

## Article

# Homomorphic Evaluation of Neural Networks Using Functional Bootstrapping with CKKS

Mona Scheerer<sup>1,2</sup>  and Yogachandran Rahulamathavan<sup>1,\*</sup> 

<sup>1</sup> Institute for Digital Technologies, Loughborough University, London E20 3BS, UK; mona.scheerer@stud.uni-heidelberg.de

<sup>2</sup> Institute for Mathematics, Heidelberg University, 69120 Heidelberg, Germany

\* Correspondence: y.rahulamathavan@lboro.ac.uk

## Abstract

In this work, the newly developed functional bootstrapping (FBT) for the Cheon–Kim–Kim–Song (CKKS) scheme is used for the first time to homomorphically evaluate an encrypted neural network. The advantage of FBT over previous approaches for the homomorphic evaluation of non-linear activation functions is that it combines bootstrapping and homomorphic function evaluation. For this purpose, FBT for CKKS is extended to be applied to real input values by evaluating the first order Hermite interpolation function not only on its interpolation points but on the entire domain  $[0, 1]$ . For the sigmoid function, to respect the internal representation of negative values in CKKS and the convergence behaviour of trigonometric interpolation, a glueing of shifted and reflected sigmoid functions that is periodic and continuous is used as an input function for FBT. The experimental results yield an accuracy of 97.33% with a relative loss of 0% compared to the Hermite plaintext counterpart that were obtained with a fully connected neural network with 100 hidden neurons on the MNIST test set at a security level of 128 bits. The current implementation required approximately 1.66 s per image (amortised time) and about 201 GB RAM.

**Keywords:** fully homomorphic encryption; CKKS; functional bootstrapping; privacy-preserving machine learning; MNIST

## 1. Introduction

### 1.1. Homomorphic Encryption

Homomorphic encryption (HE) is a field in cryptography where computations are performed on encrypted data without having decryption. From a security point of view, the main advantage of common homomorphic security schemes such as BGV, BFV, TFHE, CKKS is that the underlying mathematical problems such as learning with errors (LWE) [1], and ring learning with errors (RLWE) [2] are believed to be quantum secure, making these schemes a promising area for future research. Furthermore, their arithmetic capability makes them suitable for privacy-preserving machine learning (PPML), such as client–server applications when dealing with sensitive data.

Within the field of homomorphic encryption, a somewhat homomorphic encryption (SHE) scheme supports evaluating a circuit of limited depth, meaning that the scheme is capable of evaluating a limited number of compositions of operations before the decryption result deviates too much from the intended outcome. The first SHE scheme was introduced by [3], where this limitation of SHE is overcome by a modification with a method called bootstrapping. This means that the ciphertext's own decryption circuit is homomorphically



Academic Editor: Carlo Blundo

Received: 11 February 2026

Revised: 21 May 2026

Accepted: 26 May 2026

Published: 28 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 \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

evaluated, enabling Gentry's scheme to become a fully homomorphic encryption (FHE) scheme, meaning that it can evaluate a circuit of arbitrary depth.

Further schemes were introduced with BGV [4] and BFV [5,6], which support the encryption of vectors of integers, and TFHE [7], a bit-wise encryption scheme. The latter came with a new method called functional bootstrapping, which evaluated arbitrary functions via a look-up table (LUT) while simultaneously performing bootstrapping.

The next development was the Cheon–Kim–Kim–Song (CKKS) scheme [8], which is an approximate scheme that supports the encryption of complex-valued vectors. CKKS requires large parameters defining the internal structure of the scheme to ensure a security standard [9]. Thus, sequential homomorphic operations with CKKS are computationally expensive. The SHE scheme of CKKS was extended to a fully homomorphic scheme with the introduction of its bootstrapping procedure by [10].

In [11], functional bootstrapping was created for coefficient encoded BFV ciphertexts, so-called RLWE ciphertext, by switching from RLWE to CKKS and then applying a trigonometric interpolation function that was interpolated at the integer points of a pre-defined and finite RLWE message space. A blueprint to adapt their functional bootstrapping process to CKKS for integer inputs was given as well. This opens the possibility of evaluating functions in CKKS while simultaneously bootstrapping.

### 1.2. Privacy-Preserving Machine Learning with CKKS

In machine learning as a service (MLaaS), a client sends data to a service provider, which is then evaluated on a machine learning model. This process requires the client to trust the provider with their potentially sensitive data. Protecting this data from being exposed to the provider is the object of privacy-preserving machine learning (PPML). In this area of research, homomorphic encryption (HE) has become a prominent method. Since CKKS is an approximate encryption scheme that supports the encryption of real-valued inputs, it is suitable for evaluating neural networks, being or consisting of approximations themselves. Furthermore, CKKS supports SIMD packing, enabling the simultaneous evaluation of thousands of encrypted inputs within a single ciphertext and therefore providing significantly higher throughput compared to bit-wise FHE schemes. However, evaluating non-linear activation functions in CKKS remains challenging because such functions must typically be approximated using low-degree polynomials from numerical analysis, which introduces approximation errors, consumes multiplicative depth, and eventually requires separate bootstrapping operations for deep neural networks. Recently, functional bootstrapping (FBT) for CKKS [11] introduced a new approach that integrates arbitrary function evaluation directly into the bootstrapping procedure itself. Unlike conventional leveled CKKS evaluation, CKKS-FBT enables the evaluation of lookup table (LUT)-style functions while simultaneously refreshing ciphertext noise, thereby reducing the accumulated depth growth associated with repeated activation evaluations. In addition, the trigonometric Hermite interpolation used in CKKS-FBT provides local noise cleaning properties around interpolation points, improving robustness against approximation errors introduced during homomorphic computation. Motivated by these advantages, this work explores the application of CKKS-FBT for evaluating the sigmoid activation function over real-valued encrypted inputs.

### 1.3. Contribution of This Work

This work provides the following contributions.

- (i) A continuous and  $p$ -periodic input function  $f$  for FBT is created that approximates the sigmoid function  $\sigma$  and respects the internal representation of negative values in CKKS with

$$f : [0, p] \rightarrow [0, 1]$$

$$x \mapsto \begin{cases} \sigma(x), & x < \frac{p}{4} \\ \sigma(-x + \frac{p}{2}), & \frac{p}{4} \leq x \leq \frac{3p}{4} \\ \sigma(x - p), & x > \frac{3p}{4} \end{cases}$$

The function  $f$  approximates sigmoid on  $[0, p/2]$  and on  $[p/2, p]$ , the function  $f$  corresponds approximately to the sigmoid function on  $(-p/2, 0]$ .

- (ii) A pipeline for applying FBT for CKKS as the activation function sigmoid for homomorphic image inference is proposed. This is done by using  $f$  as an input function for FBT and choosing the FBT parameter  $p$  depending on the size of the weights of the plaintext neural network.

Experiments on two fully connected neural networks, which have the same architectures as [12], maintained very high accuracy results and serve as an initial proof of concept for the pipeline. The network architectures are employed as a demonstration workload but are not a methodological contribution of this work. The obtained results are as follows.

- (i) On an architecture with 100 hidden neurons, the accuracy of the encrypted model is 97.33% with a relative loss of  $-0.04\%$  (w.r.t. the sigmoid plaintext model) compared to an accuracy of 96.35% with a relative loss of 0.41% in [12].
- (ii) On an architecture with 30 neurons, the accuracy of the encrypted model is 95.71% with a relative loss of 0.43% (w.r.t. the sigmoid plaintext model) compared to an accuracy of 93.71% with a relative loss of 1.11% in [12].
- (iii) At 128-bit security, the amortised runtime for a single inference is approximately 1.52 s for the 30-neuron architecture and requires about 201 GB RAM.
- (iv) At 128-bit security, the amortised runtime for a single inference is approximately 1.66 s for the 100-neuron architecture and requires about 201 GB RAM.

