Journal of *Cybersecurity and Privacy*

MDPI

Article

CryptoKANs+: KAN-Inspired Self-Learning Polynomial Networks for Efficient Privacy-Preserving Machine Learning

Omar Tahmi [1,*], Chamseddine Talhi [1] and Hakima Ould-Slimane [2]

[1] Department of Software and IT Engineering, École de Technologie Supérieure (ETS), Montreal, QC H3C 1K3, Canada
[2] Department of Mathematics and Computer Science, Université du Québec à Trois-Rivières (UQTR), Trois-Rivières, QC G8Z 4M3, Canada
* Correspondence: omar.tahmi.1@ens.etsmtl.ca

Abstract

Processing sensitive data in cloud-based neural networks raises privacy concerns, which Homomorphic Encryption addresses by enabling privacy-preserving machine learning. In our previous work, we introduced CryptoKANs, enabling efficient Kolmogorov–Arnold Network (KAN) inference over encrypted data via polynomial approximation of spline-based activation functions using KAN symbolization. To avoid performance degradation, CryptoKAN required min–max scaling of pre-activation inputs to a small interval—a requirement that could negatively affect training. In addition, a direct theoretical structural comparison with Multi-Layer Perceptron (MLP)-based solutions, such as CryptoNets, was missing. In this work, we address these limitations by presenting CryptoKAN+, a KAN-inspired network integrating self-learned polynomial activations through a Fully Connected Quadratic Transformation (FCQT) layer. By enforcing polynomial activations during training, this design replaces spline functions without post-training symbolization, eliminates the need for interval scaling, absorbs subsequent linear transformations, and reduces multiplicative depth for efficient encrypted inference. Experiments show that CryptoKAN+ achieves competitive accuracy while slightly improving encrypted inference efficiency—a natural consequence of compacting weights with self-learned activations. Overall, this work provides a formal analysis of the structural relationship between KANs and MLPs and demonstrates how enforcing polynomial activations during training enables efficient encrypted inference while preserving accuracy.

Keywords: privacy-preserving machine learning; neural networks; Kolmogorov–Arnold Networks; Homomorphic Encryption; private inference

Check for updates

Academic Editors: Tao Zhang, Xiangyun Tang, Jiacheng Wang, Chuan Zhang and Jiqiang Liu

Received: 22 February 2026
Revised: 28 March 2026
Accepted: 7 April 2026
Published: 6 May 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Neural networks (NNs) have become indispensable for tasks ranging from image rendering and anomaly detection to predictive maintenance and real-time decision-making. Their widespread success across various domains stems from their ability to model intricate patterns and perform high-level computations. However, training and deploying these models demand substantial computational resources and infrastructure [1]. Cloud computing has emerged as a viable solution, providing the necessary infrastructure to implement, train, and deploy NN models efficiently [2,3]. As a result, Machine Learning as a Service (MLaaS) has emerged as a natural extension—enabling scalable, on-demand access to powerful machine learning (ML) models, especially in data-intensive environments [4].

Despite the benefits of cloud-based solutions, the reliance on third-party infrastructures introduces significant privacy concerns. Sensitive Internet of Things (IoT) data—such as medical records, financial transactions, or intrusion detection logs—must be shared with service providers for inference tasks, raising serious security risks [5–7]. For instance, in healthcare, a doctor may need to use a cloud-based model provided by a hospital—such as Company X's cancer prediction service—to analyze a patient's medical data. However, the sensitive nature of this data may prevent direct sharing due to privacy concerns [8,9]. Similarly, in intrusion detection systems (IDS) within IoT environments, encrypted network traffic may require real-time detection without exposing sensitive data to external servers. In smart homes, users may upload sensitive information, such as location data and application usage, to edge servers for predictive services; however, attackers could infer house occupancy from electricity usage patterns and launch targeted attacks [10]. Therefore, such scenarios underscore the need for privacy-preserving techniques that safeguard data confidentiality while enabling accurate and efficient NN inference.

Privacy-preserving machine learning (PPML) has emerged as a pivotal paradigm, enabling machine learning without compromising data privacy [11,12]. It encompasses a variety of privacy-preserving techniques, including Differential Privacy (DP) [13], Federated Learning (FL) [14], Secure Multi-Party Computation (SMPC) [15], and Homomorphic Encryption (HE).

DP techniques protect data by adding noise, which can significantly degrade model accuracy and limit utility. FL enables collaborative training without centralizing raw data; however, it must contend with heterogeneous data distributions, communication overhead, and potentially untrusted participants. Recent work [16] integrates both approaches to analyze how federated training and noise-based perturbation jointly affect detection accuracy and metadata privacy. From a security perspective, the proposed framework does not provide formal Differential Privacy guarantees which limits the ability to rigorously quantify privacy leakage. Moreover, the FL setting remains vulnerable to adversarial participants, such as model poisoning or inference attacks, since no secure aggregation or robust aggregation mechanisms are explicitly enforced. The absence of such defenses may expose the system to reconstruction or inference of sensitive information from shared model updates. Therefore, while the approach reduces direct data exposure, additional cryptographic protections and adversarial robustness mechanisms are required to ensure stronger end-to-end security in untrusted and potentially malicious federated environments.

MPC-based schemes, on the other hand, often incur substantial communication and computational overhead due to their interactive protocols. HE-based solutions, in contrast, allow computations directly on encrypted data without requiring interaction between the user and the server [17]. Although HE can introduce computational overhead and may slightly reduce accuracy, these trade-offs are often more favorable than the significant accuracy loss in DP or the heavy communication burden in SMPC, making it particularly suitable for efficient and secure neural network inference. In this paper, we focus on HE-based PPML approaches. Fully Homomorphic Encryption (FHE), in particular, enables arbitrary computations on encrypted data; however, its reliance on the time-consuming bootstrapping operation to reduce noise and ensure correctness renders it impractical for real-time neural network inference. Leveled FHE addresses this limitation by preconfiguring the FHE parameters required for a specific computational depth dictated by the chosen neural network architecture, thus avoiding the need for bootstrapping. In this work, we use word-wise leveled FHE schemes, specifically CKKS [18], which support SIMD operations for improved efficiency but only addition and multiplication, thereby enabling only polynomial evaluations without any non-linear operations. This limitation becomes particularly critical for Multi-Layer Perceptron (MLP)-based neural network models, where

non-linear activation functions (AFs) are essential for capturing complex patterns. AFs cannot be directly applied to encrypted data under such schemes, forcing researchers to approximate them using polynomials compatible with HE constraints. However, these approximations often degrade model accuracy or increase model complexity, highlighting the need for novel approaches that introduce non-linearity while ensuring both privacy and efficiency. Many IoT-based applications impose strict operational constraints that further motivate efficient inference. For instance, use cases such as intrusion detection systems require real-time or near real-time decision-making, where even small delays can lead to severe consequences, including delayed threat mitigation or system compromise. In such settings, latency becomes a critical factor alongside privacy and accuracy. Moreover, cloud-based inference services are often billed based on computational usage and communication overhead, meaning that inefficient protocols can significantly increase operational costs for end users. Consequently, reducing computational complexity and communication costs is highly desirable, provided that it does not significantly degrade model performance. These requirements highlight that, beyond privacy and security, efficiency, both in terms of computation and communication, plays a crucial role in the practical deployment of privacy-preserving machine learning systems in resource-constrained and latency-sensitive environments [19]. These considerations motivate the need for lightweight privacy-preserving solutions that jointly address data confidentiality, computational efficiency, and communication overhead in IoT and cloud-based deployment scenarios.

In this context, the choice of AFs involves a trade-off between expressive power and computational feasibility. Linear activations lack non-linearity and are therefore insufficient to model complex patterns, whereas higher-order polynomials (e.g., cubic or beyond) increase the multiplicative depth, leading to higher computational overhead and greater noise accumulation in leveled HE schemes. Therefore, a quadratic formulation represents a minimal yet effective compromise, as it introduces the simplest form of non-linearity beyond linear functions while maintaining a low polynomial degree. This design choice aligns with practical deployment requirements in latency-sensitive and resource-constrained scenarios, such as IoT-based intrusion detection systems and cloud-assisted inference, where real-time decision-making, limited computational resources, and reduced operational costs are critical, as discussed previously. Consequently, in this work, we restrict the activation to a quadratic form, as it enables a balanced trade-off between accuracy, efficiency, and feasibility under Homomorphic Encryption constraints. This motivates the adoption of quadratic activation functions as a practical design choice for efficient and expressive neural network inference under Homomorphic Encryption, as further detailed in the proposed methodology.

CryptoNets [1] was the pioneering work applying MLP-based NNs to encrypted data for private inference by replacing activation functions with the square function—the simplest non-linear polynomial. However, its classification performance was limited, motivating subsequent works to explore polynomial approximations of activation functions. These approximations often increase model complexity without delivering significant performance gains.

In our previous work, we proposed CryptoKANs, a privacy-preserving machine learning model that leverages the properties of Kolmogorov–Arnold Networks (KANs) [20]. CryptoKANs exploit the spline-based learnable activation functions of KANs and convert them into polynomial forms through polynomial symbolization, enabling seamless integration into Homomorphic Encryption (HE)-based models while avoiding costly HE operations. In contrast, prior attempts at encrypted KAN inference rely on direct approximation techniques that involve computationally expensive homomorphic operations, such as comparisons, as proposed in [21], which incur significant overhead during encrypted in-

ference. By symbolizing spline activations with polynomial approximations, CryptoKANs provide a more efficient alternative.

A key limitation arises during polynomial symbolization: the large input intervals required by spline activations can create discrepancies between the original KAN activations and their polynomial approximations, particularly near the edges. To mitigate this, CryptoKAN applies min–max scaling to restrict pre-activation values to a smaller interval, improving approximation accuracy. While effective, this introduces a dataset-dependent hyperparameter, and improper interval selection can degrade performance and destabilize training. These requirements highlight a structural limitation of CryptoKAN. Although CryptoKANs achieve accuracy comparable to the original KAN and only a 2× overhead relative to the best MLP-based encrypted solutions, they still lag behind MLP-based models in terms of accuracy and encrypted inference speed. This motivates a formal comparison between KAN-inspired and MLP architectures under both plaintext and encrypted settings, guiding the development of CryptoKAN+, which enforces polynomial activations directly during training to avoid symbolization while preserving accuracy and efficiency.

In this paper, we introduce CryptoKAN+, a KAN-inspired network with fully polynomial activation functions. While the original KAN authors [20] mentioned that KANs are combinations of splines and MLPs, they did not provide a formal structural analysis to clarify the nature of this combination. This leaves open questions about whether KAN represents a genuinely distinct architecture, with activations on edges rather than nodes, or is effectively an MLP with spline-based activations. To address this, we provide a formal analysis of the structural relationship between polynomial KANs and polynomial MLPs, highlighting key architectural differences. We then evaluate the polynomial KAN homomorphically and compare it with CryptoNets, an MLP-based encrypted model.

Through extensive experiments on MNIST and Fashion-MNIST, we demonstrate that CryptoKAN+ achieves competitive accuracy while improving encrypted inference efficiency, maintaining a balance between accuracy and runtime. This establishes CryptoKAN+ as an effective alternative for privacy-preserving applications where both efficiency and accuracy are critical.

1.1. Threat Model

We consider a client–server architecture where a powerful, cloud-based server provides classification services. The client, who holds sensitive data, aims to benefit from the server's predictive capabilities without compromising data confidentiality. The main challenge is to mitigate privacy risks arising from exposing client data to a semi-honest (or honest-but-curious) server. This means the server follows the protocol correctly but attempts to learn as much information as possible from the data it processes. Therefore, the client must perform private inference without disclosing its data to any external entity, ensuring that the prediction process remains confidential and protected against unauthorized access or data leakage—a concept known as privacy-preserving machine learning inference (PPML).

The process proceeds as follows: the client (e.g., a doctor) encrypts the patient's data before sending it to a third-party service provider (e.g., a hospital) for classification. The server then processes the encrypted data and returns encrypted predictions, which the client can decrypt. Throughout this process, the service provider has no access to either the patient's input data or the resulting predictions.

1.2. Contributions

The main goal of this study is to formally analyze the structural relationship between KANs and MLPs when both are constrained to use polynomial activation functions and, secondly, to investigate the differences in their formulations in privacy-preserving settings. Specifically, this study makes the following contributions:

- We provide a formal analysis of the structural differences between polynomial KANs and polynomial MLPs, clarifying the nature of combination of splines and MLPs components in the KAN architecture.
- We introduce a KAN-inspired PPML model, CryptoKAN+, which enforces polynomial activation functions through a Fully Connected Quadratic Transformation (FCQT) layer. This eliminates the need for post-training polynomial symbolization and pre-activation scaling, while maintaining sufficient non-linearity for effective learning under homomorphic encryption.
- We empirically evaluate CryptoKAN+ in encrypted inference scenarios, comparing it with polynomial MLPs (CryptoNets) on the MNIST and Fashion-MNIST datasets. Our results show that CryptoKAN+ achieves competitive accuracy while reducing encrypted inference latency, demonstrating its efficiency and practical suitability for privacy-preserving applications.

