Intelligent Beta-Based Polynomial Approximation of Activation Functions for a Robust Data Encryption System

Hanen Issaoui [1,*], Asma ElAdel [2], and Mourad Zaied [3]

[1] Research Team in Intelligent Machines (RTIM) Unit, University of Gabes, Gabes, Tunisia
[2] Higher Institute of Computer and Multimedia of Gabes, University of Gabes, Gabes, Tunisia
[3] National School of Engineers of Gabes, University of Gabes, Gabes, Tunisia
Email:hanen_issaoui78@yahoo.fr (H.I.); asma.eladel@ieee.org (A.E.); mourad.zaied@ieee.org (M.Z.)
*Corresponding author

***Abstract*—Deep neural network-based machine learning algorithms are widely used within different sectors and produce excellent results. However, their use requires access to private, often confidential, and sensitive information (financial, medical, etc). This requires precise measures and particular attention to data security and confidentiality. In this paper, we propose a new solution to this problem by integrating a proposed Convolutional Neural Network (CNN) model on encrypted data within the constraints of homomorphic encryption techniques. Specifically, we focus on the approximate activation functions ReLU, Sigmoid, and Tanh, which appear to be the key functions of CNNs. We start by developing new low-degree polynomials, which are essential for successful Homomorphic Encryption (HE). The activation functions will be replaced by these polynomials, which are based on the Beta function and its primitive. To make certain that the data is contained within a given range, the next step is to build a new CNN model using batch normalization. Finally, our methodology and the effectiveness of the proposed strategy are evaluated using Mnist and Cifar10. The experimental results support the proposed approach's efficiency.**

***Keywords*—Convolutional Neural Network (CNN), homomorphic encryption, activation function, beta function, batch normalization**

I. Introduction

As prospective developments in the field of artificial intelligence, machine learning techniques based on neural network architectures have attracted attention [1]. Neural networks seek to solve the problem of classification by correctly classifying new observations based on training data sets for which the classification is known [1–4]. Since these techniques require access to private data, several research studies have been carried out in recent years on how different machine learning algorithms, such as neural network learning algorithms, preserve data confidentiality [5, 6]. Modifying the activation function, encrypting incoming data, and running the homomorphic encryption model are all examples of network security methods [7]. The host does not have access to the keys required to decrypt the data, hence the encryption mechanism ensures its confidentiality [8]. These initiatives propose to encrypt both the model and the input data. Among these techniques, a technology known as CryptoDL [1] enables deep neural network algorithms to function on encrypted data and allows services to be given without divulging the study's data [1, 8]. Convolutional Neural Networks (CNN) and Homomorphic Encryption (HE) are the two primary parts of CryptoDL. The principle of homomorphic encryption was first introduced by Rivest *et al.* [8] in 1978. The HE method, on the other hand, employs encryption and operates on CNNs with encrypted data [9].

The complexity of the neural network, including the number of layers of each type and the complexity of the activation function, has considerable impact on the effectiveness of this method [10]. The benefit of homomorphic encryption is that it makes it possible to do computations on encrypted data without having access to the original data or the secret calculation key. Additionally, it is limited to adds and multiplications only. Additive and multiplicative homomorphic encryption methods, such as the Paillier cryptosystem [2] and the RSA or El Gamal encryption schemes [11], have been developed for many years. Later, other researchers created several additional HE methods [12]. The bulk of these encryption techniques do, however, have certain drawbacks. Some of them permit only one operation, as Paillier's encryption method [2]. Among the non-polynomial activation functions that cannot be assessed homomorphically are the ReLU, Sigmoid, and Tanh activation functions. But it's important to know whether a function can be roughly represented by a low-degree polynomial. In the solution by Xie *et al.* [13], low-degree polynomials were utilized to approximate these functions. Replace the convolutional neural network layer's neurons with a polynomial activation function. When selecting the activation function, it is important to take into account both the direct data

Manuscript received December 23, 2023; revised March 1, 2024; accepted March 13, 2024; published August 6, 2024.

transformation that the activation function does as well as its derivation, which will be utilized to modify the weight during retropropagation.

Our objective in this article is to construct a CNN that is more secure and resistant to attacks by employing an unidentified polynomial. The polynomial selected should be simple and close to the most commonly used activation functions. In addition, batch normalization has been evoked to considerably improve the learning capacity of neural networks. Ioffe and Szegedy [14] created the idea of batch normalization. To maintain a constant and predictable distribution at the input points of the activation function. The polynomial approximation only needs to be accurate over a limited and fixed range when such normalization is performed. It allows more stable data to be acquired because the data is at the same scale. This allows the neural network to train faster [15].

The main contributions of this work are as follows: Using the Beta function to approximate the ReLU, Sigmoid, and Tanh functions. The Beta function can be parameterized, which makes it possible to approximate the basic functions very closely [16]. We use the Beta function to approximate the ReLU activation function. To approximate the Sigmoid and the Tanh, we use primitives of the Beta function. The new polynomials obtained have been used as the activation function in the proposed model. We also apply batch normalizations to ensure that data from multiple sources fall within the same range. Next, we implemented the CNN model with the polynomial approximations obtained as activation functions. To analyze the performance of the modified models, training was performed on a dataset encrypted using homomorphic encryption. Comparisons are then made between the CNN model using the current ReLU, Sigmoid, and Tanh activation functions and the same model to which we have applied the approximation polynomials obtained. Finally, we present results for two datasets widely used in deep learning, Mnist and Cifar 10. The comparison is made to better appreciate the robustness of the method.

Our document is structured to provide an overall understanding of our approach and its implications. Section II provides an overview and background information. Then, in Section III, we present the Beta function for approximating activation functions and its application in our method. Section IV describes our proposed approach. Experimental results, which are based on the Mnist and Cifar 10 datasets are delineated in Section V. Finally, Section VI then presents a conclusion summarising our results and future work directions to explore potential avenues for further research.

II. Overview and Background Information

In this section, we describe the methods used to develop our contribution. We give a general overview of homomorphic encryption, activation functions, polynomial approximation, and batch normalization.