## 2. Literature Review

### 2.1. PPML Using TLWE

There have been several works [12–14] for homomorphically evaluating neural networks for image classification using TLWE, with [14] having the best accuracy results. As a bit-wise encryption scheme, TLWE supports non-arithmetic operations required for activation functions. The evaluation of the latter is either facilitated via direct Boolean circuits [13,14] or incorporated into TFHE functional bootstrapping [7]. While the work in [12–14] show significant advances in framework proposals for homomorphic image inference, the TLWE encryption requires discretisation of the weights which causes a priori accuracy losses in [12,13]. CKKS on the other hand supports the encryption of real-valued weights, hence the implementation of this work will not suffer from an a priori loss.

### 2.2. PPML Using CKKS

There have been several works that evaluated deep convolutional neural networks (CNN) with CKKS [15–18], all of which were evaluated on the CIFAR-10 or CIFAR-100 dataset and making use of RNS-CKKS bootstrapping [19].

One main focus of these works is to optimise the homomorphic convolution operations: by proposing a convolution operation that is independent of the size of the kernel by reducing a single convolution operation to a single polynomial multiplication, ref. [17] yield the best inference result for a single image among [15–18] taking 368 s for the Plain-20 architecture. The best throughput is achieved by [18] requiring 15.76 s for ResNet-20

with channel-by-channel packing for faster convolution. They also have the best accuracy preservation of their encrypted model compared to the plaintext model among [15–18].

Although refs. [15–18] achieve notable improvements on the runtime by effective homomorphic convolution operations, the approximation of ReLU and the bootstrapping component of the architecture still remain the bottleneck for their runtime. Hence, combining the approximation of the activation function and bootstrapping with functional bootstrapping (FBT) is expected to significantly reduce the runtime.

### 2.3. Approximation of the Activation Function in CKKS

To approximate the activation function in an encrypted neural network, polynomial approximation is widely applied [15–18,20,21]. Currently, the most common technique for polynomial approximation is the composition of minmax approximations [22] used in [15–18,21]. In this method, the sign function is approximated by a composition of low degree min–max polynomials generated by an improved version of Remez’ algorithm [19]. Then, by writing the maximum as a function of sign, ReLU and max pooling are approximated. While the composition of minmax approximation achieves high accuracy results in homomorphic inference [15–18,21], it is not suitable to combine it with bootstrapping to create FBT, because minmax approximations are polynomials and thus, are not periodic. The technical bootstrapping step in FBT to homomorphically remove generated overflows is solved elegantly in FBT by using an approximation that is periodic, namely trigonometric interpolation. A polynomial approximation could not achieve this without additional effort.

Trigonometric interpolation for the activation function was used in [23,24]. While both works propose different approaches on how to adapt a smooth or periodic variant of ReLU and sigmoid, using trigonometric interpolation within bootstrapping for CKKS for an activation function was not examined.

Another approach for approximating an activation function within a CKKS homomorphic inference is switching from CKKS to TFHE for nonarithmetic operations [23,25]. In particular, ref. [25] evaluated ResNet-20 in CKKS by switching to TFHE to evaluate ReLU and maxpooling in parallel for every slot by TFHE functional bootstrapping [7]. While ref. [25] showed runtime improvements in comparison to purely CKKS homomorphic inference, FBT is expected to have a better amortised runtime since FBT with CKKS can process the same number of slots at once without having to perform a scheme switch and since the initial experiments by [11] indicate that FBT for CKKS has better amortised runtime than TFHE functional bootstrapping.

## 3. Foundational Background

### 3.1. Notation

Let  $M \in \mathbb{N}_{>2}$  be a power of 2. Let  $\Phi_M$  denote the  $M$ -th cyclotomic polynomial and  $\varphi$  Euler’s totient function with  $\varphi(M) = N$ . Let the scaling factor  $\Delta$  and the modulus  $q$  be a power of 2, for the level  $L \in \mathbb{N}$ , set  $q_L = \Delta^L \cdot q$ . Let  $\mathbb{Z}_{q_L}$  denote the ring of integers modulo  $q_L$ . A residue class of  $\mathbb{Z}[X]/\Phi_M$  is denoted by Fraktur notation  $\mathfrak{f}(X) \in \mathbb{Z}[X]/\Phi_M$ . Its vector of coefficients will be denoted by  $\mathfrak{f}$ . Set  $\mathcal{R} = \mathbb{Z}[X]/\Phi_M$  and  $\mathcal{R}_q = \mathbb{Z}_q[X]/\Phi_M$ .

### 3.2. CKKS Scheme

The Cheon–Kim–Kim–Song (CKKS) scheme is a homomorphic encryption scheme that was introduced by [8]. Its security system is tied to a quantum resistant computational problem called Ring Learning with Errors (RLWE) Problem introduced by [2].

### 3.2.1. Overview

CKKS is an approximate and homomorphic encryption scheme. Approximate means that it takes a message  $m$ , encrypts it and its decryption result  $m + e$  is an approximate value of the original message  $m$ . Before the message is encrypted in CKKS, it is first brought into a suitable form for encryption. This procedure is called packing, denoted by  $\text{Pac}(\cdot)$ . The suitable form is a polynomial with integer coefficients. Then, this polynomial is encrypted, which is denoted by  $\text{Enc}(\cdot)$ :

$$\text{message(s)} \longrightarrow \text{polynomial} \longrightarrow \text{ciphertext}.$$

With this process, it can be distinguished between three spaces: a message space, a plaintext space and an encrypted domain. A technical high-level description of CKKS is given by the diagram in Figure 1.

$$\begin{array}{ccccc} \text{Message space} & \longleftrightarrow & \text{Plaintext space} & \longleftrightarrow & \text{Encrypted domain} \\ \mathbb{C}^{N/2} & \xrightarrow{\text{Pac}(\cdot)} & \mathcal{R} & \xrightarrow{\text{Enc}(\cdot)} & \mathcal{R}_{q_L}^2. \end{array}$$

**Figure 1.** High-level schematic view of CKKS.

As the message space is a multidimensional complex space, it supports encrypting and processing multiple messages at once (SIMD capability).

CKKS as a homomorphic scheme supports homomorphic arithmetic. This means that the mappings between message space, plaintext space and encrypted domain are (approximately) structure-preserving, so they respect arithmetic operations on each space (approximately). More technically speaking: for  $\mathbf{m}(X), \mathbf{r}(X) \in \mathcal{R} = \mathbb{Z}[X]/\Phi_M$ , encoded polynomials of  $m, r \in \mathbb{C}^{N/2}$  and an arbitrary arithmetic operation  $*$  for each space, it holds:

$$\mathbf{m}(X) * \mathbf{r}(X) = \text{Pac}(m) * \text{Pac}(r) = \text{Pac}(m * r), \quad (1)$$

$$\text{Enc}(\mathbf{m}(X) * \mathbf{r}(X)) = \text{Enc}(\mathbf{m}(X)) * \text{Enc}(\mathbf{r}(X)). \quad (2)$$

However, as the encryption is approximate, the homomorphic arithmetic is also approximate. Therefore, the '=' in (1) and (2) is understood as 'approximately equal'. Note, that as homomorphic multiplications progress, the space of the encrypted domain (Figure 1) changes from  $\mathcal{R}_{q_L}^2$  to  $\mathcal{R}_{q_l}^2$  with  $q_l < q_L$  caused by an operation called rescaling. For procedures in CKKS, it is always of interest to reduce problems of the encrypted domain to problems in the message space  $\mathbb{C}^{N/2}$  since proper analysis such as analytical series approximation can be done on that space. This will be seen for the FBT procedure in Section 3.3.