2. Related Work

HE enables NNs to process encrypted inputs and generate encrypted predictions without compromising data privacy—a concept referred to as PPML for the task of private or secure inference. However, implementing HE-based NN models presents several challenges, primarily due to the incompatibility of standard AFs—such as ReLU and Sigmoid—with word-wise homomorphically encrypted data. Since these non-linear functions cannot be directly applied in the HE-encrypted domain, they must be adapted to enable private inference. This typically involves replacing non-linear operations with HE-compatible approximations. The core challenge in constructing HE-based NNs, therefore, lies in the design of efficient HE implementations of NN components.

CryptoNets [1] pioneered the application of NNs to encrypted data for private inference, where the AF was replaced with a square function, the lowest-degree non-linear polynomial. However, its performance was limited, mainly due to the use of the square AF. We believe that the absence of a linear component in the square function hindered its ability to capture complex patterns, leading to performance degradation as the network depth increased. Chabanne et al. [22] proposed an approximation of the ReLU using polynomials and introduced batch normalization (BN) to ensure stable and normalized input distributions. However, BN introduces several drawbacks, such as discrepancies between training and testing. This can lead to significant performance dropouts, especially when batches are small or contain correlated samples [23]. Hesamifard et al. [24] proposed CryptoDL, in which they approximated the Sigmoid and ReLU AFs using Chebyshev polynomials, Taylor series, and a derivative-based method for ReLU over a symmetric interval. Bourse et al. [25] proposed the discretized neural network (DiNN) approach for inference over encrypted data. They chose the sign function as the AF, which was implemented using the bit-wise HE scheme Fast Fully Homomorphic Encryption over the Torus (TFHE). Faster-CryptoNets [26] uses a quantized quadratic polynomial to approximate these AFs with varying degrees of complexity and expressivity. Kim et al. [27] proposed secure logistic regression, which uses a least squares approach to find global polynomial approximations over the interval $[-8, 8]$. Addressing the same problem, Chen et al. [28] use minimax polynomial approximations over the interval $[-5, 5]$ with polynomial degrees 1 and 3. However, linear polynomials lack the non-linearity property essential for NNs.

Brutzkus et al. [29] proposed Low-Latency (Lo-La) CryptoNets to improve both latency and memory usage. Lo-La encrypts entire layers as a single message and employs different matrix–vector multiplication implementations throughout the inference process. These technical modifications enabled a significant improvement in evaluation speed compared to CryptoNets. Khan et al. [30] proposed Learning-in-the-Dark, in which they used Chebyshev polynomials to approximate AFs. Despite various approximation methods proposed to address the incompatibility of AFs with HE schemes, these approaches often rely on manually selecting and testing polynomial approximations generated by different techniques, which is labor-intensive. Moreover, polynomial approximations are typically limited to a fixed interval that encompasses all training data points and often introduces oscillations, failing to generalize well to unseen data points outside the interval, thus degrading inference performance. More critically, these methods lack guarantees of optimal performance due to their static nature, as the best-approximated polynomial may vary across datasets, necessitating additional fine-tuning to accommodate complex data.

Adaptive activation functions (AAFs) have been widely explored to enhance the expressive power of neural networks (NNs) by learning the shape of activation functions during training. Prior works have demonstrated that AAFs can reduce model bias and improve performance in both classification and regression tasks. However, designing AAFs involves a trade-off between approximation capability and model complexity, often requiring careful selection of sub-functions or parameterizations [31]. Hou et al. [31] proposed a smooth adaptive activation function (SAAF) with a piece-wise polynomial form, capable of approximating any continuous function to an arbitrary degree of accuracy while controlling model complexity through Lipschitz regularization. The SAAF parameters are trained alongside other NN parameters, enabling dynamic control over activation functions. Their results demonstrate a significant improvement over state-of-the-art regression convolutional neural networks (CNNs). While effective in standard deep learning settings, such piece-wise constructions inherently rely on segmented evaluations and conditional structures. This design is not compatible with word-wise HE schemes such as CKKS, which are known for their high-speed operations as they support Single Instruction Multiple Data (SIMD). These schemes restrict computations to polynomial operations and do not support piece-wise or branching-based function evaluations in the encrypted domain. On the other hand, bit-wise HE schemes such as TFHE support non-linear operations such as ReLU and sigmoid, but at the cost of reduced efficiency, as they do not benefit from SIMD parallelism. Hence, in this work, we focus on the use of a word-wise HE scheme (CKKS) for its superior speed performance. The piece-wise function compositions in [31] are not supported in such schemes. Thus, we adopt a fully polynomial activation formulation, enabling direct evaluation under the CKKS scheme. Overall, while prior AAF approaches primarily target improved approximation in conventional training settings, we explicitly adapt the concept of AAFs to the computational limitations of HE. Hence, the choice of polynomial-only forms is driven by the feasibility of secure inference under word-wise HE schemes.

Goyal et al. [32,33] introduced self-learning activation functions (SLAF), a class of activation functions learned during training using polynomial activation functions (PAFs), in order to train polynomial neural networks (PNNs) [34], inspired by polynomial approximation of univariate functions. In their formulation, the activation function at each neuron is expressed as a weighted sum over a predefined polynomial basis, and the resulting PNN can be fully represented as a polynomial over the input features. The activation function does not enforce a fixed polynomial degree, but instead allows an upper bound on the degree, thereby providing flexibility. Consequently, the activation can adapt to any degree less than or equal to the user-specified bound. To control the number of learnable parameters and mitigate overfitting associated with high-degree polynomials, SLAF stacks

neuronal weights with the coefficients of the polynomial activation and shares them across features within each hidden layer, leading to a compact global polynomial representation. They show that this leads to a reparameterized polynomial representation in Theorem 2 of [32], where the network output can be expressed as a linear function over a polynomial basis of the input.

This formulation is similar in spirit to our proposed framework. CryptoKAN+ serves as the quadratic counterpart aligned with KAN-style architectures. It follows a design closer to SLAF [32,33] in that coefficients of the polynomial activation and network weights are stacked; however, unlike SLAF, we do not share these parameters across features. Specifically, each feature is assigned independent quadratic activation parameters. Although this design increases the number of parameters compared to SLAF, it maintains a controlled parameter budget due to the fixed quadratic form and stacked weights/polynomial coefficients, while preserving feature-wise flexibility in the activation behavior. Our approach adopts a feature-wise parameterization that is more aligned with KAN-style architectures, enabling distinct activations per neuron and enhancing flexibility, while remaining computationally tractable due to the quadratic constraint.

Overall, SLAF [32,33] and CryptoKAN+ can be interpreted as reparameterized polynomial representations in the sense of Theorem 2 of Goyal et al. [32], where the network output is expressed as a linear function over a polynomial basis of the input. However, the difference in parameterization between them leads to distinct model expressivity and training dynamics, as the parameter spaces differ, even though both approaches can be viewed within the broader framework of polynomial activation functions.

Liu et al. [20] proposed Kolmogorov–Arnold Networks (KANs), which introduce a paradigm shift in NN design by replacing fixed activation functions with learnable univariate functions defined on the edges, parameterized using splines. This allows KANs to jointly model the compositional structure of NNs and the underlying univariate transformations, providing increased expressiveness and interpretability compared to standard MLPs, as they claim. In this work, we build upon this perspective and investigate the extent to which KAN architectures can be formally related to MLPs. Specifically, we aim to identify an equivalent or best-matching MLP formulation that captures the functional behavior of KANs, and to analyze the fundamental differences between the two architectures in terms of parameterization and expressivity induced by the choice of activation functions. To isolate this effect, we reformulate both MLP and KAN models within the framework of PNNs, where activation functions are restricted to PAFs. This enables a controlled comparison of the two architectures in terms of parameterization and expressivity under a unified polynomial setting. Furthermore, this formulation allows us to adapt the KAN-inspired design to a HE-compatible framework, where only polynomial operations are supported, similarly to early MLP-based privacy-preserving machine learning approaches such as CryptoNets [1].

Pulido et al. [35] introduced SLAFs for encrypted data by leveraging PAFs, allowing dynamic adaptation during training to build PNNs that can be seamlessly converted into encrypted form using Homomorphic Encryption. The trainable coefficients of PAFs are updated jointly with the synaptic weights at each neuron, independently, allowing the network to learn an appropriate mapping from input to output by simultaneously adjusting both the NN weights and the polynomial coefficients at each neuron. This enables the network to adapt activations to the specific problem, network structure, parameters, and dataset.

However, unlike our work and the SLAF formulation [32], the approach of Pulido et al. [35] does not stack activation coefficients with network weights. Additionally, our method further restricts the polynomial to quadratic forms. We also construct a variant similar to

Pulido et al. [35], called MLP-QL, as a baseline to compare how our work deviates from theirs. However, MLP-QL restricts the polynomial to quadratic forms in order to reduce the multiplicative depth introduced by separating weights and polynomial activation coefficients.

While prior works have explored polynomial activation functions to enable efficient homomorphic inference, the design of lightweight, efficient, and trainable polynomial activations remains largely underexplored. This aspect is particularly critical in resource-constrained environments, such as IoT systems, where both computational and memory efficiency are fundamental constraints.

Although our primary focus is on leveraging cloud-based inference for resource-constrained clients, minimizing computational and communication overhead remains essential. Uncontrolled or excessive resource usage can incur significant costs without necessarily yielding proportional performance improvements. Therefore, achieving a principled balance between model accuracy and resource consumption is paramount for practical and scalable privacy-preserving deployments. In such scenarios, the objective is to minimize overhead without compromising throughput or predictive performance.

Thus, in this context, our approach can be viewed as a formulation positioned between the SLAF framework of Goyal et al. [32] and that of Pulido et al. [35], while simultaneously aiming to better align with the KAN formulation of Liu et al. [20]. Our design also enables computationally efficient and lightweight privacy-preserving computation through the use of quadratic activations, which are specifically chosen to control multiplicative depth. This is a critical factor in Homomorphic Encryption, as it directly impacts both latency and noise growth during encrypted inference. Consequently, this restriction makes the proposed approach particularly well-suited for resource-constrained deployment scenarios, where efficient inference and reduced computational overhead are essential.

Several works have addressed private inference using Secure Multi-Party Computation (SMPC), where two or more non-colluding parties collaboratively perform inference through secret sharing. Nevertheless, MPC-based neural network inference schemes still suffer from substantial communication overhead. Hybrid solutions combining Homomorphic Encryption (HE) with MPC, such as Swift [36], have been proposed to balance computational and communication costs. However, these approaches are fundamentally ill-suited for offline private inference in single-server settings, where the assumption of multiple non-colluding parties cannot be enforced. They inherently require at least two rounds of interaction—either with the client (online) or with another server (non-collusion requirement)—in addition to the computational cost of FHE. These limitations render such hybrid schemes impractical for scenarios demanding non-interactive private inference. For this specific use case, HE-based solutions have received considerable attention and witnessed significant advancements. Nevertheless, challenges persist in reducing computational overhead and designing PPML models that are both efficient and practical in terms of accuracy and runtime. It is therefore essential to explore novel approaches that improve performance while minimizing running time.

Using the KAN architecture, prior work [21] performs homomorphic evaluation of KAN inference by approximating the first component of KAN activations—the SiLU function—using weighted least squares, while dynamically determining the approximation interval via Chebyshev's inequality. For the spline-based part of KAN activations, they propose an approximation using homomorphic comparison operations based on minimax composite polynomials. However, homomorphic comparison operations in arithmetic FHE schemes reduce accuracy and increase computational cost. Despite optimizations such as repeated packing for parallel computation, evaluating B-splines recursively over each interval remains time-consuming, motivating more efficient solutions.

To address this, our previous work introduced CryptoKANs [37], a straightforward approach for approximating KAN spline activations using polynomial symbolization. This yields affine transformations of the polynomial-symbolized functions, which can be directly integrated into encrypted contexts using simple HE operations. By symbolizing spline activations with polynomial approximations, CryptoKANs provide a more efficient alternative to [21] in terms of accuracy and inference time.

A key limitation arises during polynomial symbolization: the large input intervals required by spline activations can create discrepancies between the original KAN activations and their polynomial approximations. CryptoKAN mitigates this by using simpler splines and min–max scaling of pre-activation values to a smaller interval to improve approximation accuracy. However, the scaling interval becomes a new, dataset-dependent hyperparameter, and improper interval selection can degrade performance and destabilize training. These requirements highlight a structural limitation of CryptoKAN.

Although experimental results show that CryptoKANs achieve competitive performance in encrypted inference, they still lag behind CryptoNets in terms of accuracy and runtime. This motivates a formal comparison between KAN-inspired and MLP architectures under both plaintext and encrypted settings, guiding the development of CryptoKAN+, which enforces polynomial activations directly during training. This approach eliminates post-training symbolization and pre-activation scaling, providing more flexibility during training and preserving both accuracy and efficiency.

Specifically, CryptoKAN+ leverages a quadratic activation function applied via a Fully Connected Quadratic Transformation (FCQT) layer to enhance training while ensuring compatibility with HE. This approach removes the need for scaling hyperparameters as in CryptoKANs [37] and avoids complex HE operations with high overhead, as in [21], thereby improving encrypted inference time while maintaining competitive accuracy.

