Efficient Homomorphic Evaluation on Large Intervals

Jung Hee Cheon[1,2], Wootae Kim[1], and Jai Hyun Park(✉)[1]

[1] Department of Mathematical Sciences, Seoul National University, South Korea
{jhcheon, wo01416, jhyunp}@snu.ac.kr
[2] Crypto Lab Inc., South Korea

Abstract. Homomorphic encryption (HE) is being widely used for privacy-preserving computation. Since HE schemes only support polynomial operations, it is prevalent to use polynomial approximations of non-polynomial functions. We cannot monitor the intermediate values during the homomorphic evaluation; as a consequence, we should utilize polynomial approximations with sufficiently large approximation intervals to prevent the failure of the evaluation. However, the large approximation interval potentially accompanies computational overheads, and it is a serious bottleneck of HE application on real-world data.

In this work, we introduce domain extension polynomials (DEPs) that extend the domain interval of functions by a factor of k while preserving the feature of the original function on its original domain interval. By repeatedly iterating the domain-extension process with DEPs, we can extend with $O(\log K)$ operations the domain of a given function by a factor of K while the feature of the original function is preserved in its original domain interval.

By using DEPs, we can efficiently evaluate in an encrypted state a function that converges at infinities, i.e., $\lim_{x\to\infty} f(x)$ and $\lim_{x\to-\infty} f(x)$ exist in $\mathbb{R}$. To uniformly approximate the function on $[-R, R]$, our method exploits $O(\log R)$ operations and $O(1)$ memory. This is more efficient than the previous approach, the minimax approximation and Paterson-Stockmeyer algorithm, which uses $\Omega(\sqrt{R})$ multiplications and $\Omega(\sqrt{R})$ memory for the evaluation. As another application of DEPs, we also suggest a method to manage the risky outliers from a large interval $[-R, R]$ by using $O(\log R)$ additional multiplications.

As a real-world application, we trained the logistic regression classifier on large public datasets in an encrypted state by using our method. We exploit our method to the evaluation of the logistic function on large intervals, e.g., $[-7683, 7683]$.

1 Introduction

There have been many attempts for privacy-preserving delegation of computation these days. One of the most popular solutions for privacy-preserving computation is homomorphic encryption (HE) which supports several operations between ciphertexts without any decryption process. The privacy-preserving

delegation of machine learning (ML) algorithms based on HE, in particular, has garnered much attention since many ML algorithms are being utilized for personal data, and privacy-related issues are constantly being raised.

However, HE serves only limited types of operations: addition and multiplication. It has been a challenging problem to compute a general circuit containing sophisticated non-polynomial functions based on HE. The most prevalent solution for this is to replace non-polynomial functions with its polynomial approximations. An evaluator of HE first selects the domain of each real-valued non-polynomial function and replaces the function with its polynomial approximation on the selected domain interval. This enables us to approximately compute any given circuit in encrypted states, and now we focus on a specific problem that is to find an appropriate polynomial approximation for HE.

Meanwhile, a computation over encrypted data has several differences compared to that in an unencrypted state. First, the evaluator of HE cannot observe the input values; moreover, the intermediate values during the evaluation should not be revealed. As a consequence, there are only a few clues about the domain interval of each real-valued function. Also, all intermediate values during the evaluation should belong to the plaintext space of HE. Once a single intermediate value gets out of the plaintext space of HE, it has the potential to ruin the ciphertext and damage all evaluation results. Therefore, we should find polynomial approximations on large domain intervals for successful evaluations based on HE.

1.1 Fundamental Problem

The approximation error induced by the polynomial approximation determines the quality of computation. To use a polynomial approximation for the privacy-preserving computation based on HE, we should consider both (a) the error induced by the polynomial approximation and (b) the computational cost for the evaluation of the polynomial approximation function. For ease of discussion, in this paper, we aim to minimize the computational cost of approximate evaluation of a real-valued function in an encrypted state under a given fixed maximum error. Ultimately, the approximate evaluation with a small error and a small computational cost is the most desirable.

For the computational cost, we take the number of multiplications into account as a top priority. This is because, in HE computation, the multiplication between two ciphertexts is much heavier than the other homomorphic operations such as addition, subtraction, and constant multiplication. We stress that the evaluation of a polynomial with a higher degree does not necessarily require more number of multiplications.[3] Our goal here is to minimize the computational costs during the approximate computation; rather than to minimize the degree of an approximate polynomial.

[3] For instance, we can compute a polynomial $x^8 = ((x^2)^2)^2$ with 3 multiplications while we should use at least 4 multiplications to compute x^7.

To put it all together, the substantial question is how to evaluate a non-polynomial real-valued function over homomorphically encrypted data with the smallest number of multiplications. However, as we point out, HE evaluation requires a sufficiently large domain interval to guarantee the success of the evaluation. While there have been some general solutions for this question, they induce too much computational overhead as we select the sufficiently large domain intervals. In this paper, we focus on the following question:

On a large domain interval, how to approximately evaluate a non-polynomial real-valued function over homomorphically encrypted data with a less number of homomorphic multiplications?

In this paper, we provide a partial solution to the question. For example, our solution enables us to evaluate the logistic function at large intervals with reasonable costs in encrypted states. With the previous approach, such as the combination of minimax approximation and Paterson-Stockmeyer algorithm, we should use $\Omega(\sqrt{R})$ multiplications for the approximate evaluation with fixed maximum error on the domain interval $[-R, R]$. Our solution, *domain-extension methodology*, enables us to use $O(\log R)$ multiplications to approximately evaluate the logistic function on domain interval $[-R, R]$ with a fixed maximum error in encrypted states.