### 3.2.2. Internal Representation of Negative Values

After decryption

$$\text{Dec: } \mathcal{R}_{q_l}^2 \rightarrow \mathcal{R},$$

the plaintext initially lies in  $\mathcal{R}_q$  before being mapped to the plaintext space  $\mathcal{R}$ : internally, the representatives of the residue classes in  $\mathbb{Z}_q$  correspond to  $(-\frac{q}{2}, \frac{q}{2}] \cap \mathbb{Z}$ . In other words, in the decryption process, the coefficients of the polynomial in  $\mathcal{R}_q$ , which are elements of  $\mathbb{Z}_q$ , are mapped back to  $\mathbb{Z}$  via

$$\mathbb{Z}_q \rightarrow \left(-\frac{q}{2}, \frac{q}{2}\right] \cap \mathbb{Z}$$

$$\bar{a} \mapsto \begin{cases} a, & \frac{q}{2} \geq a \geq 0, \\ a - q, & \frac{q}{2} < a < q, \end{cases}$$

for explicit representatives  $a \in \{0, \dots, q-1\}$ , which then translates to  $\mathcal{R}_q$  and  $\mathcal{R}$ . A simplified visualisation can be found in Figure 2.

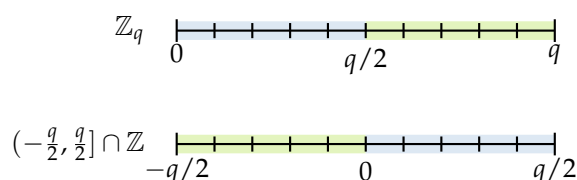


**Figure 2.** Simplified visualisation of internal representation for negative values in CKKS.

### 3.3. Functional Bootstrapping for CKKS

Functional bootstrapping (FBT) for CKKS was introduced as a blueprint by [11]. FBT performs bootstrapping of a ciphertext, meaning it extends the level of a ciphertext modulus and at the same time, it homomorphically evaluates a function. On a technical level, for a ciphertext  $ct$  at a low modulus level  $q = q_l$  for  $l < L$  that encrypts a message  $m \in \mathbb{Z}^{N/2}$  FBT does the following:

- (i) For a function  $f : \mathbb{Z}_p \rightarrow \mathbb{Z}_p$  it generates a ciphertext  $ct'$  that encrypts  $f(m)$  and that is encrypted under the same key as  $ct$ .
- (ii) The ciphertext  $ct'$  lies in an encrypted domain with a significantly higher modulus  $Q' \gg q$ .

Property (ii) then enables the scheme to perform further homomorphic multiplications. A schematic overview of the procedure can be seen in Figure 3. Note that the actual computation is performed exclusively on the encrypted domain. Since CKKS is an approximate scheme and as homomorphic operations add noise, the correspondence between elements of the encrypted domain, plaintext and message space is understood as approximate correspondence.

FBT follows the same bootstrapping idea as [10] of raising the modulus of  $\mathcal{R}_q$  (Mod-Raise) of the plaintext polynomial  $m(X)$  resulting in  $m(X) + qI(X)$  on  $\mathcal{R}_{Q'}$ , creating overflows in polynomial coefficients.

FBT furthermore follows the idea of reducing a problem to the message space to enable the use of analytical methods such as trigonometric interpolation. This means that a ciphertext having a problem is transformed into a ciphertext *encrypting* the problem, which now resides in the message space. This is done through a homomorphic matrix multiplication *CoeffToSlot* taking an encryption of a plaintext polynomial  $m(X) + qI(X)$  and creating an encryption whose decryption result after de-packing are the corresponding coefficients  $m + qI$ . The corresponding inverse transformation is *SlotToCoeff*.

The problem of interest in FBT is to remove the  $qI$  term in the raised modulus and to apply a function  $f$ , which is done by *EvalExp*. Here, a trigonometric interpolation polynomial  $T$  of  $N$ -th order is applied, satisfying

$$T\left(\frac{k}{p}\right) = f(k) \quad \text{for } k \in \{0, \dots, p-1\},$$

$$T^{(n)}\left(\frac{k}{p}\right) = 0 \quad \text{for } n \in \{1, \dots, N\},$$

where  $N \in \{1, 2, 3\}$ . Since FBT only takes input values from  $\mathbb{Z}_p$ ,  $T$  is exclusively evaluated on its  $p$  interpolation points and thus,  $T$  will always give the exact result on the message space.  $T$  computes  $f(m)$  homomorphically and due to its periodicity, removes the overflow  $qI$  at the same time. With this outline, the quality of the evaluation depends on the error growth caused by the homomorphic operations, especially the number of multiplications.

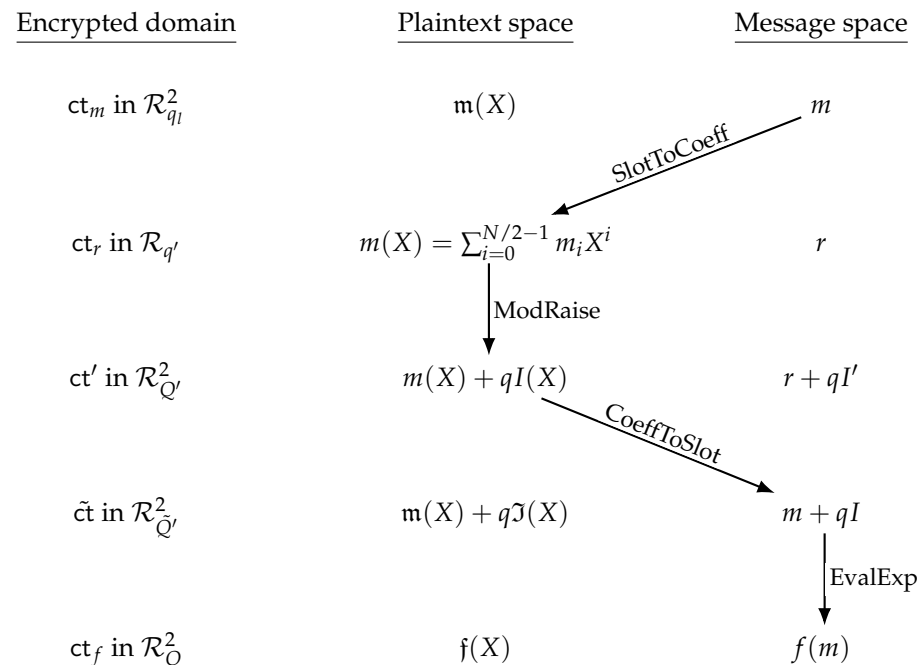


**Figure 3.** Schematic overview of functional bootstrapping for CKKS presenting the approximate impact on corresponding elements in different spaces for each step.