Table 1 summarizes the state-of-the-art PPML techniques.

Table 1. Summary of state-of-the-art privacy-preserving machine learning techniques.

Year	Model	Function Approximated	Method	Degree
MLP-based Models				
2016	CryptoNets [1]	-	Square function	2
2017	Chabanne-NN [22]	ReLU	Taylor series, Least-squares + BN	2–6
2018	CryptoDL [24]	ReLU, Sigmoid, and Tanh	Taylor, Chebyshev, Derivative of ReLU	$\geq$2
2018	DiNN [25]	Sign	TFHE + Discretizing	-
2018	Faster-CryptoNets [26]	ReLU, Swish, Softplus	Minimax and Quantization	2
2018	Kim et al. [38]	Sigmoid	Least-squares	3–7
2018	Chen et al. [28]	Sigmoid	Minimax (Remez algorithm)	1–3
2019	Lo-La [29]	-	Square function with modified representation	2
2023	Learning-in-the-Dark [30]	ReLU	Chebyshev approximation	$\geq$3
2023	SLAFs [31,32,35]	-	Self-learning polynomials	-
KAN-based Models				
2024	Efficient P-P-KAN [21]	SiLU, B-spline basis	Weighted least squares + Chebyshev inequality, HE comparisons	-
2024	CryptoKANs [37]	-	Polynomial-symbolization	2
KAN(MLP)-based Model				
2025	CryptoKANs+	-	Quadratic AF as FC layer (FCQT)	2

3. Overview and Background Information

3.1. Privacy-Preserving Kolmogorov–Arnold Networks

KANs [20] were recently introduced as a promising alternative to MLPs, inspired by the Kolmogorov–Arnold representation theorem [39,40]. The authors of KAN claim that, unlike MLPs which use fixed AFs at the nodes (neurons), KANs employ learnable AFs on the edges (weights). As a result, each weight parameter is replaced by a learnable univariate function, parameterized as a spline. Their motivation for outperforming MLPs stems from the idea that, to accurately learn a function, a model must capture both the compositional structure (external degrees of freedom) and the univariate functions themselves (internal degrees of freedom). According to the authors, KANs achieve this by combining the compositional structure of MLPs (on the outside) with the expressiveness of splines (on the inside). Thus, KANs are said to not only learn features—thanks to their structural similarity to MLPs—but also to optimize those features with greater accuracy—due to their spline-based internal representation. This design, the authors claim, leads to improvements in both accuracy and interpretability over standard MLPs.

Kolmogorov–Arnold representation theorem (see Equation (1)) established that if f is a multivariate continuous function in a bounded domain, then f can be written as a finite composition of continuous functions of a single variable and the binary operation of addition. For a smooth function $f : [0,1]^n \to \mathbb{R}$,

$$f(x) = f(x_1, \ldots, x_n) = \sum_{q=1}^{m} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right) \tag{1}$$

where $\varphi_{q,p} : [0,1] \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$.

This equation implies that the learning task is done once the appropriate univariate functions $\varphi_{q,p}$ and Φ_q (learnable activation functions for the first and second layers, respectively) are identified, forming a KAN with two layers of activations of shape $[n, m, 1]$. By stacking additional layers, a deeper KAN can be constructed [20]. Each KAN activation function is the weighted sum of a basis function b, which can be any traditional activation function such as SiLU or simply deactivated, and a spline function parameterized as a linear combination of B-splines (basis splines), as presented in Equation (2), resulting in a piece-wise polynomial function:

$$\text{spline}(x) = \sum_{m} c_m B_{m,k}(x), \tag{2}$$

where c_m are trainable parameters and $B_{m,k}(x)$ are the k-order B-spline basis functions at the knots.

The resulting form of each KAN activation function LAF can be expressed, by default, by the following equations:

$$LAF(x) = sb \cdot b(x) + sp \cdot \text{spline}(x), \tag{3}$$

$$b(x) = \text{SiLU}(x) = \frac{x}{1 + e^{-x}}. \tag{4}$$

where sb and sp are trainable weights that control the two branches.

In CryptoKANs [37], we leveraged KANs to construct a PPML solution. It exploits the properties of learnable AFs and symbolization in KANs, making them compatible with HE and thereby enabling the construction of an efficient PPML model. Symbolization replaces LAFs with known symbolic functions (e.g., x^2, sin) to make them interpretable. CryptoKANs leverage this property to align KANs with HE requirements by converting each AF from the original form into a purely polynomial-based form using a restricted

symbolization, also called polynomial symbolization, that limits the choice of symbolic functions to only polynomials, resulting in a HE-compatible KAN.

3.2. Leveled Fully Homomorphic Encryption

Leveled FHE allows a predefined number of operations (determined by the HE parameters during the encryption) on encrypted data. We use this variant in our work because the number of operations is known in advance based on the architecture of the NN. The Cheon–Kim–Kim–Song scheme (CKKS) [18], is a specific Leveled HE scheme designed for performing operations on both real and complex numbers. The main idea of this scheme is to consider the noise as part of the message we want to encrypt. To preserve separation, a scaling operation is performed with a scaling factor Δ, which determines the precision of the floating-point numbers. CKKS decryption generates an approximate value of the message with sufficient precision when the scheme is properly parameterized. Generally speaking, a larger Δ means better precision. The scheme operates within the polynomial ring $R_q = \mathbb{Z}_q[x]/f(x)$, where q is the coefficient modulus (which can be thought of as a bounded space for the ciphertext). The polynomial modulus $f(x)$ is typically chosen as $f(x) = x^n + 1$, where n represents the ring dimension (i.e., the degree of the polynomial ring). The available space for homomorphic operations in a ciphertext is determined by the parameter q, which defines the supported multiplicative depth. Each homomorphic operation consumes part of its capacity. It is important to preserve some available space to ensure the correct decryption—exceeding the limit leads to message loss. Further details on encoding, encryption, and core operations are given in Appendix A.

4. Proposed Methodology

CryptoKANs approximate the spline-based learnable activation functions of KANs by converting them into polynomial forms through the polynomial symbolization process. In this approach, KANs are trained using their original spline-based activations, and then post-training symbolization maps each activation function to a polynomial from a predefined set. This enables seamless integration with Homomorphic Encryption operations while avoiding costly non-linear computations. Although CryptoKANs achieve accuracy comparable to the original KAN, they still lag behind MLP-based encrypted solutions in terms of both accuracy and inference efficiency.

Moreover, while KAN authors [20] suggest that KANs are essentially a combination of MLPs and splines, the precise structural relationship between these components remains unclear. To clarify this and to investigate the performance gap, we guide KANs toward fully polynomial forms during training, allowing a direct comparison with MLP-based approaches that also employ polynomial activations from the outset.

From this analysis, we made an intriguing observation: when using the same type of AFs and fully connected transformation layers, a KAN is essentially equivalent to an MLP without a "linear transformation head"—that is, without the initial fully connected linear layer between the input and the activations. Conversely, an MLP can be viewed as a KAN without preprocessing the input through "head activations." Building on this insight, we propose CryptoKAN+, a KAN-inspired PPML model that enforces polynomial activations directly during training through a Fully Connected Quadratic Transformation layer. By doing so, CryptoKAN+ eliminates the need for post-training polynomial symbolization and pre-activation scaling, while maintaining sufficient non-linearity for effective learning for the task of private inference. This design also enables a fair structural comparison with MLPs and demonstrates how KAN-inspired architectures can be efficiently adapted for PPML.

In this section, we present the methodology used to modify CryptoKAN into CryptoKAN+ and demonstrate how this approach can be interpreted as a form of MLP-based PPML solution. This section is organized into three parts: CryptoKAN+ Training Optimization, CryptoKAN+ vs. CryptoNets, and CryptoKAN+ Inference.

4.1. CryptoKAN+ Training Optimization

For a direct comparison with MLP-based privacy-preserving approaches, we propose CryptoKAN+ (Quadratic KAN or KAN-Q), a modified version of CryptoKAN with fully polynomial activation functions. Instead of using the original form in Equation (3), we adopt fully polynomial activations, as shown in Equation (5), enabling natural support for polynomial-based training. Specifically, we replace the SiLU function with the identity function (as used in KANs to encourage linearity) and replace the spline branch with the squared function. This modification introduces a controllable degree of non-linearity, governed by the weighted parameters α and β, allowing the activations to learn the most appropriate combination of linearity and non-linearity through a weighted sum of x and x^2. We also include a bias term to provide additional flexibility, resulting in a quadratic activation function.

Although this represents a simplified version of KAN, we apply the same polynomial formulation to MLPs, placing both architectures on equal footing. This enables a direct and fair comparison between activation-function-independent KAN- and MLP-based models under identical polynomial constraints.

$$LAF(x) = \alpha \cdot x + \beta \cdot x^2 + \gamma \tag{5}$$

where α and β are trainable coefficients (same as sb and sp) and γ is a trainable bias parameter. The dimensions of all trainable parameters will be explained later.

This quadratic form represents a minor modification of the original KAN activation in Equation (3). Its design allows the activation to incorporate both linear and non-linear terms, unlike CryptoNets [1], thereby capturing more expressive features. We argue that relying on the square function alone is insufficient, as it implicitly assumes that the linear term is unnecessary—a condition that does not always hold. By combining both terms, the activation can represent a broader class of polynomial functions. Since polynomials are among the highest-order functions that can be efficiently evaluated under Homomorphic Encryption, adopting a quadratic form represents the minimal degree required to capture sufficient expressive capacity in encrypted contexts, whereas the square function alone covers a more limited subset. We hypothesize that this additional expressivity leads to improved classification performance compared to squared-only activations, a hypothesis that is evaluated through our methodological design and experimental results. To this end, we implement a KAN model using both squared and quadratic activations to demonstrate the enhanced capability of the quadratic form. In contrast to the SLAFs approach, which allows higher-degree polynomial activations and may introduce additional complexity without substantial benefit under HE constraints, our quadratic activation provides a simple yet effective polynomial-based non-linearity for efficient homomorphic inference. The objective is to introduce sufficient non-linearity while maintaining computational efficiency, rather than increasing model complexity at the expense of practical deployability.

The restriction to a quadratic form is motivated by the practical constraints of privacy-preserving deployment scenarios discussed in the introduction. In particular, latency-sensitive and resource-constrained applications such as IoT-based systems and cloud-assisted inference require low computational overhead under Homomorphic Encryption. Higher-degree polynomials would increase multiplicative depth, leading to greater computational cost and noise accumulation, and are therefore less suitable for such settings.

Consequently, limiting the activation to a quadratic degree provides a balanced trade-off between expressivity and efficiency while remaining compatible with leveled HE constraints. More importantly, the objective of this work is to compare KAN and MLP architectures in an efficient manner while enabling privacy-preserving machine learning inference. The structural differences observed under quadratic activations are expected to generalize to higher-degree polynomial settings, while remaining more computationally tractable. To ensure a lightweight architecture during inference and a clearer formal comparison, we therefore restrict our analysis to quadratic forms.

4.2. CryptoKAN+ vs. CryptoNets

In this section, we first show that MLPs can also be equipped with LAFs. We then place both MLP and KAN architectures under identical conditions by using the same activation functions, specifically polynomial activations, and the same fully connected transformation layers. Finally, we analyze how polynomial KANs can be represented within a polynomial MLP framework and illustrate this relationship from multiple perspectives. For clarity and readability, we will henceforth refer to polynomial MLP-based models simply as MLPs and polynomial KAN-based models as KANs.

MLPs also possess LAFs, contrary to the claims made by KAN authors. Dhiman et al. [41] demonstrates that one can interpret MLPs as having LAFs as well; it depends on the definition of what is considered an AF. Consider a two-layer MLP (excluding the input layer) with the input $x \in \mathbb{R}^n$, weight matrices $W_1 \in \mathbb{R}^{h \times n}$ and $W_2 \in \mathbb{R}^{o \times h}$, and biases $b_1 \in \mathbb{R}^h$ and $b_2 \in \mathbb{R}^o$. Here, h denotes the number of hidden units in the hidden layer and o denotes the output dimension. The function of the MLP is expressed as follows:

$$f(x) = W_2\sigma(W_1x + b_1) + b_2$$

where σ denotes the AF.

From this, we can introduce LAFs in three ways. The first way uses internal weights as learnable activation parameters, involving polynomial activations as in SLAFs. Consider σ as a polynomial of degree m with trainable parameters:

$$\text{LAF}(x) = \sigma(x) = a_0 + a_1x + a_2x^2 + \ldots + a_mx^m = \sum_{j=0}^{m} a_jx^j,$$

where $\{a_j\}_{j=0}^m$ are the learnable coefficients. The resulting MLP now has an additional $m + 1$ parameters per layer (excluding the input layer).