More generally, our domain-extension methodology serves an efficient privacy-preserving evaluation of functions that converge at infinities (i.e. the limit of $\sigma(x)$ exists as $x \to \pm\infty$). By using our methodology, we can approximately evaluate with a fixed maximum error those functions on $[-R, R]$ with $O(\sqrt{R})$ multiplications in encrypted states. To the best of our knowledge, this is computationally better than the combination of minimax approximation and Paterson-Stockmeyer algorithm, which uses $\Omega(\sqrt{R})$ multiplications.

In addition, our methodology can be utilized for the management of outliers from the outside of the approximation interval during the privacy-preserving computation. Even a few outlying data can destroy the result because the evaluation value of polynomials can easily escape the plaintext space of HE at the outside of a fixed domain interval. However, selecting a huge approximation interval to contain all possible outliers induces too many computational overheads and makes the computation impractical. By exploiting our methodology, we can accommodate the rare outliers from a sufficiently large interval $[-R, R]$ by using $O(\log R)$ additional multiplications.

1.2 Our Contributions

Our contributions can be formalized as follows.

- **Domain-extension methodology** We propose a domain-extension methodology, which efficiently extends the valid domain interval of a polynomial for HE. We first suggest the concept of *domain extension functions* (DEFs) and *domain extension polynomials* (DEPs), which extend the valid domain interval by a factor of L, the extension ratio (i.e., extend the domain interval

from $[-R, R]$ to $[-LR, LR]$). In contrast to a simple scaling, DEFs and DEPs preserve the feature of the original function on its original domain interval. Domain-extension methodology consists in using DEFs and DEPs repeatedly. After n domain-extension processes, we can extend the domain interval by a factor of L^n while preserving the feature of the original function on its original domain interval. As a consequence, once we are given a function on a small interval $[-r, r]$, we can utilize it on a wider domain interval $[-R, R]$ by using $O(\log R)$ additional operations; We note that the extended function behaves similarly on the original domain interval $[-r, r]$.

- **Uniform approximation on wide intervals**. By using domain-extension methodology, we suggest an efficient way to evaluate in encrypted state a function $f(\cdot)$ that converges at infinities, i.e., $\lim_{x\to\infty} f(x)$ and $\lim_{x\to-\infty} f(x)$ exist in $\mathbb{R}$. *Our method provides a uniform polynomial approximation of* $f(\cdot)$ *on* $[-R, R]$ *that can be evaluated by* $O(\log R)$ *multiplications and* $O(1)$ *memory in encrypted state.*
  In terms of computational complexity, this is much better than the previous approach, minimax approximation and Paterson-Stockmeyer algorithm. We observe that the degree of the minimax approximation on $[-R, R]$ is $\Omega(R)$ in cases of functions that converges at infinities. As a result, during the computation in encrypted states, it requires $\Omega(\sqrt{R})$ operations and $\Omega(\sqrt{R})$ memory space, and it is inefficient to use it for sufficiently large R. In contrast, our method can be evaluated with $O(\log R)$ operations and is more practical. We also implement and compare both methods in Section 5.1.

- **Accommodation of outliers from wide intervals**. While using HE for numerous data, a single unexpected datum has the potential to damage the entire result. This is because an evaluation of a polynomial at the out of a fixed interval can easily get out of the plaintext space of HE. This immediately contaminates the ciphertext and ruins all evaluation results. However, it seems to be inefficient to use polynomial approximations on huge approximation intervals that are capable to contain all outliers.
  On the other hand, by using the domain-extension method, we can efficiently accommodate the rare outliers from large intervals. Our method accommodates the outliers from a large interval $[-R, R]$ by using $O(\log R)$ additional operations, and prevents the rare outliers from ruining the entire process.

- **Logistic regression over large datasets in encrypted state**. We apply our method to perform the privacy-preserving logistic regression for big datasets based on HE. By applying our method to the logistic function, we efficiently evaluate the logistic function on large domain intervals in an encrypted state. There have been several similar works to this, but to the best of our knowledge, they are unlikely to be applied to big datasets. All previous works heuristically selected the domain interval of the logistic function, and we observed that those short intervals are not sufficient for many big public datasets. Moreover, previous works select the domain intervals by monitoring the process in an unencrypted state; however, this monitoring

is undesirable in HE applications since it may leak some pieces of information about the data. In this work, on the other hand, we approximate the logistic function on sufficiently large domain intervals by using DEPs. This enables us to successfully perform the logistic regression on heavy datasets with various hyper-parameters in encrypted states.

1.3 Related Work

1.3.1 Machine Learning over Encrypted Data For the privacy-preserving delegation of computation, many works have applied HE to ML algorithms. The key to HE-based solutions for privacy-preserving ML algorithms is appropriate replacement of real-valued functions by their polynomial approximations. Since logistic regression has a simple structure, there have been several works that leverage HE for the privacy-preserving logistic regression [1–4]. Kim et al. proposed a HE-based logistic regression [2]. They replace the logistic function with the least square fitting polynomial approximation on $[-8, 8]$. Chen et al. implement HE-based logistic regression by using the minimax polynomial approximation on $[-5, 5]$ [3]. To make input values of the logistic function belong to $[-5, 5]$, they perform preprocessing (average pooling) to the data. Han et al. use the least square fitting polynomial approximation on $[-8, 8]$, and also perform preprocessing (average pooling) for the experiment on MNIST dataset [4]. To the best of our knowledge, all previous works heuristically select the approximation interval of the logistic function. In some cases, preprocessing is needed to force the inputs of the logistic function to belong to the selected interval. This selection of short approximation intervals makes it difficult to perform the logistic regression on large public datasets such as the Swarm Behavior dataset. In this work, we suggest a practical solution to approximate the logistic function on substantially wide intervals, and by using it, we implement the logistic regression in an encrypted state for large datasets.

There also have been many works that utilized HE to ML algorithms other than the logistic regression. CryptoNet modified CNN models to be HE friendly by replacing the ReLU function and max-pooling by square function and sum function respectively [5]. Hesamifard et al. suggested a better polynomial replacement of the ReLU function and analyzed its efficiency [6].