### 3.4. Extension of FBT to Real-Valued Inputs

The FBT method used in this work extends the blueprint by [11] to evaluate non-integer inputs. This enables the evaluation of a real-valued neural network later without having to discretise the weights. Hence, a function  $f : [0, p) \rightarrow [0, p)$  is approximated by the trigonometric polynomial in EvalExp. Therefore, the quality of the output of FBT not only depends on the error growth caused by homomorphic operations, but also on how well  $T$  approximates  $f$  on non interpolation points  $[0, p) \setminus \{0, \dots, p-1\}$  in the message space. Hence,  $f$  has to be chosen with more care. An initial condition for the choice is given below, which can be derived from ([26], Theorem 2).

**Corollary 1.** *If the input function  $f : [0, p] \rightarrow [0, p]$  for FBT is continuous and  $p$ -periodic, then the trigonometric interpolation function in EvalExp in FBT converges uniformly to  $g$  on  $[0, 1]$  with  $g(\frac{x}{p}) = f(x)$  for  $x \in [0, p]$ .*

## 4. Homomorphic Evaluation of a Neural Network

This section outlines the theoretical framework used in this work for homomorphic inference on a fully connected neural network using FBT. For that, a model is trained in plain and then a CKKS encrypted ciphertext is fed to an adaptation of the plaintext model for homomorphic encryption. Section 4.1 explains the adaptation for linear operations, Section 4.2 for the activation function.

#### 4.1. Linear Layer

Since the neural network is trained in plain, the weight matrix  $W_i$  is available in plain. Thus, the aim of this subsection is to present how to perform a (plaintext) matrix multiplication with a ciphertext. For that, this work follows the outline of ([27], Section 4.3). Since the non-quadratic weight matrices will be padded with 0, a description for a quadratic matrix suffices in this context. Let  $v \in \mathbb{R}^n$  and let  $\eta_i : \mathbb{R}^n \rightarrow \mathbb{R}^n$  be the  $i$ -th left rotation of a vector, let  $A \in \mathbb{R}^{n \times n}$  and its generalised diagonals given by  $d_i = (A_{0,i}, A_{1,i+1}, \dots, A_{n-1,i-1})$ , so for the  $j$ -th entry of  $d_i$  it is  $d_i[j] = A_{j,j+i}$ . Then it holds

$$A \cdot v = \sum_{i=0}^{n-1} d_i \odot \eta_i(v)$$

in plain, where  $\odot$  denotes the component-wise multiplication. A schematic visualisation can be found in Figure 4. Hence, multiplying a plaintext matrix with an encrypted vector is given by

$$A \cdot \text{Enc}(\text{Pac}(v)) = \sum_{i=0}^{n-1} \text{Pac}(d_i) \odot \text{rot}(i, \text{Enc}(\text{Pac}(v))). \quad (3)$$

Note that the multiplications of diagonals with the rotations are independent of each other, thus they can be parallelised.

$$\begin{pmatrix} \text{red} & \text{blue} & \text{orange} & \text{green} \\ \text{green} & \text{red} & \text{blue} & \text{orange} \\ \text{blue} & \text{green} & \text{red} & \text{blue} \\ \text{orange} & \text{blue} & \text{green} & \text{red} \end{pmatrix} \cdot \begin{pmatrix} * \\ * \\ * \\ * \end{pmatrix} = \begin{pmatrix} \text{red} \\ \text{red} \\ \text{red} \\ \text{red} \end{pmatrix} \odot \begin{pmatrix} * \\ * \\ * \\ * \end{pmatrix} + \begin{pmatrix} \text{blue} \\ \text{blue} \\ \text{blue} \\ \text{blue} \end{pmatrix} \odot \begin{pmatrix} * \\ * \\ * \\ * \end{pmatrix} + \begin{pmatrix} \text{orange} \\ \text{orange} \\ \text{orange} \\ \text{orange} \end{pmatrix} \odot \begin{pmatrix} * \\ * \\ * \\ * \end{pmatrix} + \begin{pmatrix} \text{green} \\ \text{green} \\ \text{green} \\ \text{green} \end{pmatrix} \odot \begin{pmatrix} * \\ * \\ * \\ * \end{pmatrix}$$

**Figure 4.** Schematic plaintext matrix-vector multiplication with generalised diagonals. Each colour refers to a different diagonal in the matrix.

#### 4.2. Activation Function

This section explains how the sigmoid function is approximately evaluated with FBT for CKKS. As established in Section 3.4, FBT for CKKS evaluates a function

$$f : [0, p) \rightarrow [0, p).$$

One must consider that internally in CKKS, negative numbers are represented as the upper half of  $\mathbb{Z}_p$ , so  $\mathbb{Z}_p$  represents  $(-p/2, p/2) \cap \mathbb{Z}$ , see Section 3.2.2 for more details. Hence, to construct the sigmoid function while respecting the representation of negative values in the scheme, the function chosen to approximate in FBT is

$$f : [0, p) \rightarrow [0, 1] \quad (4)$$

$$x \mapsto \begin{cases} \sigma(x), & x < \frac{p}{4} \\ \sigma(-x + \frac{p}{2}), & \frac{p}{4} \leq x \leq \frac{3p}{4} \\ \sigma(x - p), & x > \frac{3p}{4} \end{cases} \quad (5)$$

The function  $f$  is continuous and  $p$ -periodic and thus, by Corollary 1 a suitable choice. Furthermore,  $f$  approximates sigmoid on  $[0, p/2]$  and  $f$  corresponds on  $[p/2, p)$  approx-

imately to the sigmoid function on  $(-p/2, 0]$ . Hence,  $f$  is a function that approximately represents the sigmoid function while respecting the internal representation of negative values in CKKS. A visualisation can be seen in Figure 5.

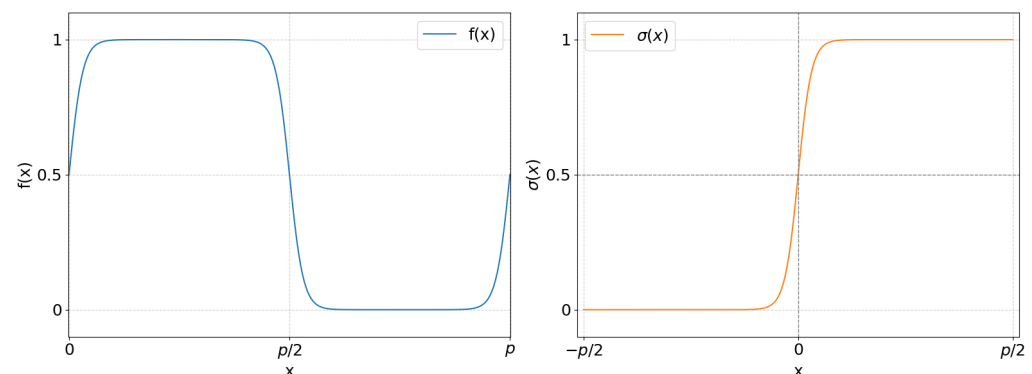


**Figure 5.** Visualisation of  $f$  on the left and the sigmoid function  $\sigma$  on the right with  $p = 64$  using matplotlib.

Note that in a neighbourhood around  $p/2$ , the function  $f$  does not approximate sigmoid well. This can be mitigated by having an input for  $f$  which does not lie within a certain neighbourhood of  $p/2$ .

## 5. Implementation Details

This section describes the implementation architecture and chosen parameters. First, two plaintext models are trained on MNIST (Section 5.1) with a feed-forward architecture (Section 5.2). Then, the models are evaluated on CKKS encrypted images from the MNIST test set where FBT is applied to homomorphically approximate the activation function (Section 5.3). Throughout the setup, the goal was to keep the input for FBT small, so that the input lies in the message space within an interval  $(-p/2, p/2]$  for a small  $p$ . This reduces the number of homomorphic operations in FBT, and thereby the error growth.