The second way uses post-weights as learnable activation parameters, by interpreting the post-weight transformation W_2 and bias b_2 as part of the LAF, in which the MLP's function becomes

$$f(x) = \text{LAF}(W_1x + b_1) \quad \text{where} \quad \text{LAF}(x) = W_2\sigma(x) + b_2.$$

The third way uses pre-weights as learnable activation parameters, in which the MLP's function becomes

$$f(x) = W_2\,\text{LAF}(x) + b_2 \quad \text{where} \quad \text{LAF}(x) = \sigma(W_1x + b_1).$$

In all three cases, the resulting MLP model incorporates LAFs. Hence, KANs can be interpreted as MLPs when using the same AFs, but without the first linear transformation layer. To examine this in detail, we set up a fair comparison with MLPs by using a

single branch in each AF of KAN. To achieve this, we remove the spline component from Equation (3) by setting $sp = 0$ and introduce a bias term b_2. This modifies the AF to

$$\text{LAF}(x) = sb \cdot \text{SiLU}(x) + b_2$$

In the KAN implementation, $sb \in \mathbb{R}^{o \times n}$ and $b_2 \in \mathbb{R}^o$, which aligns with the formulation of an MLP model with learnable post-weights, where $h = n$, $W_1 = I$ (identity matrix), $b_1 = 0$, and $\sigma = \text{SiLU}$. This equivalence shows that a KAN can be expressed as an MLP, if using the same AFs, without a pre-fully connected linear transformation layer. Since polynomial functions are both interpretable and compatible with HE, we unify the KAN and MLP concepts by employing either the square or the quadratic AF. For a direct comparison, we use a single AF layer in each model. For clarity and consistency, we will henceforth refer to polynomial MLP-based models simply as MLPs and polynomial KAN-based models as KANs.

4.2.1. KAN with Square AF (KAN-S)

A KAN with a square AF is expressed as

$$f(x) = sb \cdot x^2 + b_2$$

where sb and b_2 are the learnable parameters, corresponding respectively to β and γ in Equation (5), while α is set to zero, eliminating the linear part. Note that the parameters match those of the post-linear transformation layer in an MLP when the pre-linear transformation is removed.

4.2.2. MLP with Square AF (MLP-S)

With an MLP, the equivalent formulation with a square AF is expressed as

$$f(x) = W_2(W_1 x + b_1)^2 + b_2$$

We can see that the MLP includes an additional linear transformation applied to the inputs in the first fully connected layer. If we disable it to align the complexity scale with KAN-S by setting $h = n$, $W_1 = I$ and $b_1 = \mathbf{0}$, the two formulations become identical, with $sb = W_2$. In turn, the first activation of KAN-S transforms the input into a set of squared feature values, which can be viewed as a preprocessing step—since it is not learnable—followed by a linear transformation with learnable weights, with which a standard MLP begins.

4.2.3. KAN with Quadratic AF (KAN-Q)

From Equations (3) and (5), a KAN with a quadratic AF is formulated as

$$f(x) = sb \cdot x^2 + sp \cdot x + b_2,$$

where $\beta = sb$, $\alpha = sp$, and $\gamma = b_2$. Note that the parameters match those of the post-linear transformation layer in an MLP when the pre-linear transformation is removed. This means that the quadratic AFs are applied in the form of a Fully Connected Quadratic Transformation (FCQT) layer, eliminating the need for an additional transformation. It can be interpreted as a combination of a quadratic AFs with "post-weights as learnable activation parameters." This structure can also be viewed as a new type of fully connected layer, where element-wise operations (x and x^2) are followed by a linear transformation (i.e., a weighted sum), as elaborated in [42]. Consequently, in the KAN-Q algorithm, we compute the sum of two fully connected linear transformations within the same layer

i—one applied to x with bias (FC_{i1}) and the other to x^2 without bias (FC_{i2})—together forming the FCQT layer.

4.2.4. MLP with Quadratic AF (MLP-Q and MLP-QL)

From Equation (5), the MLP with a quadratic AF is expressed as

$$f(x) = W_2\Big(\beta \odot (W_1 x + b_1)^2 + \alpha \odot (W_1 x + b_1) + \gamma\Big) + b_2,$$

where $\alpha, \beta, \gamma \in \mathbb{R}^h$ and $\odot$ represents element-wise multiplication. This formulation is close to the proposed KAN-Q. To align them, we compact the weights, incorporating FCQT layers, by setting $\gamma = 0$, $W_2 \odot \beta = sb$, and $W_2 \odot \alpha = sp$, where the vectors α and β are expanded to match the dimensions of W_2 for element-wise multiplication. If we further disable the first fully connected linear transformation layer—by setting $h = n$, $W_1 = I$ and $b_1 = 0$—to match the complexity scale of KAN-Q, the two formulations become identical. This approach is denoted as MLP-Q. However, for comparison, we also consider an MLP with quadratic AF with internal learnable weights α, β, γ followed by a fully connected linear transformation, i.e., without compacting the weights, thereby changing the space of parameters. This variant is denoted as MLP-QL. Although MLP-Q and MLP-QL are mathematically equivalent, they do not necessarily exhibit the same behavior during training in NNs. This divergence arises from their distinct parameterizations, which can affect gradient dynamics—a phenomenon known as the linear overparameterization effect in deep learning [43].

By using unified AFs and fully connected layers, we demonstrate that KAN is essentially an MLP without the initial fully connected linear transformation layer. In other words, the KAN architecture is fundamentally equivalent to a special case of an MLP—specifically, an MLP without the head linear transformation layer—when identical AFs and linear transformations are applied. This can also be viewed as a modified MLP in which the weights of the first layer are fixed in a special manner to switch the order between the application of linear transformations and AFs, starting with the latter, as illustrated in Figure 1. Figure 1a shows a KAN perceptron with n inputs and a quadratic AF. This structure can be redesigned with MLP and viewed differently. Figure 1b illustrates how a KAN can be viewed as an MLP by first preprocessing the inputs with activations (squared function and identity function) and then applying the linear transformation. Figure 1c,d present the view of "post-weights as learnable activation parameters," which provides the same level of interpretability as in KAN [20] and Goyal [32] works.

From these perspectives, it is clear that, when using the same AFs and fully connected transformations, a KAN can be interpreted as an MLP without the first linear transformation layer, whereas an MLP can be viewed as a KAN that preprocesses inputs with activations. Consequently, KAN is not inherently more learnable than a corresponding MLP; the notion of learnability depends on how AFs are defined. Both architectures have "learnable" AFs if we include the weights as part of the activation, and "non-learnable" AFs if we consider the weights solely as part of the fully connected transformation. Importantly, learnability should not be conflated with the "symbolization" procedure proposed by the KAN authors, which interpolates post-activation values from an already trained model using predefined symbolic functions.

Similarly, KAN does not provide superior interpretability compared to an MLP. Figure 1 illustrates that both architectures can be interpreted equivalently. For example, when applied to a synthetic function-based dataset, KAN recovered the underlying functional components through LAFs and reconstructed the formula via symbolization

and dropout, independent of the learnability property. The same reconstruction process can be applied to MLPs by using comparable AFs (e.g., splines), pruning unimportant connections, and applying symbolization. Under identical AFs (such as polynomials) and parameterized transformations, KAN can therefore be regarded as an MLP without the initial fully connected linear transformation layer. This shifts the question, under a privacy-preserving context, from "Can KAN be more effective than MLP?" to "How can this small change in the MLP structure improve performance?"

To address the above question, our methodology is as follows: we compare KAN and MLP architectures under two types of AFs—the squared function and the quadratic function. First, we examine the effect of the AF itself, investigating how replacing the simple squared AF with a quadratic AF impacts training performance while keeping the architecture unchanged. Next, we study the effect of architectural modifications, specifically by including or removing the initial linear transformation layer. This results in four distinct cases for comparison. For completeness, we also include an additional MLP with a quadratic AF in which the linear transformation weights are separated from the activation weights (MLP-QL) to assess whether adding extra parameterization improves performance. MLP-QL is introduced as a baseline to compare how our work deviates from SLAF [35]. However, MLP-QL restricts the activation to a quadratic form in order to reduce the multiplicative depth introduced by separating weights and polynomial activation coefficients, while ensuring a consistent comparison setting. Overall, this yields five models for evaluation: MLP-S (CryptoNets), MLP-Q, KAN-S, KAN-Q (CryptoKANs+), and MLP-QL (quadratic version of SLAF [35]).

To ensure a fair comparison of computational complexity, we first standardize the number of activation function layers (AF-layers) across all models. Next, we align the models according to their total parameter count. The total number of parameters in a standard MLP-based neural network (MLP-S) is given by

$$P = \sum_{l=1}^{N} [(\text{inputs}_l \times \text{outputs}_l) + \text{outputs}_l],$$

where N is the total number of layers (excluding the input layer) in the network, and inputs_l and outputs_l denote the number of input and output units in layer l, respectively.

For the MLP-Q model, due to the inclusion of an extra fully connected layer (without bias) in each layer, the total parameter count is adjusted as

$$P_{\text{MLP-Q}} = \sum_{l=1}^{N} [2 \times (\text{inputs}_l \times \text{outputs}_l) + \text{outputs}_l].$$

For the MLP-QL model, due to the inclusion of additional learnable parameters in each layer, the total parameter count is adjusted as

$$P_{\text{MLP-QL}} = P_{\text{MLP-Q}} + 3 \times \sum_{l=1}^{N} \text{outputs}_l.$$

KAN-S and KAN-Q follow the same formulation as MLP-S and MLP-Q, with the initial linear transformation removed.

An architectural overview of the five models, each containing two AF-layers (excluding convolutional activation) is presented in Figure 2.

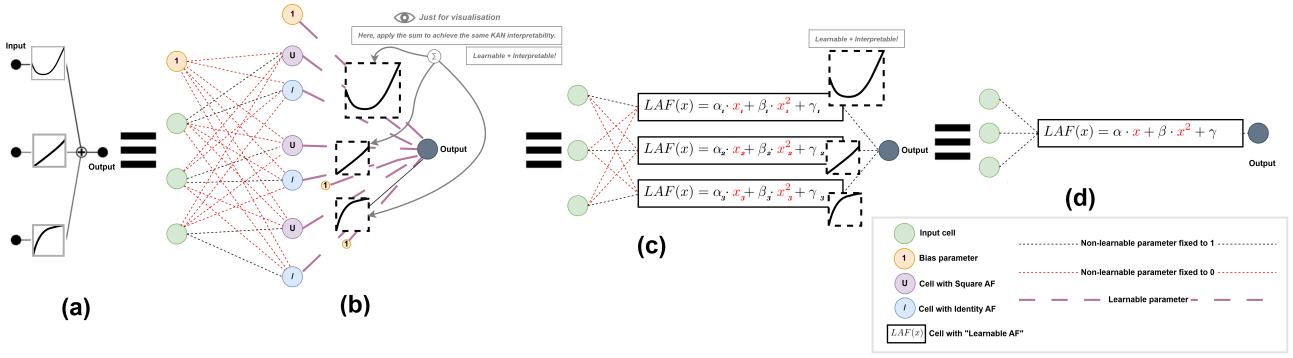


Figure 1. KAN neuron (**a**) expressed using MLP neuron with different points of view (**b**–**d**).

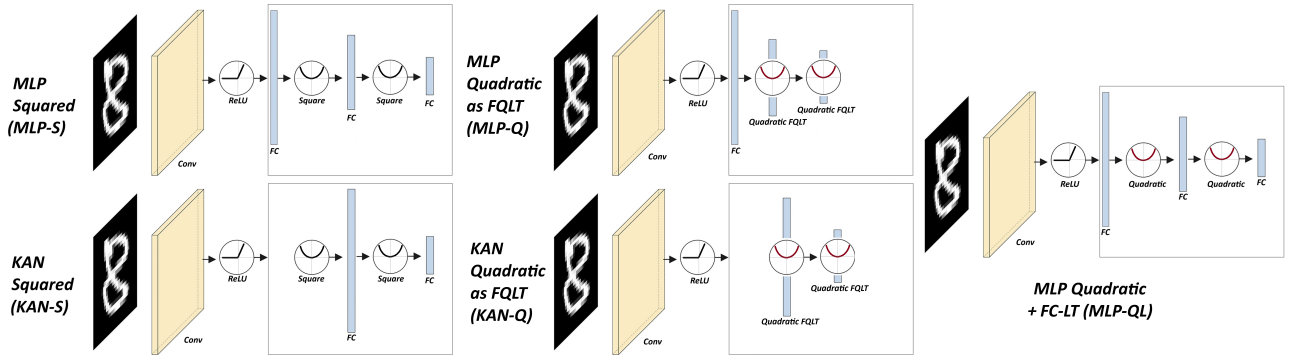


Figure 2. Architectural representations of different equivalent models with two polynomial AF layers.

To build efficient HE-encrypted models for inference, the Homomorphic Encryption parameters must be chosen based on the multiplicative depth of each model. The multiplicative depth of a model can be computed from the depths of its constituent building blocks. Specifically, the fully connected linear transformation layer (FC-LT) and the square activation function each contribute a depth of 1, while the quadratic activation function contributes a depth of 2, as illustrated in Figure 3. Accordingly, Table 2 summarizes the resulting multiplicative depths $d = (L - 1)$ for the different NN models with $m = 2$ and $m = 3$ AF layers. We provide the evaluation steps for the 1-AF-layer version of each model in Algorithm 1. We clearly can see that MLP versions always start with the transformation layer before applying the activation, whereas the KAN version starts with the latter, which can be seen as a preprocessing step. The FCQT layer is incorporated in MLP-Q and KAN-Q variants. We can clearly see through them that each neuron maintains separate weights for the linear component FC_{i1} and squared component FC_{i2} applied without bias (where i is just an index to keep the activation level in parallel between KAN and MLP). Meanwhile, it stacks the weights with the coefficients of the activations in the same layer, thus the name Fully Connected Quadratic Transformation. This corresponds to a reparameterized structure in which network weights are explicitly coupled with activation coefficients, yielding a stacked polynomial representation.