1.3.2 Computation with less multiplications When we use HE, multiplication is much heavier than the other operations such as addition and constant multiplication. Thus, the evaluation of a polynomial by using a less number of multiplications is important to mitigate the computational overhead from HE.

Paterson-Stockmeyer algorithm is an algorithm to evaluate a polynomial of degree d by using $O(\sqrt{d})$multiplications [7]. Chen et al. adopted Paterson-Stockmeyer algorithm for the HE computations, and they performed Paterson-Stockmeyer algorithm based on the Chebyshev basis to make the coefficients more consistent [8]. Chen et al.'s algorithm is being widely used for the evaluation of high-degree polynomials based on HE [9, 10].

On the other hand, while Paterson-Stockmeyer algorithm provides good performances on general polynomials, there exist some polynomials that can be evaluated with less number of multiplications compared to Paterson-Stockmeyer algorithm. For example, x^{2^n} can be evaluated with only n multiplications by repeating the squaring, rather than $2^{n/2+1}$ multiplications. Finding an approximate polynomial that can be evaluated with less multiplications can reduce the computational cost. As another example, Cheon et al. [11] proposed an iterative method to approximately evaluate the sign function on $[-1, 1]$, and they argued that their method requires relatively less number of multiplications. In the same manner, we suggest an iterative method for the polynomial approximation on a large domain interval, which can be evaluated with a less number of multiplications.

1.3.3 Homomorphic Evaluation with Bit-wise HE We are focusing on the homomorphic evaluation of functions on large domain intervals by using HE. We point out that bit-wise HE schemes, which encrypt each bit of message separately, induce a serious computational and memory overhead when applied to arithmetic operations on real-valued data on large scales. Another variant of HE schemes is programmable TFHE schemes [12, 13]. It can deal with a slightly larger interval compared to the original TFHE scheme but still suffer from the inefficiency of real-valued addition and multiplication [14]. There have been several works on conversion between TFHE and CKKS scheme to address this issue [15, 14]. However, they still have weaknesses to be applied to real-valued functions on *large domain intervals*. We describe more details in Section 4.1.2.

2 Preliminaries

2.1 Notations

In this paper, C_{lim} is the class of continuous functions that converges at infinities (i.e., $C_{\text{lim}} := \{f(\cdot) |\ f(\cdot)$ is continuous, and both $\lim_{x\to\infty} f(x)$ and $\lim_{x\to-\infty} f(x)$ exist in $\mathbb{R}\}$. When we say *multiplication*, it means a non-scalar multiplication unless we indicate. A maximum error between two function $f(\cdot)$ and $g(\cdot)$ on an interval I is $\|f - g\|_{\infty,I} := \sup\{|f(x) - g(x)| : x \in I\}$. We omit I unless it is necessary.

For the homomorphic encryption, we use the ring $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ where N is a power-of-two integer, and $\mathcal{R}_q$ denotes $\mathcal{R}/q\mathcal{R}$.

2.2 Homomorphic Encryption and CKKS scheme

HE is a cryptographic scheme that allows operations in encrypted states without any decryption process. Among various HE schemes, we adopted CKKS scheme presented by Cheon at el [16, 17], which supports arithmetic operations of approximate numbers. CKKS scheme has a strong advantage in application to machine learning algorithms since its plaintexts are real numbers. CKKS scheme

has been adopted in many implementations of privacy-preserving machine learning algorithms based on HE [4, 18, 19].

Let N be a power-of-two integer. There exists a field isomorphism $\tau : \mathbb{R}[X]/(X^N+1) \to \mathbb{C}^{N/2}$ which CKKS scheme uses for encoding and decoding messages.

CKKS scheme encrypts a (encoded) plaintext $\mathsf{m} \in \mathbb{C}^{N/2}$ into a ciphertext $\mathsf{ct} \in \mathcal{R}_Q$. For the decryption process, a secret key sk is needed; formally, the decryption process is

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}) = \mathsf{m} + \mathsf{e},$$

where e is a small error vector. We refer [16] for the detail of CKKS scheme. The main operations of CKKS scheme and their properties are the followings.

- $\mathsf{Add}(\mathsf{ct}_1, \mathsf{ct}_2)$: For ciphertexts ct_1 and ct_2 of m_1 and m_2, output a ciphertext ct of $\mathsf{m}_1 + \mathsf{m}_2$.
- $\mathsf{cMult}(\mathsf{m}, \mathsf{ct}_1)$: For a ciphertext ct_1 of m_1 and a message m, output a ciphertext ct of $\mathsf{m} \odot \mathsf{m}_1$.
- $\mathsf{Mult}(\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{evk})$: For ciphertexts ct_1 and ct_2 of m_1 and m_2, output a ciphertext ct of $\mathsf{m}_1 \odot \mathsf{m}_2$, where $\odot$ indicates the component-wise multiplication between messages in $\mathbb{C}^{N/2}$.
- $\mathsf{Bootstrap}(\mathsf{ct}, \mathsf{evk})$: For ciphertexts ct of m, output a ciphertext ct' of m with a fresh noise level.

Each multiplication between ciphertexts increases the noise in ciphertexts, so it is necessary to refresh the noise level by using Bootstrap after consuming some multiplicative levels. Also, Bootstrap and the Mult are significantly slower than other operations. Thereby, the number of multiplications and the multiplicative depth mostly determines the computational costs of circuits over CKKS ciphertexts.

To use CKKS scheme, we should select the parameters N and q_L, where N is the dimension of the ring $\mathcal{R}_{q_L}$ and q_L is the initial modulus size.

