT}_{\rho_j} \leftarrow \mathsf{CtCtAdd}(\mathsf{LCT}_\beta, \mathsf{LCT}_{\omega^+_{\cdot,j}})$.
end for
return a set of ciphertexts $\{\mathsf{LCT}_{\rho_j}\}, j \in [l]$.

Let $\pi(\cdot)$ denote the inverse mapping function that takes an integer vector as the input and outputs a polynomial. We give the detailed technical design for private evaluation for the hidden layer (i.e., Equation 4) in Algorithm 1.

Output Layer

The output layer normalizes the computed scores of all the classes into the range of $[0, 1]$. The most used function is softmax (LeCun, Bengio, and Hinton 2015; Yeganejou et al. 2023). Given the score vector $[\rho_1, \rho_2, ..., \rho_l]$, the softmax for each rule $j \in [l]$ is computed as follows:

$$P_j \leftarrow \text{softmax}(\rho_j) = \exp(\rho_j) / \sum_{i=1}^{l} \exp(\rho_i). \tag{6}$$

The evaluation of softmax is challenging as it needs to compute non-linear function exp intensively. Most existing schemes (Mohassel and Zhang 2017; Li et al. 2023; Dong et al. 2023) intend to investigate approximations to softmax, which is very expensive. Thus, in this paper, we propose to use an aggressive approximation using a quadratic function:

$$P_j \leftarrow \text{softmax}(\rho_j) = (\rho_j + c)^2 / \sum_{i=1}^{l} (\rho_i + c)^2. \tag{7}$$

c is a small random constant.

The processing of reciprocals and ciphertext-ciphertext multiplications continues to be time-consuming. PrivDNFIS optimizes the straightforward method as follows. $\mathcal{CS}_B$ first extends $(\rho_j + c)^2$ to a quadratic polynomial $\rho_j^2 + c^2 + 2c\rho_j$. Given LCT_{ρ_j}, for all $j \in [l]$ it conducts the following evaluations. ❶ $\mathsf{LCT}_1 \leftarrow \mathsf{CtCtMul}(\mathsf{LCT}_{\rho_j}, \mathsf{LCT}_{\rho_j})$. ❷ $\mathsf{LCT}_2 \leftarrow \mathsf{CtPtMul}(\mathsf{LCT}_{\rho_j}, 2c)$. ❸ $\mathsf{LCT}^*_j \leftarrow \mathsf{CtCtAdd}(\mathsf{LCT}_1, \mathsf{CtPtAdd}(\mathsf{LCT}_2, c^2))$. Apparently, LCT^*_j is a ciphertext of $\rho_j^2 + c^2 + 2c\rho_j$. ❹ Then $\mathcal{CS}_B$ can repeatedly invokes $O(\log(l))$ times function $\mathsf{CtCtAdd}(\cdot)$ to obtain a ciphertext of $\sum_{i=1}^{l}(\rho_i + c)^2$ denoted as LCT^*_ρ. Intuitively, in the next step, $\mathcal{CS}_B$ should compute the reciprocal of $\sum_{i=1}^{l}(\rho_i + c)^2$ over the ciphertext domain. To reduce the computational cost, PrivDNFIS let $\mathcal{CS}_B$ send the ciphertext back to $\mathcal{MU}$ for decryption. Then the extra cost is merely one LWE ciphertext transmission (roughly 100KB in practice). In exchange, the overall time costs are substantially reduced as k times reciprocal processing is moved to the $\mathcal{MU}$. And it is computed over the plaintext domain. Here, the k is number of the returned ciphertexts $\mathsf{LCT}^*_j, j \in [l]$ and $k < n$.

At last, the service provider $\mathcal{CS}_B$ needs to send the top-k $P_j, j \in [l]$ to user $\mathcal{MU}$. In PrivDNFIS, it returns the top-k $(\rho_j + c)^2$, i.e., their ciphertexts LCT^*_j. For the implementation of homomorphic sorting, we adopt the existing scheme (Hong et al. 2021; Viand, Jattke, and Hithnawi 2021). After receiving the k LWE ciphertexts, $\mathcal{MU}$ simply decrypts them and divides the plaintexts by $\sum_{i=1}^{l}(\rho_i + c)^2$.