Table 2. Required HE parameters to achieve a minimum of 25 bits of floating-point HE precision and the same security level.

Model	Two AF-Layers		Three AF-Layers	
	L	n	L	n
MLP-S	6	13	8	14
KAN-S	5	13	7	13
MLP-Q	6	13	8	14
KAN-Q	5	13	7	13
MLP-QL	8	14	11	14

Algorithm 1 Evaluation steps for 1-AF-layer architectures

1: **MLP-S:** $\mathrm{FC}_1(x) \rightarrow x^2 \rightarrow \mathrm{FC}_2(x)$
2: **KAN-S:** $x^2 \rightarrow \mathrm{FC}_2(x)$
3: **MLP-Q:** $\mathrm{FC}_1(x) \rightarrow \mathrm{FC}_{21}(x) + \mathrm{FC}_{22}(x^2)$ (FC_{22} is without bias)
4: **KAN-Q:** $\mathrm{FC}_{21}(x) + \mathrm{FC}_{22}(x^2)$ (FC22 is without bias)
5: **MLP-QL:** $\mathrm{FC}_1(x) \rightarrow \beta x^2 + \alpha x + \gamma \rightarrow \mathrm{FC}_2(x)$

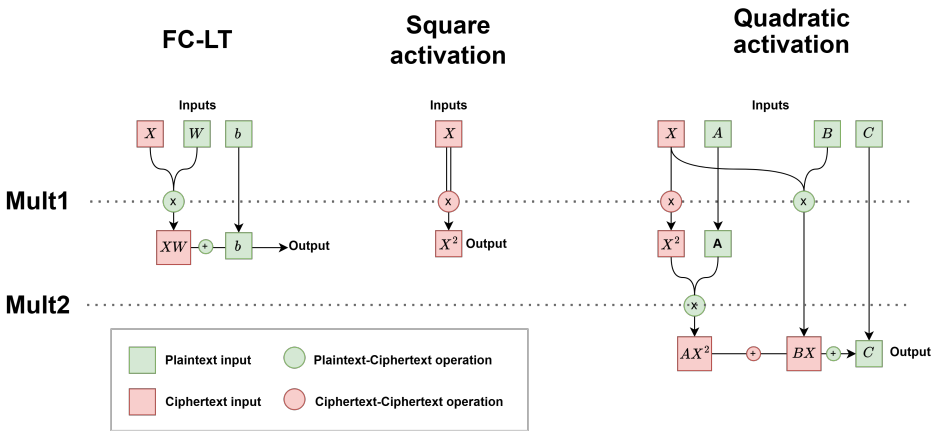


Figure 3. Multiplication depth of NN components.

4.3. CryptoKAN+ Inference

It is worth noting that we do not adopt the KAN symbolization mechanism in our approach for two main reasons. First, symbolization is unnecessary because the AFs are already learned in polynomial form, which is natively supported by HE. Second, and more importantly, symbolization introduces substantial computational overhead during encrypted inference. If symbolization were applied, each learned AF would be approximated by an affine transformation of a symbolic function f, selected from the set $\{x, x^2\}$ to enforce a polynomial form of degree up to 2 to keep the same complexity or less. This transformation is carried out using the `fit_params` method, which identifies the optimal affine parameters $(a_{ij}, b_{ij}, c_{ij}, d_{ij})$ for each input–output pair (i, j), where $i \in [1, d_{\text{in}}]$ represents the index of the input neuron, and $j \in [1, d_{\text{out}}]$ represents the index of the output neuron. Here, d_{in} and d_{out} denote the number of input and output neurons in the layer, respectively. The optimization objective minimized for this transformation is given by

$$\min_{a,b,c,d} \left\| y_{ij} - \left(c_{ij} \cdot f(a_{ij}x + b_{ij}) + d_{ij}\right) \right\|^2$$

The symbolic function f that yields the highest coefficient of determination (R^2) is selected, and the corresponding affine transformation $c_{ij} \cdot f(a_{ij}x + b_{ij}) + d_{ij}$ is used to replace the original LAF. As a result, the symbolization step extends the input vector of dimension d_{in} to $d_{\text{in}} \times d_{\text{out}}$ dimensions before applying the symbolic functions on them,

since each input must first be affinely transformed using different values of a_{ij} and b_{ij}. Consequently, this leads to a significantly larger number of distinct input ciphertexts on which symbolic functions must be evaluated, resulting in computationally expensive encrypted inference. In contrast, our method without symbolization can be evaluated directly over a compact set of encrypted inputs of dimension d_{in} (since in KANs, the process starts with applying the activations). In other words, the batching property that we can leverage from the transition from Figure 1c to Figure 1d would not be possible if we converted to an affine representation. Hence, evaluation without symbolization eliminates substantial computational overhead, leading to more efficient private inference. Therefore, we use native quadratic LAFs without symbolization for better runtime.

5. Experimental Results

PPML private inference tasks hinge on two key factors: classification performance and inference time. In this work, we focus primarily on these metrics, comparing the performance of different models under encrypted settings. Before presenting the results, we first describe the experimental setup.

5.1. Experimental Setup

To conduct the experiments, we utilized two publicly available computer vision datasets: MNIST and Fashion-MNIST. For feature extraction, we incorporated a convolutional layer with four kernels of size 7×7, followed by a ReLU activation function in all models, ensuring direct comparability with SLAFs [35] and Lo-La [29].

The convolutional layer and its activation were excluded from encrypted inference in this study to isolate and evaluate the effects of the fully connected layers and polynomial activations on a flattened input. This exclusion allows a direct assessment of the proposed CryptoKAN+ modifications without confounding factors from convolutional operations.

This setup is known as split inference, a framework that has been leveraged to enable DNN inference on resource-constrained edge devices with the assistance of a server equipped with relatively abundant computational resources [44]. In this setup, a pre-trained DNN is divided into edge-side and server-side networks, which collaboratively perform forward propagation. While this mitigates resource limitations at the edge, it raises privacy concerns if raw intermediate data is shared. Privacy-preserving approaches, such as Homomorphic Encryption, can secure the transferred features. In HE-based split inference, the performing part of the network on the client side reduces the multiplicative depth on the server and thus lowers computational overhead and latency. This is exactly the approach we adopt, where the convolution is executed in plaintext at the client, and the remaining inference is completed in the encrypted domain at the server.

Nevertheless, convolutional operations can also be implemented within KAN-based architectures, as proposed in [45,46], suggesting that future extensions of CryptoKAN+ could incorporate convolutional layers.

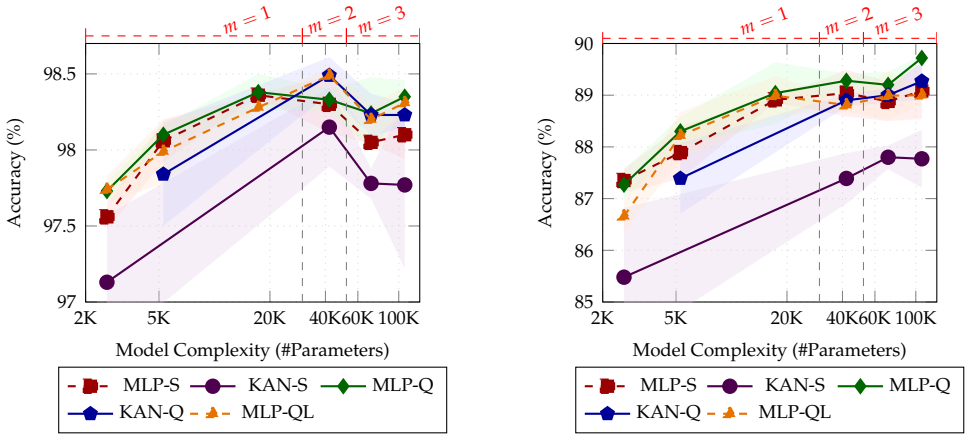
The choice of a computer vision dataset is motivated by its status as a standard benchmark. This exclusion allows a direct assessment of the proposed CryptoKAN+ modifications without confounding factors from convolutional operations. However, it is worth noting that convolutional operations can also be implemented within KAN-based architectures, as proposed in the works [45,46], suggesting that future extensions of CryptoKAN+ could incorporate convolutional layers while maintaining privacy and compatibility with polynomial-based activations.

To ensure fairness, the same weight initialization method with exactly the same values was used across all models. Next, we constructed our models starting with a small architecture (one AF layer) and progressively increased their complexity while maintaining

approximately the same number of learnable parameters across models. For all experiments, we employed the L-BFGS optimizer with an adaptive learning rate starting at 0.5, and applied an early stopping mechanism to prevent overfitting. For the implementation of HE, we used the TenSEAL library [47], which supports the CKKS scheme. All experiments were conducted on a workstation equipped with a 12th Gen Intel® Core™ i9-12900 processor and 32 GB of RAM.

5.2. Classification Performance Analysis

To evaluate the classification performance of the models, we computed four standard metrics: accuracy, precision, recall, and F1-score. Since we balanced the datasets, these metrics produced nearly identical results across all models; therefore, we report only the accuracy results. Each experiment was repeated three to five times with different random seeds to ensure robust evaluation. We then calculated the average accuracy, the standard deviation of the accuracy, and the average generalization error. Figure 4a,b show the evaluation accuracy versus model complexity for different models on the MNIST and Fashion-MNIST datasets, respectively. Tables 3 and 4 provide detailed values of accuracy, standard deviation, and generalization for each model and each network structure.



(**a**) (**b**)

Figure 4. Accuracy vs. model complexity for MNIST (**a**) and Fashion MNIST (**b**). Shaded areas represent the standard deviation around the mean accuracy.

In both datasets, we observe that the MLP-Q model consistently achieves the highest performance, regardless of the network size, surpassing state-of-the-art approaches such as Lo-La [29], which utilizes square AFs, and SLAFs [35], which employ cubic-degree AFs. A minor exception occurs in models with $m = 2$ activation layers on the MNIST dataset, where MLP-QL and KAN-Q achieve higher performance, yet still always employ quadratic activation functions. This observation indicates that simpler AFs, with complexity lower than quadratic, are generally insufficient to capture complex patterns: linear AFs lack non-linearity, while square AFs lack sufficient compositional depth due to the absence of an internal linear component. Conversely, more complex AFs may introduce unnecessary complexity without improving performance and can even contribute to overfitting. In contrast, a quadratic AF applied in the FCQT layer provides an optimal balance between linearity and non-linearity, without additional complexity.

Table 3. Classification performance comparison on MNIST dataset. Bold values indicate the two best performing results among the compared methods for each metric and architecture.

ML Structure	Metric	MLP-S	KAN-S	MLP-Q	KAN-Q	MLP-QL
[256, 9, 10]	Acc	0.9756	0.9713	**0.9773**	-	**0.9774**
	Std	**0.0008**	0.0044	**0.0006**	-	0.0009
	Gen	**0.0055**	**0.0036**	0.0075	-	0.0058
[256, 19, 10]	Acc	**0.9806**	-	**0.9810**	0.9784	0.9799
	Std	**0.0014**	-	**0.0006**	0.0034	0.0021
	Gen	0.0115	-	0.0126	**0.0068**	**0.0111**
[256, 64, 10]	Acc	**0.9836**	-	**0.9838**	-	0.9828
	Std	**0.0001**	-	0.0012	-	**0.0003**
	Gen	0.0163	-	**0.0159**	-	**0.0160**
[256, 128, 64, 10]	Acc	0.9830	0.9815	0.9833	**0.9849**	**0.9840**
	Std	0.0019	0.0026	**0.0007**	0.0012	**0.0006**
	Gen	0.0170	0.0173	0.0167	**0.0149**	**0.0158**
[256, 128, 64, 32, 10]	Acc	0.9805	0.9778	**0.9824**	**0.9823**	0.9820
	Std	**0.0001**	**0.0010**	0.0024	0.0022	**0.0010**
	Gen	0.0192	0.0194	**0.0174**	**0.0174**	**0.0175**
[256, 256, 128, 64, 10]	Acc	0.9810	0.9777	**0.9835**	0.9823	**0.9831**
	Std	0.0016	0.0055	0.0011	**0.0008**	**0.0010**
	Gen	0.0181	0.0195	**0.0162**	0.0179	**0.0165**

Table 4. Classification performance comparison on Fashion MNIST dataset. Bold values indicate the two best performing results among the compared methods for each metric and architecture.