2.3 Minimax Polynomial Approximation and Paterson-Stockmeyer Algorithm

For a given continuous function $f(\cdot)$, an interval $[a, b]$ and a positive integer d, a *minimax polynomial approximation* $p(\cdot)$ is a polynomial of degree at most d such that minimize the *maximum error*, $\|f - p\|_{\infty,[a,b]}$. Many algorithms finding the minimax polynomial approximation have been suggested such as Remez algorithm and Barycentric-Remez algorithm [20, 21].

Minimax approximation guarantees a good quality of approximation at each point of the approximation interval. Hence, it is plausible to be adopted in the polynomial approximation for HE computations which hide the inputs of each function to the HE evaluator.

The evaluation of a high-degree polynomial requires lots of multiplications and additions. There are several evaluation algorithms such as Horner's method [22] and Paterson-Stockmeyer algorithm [23] to decrease the computational costs of

the evaluation. Horner's method requires $O(d)$ multiplications, and Paterson-Stockmeyer algorithm requires $O(\sqrt{d})$ multiplications. Paterson-Stockmeyer algorithm is often being adopted when multiplication is substantially more expensive than addition operation; such as circuits over matrix and circuits over HE.

Chen et al. [8] suggest Paterson-Stockmeyer algorithm with Chebyshev basis. They pointed out that in the case of many known polynomial approximation techniques, the coefficient with respect to the Chebyshev basis is more consistent compared to that with respect to the power basis. The modified Paterson-Stockmeyer algorithm is currently widely being used for the evaluation of high degree polynomials based on HE.

2.4 Logistic Regression

The logistic regression algorithm is a well-known ML algorithm that solves binary classification problems. A logistic regression classifier consists of a weight w and a bias b. As in [2, 24], for the ease of discussion, we use each datum with an additional feature with the value of 1 for the bias term. To be more detailed, we represent each datum x in the form of $z = (x, 1)$, and the logistic regression model in the form of $W = (w^T, b)^T$. Then, $w^T x + b = W^T z$.

For the inference of the logistic regression, a model W classifies each datum x into a class of either one of 1 or -1 by addressing the value of the following.

$$\Pr\Big(\textit{the label of } x \textit{ is } 1\Big) = \sigma(W^T z)$$

where $\sigma(t) = 1/(1 + e^{-t})$ is the logistic function.

For the training of logistic regression, we consider a cost function

$$J(W) = \frac{1}{n} \sum_{(z,y)} \log(1 + \exp(-y \cdot W^T z))$$

where each z is a datum with the class of $y \in \{-1, 1\}$, and n is the number of data. By using gradient descent method to this cost function, the training process seeks the appropriate weights and bias that minimize the cost function for given training dataset. Explicitly, for a given learning rate α, we update the weight and bias as follows.

$$W \leftarrow W - \alpha \nabla J(W) = W + \frac{\alpha}{n} \sum_{(z,y)} \sigma(-y \cdot W^T z) \cdot (yz)$$

3 Domain-extension Methodology

HE supports only polynomial operations: addition and multiplication. When we compute a circuit over encrypted data based on HE, we need to replace the non-polynomials by their polynomial approximations on each of its estimated

domain interval, *the approximation interval*. Once the intermediate value during the computation gets out of the plaintext space of HE, the outlier immediately ruins the ciphertext and contaminates the entire computation. To avoid this, we need to select the approximation interval of each polynomial approximation as large enough. Unfortunately, a larger approximation interval requires a bigger degree of the approximate polynomial, and results in considerable computational overhead.

We can efficiently evaluate low-degree polynomials, but in general, a low-degree polynomial is useful only on relatively small domain intervals and behaves unexpectedly on the out of the small interval. In this work, we aim to utilize a low-degree polynomial to large intervals by using a few more number of operations. We call this process by *domain-extension* process. By the domain-extension process, we use a low-degree polynomial similarly on its original (small) domain interval; meanwhile, on a larger interval, we maintain the size of function values bounded and prevent it from behaving unexpectedly.

For ease of discussion, during the domain-extension process, we distinguish two domain intervals of a given polynomial, *the interval type I and II*. We denote the interval type I as the interval that the function after the domain-extension process behaves similar to the original polynomial. We denote the interval type II as the interval that the size of function values is reasonably bounded. More precisely, We aim to extend the interval type II during the domain-extension process. The maximum norm (i.e., l_∞-norm) of the function after the domain-extension process on the extended interval type II is bounded by that of the original function on the original interval type II. In short, we regard interval type I as where the most of inputs come from, and interval type II, on the other hand, is the interval where we want to prevent the function value from behaving unexpectedly.

3.1 Motivation

Domain Extension Function. We first introduce *domain extension functions* (DEFs). For a given $L > 1$, we define a function $D : [-L, L] \to [-1, 1]$ by

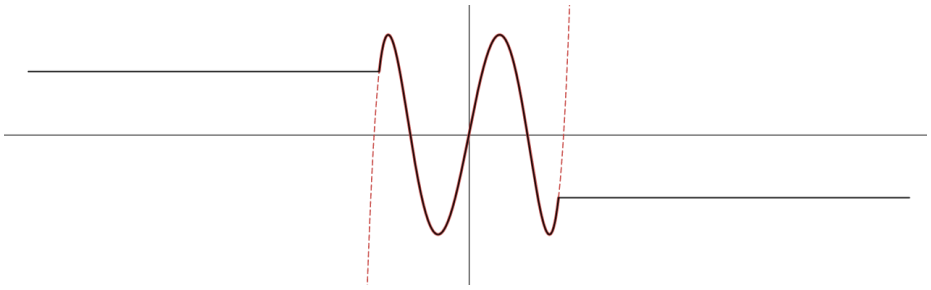

Fig. 1: An example of domain-extension process by using a DEF.

$D(x) = \frac{1}{2}\big(|x+1| - |x-1|\big)$. We will call this function a DEF with an extension ratio L.