The entire implementation is built on the outline by [12], the difference being that the network is evaluated in CKKS using FBT for CKKS.

### 5.1. MNIST

The MNIST dataset consists of handwritten numbers from 0-9 with labels corresponding to the respective digit [28]. Both sets were normalised and standardised, reducing the size of the initial input and thereby the size of the weight matrices.

### 5.2. Plaintext Neural Network

All plaintext architectures were set up, which only differ in the size of the hidden layer and are as follows.

- Input layer:  $28 \times 28 = 784$  neurons;
- Hidden layer: 30, 60 or 100 neurons;
- Output layer: 10 neurons.

Instead of hard sigmoid as in [12], sigmoid was chosen as an activation function for the hidden layer. The network was implemented using keras [29] and tensorflow [30] on Google Colab. After some testing with the hyperparameters, the batch size was set to 64, epochs to 60, dropout rate to 0.1 and the Adam optimiser was chosen.

**Notation 1.** From now on, 30-neuron, 60-neuron and 100-neuron architectures will refer to the architectures with 30 hidden neurons, 60 hidden neurons, and 100 hidden neurons, respectively.

### 5.3. Encrypted Neural Network

#### 5.3.1. Homomorphic Evaluation

The encrypted neural network loads the weights and biases that were generated by the plain neural network.

**Notation 2.** The linear map in the  $i$ -th layer in the neural network will be denoted by a single matrix  $W_i$ . In the practical implementation however, the bias will be added separately to save one homomorphic rotation and multiplication.

A single image inference has the following process.

1. Encrypt a normalised and standardised  $28 \times 28$  image from the MNIST dataset as a CKKS ciphertext.
2. Evaluate the linear map with the weight matrix multiplication  $W_1 \text{ct}_0$  as in (3).
3. Apply FBT with CKKS with function  $f$  as in (4).
4. Evaluate the linear map with the weight matrix multiplication  $W_2 \text{ct}_1$  as in (3).
5. Decrypt the CKKS ciphertext.
6. Apply argmax to get the classified label.

A visual description can be found in Figure 6, which was inspired by ([12], Figure 6.1). Note that contrary to the visual description, multiple images will be processed at once in the implementation due to the SIMD capability of CKKS.

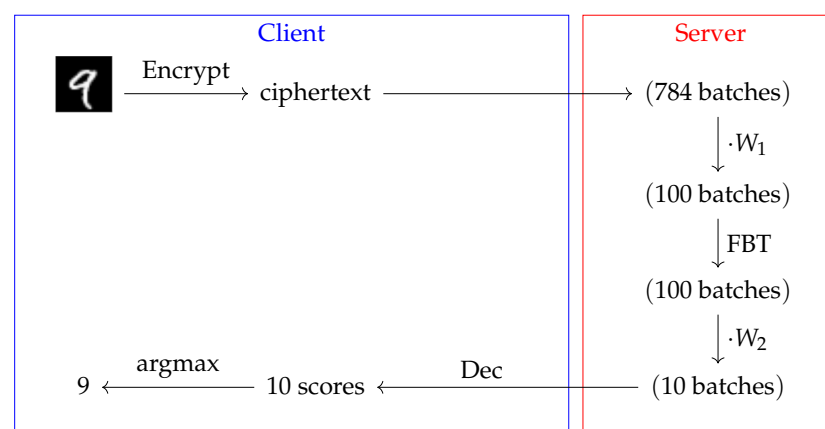


**Figure 6.** Homomorphic evaluation process for FBT-CKKS-100.

**Notation 3.** FBT-CKKS-30, FBT-CKKS-60 and FBT-CKKS-100 will refer to the encrypted architectures.  $\mathcal{L}_1$ - $k$  neurons will refer to the first layer of the architecture of  $k = 30, 60, 100$  neurons.

#### 5.3.2. FBT Parameters

To set the parameters for FBT, the size of  $p$  for the message interval  $[0, p)$  of the function  $f$  had to be determined by examining the maximal required input size for FBT. For that, the maximal L1 norm for the input for sigmoid (and thereby for FBT) was computed, which is the output of the weight matrix  $W_1$  in the first layer was evaluated on the test set. Based on these results, values for  $p$  were derived which have to be twice as big as the input size to support negative values (see Section 3.2.2). Table 1 presents the experimentally derived sizes of  $p$  based on the upper bounds of the input vector size.

**Table 1.** Experimentally derived size of  $p$  for FBT based on L1 norm of output of  $W_1$  on MNIST test set.

|                              | $\max_{x \in \text{Test}} \ W_1 x\ _1$ | exp. $p$ |
|------------------------------|----------------------------------------|----------|
| $\mathcal{L}_1$ -30-neurons  | 18                                     | 64       |
| $\mathcal{L}_1$ -60-neurons  | 34                                     | 128      |
| $\mathcal{L}_1$ -100-neurons | 52                                     | 128      |

Table 2 shows the FBT parameters used for both encrypted models. Note that although Table 1 indicated a required size of  $p = 64$  for 30-neurons,  $p = 128$  was set for both architectures for better comparability between the models and slightly more precision since  $p$  is also the number of interpolation points in FBT. For an initial network setup, a Hermite order of 1 was chosen.

**Table 2.** FBT parameters for FBT-CKKS-30, FBT-CKKS-60 and FBT-CKKS-100.

| $f$ | $p$ | $\log(p)$ | Hermite Order |
|-----|-----|-----------|---------------|
| (4) | 128 | 7         | 1             |

### 5.3.3. CKKS Parameters

For a 128-bit security implementation, suitable CKKS parameters were used which can be found in Table 3. The ciphertext modulus is denoted by  $Q$  and modulus required for relinearisation for key-switching by  $P$ . With an input size of 784 for a single image, the Ring dimension of  $2^{17}$  allowed to process 83 images at once in SIMD fashion.

**Table 3.** CKKS parameters for setting up FBT-CKKS-30, FBT-CKKS-60 and FBT-CKKS-100.

| Security Level | Hamming Weight | Ring Dimension | Modulus $QP$ | Base Modulus $q$ | Scaling Factor $\Delta$ | Mult. Depth |
|----------------|----------------|----------------|--------------|------------------|-------------------------|-------------|
| 128 bit        | 64             | $2^{17}$       | 1262 bit     | 41 bit           | 40 bit                  | 23          |

### 5.4. Library Sec-Ckks

At the time of setting up the encrypted evaluation, an implementation of FBT for CKKS was not available in any official homomorphic encryption library. Therefore, the encrypted neural network was implemented using sec-ckks [31], a fork of OpenFHE [32]. Currently, the implementation supports FBT for CKKS for full packing, meaning it processes all  $N/2$  slots, FIXEDMANUAL scaling mode and SPARSE\_TENARY as the secret key distribution.

### 5.5. System Specification

OpenMP 5.1 [33] was used to parallelise the processes of the fork and the matrix multiplication. The programme was running on an AWS EC2 instance with an Intel(R) Xeon(R) Platinum 8375C CPU@395 2.90 GHz, 32 logical CPUs, and 247 GiB RAM.