ML Structure	Metric	MLP-S	KAN-S	MLP-Q	KAN-Q	MLP-QL
[256, 9, 10]	Acc	**0.8735**	0.8548	**0.8727**	-	0.8666
	Std	0.0023	0.0134	**0.0029**	-	**0.0029**
	Gen	**0.0193**	**0.01417**	0.0212	-	0.0221
[256, 19, 10]	Acc	0.8789	-	**0.8830**	0.8739	**0.8822**
	Std	0.0051	-	**0.0028**	0.0068	**0.0046**
	Gen	**0.0230**	-	0.0271	**0.0203**	0.0256
[256, 64, 10]	Acc	0.8891	-	**0.8904**	-	**0.8900**
	Std	**0.0005**	-	0.0060	-	**0.0029**
	Gen	**0.0359**	-	0.0376	-	**0.0348**
[256, 128, 64, 10]	Acc	**0.8904**	0.8739	**0.8928**	0.8890	0.8881
	Std	0.0045	0.0049	**0.0014**	**0.0012**	0.0038
	Gen	0.0419	**0.0393**	0.0416	0.0455	**0.0413**
[256, 128, 64, 32, 10]	Acc	0.8888	0.8780	**0.8920**	**0.8900**	0.8899
	Std	0.0038	**0.0023**	0.0025	**0.0014**	0.0047
	Gen	**0.0387**	**0.0379**	0.0447	0.0440	0.0400
[256, 256, 128, 64, 10]	Acc	0.8908	0.8777	**0.8972**	**0.8927**	0.8900
	Std	0.0054	0.0056	**0.0005**	0.0036	**0.0017**
	Gen	**0.0431**	0.0436	0.0440	0.0495	**0.0399**

For smaller networks with $m = 1$ activation layers, the MLP-Q ranks first, followed by MLP-S and MLP-QL. In contrast, KAN models (KAN-S and KAN-Q) perform relatively poorly, indicating that the MLP structure is more efficient than the KAN structure for smaller networks. This suggests that the first FC-LT is crucial for the performance of smaller networks. Omitting this component significantly impacts model performance. As the number of parameters and AF-layers increases ($m > 1$), quadratic models begin to outperform squared models, with the MLP-Q model maintaining its lead. For the MNIST dataset, KAN-Q achieves the highest average accuracy across all models (98.49%) and the best average generalization error of only 0.0149, showing that the model generalizes well at its peak performance. For the Fashion MNIST dataset, MLP-Q achieves the highest average accuracy (89.72%) and continues to improve, while showing the most consistent performance, as indicated by the lowest average standard deviation of accuracy. These results highlight the exceptional performance stability and robustness of quadratic models

compared to the relatively poor classification performance and degradation exhibited by squared models as model complexity increases. Overall, we observe that MLP-Q consistently outperforms all models, regardless of network size. In contrast, the KAN structure demonstrates its efficiency only when paired with quadratic AFs (KAN-Q) in larger networks. This suggests that quadratic AFs applied as FQLT significantly enhance the performance of MLP architecture, preventing the degradation observed with square activations as model complexity increases. They also effectively mitigate the performance drop in KAN models caused by the removal of the initial FC-LT. Consequently, the KAN-Q (CryptoKAN+) approach slightly reduces the model's complexity (multiplicative depth) without compromising performance. In contrast, the MLP-QL model does not necessarily yield performance gains, as observed on the Fashion-MNIST dataset. Ultimately, these observations reaffirm the superior performance of models with quadratic AFs applied as FCQT—even when the initial linear transformation layer is removed—compared to those utilizing squared AFs.

Maintaining the same classification performances when switching to encrypted evaluation was achieved by choosing a scaling factor of $\Delta = 25$ bits of floating-point HE precision, yielding a maximum error of 10^{-4} in accuracy for all models, effectively maintaining the same order in terms of classification performance.

5.3. Inference Time Analysis

Inference time is critical for PPML, especially in latency-sensitive tasks such as intrusion detection. To evaluate this, we constructed five models with two and three AF-layers ($m = 2$ and $m = 3$) and measured their encrypted inference time. Table 2 reports the required FHE circuit length L (corresponding to its multiplicative depth of $m = L - 1$) and the parameter n for each model, ensuring at least 25 bits of FHE floating-point precision while maintaining the same security level across all models.

Figure 5a,b show the encrypted inference time for different models with two and three AF-layers, respectively, as the number of parameters varies. For models with two AF-layers, the MLP-QL model exhibits the worst inference time due to its multiplicative depth of 8, which necessitates a higher polynomial ring dimension ($n = 14$) compared to other models ($n = 13$). This is expected and renders MLP-QL unsuitable for applications requiring near-real-time predictions, especially since it does not provide any additional performance benefits, as shown in our experiments.

For smaller models (fewer AF-layers, $m = 2$, and fewer parameters, less than 5K), MLP models show a slight advantage over KAN models. This is because KAN models apply a higher number of AFs in the input layer, leading to increased computation time due to the larger number of inputs (256). In contrast, MLP models apply AFs only in the hidden layers, which are smaller than the input layer.

As the network size increases, KAN models reverse this trend and demonstrate lower inference times due to their reduced multiplicative depth. This advantage becomes even more pronounced for networks with three AF-layers. In such cases, MLP models must increase their ring dimension to $n = 14$, whereas KAN models can maintain a lower ring dimension. Consequently, KAN models achieve superior encrypted inference times, and their advantage grows as both the number of parameters and AF-layers increase.

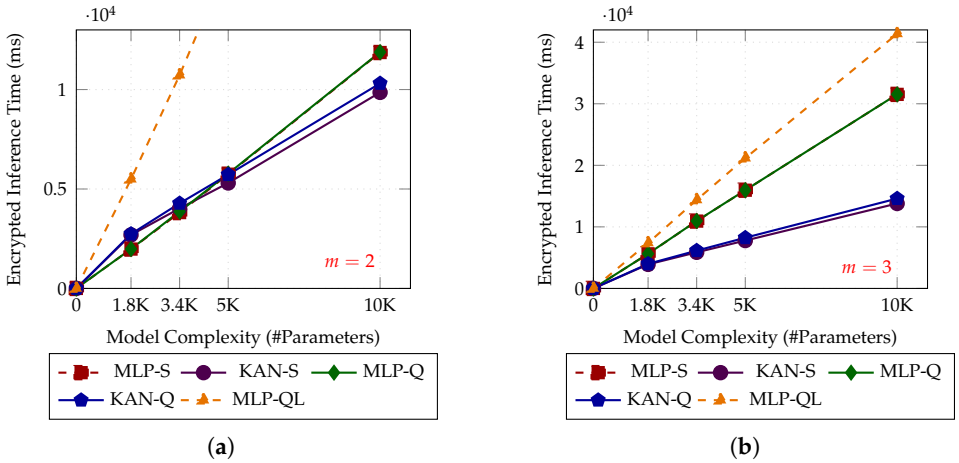


Figure 5. Encrypted inference time vs. model complexity for different architectures. (**a**) Two-AF-layer models ($m = 2$). (**b**) Three-AF-layer models ($m = 3$).

These results indicate that the MLP-Q model consistently achieves the highest accuracy across all configurations while maintaining the second-fastest encrypted inference time. In contrast, the KAN-Q (CryptoKAN+) model delivers the fastest inference times, particularly for larger networks, while ranking second in classification performance. The MLP-S model (CryptoNets) is effective only for small networks, whereas MLP-QL and KAN-S models are largely suboptimal: MLP-QL suffers from excessive inference time, and KAN-S exhibits limited classification performance.

From these observations, we conclude that minimizing the multiplicative depth—even by a single level—without sacrificing accuracy is critical in encrypted contexts. This is realized through a KAN-like architecture combined with quadratic activation functions applied as FCQT layers, which substantially reduces inference time while preserving performance. As a result, the CryptoKAN+ model emerges as a compelling solution for privacy-preserving applications requiring near-real-time predictions on resource-constrained systems, delivering competitive accuracy. Conversely, the MLP-Q model with FCQT layers is better suited for applications that prioritize detection precision, underscoring the advantage of quadratic activation functions in this configuration. In Table 5, we summarize the findings.

Table 5. Comparison summary of different PPML models. "Acc. S" refers to accuracy for small NNs, and "Acc. L" refers to accuracy for large NNs.

Model	Forward Layer Form	Depth	Acc. S	Acc. L	Encrypted Eval. Time
MLP-S	$W_2(W_1x + b_1)^2 + b_2$	Medium	2nd	2nd	2nd
MLP-Q	$W_{21}(W_1x + b_1)^2 + W_{22}(W_1x + b_1) + b_2$	Medium	1st	1st	2nd
KAN-S	$W_2x^2 + b_2$	Low	3rd	3rd	1st
KAN-Q	$W_{21}x^2 + W_{22}x + b_2$	Low	3rd	2nd	1st
MLP-QL	$W_2\left(\beta \odot (W_1x + b_1)^2 + \alpha \odot (W_1x + b_1) + \gamma\right) + b_2$	High	2nd	2nd	3rd

6. Discussion and Future Work

KANs [20] were introduced as a promising alternative to MLPs, inspired by the Kolmogorov–Arnold representation theorem. Unlike MLPs, which use fixed AFs at nodes, KANs employ learnable univariate functions on edges (weights), typically parameterized as splines. While described as a combination of splines and MLPs, the original work

lacks a formal structural analysis, leaving the open question of whether KANs constitute a fundamentally new architecture or an alternative parameterization of MLPs with different activations.

In this work, we address this question by providing a formal analysis of the relationship between polynomial KANs and polynomial MLPs under identical conditions, using the same class of activation functions. We further evaluate polynomial KANs in a Homomorphic Encryption setting and compare them with CryptoNets [1], which pioneered the use of polynomial activations (e.g., x^2) for encrypted inference.

Although prior work, including SLAF-based approaches [32,33,35], has explored polynomial activations for efficient and/or private inference, the design of lightweight and efficient polynomial activations for private inference remains underexplored. This is particularly important in resource-constrained environments, including IoT systems and cloud-based deployments, where both computational and memory efficiency are critical.

Both our approach and SLAF-based methods can be interpreted as reparameterized polynomial neural networks. However, differences in parameterization lead to distinct expressivity and training behavior. Our formulation lies between the SLAF frameworks of Goyal et al. [33] and Pulido et al. [35], while aiming to better align with the KAN formulation of Liu et al. [20]. In particular, we introduce a quadratic restriction and a specific reparameterization (FCQT), which together enable efficient and lightweight privacy-preserving computation by controlling multiplicative depth. To isolate the effect of FCQT reparameterization, we introduce MLP-QL, a variant closest to Pulido et al. [35], but restricted to quadratic activations. This allows a controlled comparison between stacked and non-stacked formulations under the same polynomial constraints.

Overall, our contributions go beyond simple reparameterization. We provide a formal characterization of the structural relationship between polynomial KANs and polynomial MLPs, clarifying the nature of the MLP components implicitly used in KANs when operating under identical activation functions. This analysis reveals that the closest matching MLP formulation corresponds to a specific reparameterization, referred to as FCQT, which differs from prior SLAF-based designs.

Experimental results highlight a clear trade-off between classification accuracy, multiplicative depth, and inference efficiency under Homomorphic Encryption. MLP-Q achieves the highest accuracy, demonstrating the effectiveness of quadratic activations compared to square activations used in CryptoNets. At the same time, limiting the polynomial degree to quadratic is essential to control multiplicative depth, a critical factor in HE that governs noise growth, latency, and overall system performance.

The importance of FCQT parameterization is evidenced by MLP-QL, which does not stack weights and activation coefficients. Despite using quadratic activations, it incurs higher latency due to increased multiplicative depth and does not consistently improve accuracy. This confirms that deeper polynomial compositions do not necessarily yield better performance and highlights the effectiveness of the FCQT design.

This trend is reinforced by CryptoKAN+ (KAN-Q), which achieves the best inference efficiency, particularly for larger models, due to reduced multiplicative depth, while maintaining competitive accuracy. This demonstrates that appropriate parameterization can recover much of the expressive power of deeper models while significantly improving efficiency.

Overall, our results show that quadratic activations implemented via FCQT provide an effective balance between accuracy and efficiency, and that controlling multiplicative depth is central to practical encrypted inference. CryptoKAN+ (KAN-Q) offers a favorable trade-off for latency-sensitive applications, whereas MLP-Q remains preferable when accuracy is the primary objective.

For future work, extending this analysis to deeper architectures, alternative polynomial bases, and adaptive degree selection could further improve the balance between expressivity and efficiency in privacy-preserving settings.

7. Conclusions

In this work, we provided a formal analysis of the structural differences between Kolmogorov–Arnold Networks (KANs) and Multi-Layer Perceptrons (MLPs), clarifying how the combination of MLP and spline components shapes the KAN architecture. This analysis guided the design of a KAN-inspired privacy-preserving model, CryptoKAN+, which integrates polynomial activation functions through a Fully Connected Quadratic Transformation (FCQT) layer. By enforcing polynomial activations during training, CryptoKAN+ eliminates the need for post-training polynomial symbolization while maintaining sufficient non-linearity for effective private inference under homomorphic encryption.

Extensive experiments on MNIST and Fashion-MNIST demonstrate that CryptoKAN+ achieves competitive classification accuracy while significantly reducing encrypted inference latency compared to conventional polynomial MLPs (CryptoNets). This advantage stems from the FCQT layer, which efficiently combines weights and activations to minimize multiplicative depth, enhancing encrypted inference efficiency without compromising accuracy.