Now assume that we are given a function $P(\cdot)$ with the domain $[-1, 1]$. Then, on $[-L, L]$, the composition $P \circ D(\cdot)$ behaves as followings.

$$P \circ D(x) = \begin{cases} P(-1), & \text{if } -L \le x \le -1 \\ P(x), & \text{if } -1 \le x \le 1 \\ P(1), & \text{if } 1 \le x \le L \end{cases}$$

This means that interval type II of $P \circ D$ is extended to $[-L, L]$ while the function values do not change on the interval type I, $[-1, 1]$.

Figure 1 illustrates the feature of DEF with extension factor 5, and how it extends the small interval type II $[-1, 1]$ of a polynomial $4x^3 - 3x$ to a wide interval $[-5, 5]$. In particular, it extends the interval type II by simply stretching the endpoints, and it exactly preserves the features on the interval type I.

Domain-extension Methodology. We can extend the size of the interval type II exponentially by repeatedly using the argument above. To be more precise, let $D(\cdot)$ be a DEF with the extension factor L that extends interval type II $[-1, 1]$ to $[-L, L]$. We may consider the scaled function $D_n(x) := L^n D(x/L^n)$ as another DEF with extension ratio L, which extends the interval type II from $[-L^n, L^n]$ to $[-L^{n+1}, L^{n+1}]$. Applying $D_i(\cdot)$ n times sequentially, we can extend interval type II of $f(\cdot)$ from $[-1, 1]$ to $[-L^n, L^n]$. In other words, we extend the size of interval type II by factor of R with $O(\log R)$ domain-extension processes. We also point out that the function value on the original domain, $[-1, 1]$, does not change.

We name this iterative strategy that extends the interval type II while preserving the features of the target function on the interval type I as *domain-extension methodology*. Unfortunately, DEFs may not be directly used for privacy-preserving computations based on HE since they cannot be evaluated by using only HE operations. Thus, we define domain extension polynomials (DEPs) in Section 3.2, and we explain the domain-extension methodology with DEPs in Section 3.3.

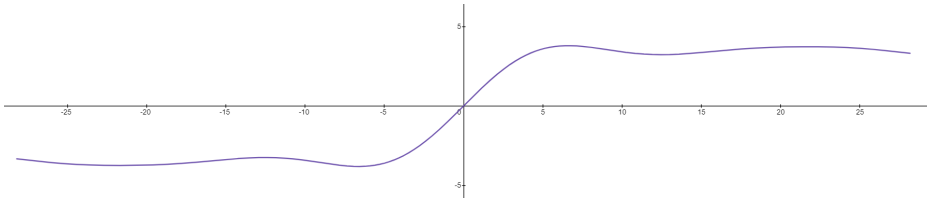

Fig. 2: An example of DEP.

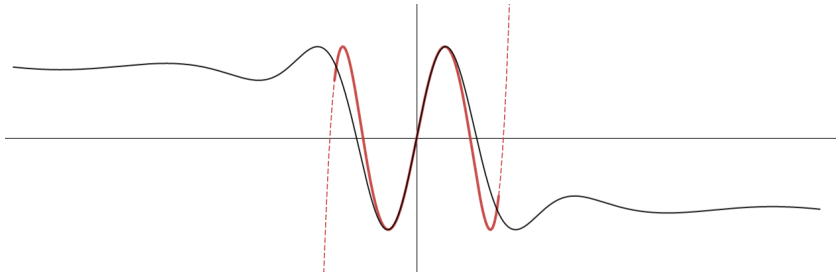

Fig. 3: An example of domain-extension process by using a DEP.

3.2 Domain Extension Polynomial

To utilize domain-extension methodology for the HE-based computations, we need polynomials that can take the role of DEFs. The crux is to exploit the approximate polynomials of DEFs. For ease of discussion, we define a narrow definition of DEPs as follows.

Definition 1 (Domain Extension Polynomials). *For given constants $R_2 > R_1 > r > 0$ and small error $\delta > 0$, we define $\mathscr{D}(\delta, r, R_1, R_2)$ as a class of polynomials $d(\cdot)$ satisfying:*
(a) $|x - d(x)| \leq \delta |x|^3 \ \forall x \in [-r, r]$
(b) $0 \leq d'(x) \leq 1 \ \forall x \in [-r, r]$
(c) $d(r) < d(x) < R_1 \ \forall x \in [r, R_2]$
(d) $-R_1 < d(x) < d(-r) \ \forall x \in [-R_2, -r]$.
We call R_2/R_1 as the extension ratio of $d(\cdot)$. We also denote $d(\cdot)$ is a DEP extending $[-R_1, R_1]$ to $[-R_2, R_2]$ preserving $[-r, r]$ if $d(\cdot) \in \mathscr{D}(\delta, r, R_1, R_2)$.

Figure 2 visualizes a graph of an example of DEPs. We point out that it approximates a DEF and it may behave similarly to a DEF. Figure 3 visualizes how a DEP (approximately) extends the interval type II of a given polynomial. We can observe that a DEP can extend the interval type II of a polynomial while preserving the function values on the narrow interval type I with minor differences. More precisely, we argue that a DEP $d(\cdot) \in \mathscr{D}(\delta, r, R_1, R_2)$ can extend the interval type II of a polynomial from $[-R_1, R_1]$ to $[-R_2, R_2]$ while preserving the polynomial on a narrow subinterval of type I, $[-r, r]$. We formally state this in Theorem 1.