A. Homomorphic Encryption

Homomorphic encryption is a robust protection barrier for data confidentiality. It is also an advanced encryption technique that allows operations to be carried out on encrypted data without first decrypting it. This unique property paves the way for the secure handling of sensitive data, even in environments where confidentiality is crucial. By integrating homomorphic encryption into the deep learning process, researchers can ensure that sensitive information remains encrypted throughout the learning and inference process, offering better protection against privacy breaches [17]. Homomorphic Encryption (HE) is a leading method for protecting and maintaining data confidentiality. Data can be processed even when it is encrypted thanks to this type of encryption. Homomorphic encryption refers to a class of cryptographic methods that meet the homomorphic property. In mathematics, a transformation that preserves the original structure is known as a homomorphism [3]. The principle of homomorphic encryption is illustrated in the Fig. 1.

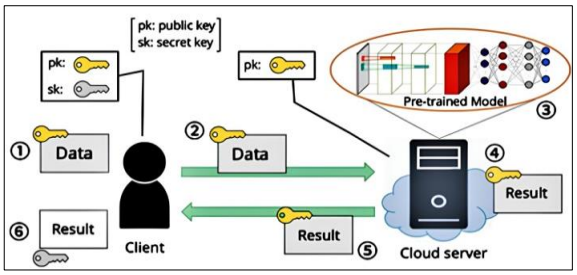


Fig. 1. Outline of inference on HE [18].

Some operations, such as addition and multiplication, can be performed directly on the encrypted data to ensure that the result of decryption is the same as if the operation had been performed on the original data. This is a special feature that conventional encryption methods do not take into account [14]. Many mathematical models deal with addition, multiplication, or both [12, 16, 17]. It seems strange that the data must first be decoded before accurate calculations can be made using traditional procedures. The confidentiality of the data is jeopardized as soon as decryption is complete. It is therefore essential to know the most effective approach for HE to overcome this contradiction. The fact that it has taken the cryptographic community over 30 years to develop a solution shows just how difficult the answer is. Craig Gentry presented the first answer in 2009, although it was purely theoretical [18]. To make these ideas more applicable, several research projects have been carried out since then, for example [19–23].

B. Activation Functions

Each convolutional layer in a neural network is typically succeeded by an activation layer, which serves as a non-linear function. Activation functions are essential for introducing non-linearity between consecutive layers in a network. They help in cascading linearity between layers if the activation function relates to cached layers of the network and in constraining the values of a layer within a specified range if the activation function is applied to the final layer [24, 25]. In practical applications, widely-used activation functions such as ReLU, Sigmoid, and Tanh are commonly employed [1, 26]. Each activation function

operates on a single number and performs a predetermined mathematical operation. Below, we list the activation functions discussed in this paper, along with their corresponding mathematical expressions.

- Rectified Linear Unit (ReLU). ReLU replaces negative values with zero, resulting in a simple yet effective activation function. The ReLU function is defined as:

$$\text{ReLU}(x) = \max(0, x) \quad (1)$$

- Sigmoid. Sigmoid squashes the input values between 0 and 1, making it suitable for binary classification tasks. The sigmoid function is defined as:

$$\sigma(x) = 1/(1+e^{-x}) \quad (2)$$

- Tanh (Hyperbolic Tangent). Tanh squashes the input values between −1 and 1, allowing a better representation of negative values compared to sigmoid. The hyperbolic tangent function is defined as:

$$\text{Tanh}(x) = \frac{2}{1+e^{-2x}} - 1 \quad (3)$$

These activation functions are fundamental building blocks in neural networks, each offering unique properties and suitability for different tasks. Our contribution in this paper is based on the approximation of polynomials for these functions, which only use addition and multiplication operations since we cannot compute them on encrypted values [25]. We focus on neural activity because our main goal is to adopt a CNN capable of operating with, HE restrictions. In addition, the construction of activation functions in neurons is prohibited by the impossibility of using encrypted data (addition and multiplication). So, to act on encrypted data, we need to discover equivalent replacement functions. Polynomials that can be implemented using simple addition and multiplication operations can be used to approximate the majority of functions, including the activation functions used in CNNs. We study polynomial approximations of the common CNN, ReLU, Sigmoid, and Tanh activation functions, and choose the one that most closely approximates each activation function. To find the best approximation, we used the Beta function to approximate the ReLU activation function. To approximate the Sigmoid and Tanh, we used the primitives of the Beta function for the hidden layers of the CNN. When choosing the optimal activation function, we take into account the direct transformation that the activation function applies to the data, as well as its derivative, which will be used to modify the weights during backpropagation [26].

C. Polynomial Approximation

For many applications, including signal processing, multimedia, and neural networks, an accurate approximation of the activation functions is required [12]. In general, we can approximate activation functions with a variety of polynomials. Higher-level polynomials provide a more accurate approximation and, when they replace the activation function in a CNN, they improve the performance of the model created. However, when operations are performed on encrypted data, higher-degree polynomials result in very slow calculations. A compromise must therefore be found between the performance of the model and the degree of polynomial approximation [1]. We propose a solution for the polynomial approximation of the ReLU, Sigmoid, and Tanh activation functions. Then, using these polynomials, we created a CNN model and compared the results with those of models using the original activation functions. In the literature, several studies have been carried out in this context. Chabanne *et al.* [5] used the Taylor series to approximate the sigmoid function using a polynomial of degree 2. According to H. Ehsan in [1], the sigmoid function has also been approximated by polynomials. In their experience, 99.73% of the values lie in the interval [−3, 3]. Cheon *et al.* [26] proposed an approximation method that minimized the mean square error. They approximated the sigmoid function on the interval [−8, 8] with different degrees (3 and 7) [27]. The choice of an appropriate approximate polynomial can affect the complexity and depth of the activation layers. According to Xie *et al.* [13], the technique is to approximate these functions using low-degree polynomials. Numerous ReLU function approximations are discussed in the literature [13], including, for example, the Taylor series and the Chebyshev polynomial [3, 6, 22]. In this paper, we present a formal approach for approximating ReLU, Sigmoid, and Tanh functions based on polynomial functions. Although the activation function is crucial for the learning phase, it is also well known that it opens the door to attacks on the network.

Furthermore, the use of HE in our case is incompatible with non-polynomial activation functions. The ReLU, Sigmoid, and Tang functions belong to the category of non-polynomial functions. This is why we suggest using a new activation function.