### 5.6. Baseline Hermite Plaintext Model

Apart from the plaintext model with sigmoid as an activation function, a plaintext model reflecting the FBT pipeline was set up to act as a baseline for the accuracy evaluation of the encrypted models. Instead of sigmoid as an activation function, it used the same Hermite interpolation function as FBT in the encrypted model. The coefficients were generated based on ([11], Corollary 1) and the specifications in Section 5.3.2. It received the

same pre-trained weights as the encrypted models that were trained on the plaintext model with sigmoid as an activation function (see Section 5.2). Before passing layer 1 outputs to the Hermite interpolation function, they were shifted by  $p$  to accommodate the internal representation of negative values that was necessary for FBT (see Section 4.2).

## 6. Experimental Results

### 6.1. Accuracy

Table 4 shows the accuracies of the plaintext models and of the FBT-CKKS-NN, which slightly drop for the 30-neuron and 60-neuron architecture and have identical accuracy results for the 100 neurons architecture. The accuracy was evaluated for all architectures on the MNIST test set. The classification accuracy for FBT-CKKS-30 is 95.71% with a relative loss of 0.48%, for FBT-CKKS-60 it is 96.69% with a relative loss of 0.32% and for FBT-CKKS-100 it is 97.33% with a relative loss of 0%. All encrypted networks maintain a high accuracy overall.

**Table 4.** Accuracy results of plaintext neural networks (Sigmoid vs. Hermite activations) and the CKKS encrypted neural network with FBT evaluated on the MNIST test set at 128-bit security.

| Network Architecture | Accuracy            |                     |                 | Relative Loss (Enc. vs Hermite) |
|----------------------|---------------------|---------------------|-----------------|---------------------------------|
|                      | Plaintext (Sigmoid) | Plaintext (Hermite) | Encrypted (FBT) |                                 |
| 30-neurons           | 96.12%              | 96.18%              | 95.71%          | 0.48%                           |
| 60-neurons           | 96.98%              | 97.00%              | 96.69%          | 0.32%                           |
| 100-neurons          | 97.29%              | 97.33%              | 97.33%          | 0%                              |

Table 5 presents the agreement ratio, which is the proportion of the number of images that are classified identically by both (plain and encrypted) networks. It is very high for 30-neurons (99.29%), for 60-neurons (99.47%) and 100-neurons (99.75%). Furthermore, it gives the proportion (with respect to the test set) of images that were correctly classified by precisely one model. All architectures show that they correctly classify a very small number of images that were not correctly classified by their plaintext counterpart. Note that since the models classify an image from 0–9, meaning they have more than two options for classification, a disagreement does not imply that either of the models needs to have a correct prediction.

**Table 5.** Agreement ratio between Hermite plaintext and encrypted model and ratio of a correct prediction by precisely one model among all predictions.

|             | Agreement Ratio | Plain Correct (Enc. Wrong) | Enc. Correct (Plain. Wrong) |
|-------------|-----------------|----------------------------|-----------------------------|
| 30-neurons  | 99.29%          | 0.58%                      | 0.11%                       |
| 60-neurons  | 99.47%          | 0.39%                      | 0.08%                       |
| 100-neurons | 99.75%          | 0.12%                      | 0.12%                       |

### 6.2. Amortised Runtime and Memory Consumption

To complement the 128-bit measurements, additional encrypted inference runs were carried out for a better comparability with [12]. For the 30-neuron model, and the 100-neuron model, the accuracies of the 80-bit model are the same as the accuracies of 128-bit security. The high ring dimension for both security levels allowed to evaluate multiple images at once (SIMD based packing). The resulting runtime and memory measurements are summarised in Table 6.

**Table 6.** Runtime and memory measurements for FBT-CKKS inference at 80-bit, and 128-bit security.

| Security | Model       | Ring Dim. | Depth | Time/Image | Memory (GB) |
|----------|-------------|-----------|-------|------------|-------------|
| 80-bit   | 30-neurons  | $2^{16}$  | 23    | 1.38 s     | 95.84       |
| 80-bit   | 100-neurons | $2^{16}$  | 23    | 1.45 s     | 95.73       |
| 128-bit  | 30-neurons  | $2^{17}$  | 23    | 1.52 s     | 201.01      |
| 128-bit  | 100-neurons | $2^{17}$  | 23    | 1.66 s     | 200.99      |

The measurements show that the encrypted inference is feasible at standard-security settings with amortised time largely unchanged, but at a substantial memory cost. Increasing the security level from 80-bit to 128-bit increases the amortised runtime slightly from 1.38 s to 1.52 s per image for the 30-neuron model, increases the ring dimension from  $2^{16}$  to  $2^{17}$  while increasing memory usage from 95.84 GB to approximately 201 GB. These measurements quantify the practical deployment cost of moving to standard security levels in the current implementation.

### 6.3. FBT Versus Polynomial Sigmoid Approximation

As polynomial approximation is the standard approach for activation evaluation in CKKS-based inference, an additional comparison was carried out against a non-FBT polynomial baseline on the 30-neurons architecture. The best polynomial identified in the plaintext study was the degree-9 approximation

$$\sigma(x) \approx 0.5 + 0.05612553835x - 7.301552715625 \times 10^{-5}x^3 + 3.573702564453125 \times 10^{-8}x^5 - 6.62557412109375 \times 10^{-12}x^7 + 4.04203385925293 \times 10^{-16}x^9.$$

On the MNIST test images in plaintext, this approximation achieved an accuracy of 94.85%, compared with 98.33% for the original sigmoid network.

For encrypted inference at 128-bit security, the polynomial baseline used the same ring dimension as the CKKS-FBT pipeline but a smaller multiplicative depth of 14 instead of 23. On the first 100 MNIST test images, the degree-9 polynomial baseline achieved an accuracy of 94.85%, whereas the 128-bit CKKS-FBT approach achieved 98.77% on the same images on the AWS EC2 instance. For the 30-neuron network in the plaintext domain, the corresponding accuracies are 98.33% with the true sigmoid activation and 94.85% with the polynomial approximation. Closing this gap further would require a more precise polynomial approximation, typically with smaller coefficients. However, this creates a practical difficulty in the encrypted domain: CKKS preserves only a limited number of most significant bits, so very small coefficients and lower-significance terms are more vulnerable to numerical truncation. As a result, even if an almost perfect plaintext approximation to the sigmoid is obtained, the encrypted-domain accuracy may still degrade because the contribution of the smaller terms is lost or distorted during homomorphic evaluation. Table 7 summarises this comparison.

**Table 7.** Comparison between the 128-bit FBT pipeline and the 128-bit polynomial sigmoid approximation baseline.

| Method          | Depth | Accuracy |
|-----------------|-------|----------|
| FBT-CKKS        | 23    | 98.33%   |
| Polynomial CKKS | 14    | 94.85%   |

This comparison shows that, in the current implementation, the polynomial approach is less accurate. The main reason is that the hidden-layer pre-activations span a wide range, and even a tuned low-degree polynomial cannot approximate the sigmoid accurately over the full operating interval. Note that the polynomial approximation is efficient than the FBT-CKKS due to the depth. However, FBT-CKKS will be efficient when the underlying neural network has more than one hidden layer. This is a potential future direction for FBT-CKKS based PPML.