Security Analysis

We prove the security of PrivDNFIS formally following the simulation paradigm (Lindell 2017). Specifically, we need to prove that there exists a probabilistic polynomial time (PPT) simulator that interacts with a PPT adversary, and the adversary can not distinguish the view generated by the simulator from the real protocol execution. Any PPT adversary can not distinguish the view produced by the simulator from the real protocol execution. The simulator can observe the input/output of the adversary. According to the theory of provable security (Lindell 2017), if we can construct such a simulator, then PrivDNFIS is semantic secure (i.e., provable secure). PrivDNFIS is secure against the semi-honest PPT $\mathcal{A}$, which is formalized as following theorem.

Theorem 1 (Security of PrivDNFIS)**.** *If the applied RLWE/LWE SHE schemes used in* PrivDNFIS *are semantically secure against the semi-honest PPT adversaries, then the proposed protocol* PrivDNFIS *is secure against the semi-honest PPT $\mathcal{A}$.*

Proof sketch. As the user $\mathcal{MU}$ is fully trusted, the main task to prove Theorem 1 is to prove the security of PrivDNFIS against the semi-honest $\mathcal{A}$ ($\mathcal{CS}_B$). To achieve this, we should construct a simulator Sim_0 that makes the simulated view indistinguishable from the real execution of PrivDNFIS. Due to the space limitation, the concrete hybrid arguments in Sim_0 will be provided in the full version of this paper. The existence of Sim_0 confirms that PrivDNFIS provides semantic security under the given threat model.

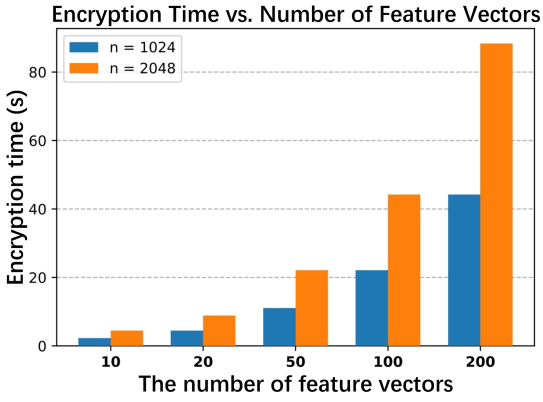

Figure 3: Encryption time cost on $\mathcal{MU}$.

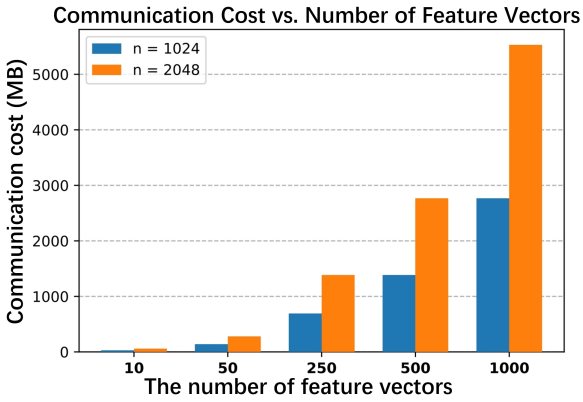

Figure 4: Communication cost on $\mathcal{MU}$.

Performance Evaluation

Implementation settings. Experiments are conducted on a computing machine with Intel (R) Xeon (R) CPU E5-26800 @ 2.70GHz processor with 8 cores, 4 GB RAM storage, and Ubuntu 20.04 operation system. This server is used to simulate $\mathcal{CS}_B$ and we fully utilize the server's computing power. When simulating the end user $\mathcal{MU}$, we use two cores to run the encryption/decryption algorithms. The network delay is set to 60ms (simulate the WAN environment.) and the network bandwidth is set to 2Gbps when simulating the end-to-end time costs. In PrivDNFIS we use the classic RLWE-based HE scheme BFV (Fan and Vercauteren 2012) to encrypt the private input and the homomorphic evaluations are naturally operated over the BFV ciphertexts. The ciphertext extraction function $\mathsf{Extract}(\cdot)$ is implemented by (Chen et al. 2021; Huang et al. 2022), we use their open-sourced code. To implement BFV, the RLWE/LWE FHE library SEAL (Laine, K.; Cruz, R.; Boemer, F.; Angelou, N.; and et al 2015) is applied with the parameters that guarantee 128-bit security. The results presented are the average of 10 trials.