D. Batch Normalization

Another improvement we are making to the proposed model is to incorporate batch normalization between framed layers. The aim is to ensure that the data is contained within a specific interval. This makes the model reliable and efficient [6, 26]. Loffe and Szegedy [14] introduced the concept of batch normalization as a regularisation strategy in 2015. Batch normalization is used in the context of deep learning [12, 27], it is one of the most effective regularisation techniques in Deep Learning, to improve network performance. Batch normalization is a preprocessing technique that allows data from many sources to fit into a single range. Normalizing data before learning can increase the effectiveness and efficiency of the process [25]. We include batch normalization functionality between neural network layers [28–30]. Before sending it to the next layer, it takes the output of the previous layer and normalizes it.

III. The Beta Function for Approximation of Activation Functions

In our work, we used the beta function as the basis for our approximation [31]. It was used to approximate the ReLU activation function. The Beta distribution is a type of probability distribution that represents all possible outcomes of the data set. The general formula for the Beta function is given by Eq. (4).

$$Beta(x) = \left(\frac{x-a}{c-a}\right)^{p} \times \left(\frac{b-x}{b-c}\right)^{q} \quad (4)$$

where:
$c = \frac{ap+bq}{p+q}$
p, q: natural numbers > 0, the shape parameters.
a, b: upper and lower bounds where $a \leq x \leq b$.

To approximate the Sigmoid and Tanh activation function, we used the primitive of the Beta function.
The formula for the primitive of Beta is given by the following Eq. (5):

$$\left[\sum_{k=1}^{q+1} \frac{q!}{(q-k+1)!} \times \frac{p!}{(p+k)!} \times (x-a)^{p+k} \times (b-x)^{q-k+1}\right] \times \frac{1}{(c-a)^{p} \times (b-c)^{q}} \quad (5)$$

The parameters a, b, p, and q of the Beta function and its primitives are adjusted so that it is as close as possible to the approximation functions studied in this paper. To adjust the parameters, we used the gradient descent algorithm. The principle is described in the following algorithm:

Algorithm: Adjustment of the parameters a, b, p, and q of the Beta function using the gradient descent algorithm.

Result: Optimized parameters a, b, p, q
//Initialize parameters a, b, p, q with random values
$a \leftarrow$random_value
$b \leftarrow$random_value
$p \leftarrow$random_value
$q \leftarrow$random_value
//Initialize learning rate alpha
$\alpha \leftarrow 0.01$
//Initialize maximum number of iterations max_iter
max_iter$\leftarrow$1000
//Initialize previous error with a high value
preferred$\leftarrow$10000
//Initialize iteration counter
iter_count$\leftarrow$0
//Gradient Descent
while (iter_count ≤ max_iter) and (error ≥ tolerance) **do**
//Compute current error
//Check for convergence
If (current error < tolerance) **then**
Exit the loop
else
//Update Parameters
For each parameter (a, b, p, q) **do**
//Compute current error using the previously defined error function
//Update the parameter based on the gradient and α:
$a \leftarrow a - \alpha$*gradient(a)
$b \leftarrow b - \alpha$*gradient(b)
$p \leftarrow p - \alpha$*gradient(p)
$q \leftarrow q - \alpha$*gradient(q)
end
//Handle learning rate α adjustment
//Increment iter_count
If (current error > previous error) **then**
Reduce learning rate α
Update previous error
end
end
end
Return the optimized parameters a, b, p, q

IV. Proposed Approach

Access to private data is critical, making CNN vulnerable to attack. However, we propose a method for securing the model that consists of modifying the activation function without causing a malfunction of the neuron's internal activity [32]. To improve the accuracy of classification by a CNN network on data encrypted using HE homomorphic encryption, we approximated the ReLU activation function by a polynomial using the Beta function, and for the Sigmoid and Tanh activation functions, we used the Beta primitive to approximate them. The various steps are illustrated in Fig. 2.

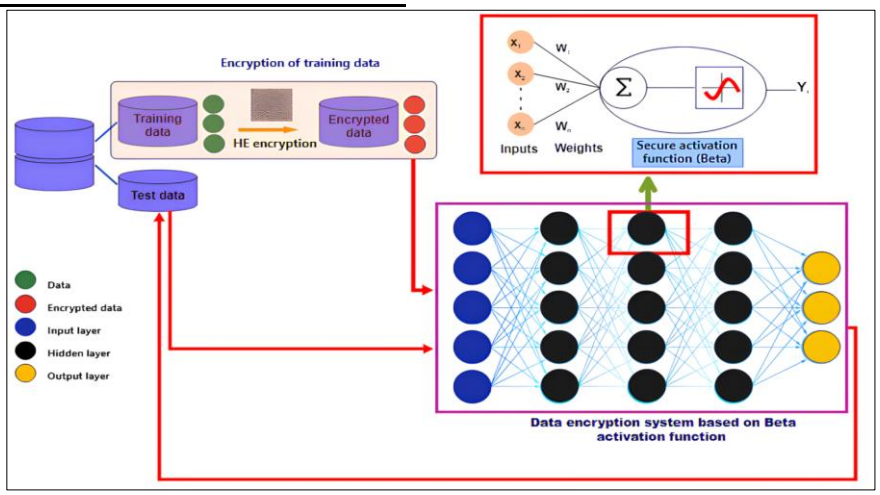


Fig. 2. CNN using Beta activation functions approximations on homomorphic encryption.

Neural networks require the use of an activation function at the output of each neuron [15]. As a first step, we propose to develop a new low-degree polynomial to be applied to the different convolutional layers of the CNN network [16–18]. As this polynomial is unknown, so it will be difficult to attack the network.

Our approach is based on the use of the Beta function Eq. (4) and its corresponding primitive Eq. (5). These functions can be parametrically tuned, allowing their parameters (a, b, p, q) to be modified to obtain the most optimal approximation in each case.

By adjusting the parameter values, we achieve optimal approximations for each activation function. The approximation expressions for each activation function, based on the optimal values of the adjusted parameters, along with the resulting approximation polynomials, are presented below.