Overall, our findings highlight that careful architectural choices—grounded in a formal understanding of KAN structures and the integration of polynomial FCQT activations—can simultaneously optimize encrypted inference performance while maintaining competitive accuracy. CryptoKAN+ thus represents a practical and efficient solution for privacy-preserving machine learning on resource-constrained systems, such as embedded intelligent IoT devices, delivering fast and reliable encrypted inference while sustaining high classification performance.

Author Contributions: Conceptualization, O.T.; methodology, O.T.; software, O.T.; validation, O.T., C.T. and H.O.-S.; formal analysis, O.T.; investigation, O.T., C.T. and H.O.-S.; data curation, O.T.; writing—original draft preparation, O.T.; writing—review and editing, O.T., C.T. and H.O.-S.; resources, C.T. and H.O.-S.; supervision, C.T. and H.O.-S.; project administration, C.T. and H.O.-S.; funding acquisition, C.T. and H.O.-S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by funding from the Innovation for Defence Excellence and Security (IDEaS) program of the Department of National Defence (DND), MITACS, and StreamScan Inc., whose financial support was crucial in advancing this work.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available on request from the corresponding author.

Acknowledgments: We thank the team of StreamScan Inc. for their valuable collaboration, which greatly enhanced the quality and impact of this work.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following notations are used in this manuscript:

σ	Activation function used in the MLP
W_i	Trainable weight matrix for layer i
I	Identity matrix
b_i	Trainable bias vectors for layer i
α, β, γ	Trainable vectors
sb, sp	KAN trainable branch weights
c_i	Control point coefficients used in KAN
B_i	B-spline basis functions used in KAN
$f(x_1, \ldots, x_n)$	Multivariate function
$\varphi_{q,p}, \Phi_q$	KAN univariate functions (LAFs)
$\text{base}(x)$	KAN basis activation function (e.g., SiLU)
$\text{spline}(x)$	KAN spline function
R_q	Polynomial ring used in CKKS scheme
q	Coefficient modulus defining the ciphertext space
n	Ring dimension (degree of the polynomial ring)
Δ	Scaling factor in CKKS controlling precision
q_l	Coefficient modulus at level l in CKKS
$[\cdot]_{q_l}$	Modular reduction with respect to q_l
L	Initial level of freshly encrypted ciphertext

Appendix A

Appendix A.1. Homomorphic Encryption: Encoding, Encryption, and Core Operations

Before encryption, an encoding operation is performed to map a vector of complex/real numbers into a plaintext polynomial M ready for encryption. Decoding reverses this process, reconstructing the original vector. For encryption, the public key K_P is defined as $K_P = (K_{P_1} = [-a \cdot K_S + e]_{q_l},\ K_{P_2} = a)$, where K_S is the secret key, and a and e are random polynomials. The notation $[\cdot]_{q_l}$ denotes arithmetic modulo q_l. A plaintext message M is encrypted at any level l in a ciphertext $C = (C_1, C_2)$ as follows:

$$C_1 = [K_{P_1} \cdot u + e_1 + M]_{q_l}, \quad C_2 = [K_{P_2} \cdot u + e_2]_{q_l}$$

where u, e_1, and e_2 are random polynomials. Decryption is performed by evaluating the ciphertext at level $l \geq 1$ using the secret key to generate an approximate value of the plaintext message:

$$\hat{M} = [C_1 + C_2 \cdot K_S]_{q_l} \approx M$$

The main HE operations are defined as follows:

$$\begin{aligned}
\mathbf{C}_{\text{AddPlain}}(C, M^*) &= \left([C_1 + M^*]_{q_l}, [C_2]_{q_l}\right), \\
\mathbf{C}_{\text{Add}}(C, C^*) &= \left([C_1 + C_1^*]_{q_l}, [C_2 + C_2^*]_{q_l}\right), \\
\mathbf{C}_{\text{MultPlain}}(C, M^*) &= \left([M^* \cdot C_1]_{q_l}, [M^* \cdot C_2]_{q_l}\right), \\
\mathbf{C}_{\text{Mult}}(C, C^*) &= \begin{pmatrix} [C_1 \cdot C_1^*]_{q_l}, \\ [C_1 \cdot C_2^* + C_2 \cdot C_1^*]_{q_l}, \\ [C_2 \cdot C_2^*]_{q_l} \end{pmatrix}.
\end{aligned}$$

Notice that the homomorphic multiplication C_{Mult} produces a three-term ciphertext; relinearization can then be applied to reduce the ciphertext size back to two elements.

After each multiplication, the ciphertext undergoes a *RESCALE* operation, which reduces the accumulated noise by scaling down the ciphertext by a factor of Δ. This rescaling step compensates for the growth in the scaling factor (from Δ to Δ^2) by aligning it back to Δ, but it also consumes a portion of q. Consequently, the coefficient modulus at level l is denoted by q_l, where $1 \leq l \leq L$. Here, L represents the initial level of a freshly encrypted ciphertext, which can support up to $L-1$ homomorphic multiplications. At the final level $l = 1$, the ciphertext can no longer support further operations.

The scheme parameters q, n, and Δ control the security level, the precision of the calculation, and the multiplicative depth of the homomorphic computations. Hence, these parameters must be chosen carefully. Notably, q must be large enough to accommodate the desired multiplicative depth of the evaluated circuit. In Microsoft SEAL (an open-source HE library), q is implemented as a vector of small modulus elements (moduli), each with an arbitrary bit-length of up to 60 bits. The product of these moduli constitutes the encryption coefficient modulus. In our implementation, we use Δ for all small modulus elements, except for the first and last elements of the vector, in order to preserve a small available space before decryption. The bit length of the vector q, $|q| = L$, supports at most a multiplicative depth of $L-1$ for the target circuit, leaving one remaining level for decryption (at level q_1). Once q is fixed, the parameter n is chosen to ensure an adequate security level.

The most important HE parameters affecting inference evaluation time are the coefficient modulus q and the polynomial ring dimension n. A higher multiplicative depth in the architecture necessitates a higher modulus q to perform these multiplications, which may, in turn, require a larger polynomial ring dimension to maintain the same security level. An increase in q raises inference time linearly, while a joint increase in q and n causes exponential growth in inference time.

The multiplication depth of each model can be inferred from the depths of its basic building blocks. The fully connected linear transformation layer (FC-LT) and the square AF each require a multiplication depth of 1, while the quadratic AF requires a depth of 2. Hence, to construct a NN with m AF-layers, the MLP-S and MLP-Q models require $L = 2m + 2$ moduli (i.e., the bit-length of q), allowing for $2m + 1$ multiplication levels. In contrast, the KAN-S and KAN-Q models require $L = 2m + 1$ moduli due to having one fewer hidden layer, allowing for $2m$ multiplication levels. The MLP-QL model requires $L = 3m + 2$ moduli to account for both internal AF weights and FC-LT weights applied at each layer, allowing for $3m + 1$ multiplication levels. An example for $m = 2$ and $m = 3$ is presented in Table 2. Based on this, our hypothesis regarding encrypted inference time suggests that KAN models will exhibit the lowest inference time (due to the smallest L), followed by MLP-S and MLP-Q. Meanwhile, MLP-QL is expected to have the highest inference time due to its significantly higher multiplication depth (i.e., a substantially larger L) which in turn requires a higher n.

For each model structure, we varied the number of parameters to examine how it affects inference time. It is important to note that encrypted inference time depends heavily on the selected HE parameters q and n, where the length of q (i.e., L) is tailored to the model's multiplicative depth of the circuit ($L-1$), as discussed earlier. Next, a Δ of 25 bits of floating-point HE precision was fixed in middle moduli elements of q to maintain precision. Once q is fixed, the parameter n is chosen to ensure the same security level across all models.

Appendix A.2. Threat Model and Security Analysis

In this section, we formalize the threat model adopted in CryptoKAN+ and provide a security analysis demonstrating that the protocol guarantees both data and model confidentiality.

In the CryptoKAN+ setting, the *client* is assumed to be fully trusted, whereas the *server* is modeled as an honest-but-curious (semi-honest) adversary. The server correctly follows all prescribed steps of the protocol but may attempt to infer partial information about the client's input data or model parameters from the ciphertexts. We emphasize that the client is the sole owner of both the input data and the model parameters, and that both are encrypted before being sent to the server. Consequently, the server does not possess any private information that requires protection from the client. The primary security objective is therefore to ensure that the server learns nothing about the client's input or model parameters, which remain encrypted throughout the computation.

Theorem A1. *The CryptoKAN+ protocol relies on the CKKS FHE scheme, which is IND-CPA secure under the hardness assumption of the Ring Learning With Errors (RLWE) problem. Consequently, CryptoKAN+ is secure against an honest-but-curious (semi-honest) server.*

Proof. Let the client possess the input data M and model parameters W. The client generates a CKKS key pair (K_S, K_P) and shares only the public key K_P with the server. The encryptions of M and W are given by

$$\begin{aligned} C_M = (C_{1M}, C_{2M}) = \Big(& [K_{P_1} \cdot u_M + e_{1M} + M]_{q_l}, \\ & [K_{P_2} \cdot u'_M + e_{2M}]_{q_l} \Big), \\ C_W = (C_{1W}, C_{2W}) = \Big(& [K_{P_1} \cdot u_W + e_{1W} + W]_{q_l}, \\ & [K_{P_2} \cdot u'_W + e_{2W}]_{q_l} \Big), \end{aligned}$$

where $u_M, u'_M, u_W, u'_W, e_{1M}, e_{2M}, e_{1W}, e_{2W}$ are random polynomials, independently sampled for each ciphertext. This randomness ensures that encryption is probabilistic, i.e., encrypting the same plaintext twice produces different ciphertexts.

The security of CKKS, based on the RLWE assumption, satisfies the IND-CPA property, meaning that no polynomial-time adversary can distinguish between encryptions of two chosen plaintexts with non-negligible advantage.

Formally, let $\mathrm{SE} = (K, E, D)$ be a symmetric encryption scheme and $\mathcal{A}$ a probabilistic polynomial-time (PPT) adversary. The IND-CPA experiment is defined as

$$\begin{aligned} \mathrm{Exp}^1_{\mathrm{SE}}(\mathcal{A}) &: K \xleftarrow{\$} K;\ d \xleftarrow{\$} \mathcal{A}^{E_K(\mathrm{LR}(\cdot,\cdot,1))};\ \text{return } d, \\ \mathrm{Exp}^0_{\mathrm{SE}}(\mathcal{A}) &: K \xleftarrow{\$} K;\ d \xleftarrow{\$} \mathcal{A}^{E_K(\mathrm{LR}(\cdot,\cdot,0))};\ \text{return } d, \end{aligned}$$

where the left-or-right oracle $\mathrm{LR}(M_0, M_1, b)$ returns $E_K(M_b)$ for a random bit $b \in \{0, 1\}$. The adversary's advantage is

$$\mathrm{Adv}^{\text{ind-cpa}}_{\mathrm{SE}}(\mathcal{A}) = \left| \Pr\left[\mathrm{Exp}^1_{\mathrm{SE}}(\mathcal{A}) = 1\right] - \Pr\left[\mathrm{Exp}^0_{\mathrm{SE}}(\mathcal{A}) = 1\right] \right|.$$

A scheme is IND-CPA secure if this advantage is negligible in the security parameter λ. Since CKKS satisfies this property [7,48–50], the encryptions of M and W reveal no partial information. □

Next, we show that the CryptoKAN+ protocol does not leak any additional information beyond what CKKS already hides. Since CryptoKAN+ can be represented as an equivalent MLP architecture, the security of the protocol follows from the security of the MLP executed under the same encryption scheme. A multi-layer MLP is simply a composi-

tion of single-layer functions, so the analysis can be reduced to the one-layer case. Then, the layer φ can be expressed as

$$\varphi = \mathrm{LAF}(W, M),$$

where M is the input, W is the list of weights of the activation function, and LAF denotes the activation function.

The server sees the encrypted W parameters, the activation function LAF is clear, and the input M is encrypted.

We analyze this using simulation-based security, following the standard Real/Ideal-world paradigm, to prove that whatever an adversary can achieve in the real world can also be simulated in the ideal world.

- **Real World:** The adversary $\mathcal{A}$ is given the public key K_P and receives the actual ciphertexts:
$$C^{\text{real}} = (C_M, C_W),$$
where C_M and C_W use random polynomials sampled in the real execution.
- **Ideal World:** A simulator $\mathcal{S}$, given only the public key K_P, generates ciphertexts encrypting zero:
$$C^{\text{ideal}} = (C_{0M}, C_{0W}),$$
with independently sampled random polynomials for each ciphertext:
$$C_{0M} = ([K_{P_1} \cdot u' + e'_1 + 0]_{q_l}, [K_{P_2} \cdot u'' + e'_2]_{q_l}),$$
$$C_{0W} = ([K_{P_1} \cdot \tilde{u} + \tilde{e}_1 + 0]_{q_l}, [K_{P_2} \cdot \tilde{u}' + \tilde{e}_2]_{q_l}).$$

Since the private key is only known to the client, the adversary cannot decrypt any ciphertext. Moreover, due to the probabilistic nature of CKKS, the real and ideal ciphertexts are computationally indistinguishable,

$$\left| \Pr\left[\mathcal{A}(C^{\text{real}}) = g\right] - \Pr\left[\mathcal{A}(C^{\text{ideal}}) = g\right] \right| \leq \mathrm{negl}(\lambda),$$

for any efficiently computable function g representing partial information about the inputs M and W. In other words, the adversary cannot learn any information beyond what g specifies from the ciphertexts.