Theorem 1. *Assume that we are given a DEP $d(\cdot) \in \mathscr{D}(\delta, r, R_1, R_2)$, and a polynomial $p(\cdot)$ such that $\sup\{|p'(x)| : -r \leq x \leq r\} \leq M$. Then,*

$$\|p \circ d\|_{\infty,[-R_2,R_2]} \leq \|p\|_{\infty,[-R_1,R_1]},$$

$$\|p \circ d - p\|_{\infty,[-r,r]} \leq Mr^3\delta,$$

and

$$\sup\{|(p \circ d)'(x)| : -r \leq x \leq r\} \leq M.$$

Proof. For each $x \in [-r, r]$, there exist x_* between $d(x)$ and x such that

$$|p \circ d(x) - p(x)| = |p'(x_*)||x - d(x)| \leq M\delta|x|^3 \leq Mr^3\delta.$$

Also,

$$|(p \circ d)'(x)| = |p'(d(x))d'(x)| \leq M.$$

Note that $0 \leq d' \leq 1$ on $[-r, r]$ insists that $-r \leq -|x| \leq d(x) \leq |x| \leq r$. □

To put it all together, a DEP behaves similar to a DEF. A DEP extends the interval type II from $[-R_1, R_1]$ to $[-R_2, R_2]$, while preserving the features of the original function on a small subinterval of type I, $[-r, r]$.

3.3 Domain-extension Methodology with DEPs

We now describe an iterative domain-extension by using a DEP.

We begin with the observation that if $d(\cdot)$ is a DEP that extends $[-R_1, R_1]$ to $[-R_2, R_2]$, then its scaled polynomial $kd(\cdot/k)$ is a DEP that extends $[-kR_1, kR_1]$ to $[-kR_2, kR_2]$.

Theorem 2. *If $d(\cdot) \in \mathscr{D}(\delta, r, R_1, R_2)$ and $L > 1$, then*

$$D(x) := Ld\Big(\frac{x}{L}\Big) \in \mathscr{D}\Big(\frac{\delta}{L^2}, r, LR_1, LR_2\Big).$$

Proof. For each $x \in [-r, r]$, let $z = x/L \in [-r, r]$. Then,

$$|x - D(x)| = |Lz - Ld(z)| \leq L\delta|z|^3 = \frac{\delta}{L^2}|x|^3$$

holds. Also,

$$D'(x) = \frac{d}{dx}\left(Ld\left(\frac{x}{L}\right)\right) = d'(x) \in [0, 1].$$

For each $x \in [r, LR_2]$, let $z = x/L \in [-r/L, R_2]$. If $z \leq r$, $d\left(\frac{r}{L}\right) \leq d(z) \leq d(r) \leq R_1$ holds since $0 \leq d' \leq 1$ on $x \in [0, r]$. Meanwhile, if $z > r$, $d\left(\frac{r}{L}\right) \leq d(r) \leq d(z) \leq R_1$ holds. Thus, in both cases,

$$D(r) = Ld\left(\frac{r}{L}\right) \leq Ld(z) = D(x) \leq LR_1$$

holds, and similar result holds for $x \in [-LR_2, -r]$.
Therefore, $D(x) \in \mathscr{D}\Big(\frac{\delta}{L^2}, r, LR_1, LR_2\Big)$. □

By using Theorem 2, we can generate a sequence of DEPs, $B_n(x) = L^n B(x/L^n)$, from a base DEP $B(x)$ that extends $[-R, R]$ to $[-LR, LR]$. Each DEP $B_n(\cdot)$ extends $[-L^n R, L^n R]$ to $[-L^{n+1}R, L^{n+1}R]$, so we can utilize them sequentially. To apply DEPs $B_i(\cdot)$ sequentially to a given polynomial that has interval type II

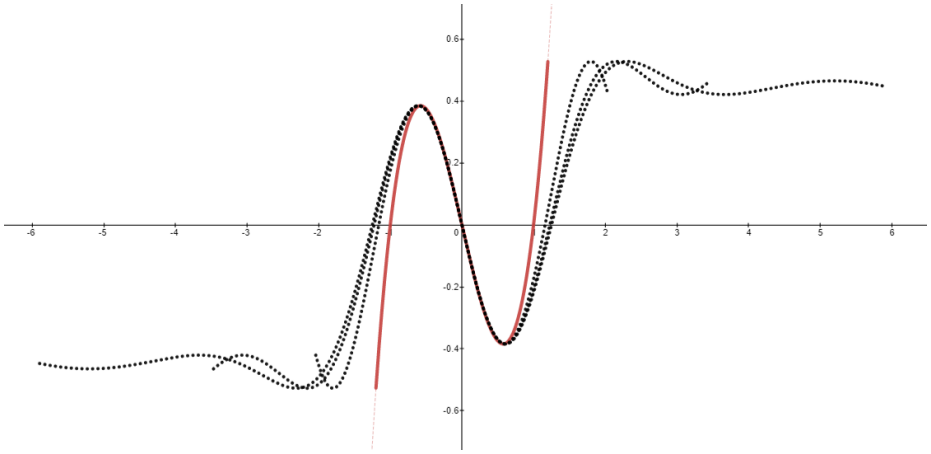

Fig. 4: The functions with extended intervals type II after $1, 2, 3$ iterations of domain-extension method to the red curve. Note that the feature of the original function is preserved on $[-0.5, 0.5]$

$[-R, R]$, the interval type II would be extended to $[-L^n R, L^n R]$ after n iterations. Hence, we can extend the interval type II by a factor of L' with $O(\log L')$ iterations. Figure 4 visualizes how the sequence of DEPs extends the interval type II for each iteration. We will call this methodology as *domain-extension methodology*. We argue that domain-extension methodology with DEPs (1) extends the interval type II efficiently, and (2) preserves the feature of the original function on the interval type I.