1) Approximation of the ReLU activation function

The expression for the approximation of the ReLU activation function with the Beta function is given by the following expression:

$$0.74 \times \text{Beta}\left(\frac{(x-0.99)}{6.8}\right). \quad (6)$$

The optimal values of the parameters:

$$(a, b, c) = (-0.7, -5, -3.57),$$
$$(p, q) = (1, 2).$$

The mathematical expression of the approximation polynomial (Polynomial 1) of the ReLU activation function with the Beta function, obtained by developing Eq. (6) according to the parameters previously introduced, is represented by Eq. (7) below:

Polynomial 1

$$25\times10000\times\left(-\frac{5\times\left(x-\frac{99}{100}\right)}{34}-5\right)^2\times\left(\frac{5\times\left(x-\frac{99}{100}\right)}{34}-\frac{9}{10}\right)\Bigg/1819961 \quad (7)$$

The approximation of the ReLU activation function based on the Beta function according to Polynomial 1 is illustrated in Fig. 3.

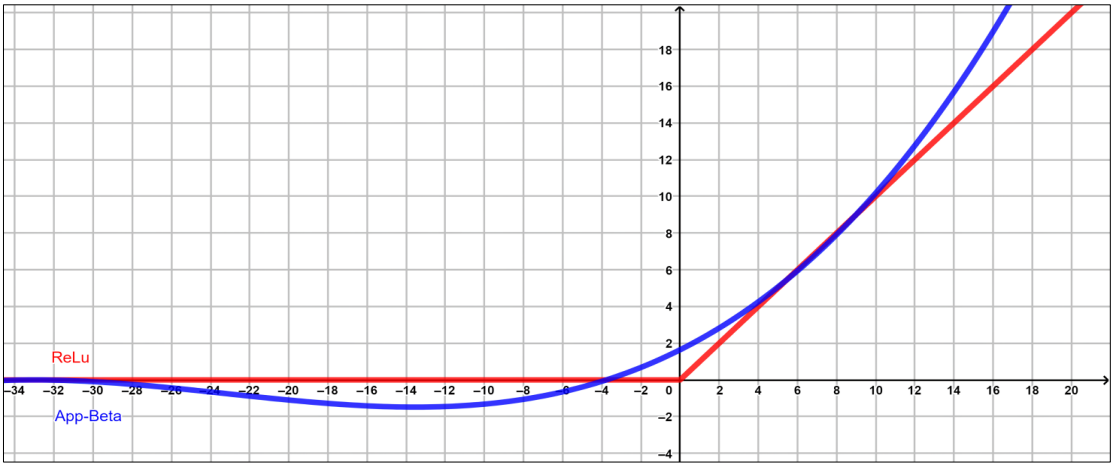


Fig. 3. Polynomial approximation of the ReLU activation function using Polynomial 1 (Eq. (7)).

2) Approximation of the Sigmoid activation function

To approximate the Sigmoid activation functions, we relied on the primitive of the Beta function. The approximation expression with the primitive of the Beta function is given by the following expression:

$$0.5 + 1.125 \times \text{Primitive\_Beta}\,(0.6x) \quad (8)$$

The optimal values of the parameters:

$$(a, b, c) = (-0.9, 0.9, 0),$$
$$(p, q) = (2, 2).$$

The mathematical expression of the approximation polynomial (Polynomial 2) of the Sigmoid activation function with the primitive of the Beta function, obtained by developing Eq. (8) according to the parameters previously introduced, is represented by Eq. (9) below:

Polynomial 2

$$\frac{67\times\left(\frac{125\times x^5}{64}-\frac{675\times x^3}{8}+\frac{6561\times x}{4}\right)}{437400}+\frac{1}{2} \quad (9)$$

The approximation of the Sigmoid activation function based on the primitive of the Beta function according to Polynomial 2 is illustrated in Fig. 4.

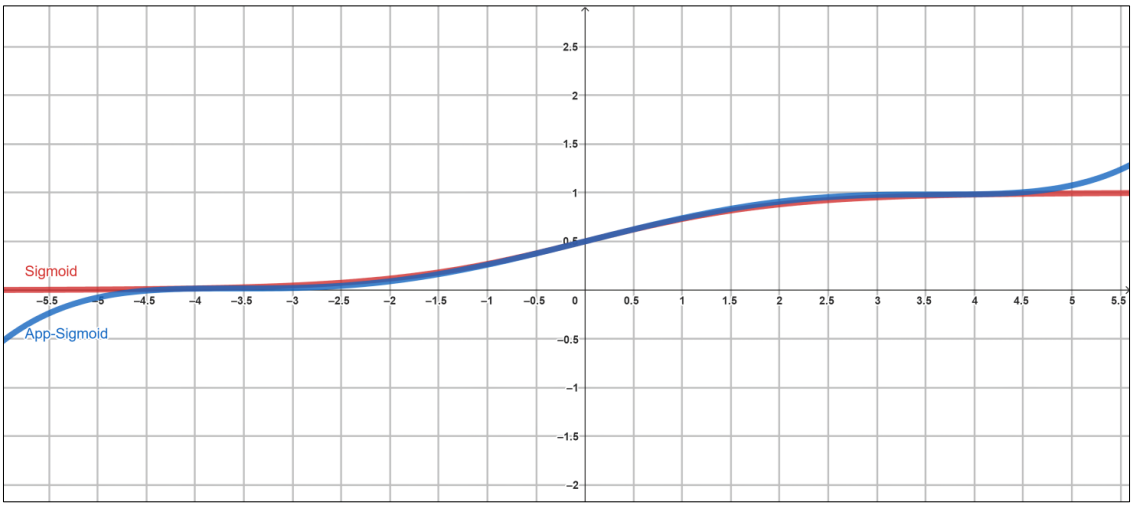


Fig. 4. Polynomial approximation of the Sigmoid activation function using Polynomial 2 (Eq. (9)).

3) Approximation of the Tanh activation function

We use the primitive of the Beta function to approximate the Tanh activation function. The approximation expression with the primitive of the Beta function is given by the following expression:

$$1.5 \times \text{Primitive\_Beta}\,(0.6x). \tag{10}$$

The optimal values of the parameters:

$$(a, b, c) = (-0.9, 0.9, 0),$$
$$(p, q) = (2, 2).$$

The mathematical expression of the approximation polynomial (Polynomial 3) of the Sigmoid activation function with the primitive of the Beta function, obtained by developing Eq. (10) according to the parameters previously introduced, is represented by Eq. (11) below:

Polynomial 3

$$\frac{\frac{125\times x^5}{64} - \frac{675\times x^3}{8} + \frac{6561\times x}{4}}{4374} \tag{11}$$

Polynomial 3 represents the function of the approximation obtained and Fig. 5 shows the shape of the curve.