During encrypted inference, in the real world, the server performs homomorphic operations on ciphertexts, producing

$$C^{\text{real}}_{\text{output}} = LAF(C_W, C_M).$$

In the ideal world, the simulator $\mathcal{S}$ produces

$$C^{\text{ideal}}_{\text{output}} = LAF(C_{0W}, C_{0M}).$$

Since the ciphertexts in both worlds are indistinguishable, so are their homomorphic outputs. The adversary's distinguishing advantage remains negligible throughout the protocol. The CryptoKAN+ protocol satisfies semantic security; any information that can be efficiently computed from ciphertexts can also be computed without access to them. Thus, under the honest-but-curious model and assuming IND-CPA security of CKKS, the client's data and model parameters remain fully confidential during encrypted inference.

Remark A1. *It is worth noting that the use case in which the IND-CPA security model is sufficient corresponds to scenarios where the client is the sole owner of the model and only shares its encrypted version—i.e., the encrypted model parameters—with the server, delegating the private inference*

task. However, if the model parameters are publicly available or owned by an untrusted party, the IND-CPA model becomes insufficient. This situation arises, for example, when multiple data providers encrypt their data under a common public key and delegate computation to a server that returns an unencrypted model computed from the aggregated datasets. Another example occurs when the server itself owns and trains the model using its own data and offers inference as a service. In such configurations, an adversarial server with access to the unencrypted model—that is therefore capable of observing decrypted outputs for chosen plaintexts—may recover the scheme's secret key and subsequently decrypt other clients' data during inference. To ensure stronger security guarantees in these cases, additional protective mechanisms should be employed, such as the IND-CPA^D adversarial model (Indistinguishability under Chosen Plaintext Attacks with Decryption Oracles) [48].

References

1. Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.; Naehrig, M.; Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2016; pp. 201–210.
2. Qu, Y.; Xu, C.; Gao, L.; Xiang, Y.; Yu, S. FL-SEC: Privacy-preserving decentralized federated learning using SignSGD for the Internet of Artificially Intelligent Things. *IEEE Internet Things Mag.* **2022**, *5*, 85–90. [CrossRef]
3. Falcetta, A.; Roveri, M. Privacy-preserving deep learning with homomorphic encryption: An introduction. *IEEE Comput. Intell. Mag.* **2022**, *17*, 14–25. [CrossRef]
4. Li, K.; Shi, Y. Towards Incentive with Privacy Preserving Machine Learning as a Service for Crowdsensed Data Trading. *IEEE Internet Things J.* **2024**, *11*, 36494–36507. [CrossRef]
5. Wood, A.; Najarian, K.; Kahrobaei, D. Homomorphic encryption for machine learning in medicine and bioinformatics. *ACM Comput. Surv. (CSUR)* **2020**, *53*, 1–35. [CrossRef]
6. Zhang, L.; Xu, J.; Vijayakumar, P.; Sharma, P.K.; Ghosh, U. Homomorphic encryption-based privacy-preserving federated learning in IoT-enabled healthcare system. *IEEE Trans. Netw. Sci. Eng.* **2022**, *10*, 2864–2880. [CrossRef]
7. Phong, L.T.; Aono1, Y.; Hayashi1, T.; Wang, L.; Moriai, S. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 1333–1345. [CrossRef]
8. Panzade, P.; Asl, J.R.; Takabi, D.; Cai, Z. BlindTuner: On Enhancement of Privacy-Preserving Fine-tuning of Transformers based on Homomorphic Encryption. *IEEE Internet Things J.* **2025**, *12*, 39211–39222. [CrossRef]
9. Xie, B.; Xiang, T.; Liao, X.; Wu, J. Achieving privacy-preserving online diagnosis with outsourced SVM in internet of medical things environment. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 4113–4126. [CrossRef]
10. Yang, W.; Wang, H.; Li, Z.; Niu, Z.; Wu, L.; Wei, X.; Su, Y.; Susilo, W. Privacy-preserving machine learning in cloud-edge-end collaborative environments. *IEEE Internet Things J.* **2024**, *12*, 419–434. [CrossRef]
11. Gao, Y.; Deng, H.; Zhu, Z.; Chen, X.; Xie, Y.; Duan, P.; Chen, P. PEAFOWL: Private Entity Alignment in Multi-Party Privacy-Preserving Machine Learning. *IEEE Trans. Inf. Forensics Secur.* **2025**, *20*, 2706–2720. [CrossRef]
12. Qin, H.; He, D.; Feng, Q.; Khan, M.K.; Luo, M.; Choo, K.K.R. Cryptographic primitives in privacy-preserving machine learning: A survey. *IEEE Trans. Knowl. Data Eng.* **2023**, *36*, 1919–1934. [CrossRef]
13. Wei, K.; Li, J.; Ding, M.; Ma, C.; Yang, H.H.; Farokhi, F.; Jin, S.; Quek, T.Q.; Poor, H.V. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3454–3469. [CrossRef]
14. Agrawal, S.; Sarkar, S.; Aouedi, O.; Yenduri, G.; Piamrat, K.; Alazab, M.; Bhattacharya, S.; Maddikunta, P.K.R.; Gadekallu, T.R. Federated learning for intrusion detection system: Concepts, challenges and future directions. *Comput. Commun.* **2022**, *195*, 346–361. [CrossRef]
15. Soykan, E.U.; Karaçay, L.; Karakoç, F.; Tomur, E. A survey and guideline on privacy enhancing technologies for collaborative machine learning. *IEEE Access* **2022**, *10*, 97495–97519. [CrossRef]
16. Alomari, Z. Privacy-Preserving Intrusion Detection Using Federated Learning and Differential-Privacy-Inspired Noise. In *Proceedings of the 2025 4th International Conference on Computing, Management and Telecommunications (ComManTel)*; IEEE: Piscataway, NJ, USA, 2025; pp. 106–112.
17. Aziz, R.; Banerjee, S.; Bouzefrane, S.; Le Vinh, T. Exploring homomorphic encryption and differential privacy techniques towards secure federated learning paradigm. *Future Internet* **2023**, *15*, 310. [CrossRef]
18. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In *Proceedings of the Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017*; Proceedings, Part I 23; Springer: Berlin/Heidelberg, Germany, 2017; pp. 409–437.

19. Li, S.; Zhao, S.; Min, G.; Qi, L.; Liu, G. Lightweight privacy-preserving scheme using homomorphic encryption in industrial internet of things. *IEEE Internet Things J.* **2021**, *9*, 14542–14550. [CrossRef]
20. Liu, Z.; Wang, Y.; Vaidya, S.; Ruehle, F.; Halverson, J.; Soljačić, M.; Hou, T.Y.; Tegmark, M. Kan: Kolmogorov-arnold networks. *arXiv* **2024**, arXiv:2404.19756.
21. Lai, Z.; Zhou, Y.; Zheng, P.; Chen, L. Efficient Privacy-Preserving KAN Inference Using Homomorphic Encryption. *arXiv* **2024**, arXiv:2409.07751. [CrossRef]
22. Chabanne, H.; De Wargny, A.; Milgram, J.; Morel, C.; Prouff, E. Privacy-preserving classification on deep neural network. *Cryptol. ePrint Arch.* **2017**.
23. Rivoir, D.; Funke, I.; Speidel, S. On the pitfalls of Batch Normalization for end-to-end video learning: A study on surgical workflow analysis. *Med. Image Anal.* **2024**, *94*, 103126. [CrossRef]
24. Hesamifard, E.; Takabi, H.; Ghasemi, M.; Wright, R.N. Privacy-preserving machine learning as a service. *Proc. Priv. Enhancing Technol.* **2018**, *2018*, 123–142. [CrossRef]
25. Bourse, F.; Minelli, M.; Minihold, M.; Paillier, P. Fast homomorphic evaluation of deep discretized neural networks. In *Proceedings of the Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2018*; Proceedings, Part III 38; Springer: Berlin/Heidelberg, Germany, 2018; pp. 483–512.
26. Chou, E.; Beal, J.; Levy, D.; Yeung, S.; Haque, A.; Li, F.-F. Faster cryptonets: Leveraging sparsity for real-world encrypted inference. *arXiv* **2018**, arXiv:1811.09953.
27. Kim, M.; Song, Y.; Wang, S.; Xia, Y.; Jiang, X. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR Med. Inform.* **2018**, *6*,e8805. [CrossRef] [PubMed]
28. Chen, H.; Gilad-Bachrach, R.; Han, K.; Huang, Z.; Jalali, A.; Laine, K.; Lauter, K. Logistic regression over encrypted data from fully homomorphic encryption. *BMC Med. Genom.* **2018**, *11*, 3–12. [CrossRef]
29. Brutzkus, A.; Gilad-Bachrach, R.; Elisha, O. Low latency privacy preserving inference. In *Proceedings of the International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2019; pp. 812–821.
30. Khan, T.; Michalas, A. Learning in the Dark: Privacy-Preserving Machine Learning using Function Approximation. In *Proceedings of the 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*; IEEE: Piscataway, NJ, USA, 2023; pp. 62–71.
31. Hou, L.; Samaras, D.; Kurc, T.; Gao, Y.; Saltz, J. Convnets with smooth adaptive activation functions for regression. In *Proceedings of the Artificial Intelligence and Statistics*; PMLR: Cambridge, MA, USA, 2017; pp. 430–439.
32. Goyal, M.; Goyal, R.; Lall, B. Learning activation functions: A new paradigm for understanding neural networks. *arXiv* **2019**, arXiv:1906.09529.
33. Goyal, M.; Goyal, R.; Lall, B. Improved polynomial neural networks with normalised activations. In *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN)*; IEEE: Piscataway, NJ, USA, 2020; pp. 1–8.
34. Livni, R.; Shalev-Shwartz, S.; Shamir, O. On the computational efficiency of training neural networks. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 855–863.
35. Pulido-Gaytan, B.; Tchernykh, A. Self-learning activation functions to increase accuracy of privacy-preserving Convolutional Neural Networks with homomorphic encryption. *PLoS ONE* **2024**, *19*, e0306420. [CrossRef]
36. Fu, Y.; Tong, Y.; Ning, Y.; Xu, T.; Li, M.; Lin, J.; Feng, D. Swift: Fast Secure Neural Network Inference With Fully Homomorphic Encryption. *IEEE Trans. Inf. Forensics Secur.* **2025**,*20*, 2793–2806. [CrossRef]
37. Tahmi, O.; Talhi, C.; Ould-Slimane, H. CryptoKANs: Enhancing Privacy-Preserving Machine Learning in IoT Environments using Kolmogorov-Arnold Networks over Encrypted Data. *Authorea Preprints* **2024**.
38. Kim, A.; Song, Y.; Kim, M.; Lee, K.; Cheon, J.H. Logistic regression model training based on the approximate homomorphic encryption. *BMC Med. Genom.* **2018**, *11*, 23–31. [CrossRef] [PubMed]
39. Kolmogorov, A.N. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Proceedings of the Doklady Akademii Nauk*; Russian Academy of Sciences: Moscow, Russia, 1957; Volume 114, pp. 953–956.
40. Braun, J.; Griebel, M. On a constructive proof of Kolmogorov's superposition theorem. *Constr. Approx.* **2009**, *30*, 653–675. [CrossRef]
41. Dhiman, V. Kan: Kolmogorov–arnold networks: A review. *arXiv* **2024**, arXiv:2404.19756.
42. Yu, R.; Yu, W.; Wang, X. Kan or mlp: A fairer comparison. *arXiv* **2024**, arXiv:2407.16674. [CrossRef]
43. Guo, S.; Alvarez, J.M.; Salzmann, M. Expandnets: Linear over-parameterization to train compact convolutional networks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1298–1310.
44. Lee, J.; Lee, H.; Choi, W. Wireless channel adaptive DNN split inference for resource-constrained edge devices. *IEEE Commun. Lett.* **2023**, *27*, 1520–1524. [CrossRef]
45. Mohan, K.; Wang, H.; Zhu, X. Kans for computer vision: An experimental study. *arXiv* **2024**, arXiv:2411.18224. [CrossRef]

46. Abd Elaziz, M.; Fares, I.A.; Aseeri, A.O. Ckan: Convolutional kolmogorov–arnold networks model for intrusion detection in iot environment. *IEEE Access* **2024**, *12*, 134837–134851. [CrossRef]
47. Benaissa, A.; Retiat, B.; Cebere, B.; Belfedhal, A.E. TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption. *arXiv* **2021**, arXiv:2104.03152. [CrossRef]
48. Li, B.; Micciancio, D. On the security of homomorphic encryption on approximate numbers. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 648–677.
49. Phong, L.T.; Phuong, T.T. Privacy-preserving deep learning via weight transmission. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 3003–3015. [CrossRef]
50. Lyubashevsky, V.; Peikert, C.; Regev, O. On ideal lattices and learning with errors over rings. *J. ACM (JACM)* **2013**, *60*, 1–35. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.