Performance evaluation of $\mathcal{MU}$

$\mathcal{MU}$ first needs to encode and encrypt the private input and generate a pair of RLWE ciphertexts. Compared to encryption, the cost of mapping the plaintext messages onto the coefficients of a polynomial is negligible. As shown in Fig. 3, the cost of encryption increases linearly as the number of input vectors grows. Note that, the length of the input feature vector n and the number of the ciphertext slots N should satisfy $n^2 < N$. Here we set $N = 4096$, then $n \leq 64$. Thus in the real implementation, we need to segment the large feature vector into small vectors with 64 elements. Let T_E be the amortized RLWE SHE encryption time, then the time cost for one feature vector encryption is roughly $2\left\lceil \frac{n}{64} \right\rceil \cdot T_E$. We report the concrete running times for encryption in Fig. 3 by varying the number and length of feature vectors. Specifically, for $n = 2048$, the total time cost for encrypting 100 feature vectors is roughly 44.15s.

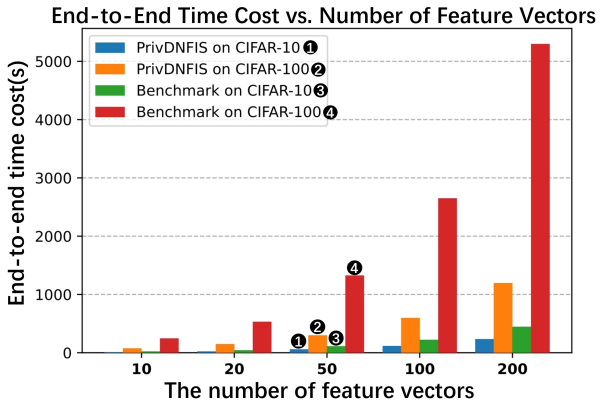

Figure 5: End-to-end running time comparison.

The second part is the communication overhead. Assume that the size of one RLWE SHE ciphertext is S_R, then the total communication overhead for uploading a pair of ciphertexts is $2\left\lceil \frac{n}{64} \right\rceil \cdot S_R$. As shown in Fig. 4, the total communication load increases linearly with the number of issued queries. When the amount of queries reaches 10^3, and $n = 2048$, the total communication cost is around 5.4GB.

Number of ciphertexts (k)	2	5	10	15	20
Decryption time (ms)	19	21	30	45	55

Table 1: Decryption time cost on $\mathcal{MU}$.

The third part is decrypting the LWE ciphertext of $\sum_{i=1}^{l}(\rho_i+c)^2$. k LWE ciphertexts also need to be decrypted. Let T_D be the amortized LWE ciphertext decryption time, then the total decryption time is around $(k+1) \cdot T_D$. We report the concrete running time with different k in Table 1. When examining the total time expenditure, this cost is considered trivial.

Performance evaluation of $\mathcal{CS}_B$

For the fuzzification layer, the main task is evaluating the Equation 3. Let $T_\alpha, T_\beta, T_\gamma, T_\delta, T_\varepsilon$ be the amortized time cost on one-time evaluation of function $\mathsf{CtPtAdd}(\cdot)$, $\mathsf{CtCtAdd}(\cdot)$, $\mathsf{CtPtMul}(\cdot)$, $\mathsf{CtCtMul}(\cdot)$, and $\mathsf{Extract}(\cdot)$; respectively. Given an encrypted feature vector

with n elements, the asymptotic time cost of the *fuzzification layer* is $O(l\left\lceil\frac{n}{64}\right\rceil(T_\alpha+T_\beta+2T_\gamma))$. In the rule layer, the main task is to compute the aggregation of the polynomial coefficients over the received ciphertexts. Specifically, it invokes one time $\mathsf{CtPtMul}(\cdot)$ and $\mathsf{Extract}(\cdot)$ function for each ciphertext received from the previous layer. Note that when $n > 64$, we can aggregate the extracted LWE ciphertext by simply invoking $l\cdot\log(\left\lceil\frac{n}{64}\right\rceil)$ times of function $\mathsf{CtCtAdd}(\cdot)$. In sum, the asymptotic time cost of *rule layer* is $O(l(\left\lceil\frac{n}{64}\right\rceil(T_\gamma+T_\varepsilon)+\log(\left\lceil\frac{n}{64}\right\rceil)T_\beta))$.