In this paper, we first use the polynomials obtained as activation functions in the proposed model. The activation functions ReLU, Sigmoid, and Tanh are replaced by the polynomials given by Eqs. (7), (10) and (11), respectively. Secondly, the addition of batch normalization between the CNN layers improves the efficiency and performance of the network. This Batch Normalization (BN) was used to ensure that the data fell within the defined range.

The new model was then trained on an encrypted training set using homomorphic encryption. Finally, we evaluate our method using the Mnist and Cifar 10 datasets, two datasets frequently used in deep learning.

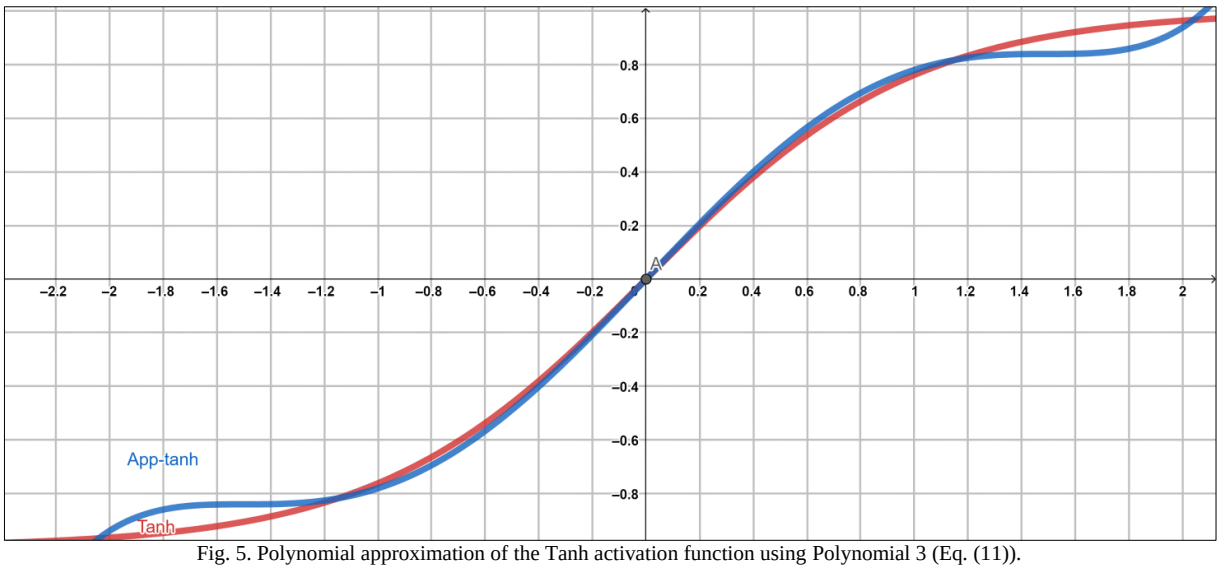


Fig. 5. Polynomial approximation of the Tanh activation function using Polynomial 3 (Eq. (11)).

V. Experimental Results

We have conducted experiments on the Mnist [33] and Cifar-10 [34] datasets. Mnist is a collection of 60,000 images depicting handwritten digits. Each image is represented as a 28×28 pixel array, where the gray level of each pixel ranges from 0 to 255. In our study, we utilized the training portion of the Mnist dataset, which consists of 50,000 images, to train a neural network. The remaining 10,000 images were employed for testing purposes. The Cifar-10 dataset comprises 60,000 images, with 50,000 images designated for training and 10,000 for testing. Each image in the Cifar-10 dataset is an RGB image consisting of 3×32×32 pixels. Additionally, each image is assigned a label corresponding to one of ten classes.

A first comparison is made using our proposed CNN model architecture. This comparison is made using the original activation functions on the one hand, and the approximation polynomials proposed using our approach on the other. The results of the evaluation of Mnist are presented in Table I.

TABLE I. Performance of the Proposed CNN Using Activation Functions and Their Polynomial Approximations Using the Mnist Dataset

Activation function	Original Model	Model with Our Approach (MNIST)
ReLU	98.48%	98.86%
Sigmoid	98.67%	98.78%
Tanh	98.01%	97.76%

We examined the impact of polynomial approximations of activation functions on the MNIST dataset. We observed that the polynomial approximation of the ReLU activation function, obtained from the beta function, produced the best results when integrated into the layers of our proposed architecture. These results indicate that this combination offers a significant performance improvement on MNIST. In addition, the use of the polynomial approximation of the Sigmoid activation function, based on the beta primitive, also led to a slight improvement in performance. However, the performance of the polynomial approximation of the Tanh activation function, also derived from the beta primitive, was slightly lower.

To further test our approach, another evaluation was carried out on the Cifar-10 dataset. The results are presented in Table II.

TABLE II. Performance of the Proposed CNN Using Activation Functions and Their Polynomial Approximations Using the Cifar-10 Dataset

Activation function	Original Model	Model with Our Approach (Cifar 10)
ReLU	97.87%	97.94%
Sigmoid	96.32%	96.13%
Tanh	95.22%	93.06%

Using Cifar10, we find that the approximation polynomials for the ReLU and Sigmoid activation functions obtained perform better when used in the layers of the CNN architecture we propose. Whereas, the approximation polynomial for the Tanh activation function performs worse.

In the experimental part, we use the Mnist and Cifar-10 datasets encrypted by homomorphic encryption to evaluate the new model, and the results indicate that the classification accuracy of the encrypted Mnist dataset is better than that of the Cifar-10 dataset. The results show that the classification accuracy of the Mnist dataset can reach 98.86% for the ReLU function, which is interesting because the use of unknown polynomials is an important aspect of the security and protection of machine learning models. The experimental results demonstrate the merits of the approach, which preserves privacy in predictions.

To better assess the effectiveness of our approach, a second comparison was carried out. Table III presents the results of the comparison between the results presented in the literature and our own. The polynomials presented concern the approximation of the ReLU function on the Mnist dataset, each using its polynomial [35, 36]. The values obtained are given in Table III.