## 7. Discussion

### 7.1. Accuracy

All encrypted models maintain an overall high absolute accuracy. This stems mostly from the correctness of the Hermite plaintext model, which is evident by the high agreement ratio. Only a small margin of random correct predictions that the plaintext model missed adds to the absolute accuracy. It was expected that a small number of encrypted predictions would deviate from the plaintext model due to homomorphic error growth. Some of these disagreements predicted the right outcome, which is stochastically to be expected. The very slight accuracy loss for the 30-neurons and 60-neurons architectures could stem from an approximation error of the trigonometric Hermite interpolation or from the error growth for homomorphic operations. Since the Hermite plaintext model shows no drop in accuracy compared to the sigmoid plaintext model, it is evident that the error growth causes the slight drop. This includes the matrix-vector multiplications with their rotations and expensive key-switching operations as well as the evaluation of FBT. The EvalExp component of FBT requires evaluating a circuit of high multiplicative depth where every multiplication needs a key switch for its relinearisation adding to the error.

There is a very slight drop in absolute and relative accuracy in the 30-neurons and 60-neurons architectures compared to the 100-neurons architecture. All architectures have the same FBT parameters and therefore the same interpolation polynomial, however, the entries of the weight matrices are different, which could attribute to the error growth.

Compared to the work of [12], the encrypted models in this work achieve better absolute accuracy results (Table 8). Furthermore, the relative losses w.r.t. the originally trained networks having sigmoid as the activation function of this work compared to the real valued plaintext model are better for both architectures. However, the relative accuracy loss in [12] comes almost entirely from an a priori loss by discretising the network in plain.

Overall, the accuracy results showcase that even for a small  $p$  using FBT for a simple fully connected neural network architecture maintains almost the same accuracy as the plaintext model and yields a high agreement ratio for a standard security level of 128-bit. This emphasises that FBT for CKKS is a promising method for homomorphically evaluating activation functions in neural networks.

**Table 8.** Comparison of accuracy results and relative loss w.r.t. to the original plaintext models with sigmoid of this work to those of [12].

|       | Encrypted Model |               | This Work    | Relative Loss                   |                                 |
|-------|-----------------|---------------|--------------|---------------------------------|---------------------------------|
|       | This Work       | Bourse et al. |              | Bourse et al.<br>Real-Valued NN | Bourse et al.<br>Discretised NN |
| 30-n  | <b>95.71%</b>   | 93.71%        | <b>0.43%</b> | 1.11%                           | −0.17%                          |
| 100-n | <b>97.33%</b>   | 96.35%        | −0.04%       | 0.41%                           | 0.08%                           |

### 7.2. Runtime

Both models have almost the same runtime, because they work with the same CKKS parameters and their computational effort is the same for the first layer and the FBT,

which are the dominating components of the architecture. The effort for FBT is the same, because the library currently only supports sparse packing.

Additionally, high-security runs show that practical security is feasible with a well-amortised runtime per image but that it comes with a steep memory cost in the current implementation. For the 30-neuron model, moving from 80-bit to 128-bit security increased the runtime from 1.38 s to 1.52 s per image and increased the memory usage from 95.84 GB to 201 GB. Comparing the runtime for 80-bit security to [12] in Table 9, it can be seen that the runtime is better for the 30-neuron architecture in [12], but faster for the 100-neuron architecture in this work.

**Table 9.** Comparison of amortised runtime at 80-bit security level with [12].

|       | <b>This Work</b> | <b>Bourse et al.</b> |
|-------|------------------|----------------------|
| 30-n  | 1.38 s           | <b>0.515 s</b>       |
| 100-n | <b>1.45 s</b>    | 1.64 s               |

The additional comparison with a polynomial sigmoid approximation also provides a useful baseline for interpreting the benefit of FBT. As mentioned earlier, polynomial approximation is slightly efficient than FBT-CKKS due to smaller depth. However, FBT-CKKS will be significantly efficient when the underlying neural network has more than one hidden layer as the FBT will refresh the ciphertext. Theoretically, FBT-CKKS-based PPML can evaluate a deep neural network with only 23 multiplicative depth (if it is based on Sigmoid activation) regardless of the number of hidden layers. This is one of the avenues for future research.

## 8. Limitations and Future Work

### 8.1. Library Limitations

The current implementation of FBT for CKKS in sec-ckks has some drawbacks. First, it computes the Hermite coefficients for every single FBT evaluation. From a methodological perspective, this is not necessary for FBT and could be optimised by pre-computing the coefficients. In addition to that, enabling a sparsely packed support for FBT, meaning taking into account if a smaller number of slots is encoded, could speed up the linear transformations in CoeffToSlot and SlotToCoeff ([10], Section 4.4). Moreover, with the scaling technique FIXEDMANUAL, the rescaling for the matrix–vector multiplication is performed after each multiplication, but there are scaling techniques that better reduce the error such as FLEXIBELAUTOEXT [32,34]. Finally, the implemented method currently only works for small inputs up to 10 bits. However, there are current developments for FBT algorithms that can evaluate larger inputs [35].

It would be interesting to see whether the release of version 1.5 of OpenFHE which will come with FBT for CKKS, would improve on the model’s performance.

### 8.2. Scalability Constraints

A main limitation of the proposed pipeline is its lack of scalability. The experimental results showed a steep memory growth associated with higher-security CKKS settings, due to the need for larger parameters at a stronger security. This would result in a security–performance trade-off for practical deployment. Moreover, choosing the size of  $p$  depending on the size of the model’s weights comes with the disadvantage of having to adjust the parameters for every new model. Hence, an object of future research could be to develop a framework to have encryption parameters that are independent of the concrete plaintext model.

Furthermore, it has to be noted that the current experiments are restricted to shallow fully connected neural networks. However, the proposed FBT pipeline could be in principle extended to deeper or convolutional architectures, since from algorithmic point of view, the computation of linear operations and an activation function approximation through FBT are independent. The challenge of adjusting  $p$  to support a message space that is large enough for the inputs of the non-linear layer would translate to deeper networks as well. As the size of  $p$  can be chosen individually for each layer, this is a general restriction of the pipeline itself independent of the depth or specific linear operations of the architecture. A new challenge that would arise with deeper architectures would come with a larger error growth and thereby with the question of how well FBT reduces the homomorphic error growth. This has to be the object of future research.

### 8.3. Framework Drawbacks

Another limitation is the current choice of  $f$  in (4) for the sigmoid function. While it is  $p$ -periodic and continuous, and therefore the trigonometric interpolation will converge, it would converge faster if it were continuously differentiable or even smooth ([36], Section 2). Thus, a smooth variant of the function in (4) would likely provide better accuracy. Furthermore, it would be interesting to use FBT to evaluate a model with a different activation function such as ReLU. The idea of [23] for a smooth variant of ReLU could be adapted to the internal representation of negative numbers in CKKS.

It would also be interesting to see whether discretising the neural network and thereby evaluating the activation function as a LUT with FBT like the blueprint in [11] would result in an overall better accuracy. This would imply having an a priori accuracy loss but no approximation error caused by the trigonometric interpolation in plain, as only the interpolation points would be evaluated. The discretisation loss might be mitigated by fixed-point encoding like in [14].

## 9. Conclusions

For the first time, FBT for CKKS was used to perform homomorphic inference on a fully connected neural network. The blueprint by [11] for FBT was adapted to real-valued inputs by evaluating a trigonometric interpolation function. As an input function for FBT in the neural network, a  $p$ -periodic and continuous function resembling sigmoid was constructed that respects the internal representation of negative values in CKKS. The experimental results show that an architecture with 100 hidden neurons achieves the same accuracy as its plaintext counterpart.

This work opens the possibility of utilising FBT for CKKS in privacy-preserving machine learning (PPML). The ideas on how to evaluate sigmoid with FBT could be extended to other activation functions such as ReLU. Furthermore, it would be interesting to examine the effectiveness of the two approaches of discretising neural networks with an exact evaluation at the trigonometric interpolation points or an approximation by trigonometric interpolation functions on a real-valued neural network. Finally, future work could explore how FBT with CKKS performs on more advanced architectures with convolution such as ResNet-20.