For *Hidden layer*, $\mathcal{CS}_B$ needs to evaluate Equation 4. If $n > 64$, for all $\left\lceil\frac{n}{64}\right\rceil$ segmented encrypted feature vectors, $\mathcal{CS}_B$ repeats the same evaluation protocol and conducts addition aggregation over the output of the Algorithm 1. Theoretically, for all l rules, the asymptotic time cost of *hidden layer* is $O(l(T_\alpha+(\log(\left\lceil\frac{n}{64}\right\rceil)+1)T_\beta+\left\lceil\frac{n}{64}\right\rceil(T_\gamma+T_\varepsilon)))$.

In the last layer, $\mathcal{CS}_B$ evaluates the Equation 7. Specifically, to reduce the overall computational cost, $\mathcal{CS}_B$ returns a LWE ciphertext of $\sum_{i=1}^{l}(\rho_i+c)^2$ instead of computing its reciprocal. PrivDNFIS adopts the SOTA homomorphic sorting method (Hong et al. 2021) to obtain the top-k classification scores. Any advanced scheme can be plugged into our scheme. We use T_S to denote its time cost and we omit the technical details for simplicity. In theory, the asymptotic time cost of *output layer* is $O((\log(l)+l)T_\beta+l(T_\alpha+T_\gamma+T_\delta)+T_S)$. We have the total computational complexity Compx as shown in Equation 8, where $n'=\left\lceil\frac{n}{64}\right\rceil$.

$$\begin{aligned}\mathsf{Compx}\cong& O((2+n')l\cdot T_\alpha\\&+(2l+ln'+(l+1)\log n'+\log l)\cdot T_\beta\\&+(4ln'+l)\cdot T_\gamma+l\cdot T_\delta\\&+(l+1)n'\cdot T_\varepsilon+T_S).\end{aligned}\tag{8}$$

Parameter and dataset	Running time (ms)
$n=1024$, Dataset: CIFAR-10	$431.47+T_S$
$n=2048$, Dataset: CIFAR-10	$711.16+T_S$
$n=1024$, Dataset: CIFAR-100	$3850.90+T_S$
$n=2048$, Dataset: CIFAR-100	$7003.87+T_S$

Table 2: Running time on $\mathcal{CS}_B$.

In Table. 2, we report the concrete running time on $\mathcal{CS}_B$. We use T_S to represent the cost of the sorting operation rather than give the concrete cost. When we process classification with a few categories like 10, simply downloading all 10 ranking scores may only introduce a few milliseconds that are negligible. In this case, it is unnecessary to run a secure sorting protocol. For reference, it takes roughly two minutes to find the top five among 100 HE ciphertexts (Hong et al. 2021). However, it only needs a few seconds ($\approx$ 3s) to download the 100 HE ciphertexts. Furthermore, any latest secure sorting protocol can be seamlessly plugged into PrivDNFIS if we have to manage large-scale datasets.

The communication load on $\mathcal{CS}_B$ depends on k chosen by $\mathcal{MU}$. Totally, $\mathcal{CS}_B$ needs to return $k+1$ LWE ciphertexts back to $\mathcal{MU}$. It roughly takes 64ms to download 11 ($k=10$) HE ciphertexts under our network setting that are negligible.

End-to-end time cost

The total running time includes the local query encryption, the server-side evaluation, and the upload & download communication overhead. Fig. 5 shows the total end-to-end time costs by varying the amount of queries. When processing 200 queries on CIFAR-100 (Krizhevsky, A.; Nair, V.; and Hinton, G 2013), PrivDNFIS takes 1194.62s in total. On average, it merely needs roughly 6s to process one query. To demonstrate the high efficiency of PrivDNFIS, we also simulate the performance of a previously proposed scheme CryptFlow (Rathee et al. 2020) as the benchmark that uses rotation to evaluate the vector inner product (Viand, Jattke, and Hithnawi 2021; Rathee et al. 2020). As shown in Fig. 5, compared with the benchmark scheme (Rathee et al. 2020) PrivDNFIS has compressed roughly $1.9\times$, $4.4\times$ running time on CIFAR-10 and CIFAR-100 (Krizhevsky, A.; Nair, V.; and Hinton, G 2013), respectively.