Table III shows the different results obtained for the Mnist dataset using different approaches. At the end of the evaluation, we obtained a classification accuracy of 98.86%, which is very close to the CryproDL (99.52%) and CryptoNets (98.95%) approaches. On the other hand, our accuracy is better than the three methods [19] (98.82%), [37] (93.40%) and [38] (98.44%).

TABLE III. Comparison of the Effectiveness of Our Approach with Other Methods Proposed Using the Mnist Data Set and Polynomial Approximations of the Related Activation Function

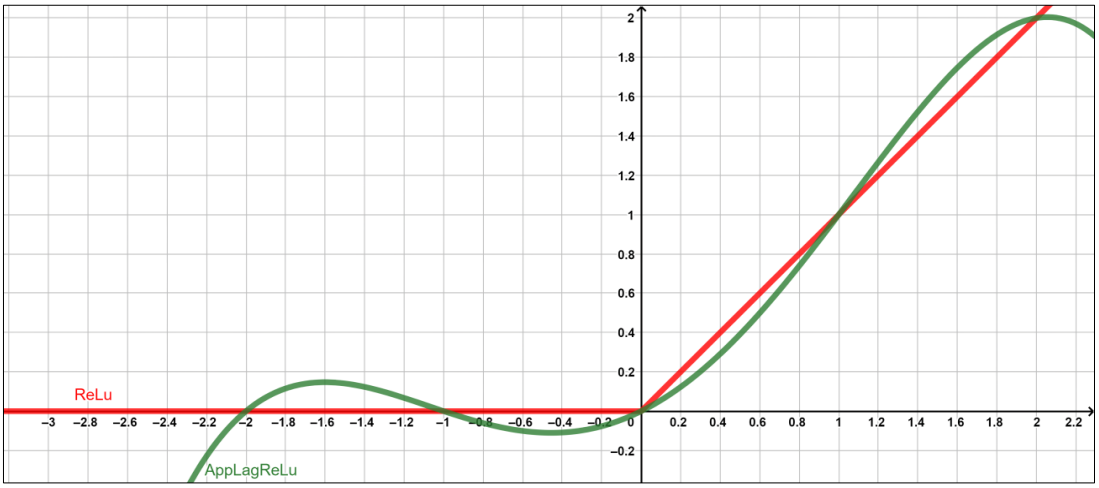
Proposed method	Accuracy
CryptoDL [1]	99.52%
CryptoNets [3]	98.95%
Ishiyama *et al.* [19]	98.82%
SecureML [39]	93.40%
Ana *et al.* [40]	98.44%
Our approach	98.86%

TABLE IV. The Approximation Polynomials Obtained by Several Points Equal to 5 and in the Range [−2, 2]

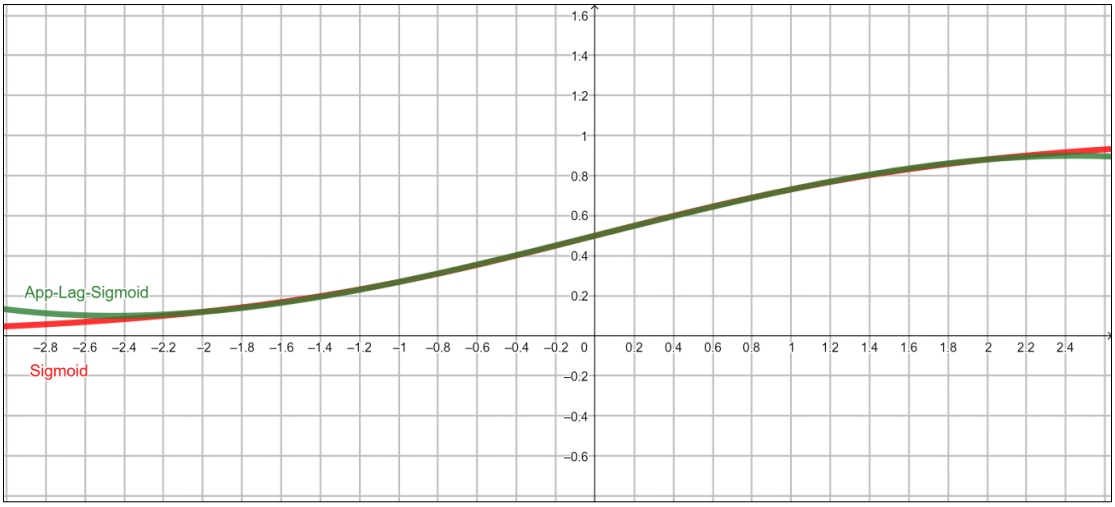
Activation function	Polynomial approximations using Lagrange interpolation
ReLU	$-0.083333333333333x^4 + 0.583333333333333x^2 + 0.5x$
Sigmoid	$0.000021764439479x^4 - 0.013605563373428x^3 + 0.000043182201679x^2 + 0.244733041370182x + 0.5$
Tanh	$0.000396751423111x^4 - 0.094230782415271x^3 - 0.001587005692445x^2 + 0.858936919698991x$

To further test our approach, a third comparison is carried out. We approximate the ReLU, Sigmoid, and Tanh activation functions using Lagrange interpolations [33, 34] and then compare the results obtained with the polynomials we have already proposed. The approximation polynomials obtained with several points equal to 5 and in the interval [−2, 2] are presented in Table IV.