**Author Contributions:** Conceptualization, M.S.; methodology, M.S.; software, M.S. and Y.R.; validation, M.S.; formal analysis, M.S.; investigation, M.S.; resources, M.S.; data curation, M.S.; writing—original draft preparation, M.S.; writing—review and editing, M.S. and Y.R.; visualization, M.S.; supervision, Y.R.; project administration, Y.R. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding

**Data Availability Statement:** The code and results used in the study are openly available at <https://github.com/ms-git-code/git-fbt-code.git>, accessed on 25 May 2026.

**Acknowledgments:** This article is derived from a master thesis within a postgraduate programme that was funded by the German Academic Scholarship Foundation.

**Conflicts of Interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

1. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM (JACM)* **2009**, *56*, 1–40. [\[CrossRef\]](#)
2. Lyubashevsky, V.; Peikert, C.; Regev, O. On ideal lattices and learning with errors over rings. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 1–23.
3. Gentry, C. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
4. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory (TOCT)* **2014**, *6*, 1–36. [\[CrossRef\]](#)
5. Fan, J.; Vercauteren, F. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, Paper 2012/144, 2012. Available online: <https://ia.cr/2012/144> (accessed on 16 August 2025).
6. Brakerski, Z. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Proceedings of the Annual Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 868–886.
7. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachene, M. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 3–33.
8. 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.
9. Albrecht, M.; Chase, M.; Chen, H.; Ding, J.; Goldwasser, S.; Gorbunov, S.; Halevi, S.; Hoffstein, J.; Laine, K.; Lauter, K.; et al. *Homomorphic Encryption Security Standard*; Technical Report; HomomorphicEncryption.org: Toronto, ON, Canada, 2018.
10. Cheon, J.H.; Han, K.; Kim, A.; Kim, M.; Song, Y. Bootstrapping for approximate homomorphic encryption. In *Proceedings of the Advances in Cryptology—EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, 29 April–3 May 2018*; Proceedings, Part I 37; Springer: Berlin/Heidelberg, Germany, 2018; pp. 360–384.
11. Alexandru, A.; Kim, A.; Polyakov, Y. General functional bootstrapping using CKKS. In *Proceedings of the Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2025; pp. 304–337.
12. 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.
13. Sanyal, A.; Kusner, M.; Gascon, A.; Kanade, V. TAPAS: Tricks to accelerate (encrypted) prediction as a service. In *Proceedings of the International Conference on Machine Learning*, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 4490–4499.
14. Lou, Q.; Jiang, L. She: A fast and accurate deep neural network for encrypted data. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 10035–10043.
15. Lee, J.W.; Kang, H.; Lee, Y.; Choi, W.; Eom, J.; Deryabin, M.; Lee, E.; Lee, J.; Yoo, D.; Kim, Y.S.; et al. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access* **2022**, *10*, 30039–30054. [\[CrossRef\]](#)
16. Lee, E.; Lee, J.W.; Lee, J.; Kim, Y.S.; Kim, Y.; No, J.S.; Choi, W. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In *Proceedings of the International Conference on Machine Learning*, PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 12403–12422.
17. Kim, D.; Guyot, C. Optimized privacy-preserving cnn inference with fully homomorphic encryption. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 2175–2187. [\[CrossRef\]](#)
18. Cheon, J.H.; Kang, M.; Kim, T.; Jung, J.; Yeo, Y. Batch Inference on Deep Convolutional Neural Networks With Fully Homomorphic Encryption Using Channel-By-Channel Convolutions. *IEEE Trans. Dependable Secur. Comput.* **2025**, *22*, 1674–1685. [\[CrossRef\]](#)
19. Lee, J.W.; Lee, E.; Lee, Y.; Kim, Y.S.; No, J.S. High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 618–647.

20. Ishiyama, T.; Suzuki, T.; Yamana, H. Highly accurate CNN inference using approximate activation functions over homomorphic encryption. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; pp. 3989–3995.
21. Lee, J.; Lee, E.; Lee, J.W.; Kim, Y.; Kim, Y.S.; No, J.S. Precise approximation of convolutional neural networks for homomorphically encrypted data. *IEEE Access* **2023**, *11*, 62062–62076. [CrossRef]
22. Lee, E.; Lee, J.W.; No, J.S.; Kim, Y.S. Minimax approximation of sign function by composite polynomial for homomorphic comparison. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 3711–3727. [CrossRef]
23. Boura, C.; Gama, N.; Georgieva, M.; Jethchev, D. Chimera: Combining ring-lwe-based fully homomorphic encryption schemes. *J. Math. Cryptol.* **2020**, *14*, 316–338. [CrossRef]
24. Boura, C.; Chillotti, I.; Gama, N.; Jethchev, D.; Peceny, S.; Petric, A. High-precision privacy-preserving real-valued function evaluation. In *Proceedings of the International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 183–202.
25. Agrawal, R.; Chandrakasan, A.; Joshi, A. Heap: A fully homomorphic encryption accelerator with parallelized bootstrapping. In Proceedings of the 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA), Buenos Aires, Argentina, 29 June–3 July 2024; pp. 756–769.
26. Sharma, A.; Varma, A.K. Trigonometric interpolation. *Duke Math. J.* **1965**, *32*, 341–357. [CrossRef]
27. Halevi, S.; Shoup, V. Algorithms in helib. In *Proceedings of the Annual Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 554–571.
28. LeCun, Y.; Cortes, C.; Burges, C. MNIST Handwritten Digit Database. ATT Labs. 2010, Volume 2. Available online: <http://yann.lecun.com/exdb/mnist> (accessed on 26 July 2025).
29. Keras. 2015. Available online: <https://keras.io> (accessed on 4 August 2025).
30. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: <http://tensorflow.org> (accessed on 5 August 2025).
31. Dumezy, J. Sec-Ckks. 2025. Available online: <https://github.com/jdumezy/sec-ckks> (accessed on 18 August 2025).
32. Badawi, A.A.; Alexandru, A.; Bates, J.; Bergamaschi, F.; Cousins, D.B.; Erabelli, S.; Genise, N.; Halevi, S.; Hunt, H.; Kim, A.; et al. OpenFHE: Open-Source Fully Homomorphic Encryption Library. Cryptology ePrint Archive, Paper 2022/915. 2022. Available online: <https://eprint.iacr.org/2022/915> (accessed on 4 September 2025).
33. OpenMP Architecture Review Board. OpenMP Application Programming Interface, Version 5.1. 2020. Available online: <https://www.openmp.org/specifications/> (accessed on 22 August 2025).
34. Kim, A.; Papadimitriou, A.; Polyakov, Y. Approximate homomorphic encryption with reduced approximation error. In *Proceedings of the Cryptographers' Track at the RSA Conference*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 120–144.
35. Dumezy, J.; Alexandru, A.; Polyakov, Y.; Clet, P.E.; Chakraborty, O.; Boudguiga, A. Evaluating Larger Lookup Tables Using CKKS. Cryptology ePrint Archive 2025. Available online: <https://ia.cr/2025/1301> (accessed on 27 August 2025).
36. Hesthaven, J.S.; Gottlieb, D.I.; Gottlieb, S. *Spectral Methods for Time-Dependent Problems*; Cambridge University Press: Cambridge, UK, 2007; Volume 21.

**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.