Inference accuracy

In PrivDNFIS, the network architecture of the fuzzy classifier is similar to the latest work (Yeganejou et al. 2023) except for the last layer. Recall that we replace the last layer (i.e., softmax) with a quadratic function for efficiency improvement. Fortunately, it seems that we have a free lunch for such an approximation, and the classification results are the same as the original results. PrivDNFIS only changes the last layer, and *the score ranking is not changed* which leads to the same inference accuracy. For two testing datasets CIFAR-10 and CIFAR-100, the accuracy of the non-private scheme DCNFIS (Yeganejou et al. 2023) and PrivDNFIS are the same. Specifically, they are 93.02% on CIFAR-10 and 74.52% on CIFAR-100, respectively.

In the future, We plan to develop more efficient schemes tailored to DNFIS, drawing on the latest advancements in schemes (Bian et al. 2024; Lu et al. 2025; Zeng et al. 2023).

Conclusion

This paper initiates the investigation into the unexplored territory of PI over DNFIS, introducing PrivDNFIS, a scheme that melds DNFIS network architecture with advanced cryptographic primitive lattice-based SHE. PrivDNFIS offers formal security guarantees and efficiency improvements by investigating several concrete optimizations, addressing several pressing technical challenges. It presents a foundation for future research in the critical domain of PI over DNFIS.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC Grant No. 62402331, and No. 62202320), the Fundamental Research Funds for the Central Universities (Grant No. YJ202429), the Fundamental Research Funds for the Central Universities (No. SCU2024D012), the Science and Engineering Connotation Development Project of Sichuan University (No. 2020SCUNG129), the Natural Science Foundation of Sichuan Province (Grant No. 2024NSFSC1450).

References

Bai, K.; Zhu, X.; Wen, S.; Zhang, R.; and Zhang, W. 2021. Broad learning based dynamic fuzzy inference system with adaptive structure and interpretable fuzzy rules. *IEEE Transactions on Fuzzy Systems*, 30(8): 3270–3283.

Bian, S.; Zhao, Z.; Zhang, Z.; Mao, R.; Suenaga, K.; Jin, Y.; Guan, Z.; and Liu, J. 2024. HEIR: A Unified Representation for Cross-Scheme Compilation of Fully Homomorphic Computation. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. Internet Society.

Chen, H.; Dai, W.; Kim, M.; and Song, Y. 2021. Efficient homomorphic conversion between (ring) LWE ciphertexts. In *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS)*, 460–479. Springer.

Dong, Y.; Lu, W.-j.; Zheng, Y.; Wu, H.; Zhao, D.; Tan, J.; Huang, Z.; Hong, C.; Wei, T.; and Chen, W. 2023. Puma: Secure inference of llama-7b in five minutes. *arXiv preprint arXiv:2307.12533*.

Fan, J.; and Vercauteren, F. 2012. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*.

Gentry, C.; Sahai, A.; and Waters, B. 2013. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Proceedings of the Annual Cryptology Conference (CRYPTO)*, 75–92. Springer.

Hastings, M.; Hemenway, B.; Noble, D.; and Zdancewic, S. 2019. Sok: General purpose compilers for secure multi-party computation. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, 1220–1237. IEEE.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Hong, S.; Kim, S.; Choi, J.; Lee, Y.; and Cheon, J. H. 2021. Efficient sorting of homomorphic encrypted data with k-way sorting network. *IEEE Transactions on Information Forensics and Security*, 16: 4389–4404.

Huang, Z.; Lu, W.-j.; Hong, C.; and Ding, J. 2022. Cheetah: Lean and fast secure {Two-Party} deep neural network inference. In *Proceedings of the USENIX Security Symposium*, 809–826.

Jang, J.-S. 1993. ANFIS: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3): 665–685.

Jang, J.-S. R.; Sun, C.-T.; and Mizutani, E. 1997. Neuro-fuzzy and soft computing-a computational approach to learning and machine intelligence [Book Review]. *IEEE Transactions on Automatic Control*, 42(10): 1482–1484.