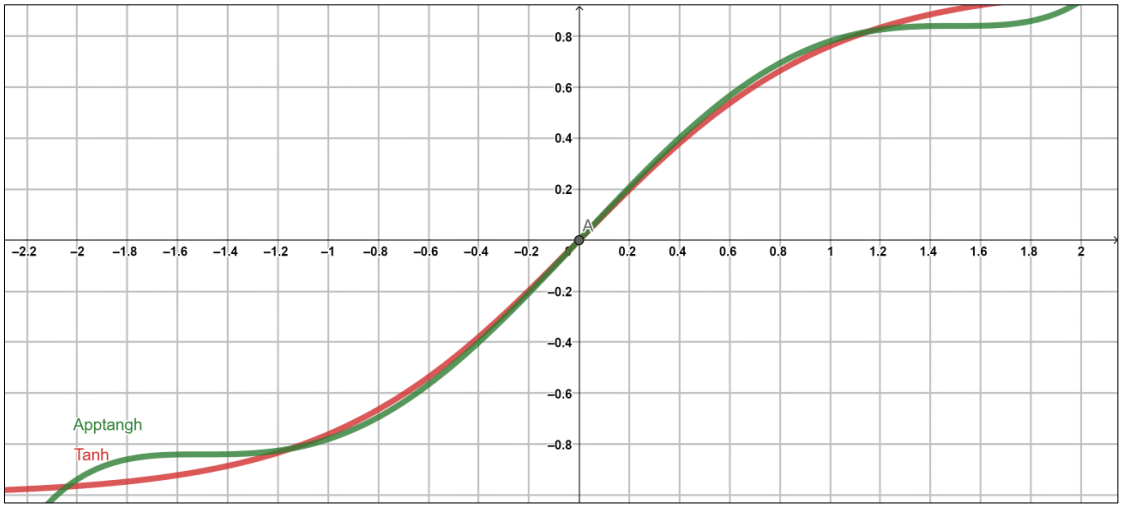
Fig. 6(a)–(c) shows successively the ReLU, Sigmoid, and Tanh activation functions and their polynomial approximations of our approach, as well as the approximations obtained by Lagrange interpolation in the interval [–2, 2].



(a)



(b)



(c)

Fig. 6. The ReLU, Sigmoid, and Tanh activation functions and their polynomial approximations, (a) Our proposed polynomial approximation (Polynomial 1) and the Lagrange interpolation approximation of the ReLU activation function, (b) Our proposed polynomial approximation (Polynomial 2) and the Lagrange interpolation approximation of the Sigmoid activation function, (c) Our proposed polynomial approximation (Polynomial 3) and the Lagrange interpolation approximation of the Tanh activation function.

We implement the proposed new model by changing, each time, the ReLU, Sigmoid, and Tanh activation functions by polynomial approximations obtained using Lagrange interpolation. The comparison is made using the mnist dataset (Table V).

TABLE V. THE CNN'S PERFORMANCE USING A POLYNOMIAL APPROXIMATION WITH LAGRANGE INTERPOLATION OF THE ACTIVATION FUNCTIONS ON MNIST DATA SET

Activation Function	Approximation Lagrange	Model with Our approach
ReLU	98.18%	98.86%
Sigmoid	97.85%	98.78%
Tanh	97.92%	97.76%

On the Mnist dataset, we find that the ReLUactivation function approximation polynomial, obtained using the Beta function, gives the best results when used in the layers of our proposed architecture. On the other hand, the approximation polynomial of the Sigmoid activation function obtained using the Beta primitive, slightly improves performance compared to the polynomial obtained by approximation using Lagrange interpolation. For the polynomial approximating the Tanh function, also obtained using the Beta primitive, performance is slightly lower.

VI. CONCLUSION AND FUTURE WORK

In this article, we proposed a new strategy to enhance the privacy and security of data in neural networks. We introduced innovative approaches to approximate the ReLU activation function using the Beta function and its primitive for Sigmoid and Tanh approximations. This method offers increased flexibility due to the parametrizability of the Beta function, allowing for better approximation and making the model more resilient to attacks. Concurrently, we optimized the model's performance by adding additional layers of batch normalization in a CNN. The utilization of the homomorphic encryption algorithm on encrypted data was also implemented to reinforce the confidentiality of processed information. To evaluate our approach, we utilized the Mnist and Cifar 10 databases, demonstrating that our method yields accurate, precise, and scalable predictions. The experimental results validate the merits of our approach. By combining the use of unknown polynomials with the flexibility of the Beta function, we significantly reduce the likelihood of attacks and enhance model confidentiality. Integrating these unknown polynomial approximations into activation functions adds an extra layer of security, particularly crucial in sensitive domains where data protection is paramount. In conclusion, our approach not only improves model performance but also strengthens security by providing additional defense against attacks. The flexibility of the Beta function used as an approximation base allows for precise adaptation, constituting a major asset for achieving maximal approximation.

For future work, a deeper exploration of the possibilities offered by the Beta function in approximating activation functions could be pursued. Additionally, investigating the impact of different normalization and encryption methods on model performance and security would be worthwhile. Moreover, extending this approach to other types of neural networks and more complex datasets could open up new avenues in the field of data protection and privacy in machine learning.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Hanen Issaoui proposed the article's idea, implemented it, conducted experiments, wrote and revised the manuscript, designed the experiments and analyzed and interpreted the results; Asma ElAdel contributed to the methodology's structure, participated in article writing, revised the manuscript, helped refine the research results, and contributed to the analysis and interpretation of the data; Mourad Zaie led the research, assisted in refining the research results, analyzed and interpreted the data, approved its content, and provided guidance; All authors had approved the final version.

REFERENCES

[1] E. Hesamifard, H. Takabi, and M. Ghasemi, "CryptoDL: Deep neural networks over encrypted data," arXiv preprint arXiv: 1711. 05189, 2017.

[2] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," *Springer eBooks*, 2007, pp. 223–238. doi: 10.1007/3-540-48910-x_16

[3] N. Dowlin, R. G. Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. International Conference on Machine Learning*, Jun. 2016, pp. 201–210.

[4] N. Chaibi, A. ElAdel, and M. Zaied, "SR-Net: A super-resolution image based on DWT and DCNN," *Hybrid Intelligent Systems. HIS 2022*, pp. 291–301, Jan. 2022, doi: 10.2139/ssrn.4113833.

[5] H. Chabanne, A. De Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving classification on deep neural network," *IACR Cryptologye Print Archive*, vol. 2017, 35, Jan. 2017.

[6] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," *Lecture Notes in Computer Science*, 1999, pp. 319–345. doi: 10.1007/3-540-46805-6_19

[7] H. Issaoui, A. E. Adel, and M. Zaied, "B-CNN: Beta deep convolutional neural network over encrypted data," in *Proc. International Conference on Machine Vision*, Jun. 2023. doi: 10.1117/12.2679393

[8] R. L. Rivest and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations on Secure Computation, Academia Press*, pp. 1–11, Jan. 1978.