Theorem 3 (Domain-extension methodology). *Assume that $B(\cdot) \in \mathscr{D}(\delta, r, R, LR)$ and a polynomial $p(\cdot)$ on $[-R, R]$ is given. For each non-negative integer n, let*

$$B_n(x) := L^n B\Big(\frac{x}{L^n}\Big).$$

Then, after n sequential composition of $B_i(\cdot)$ to $p(\cdot)$,

$$\Big\|p \circ B_0 \circ \cdots \circ B_{n-1}\Big\|_{\infty,[-L^n R, L^n R]} \leq \|p\|_{\infty,[-R,R]},$$

and

$$\Big\|p - p \circ B_0 \circ \cdots \circ B_{n-1}\Big\|_{\infty,[-r,r]} \leq M r^3 \frac{L^2}{L^2 - 1}\delta$$

holds where $M = \sup\{|p'(x)| : -r \leq x \leq r\}$.

Proof. By Theorem 2, $B_n(\cdot) \in \mathscr{D}\Big(\frac{\delta}{L^{2n}}, r, L^n R, L^{n+1}R\Big)$. By using Theorem 1, we conclude that

$$\begin{aligned}
&\Big\|p - p \circ B_0 \circ \cdots \circ B_{n-1}\Big\| \\
&\leq \|p - p \circ B_0\| + \sum_{i=1}^{n-1} \|p \circ B_0 \cdots \circ B_{i-1} - p \circ B_0 \cdots \circ B_i\| \\
&\leq Mr^3\delta \sum_{i=0}^{n-1} \frac{1}{L^{2i}} \ \leq Mr^3\delta \sum_{i=0}^{\infty} \frac{1}{L^{2i}} = Mr^3 \frac{L^2}{L^2-1}\delta
\end{aligned}$$

Here, we denoted $\|\cdot\|_{\infty,[-r,r]}$ as $\|\cdot\|$. □

Theorem 3 implies that if we use a DEP with a small δ, the output polynomial of domain-extension methodology has only a small difference from the original polynomial on $[-r, r]$, and at the same time, the output polynomial is bounded on the large interval, $[-L^n R, L^n R]$.

To briefly sum up, we suggest DEPs and domain-extension methodology with it that extends the interval type II of a polynomial efficiently while preserving the approximation property on the interval type I. Note that to extend a polynomial's interval type II by a factor of L, domain-extension methodology uses $O(\log L)$ operations.

Remark 1. Note that the domain-extension methodology is done by invoking a simple base DEP repeatedly. Once we are given an appropriate base DEP $B(\cdot)$ that can be stably evaluated based on HE (e.g., it does not considerably magnify the errors induced by HE), its scaled function $B_n(x)$ would be also HE friendly. This makes the domain-extension methodology stable and HE friendly.

Example 1. $d(x) = -\frac{4}{27}x^3 + x$ is a DEP in $\mathscr{D}\Big(\frac{4}{27}, r, 1, L\Big)$ for each $1.5 < L < 1.5\sqrt{3}$ and $0 < r < 1.5$ such that $d(r) < d(L)$. Note that Theorem 2 implies that we can decrease δ as much as possible by scaling.

4 Homomorphic Evaluation on Large Intervals

In this section, we explain how domain-extension methodology can be used for privacy-preserving computations based on HE. We remark that to exploit HE, non-polynomial functions should be replaced by their polynomial approximations, and the approximation interval should be large enough to contain all input values. Since polynomial approximation on the large interval, in general, introduces a high degree and considerable computational overheads, domain-extension methodology may provide a more efficient solution to manage a large domain interval.

We suggest two applications of thedomain-extension methodology. First, we show that for several functions (e.g., the logistic, arctan, tanh, capped ReLU, and

Gaussian function), domain-extension methodology extends not only interval type II, but also interval type I. As a consequence, by using the domain-extension methodology, we can evaluate those functions based on HE over large domain intervals. We stress that this is substantially more efficient than other known polynomial approximation techniques.

Second, we explain that domain-extension methodology can effectively manage rare outlying data. Suppose there are some rare outliers from a large interval, while the most of data are from a small domain interval. The function value of known polynomial approximations soars rapidly on the outside of a fixed approximation interval. An outlier from the out of the approximation interval would become extremely large after the homomorphic evaluation of the function. Once the evaluation value does not belong to the plaintext space of HE, the ciphertext will be polluted, and the entire results will be contaminated. Hence, the approximation interval should be large, but it might be inefficient if there are only a few outliers. In Section 4.2, we suggest an efficient solution to address this issue by using the domain-extension methodology.

4.1 Uniform Approximation using DEPs

We introduce an efficient polynomial approximation technique for functions in $C_{\lim}$ (i.e., $f(\cdot)$ that is continuous and both $\lim_{x\to\infty} f(x)$ and $\lim_{x\to-\infty} f(x)$ exist in $\mathbb{R}$). We first remark that many ML algorithms utilize functions that converge at infinities; e.g., the logistic function, tanh function, arctan, capped ReLU function [25], and Gaussian function. To perform those ML algorithms in an encrypted state, it is necessary to uniformly approximate by polynomials the functions in $C_{\lim}$. However, with known approaches, the HE evaluation of such functions on sufficiently large intervals results in considerable overhead.

By utilizing our domain-extension method in Section 3, we suggest an efficient polynomial approximation technique for functions in $C_{\lim}$. The idea is simple: find an approximation on a smaller interval and extend its approximation interval by using the domain-extension method. Then, we approximately compute $f(\cdot) \in C_{\lim}$ on $[-R, R]$ with $O(\log R)$ operations. Algorithm 1 describes how to utilize domain-extension methodology for this scenario.

As in Figure 4, our domain-extension methodology stretches the graph of the given polynomial by pulling on both ends while fixing the central part (interval type I) of the original function. Thus, in the case of approximate polynomials of $f(\cdot) \in C_{\lim}$, the domain-extension method immediately extends not only the interval type II but also the interval type I, so we yield an efficient polynomial approximation $f(\cdot)$ on the extended wide interval. More precisely, Theorem 4 states that Algorithm 1 is correct unless r is too small.