Jin, H.; Che, H.; Lin, Y.; and Chen, H. 2024. Promptmrg: Diagnosis-driven prompts for medical report generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 2607–2615.

Krizhevsky, A.; Nair, V.; and Hinton, G . 2013. CIFAR-10 and CIFAR-100 datasets. https://www.cs.toronto.edu/kriz/-cifar.html. Accessed:2024-07-06.

Laine, K.; Cruz, R.; Boemer, F.; Angelou, N.; and et al. 2015. SEAL library. https://github.com/Microsoft/SEAL. Accessed:2024-06-17.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature*, 521(7553): 436–444.

Li, D.; Shao, R.; Wang, H.; Guo, H.; Xing, E. P.; and Zhang, H. 2023. Mpcformer: fast, performant and private transformer inference with mpc. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Lindell, Y. 2017. How to simulate it–a tutorial on the simulation proof technique. *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, 277–346.

Liu, B.; Ding, M.; Shaham, S.; Rahayu, W.; Farokhi, F.; and Lin, Z. 2021. When machine learning meets privacy: A survey and outlook. *ACM Computing Surveys (CSUR)*, 54(2): 1–36.

Lu, C.; Liu, B.; Zhang, Y.; Li, Z.; Zhang, F.; Duan, H.; Liu, Y.; Chen, J. Q.; Liang, J.; Zhang, Z.; et al. 2021. From WHOIS to WHOWAS: A Large-Scale Measurement Study of Domain Registration Privacy under the GDPR. In *In Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

Lu, W.; Huang, Z.; Gu, Z.; Li, J.; Liu, J.; Hong, C.; Ren, K.; Wei, T.; and Chen, W. 2025. BumbleBee: Secure Two-party Inference Framework for Large Transformers. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. Internet Society.

Mohassel, P.; and Zhang, Y. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *Proceedings of the IEEE symposium on security and privacy (SP)*, 19–38. IEEE.

Mughees, M. H.; and Ren, L. 2023. Vectorized batch private information retrieval. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, 437–452. IEEE.

Rathee, D.; Rathee, M.; Kumar, N.; Chandran, N.; Gupta, D.; Rastogi, A.; and Sharma, R. 2020. Cryptflow2: Practical 2-party secure inference. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 325–342.

Ren, H.; Xu, G.; Zhang, T.; Ning, J.; Huang, X.; Li, H.; and Lu, R. 2024. Efficiency Boosting of Secure Cross-Platform Recommender Systems Over Sparse Data. *IEEE Transactions on Dependable and Secure Computing*. doi: 10.1109/TDSC.2024.3478786.

Smart, N. P.; and Vercauteren, F. 2014. Fully homomorphic SIMD operations. *Designs, codes and cryptography*, 71: 57–81.

Talpur, N.; Abdulkadir, S. J.; Alhussian, H.; Hasan, M. H.; Aziz, N.; and Bamhdi, A. 2023. Deep Neuro-Fuzzy System application trends, challenges, and future perspectives: A systematic survey. *Artificial intelligence review*, 56(2): 865–913.

Viand, A.; Jattke, P.; and Hithnawi, A. 2021. SoK: Fully homomorphic encryption compilers. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, 1092–1108. IEEE.

Yeganejou, M.; Dick, S.; and Miller, J. 2019. Interpretable deep convolutional fuzzy classifier. *IEEE Transactions on Fuzzy Systems*, 28(7): 1407–1419.

Yeganejou, M.; Honari, K.; Kluzinski, R.; Dick, S.; Lipsett, M.; and Miller, J. 2023. DCNFIS: Deep Convolutional Neuro-Fuzzy Inference System. *arXiv preprint arXiv:2308.06378*.

Zeng, W.; Li, M.; Yang, H.; jie Lu, W.; Wang, R.; and Huang, R. 2023. CoPriv: network/protocol co-optimization for communication-efficient private inference. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*.

Zhang, W.; Deng, Z.; Zhang, T.; Choi, K.-S.; Wang, J.; and Wang, S. 2021. Incomplete multiple view fuzzy inference system with missing view imputation and cooperative learning. *IEEE Transactions on Fuzzy Systems*, 30(8): 3038–3051.