[9] E. Hesamifard, H. Takabi, and M. Ghasemi, "Deep neural networks classification over encrypted data," in *Proc. Ninth ACM Conference on Data and Application Security and Privacy*, 2019, pp. 97–108. doi: 10.1145/3292006.3300044

[10] N. Chaibi, A. ElAdel, and M. Zaied, "Deep convolutional neural network based on WaVeLet transform for super image resolution," *Advances in Intelligent Systems and Computing*, 2021, pp. 114–123. doi: 10.1007/978-3-030-73050-5_12.

[11] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985. doi: 10.1109/tit.1985.1057074

[12] P. B. Mathayo and D. K. Kang, "Beta and alpha regularizers of mish activation functions for machine learning applications in deep neural networks," *International Journal of Internet, Broadcasting, and Communication*, vol. 14, no. 1, pp. 136–141, 2022.

[13] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, and M. Naehrig, "Crypto-nets: Neural networks over encrypted data," arXiv preprint, arXiv:1412.6181, Dec. 2014.
[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Machine Learning Research*, 2015, vol. 37.
[15] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978. doi: 10.1145/359340.359342
[16] D. J. Myers and R. Hutchinson, "Efficient implementation of piecewise linear activation function for digital VLSI neural networks," *Electronics Letters*, vol. 25, no. 24, 1662, Jan. 1989. doi: 10.1049/el:19891114
[17] H. Akouaydi, S. Njah, W. Ouarda, A. Samet, M. Zaied, and A. M. Alimi, "Convolutional neural networks for online arabic characters recognition with beta-elliptic knowledge domain," in *Proc. International Conference on Document Analysis and Recognition Workshops (ICDARW)*, vol. 1, Sep. 2019. doi: 10.1109/icdarw.2019.50114
[18] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated Learning: challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, May 2020. doi: 10.1109/msp.2020.2975749
[19] T. Ishiyama, T. Suzuki, and H. Yamana, "Highly accurate CNN inference using approximate activation functions over homomorphic encryption," in *Proc. IEEE International Conference on Big Data*, 2020, pp. 10–13. doi: 10.1109/bigdata50022.2020.9378372
[20] D. Boneh, E. J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," *Lecture Notes in Computer Science*, pp. 325–341, 2005. doi: 10.1007/978-3-540-30576-7_18
[21] Everything You Should Know About Dropouts and Batch Normalization. [Online]. Available: https://analyticsindiamag.com/everything-you-should-know-about-dropouts-and-batchnormalization-in-cnn
[22] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," *Lecture Notes in Computer Science*, pp. 75–92, 2013. doi: 10.1007/978-3-642-40041-4_5
[23] A. A. Badawi *et al.*, "Towards the AlexNet moment for homomorphic encryption: HCNN, the first homomorphic CNN on encrypted data with GPUs," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1330–1343, Jul. 2021. doi: 10.1109/tetc.2020.3014636
[24] N. Chaibi, N. B. Aoun, A. ElAdel, and M. Zaied, "Image super resolution boosting using beta wavelet," *Signal, Image and Video Processing*, Dec. 2023. doi: 10.1007/s11760-023-02887-3
[25] S. Kılıçarslan, K. Adem, and M. Çelik, "An overview of the activation functions used in deep learning algorithms," *Journal of New Results in Science*, vol. 10, no. 3, pp. 75–88, Dec. 2021. doi: 10.54187/jnrs.1011739
[26] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *Electronic Colloquium on Computational Complexity*, pp 1–36, Jan. 2012. doi: 10.1145/2090236.2090262
[27] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," *Lecture Notes in Computer Science*, pp. 409–437, 2017. doi: 10.1007/978-3-319-70694-8_15
[28] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes," *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–35, Jul. 2018. doi: 10.1145/3214303
[29] A. ElAdel, M. Zaied, and C. B. Amar, "Fast DCNN based on FWT, intelligent dropout and layer skipping for image retrieval," *Neural Networks (Print)*, vol. 95, pp. 10–18, Nov. 2017. doi: 10.1016/j.neunet.2017.07.015
[30] A. ElAdel, R. Ejbali, M. Zaied, and C. B. Amar, "A new semantic approach for CBIR based on beta Wavelet network modeling shape refined by texture and color features," *Lecture Notes in* Computer *Science*, pp. 378–385, 2014. doi: 10.1007/978-3-319-10840-7_46
[31] L. J. M. Aslett, P. M. Esperança, and C. Holmes, "A review of homomorphic encryption and software tools for encrypted statistical machine learning," arXiv preprint, arXiv:1508.06574, 2015.
[32] B. Radhia, "Approximation with activation functions and applications," *African Journal of Research in Computer Science and Applied Mathematics*, vol. 32, 2021.
[33] J.P. Demailly, "Numerical analysis and differential equations," *EDP Sciences eBooks*, vol.1, 2022.
[34] M. Vlček, "Chebyshev polynomial approximation for activation sigmoid function," *Neural Network World*, vol. 22, no. 4, pp. 387–393, Jan. 2012. doi: 10.14311/nnw.2012.22.023
[35] C. F. Dunkl and Y. Xu, "Orthogonal polynomials of several variables," arXiv preprint, arXiv: 1701.02709, 2014. doi: 10.1017/cbo9781107786134
[36] W. M. Fatihia, A. Fariza, and T. Karlita, "CNN with batch normalization adjustment for offline hand-written signature genuine verification," *JOIV: International Journal on Informatics Visualization*, vol. 7, no. 1, p. 200, Feb. 2023, doi: 10.30630/joiv.7.1.1443.
[37] Y. L. Cun *and al.*, "Comparison of learning algorithms for handwritten digit recognition," *Computer Science*, pp. 53–60, Jan. 1995.
[38] V. Thakkar, S. Tewary, and C. Chakraborty, "Batch normalization in convolutional neural networks—A comparative study with CIFAR-10 data," in *Proc. Fifth International Conference on Emerging Applications of Information Technology*, vol. 2, Jan. 2018. doi: 10.1109/eait.2018.8470438
[39] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symposium on Security and Privacy*, pp. 22–27, May 2017. doi: 10.1109/sp.2017.12
[40] A. Stanojevic, E. Eleftheriou, G. Cherubini, S. Woźniak, A. Pantazi, and W. Gerstner, "Approximating ReLU networks by single-spike computation," in *Proc. 2022 IEEE International Conference on Image Processing (ICIP)*, Oct. 2022, vol. 16. doi: 10.1109/icip46576.2022.9897692