Theorem 4. *Assume that a M-Lipshitz continuous function $f(\cdot) \in C_{\lim}$ and a DEP $d(\cdot) \in \mathscr{D}(\delta, r, R, LR)$ such that*

$$|f(x) - f(y)| < \epsilon_1 \quad \text{if } x, y \le -d(r)$$
$$|f(x) - f(y)| < \epsilon_1 \quad \text{if } x, y \ge d(r).$$

Algorithm 1 Homomorphic evaluation of $C_{\lim}$ functions on large intervals

Input: $f(\cdot) \in C_{\lim}, x \in [-L^n R, L^n R]$.
Output: Approximate value of $f(x)$.
1: $P(\cdot) :=$ the minimax polynomial on $[-R, R]$
2: Take $B(\cdot) \leftarrow \mathscr{D}(\delta, r, R, LR)$
3: $y = x$
4: **for** $i \leftarrow n-1$ **to** 0 **do**
5: $\quad y = L^i B\left(\frac{y}{L^i}\right)$
6: **end for**
7: $y = P(y)$
8: **return** y

for some $\epsilon_1 > 0$. Suppose that $p(\cdot)$ on $[-R, R]$ is a polynomial approximation of $f(\cdot)$ that satisfies

$$|f(x) - p(x)| < \epsilon_2 \quad \textit{if } x \in [-R, R].$$

for some $\epsilon_2 > 0$.

Then, $p \circ d(\cdot)$ is a polynomial approximation of $f(\cdot)$ on $[-LR, LR]$ with maximum error is bounded by $max(Mr^3\delta, \epsilon_1) + \epsilon_2$.

Moreover, after Algorithm 1, we get the polynomial approximation on $[-L^n R, L^n R]$ with maximum error less than
$max(\frac{Mr^3L^2}{L^2-1}\delta, \epsilon_1) + \epsilon_2$.

Proof. From Theorem 1, for each $x \in [-r, r]$,

$$\begin{aligned} |p \circ d(x) - f(x)| &\le |p \circ d(x) - f \circ d(x)| + |f \circ d(x) - f(x)| \\ &\le \epsilon_2 + M|x - d(x)| \le \epsilon_2 + Mr^3\delta. \end{aligned}$$

Meanwhile, for each $x \in [r, LR]$,

$$\begin{aligned} |p \circ d(x) - f(x)| &\le |p \circ d(x) - f \circ d(x)| + |f \circ d(x) - f(x)| \\ &\le \epsilon_1 + \epsilon_2 \le max(Mr^3\delta, \epsilon_1) + \epsilon_2. \end{aligned}$$

And the same argument holds for $x \in [-LR. - r]$. Thus,

$$|p \circ d(x) - f(x)| \le max(Mr^3\delta, \epsilon_1) + \epsilon_2$$

for all $x \in [-LR, LR]$.

Finally, the similar argument with Theorem 3 proves that Algorithm 1 serves the uniform approximation on $[-L^n R, L^n R]$ with maximum error less than $\max(\frac{Mr^3L^2}{L^2-1}\delta, \epsilon_1) + \epsilon_2$. □

As a consequence, we can approximately evaluate a function $f(\cdot) \in C_{\lim}$ on an arbitrarily large interval $[-R, R]$ within the error less than $\epsilon = \max(\frac{Mr^3L^2}{L^2-1}\delta, \epsilon_1) +$

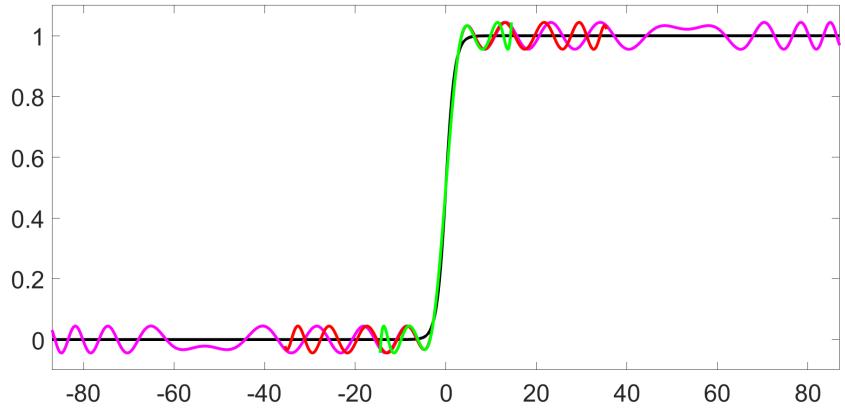

Fig. 5: The graphs of a polynomial before and after the domain-extension methodology. The black line shows the exact logistic function, the green line shows its 9^{th} order minimax approximation on $[-14.5, 14.5]$, the red line shows the polynomial after 1 domain-extension process, and the pink line shows the polynomial after 2 domain-extension processes.

# of extensions	0	1	2	3
Size of domain	29	71.05	174.07	426.48
Maximum error	0.04416	0.04447	0.04447	0.04447
# of Mult	4	6	8	10

Table 1: The maximum errors of polynomial approximations of the logistic function using Algorithm 1 with various numbers of domain-extension processes. We used the DEP $x - 16/22707x^3$, which extends $[-14.5, 14.5]$ to $[-35.525, 35.525]$.

ϵ_2 by using Algorithm 1 with $O(\log R)$ iterations of domain-extension processes. Table 1 and Figure 5 describe the maximum error after the domain-extension processes for the homomorphic evaluation of the logistic function.

Our algorithm is very efficient for large domain intervals. We measured the computational cost in terms of the number of multiplications during the computation, because the multiplication is much heavier than other operations in HE. To approximately evaluate a function in $f(\cdot) \in C_{\text{lim}}$ on sufficiently large interval $[-R, R]$, we perform domain-extension process for $O(\log R)$ times; hence, our method uses $O(\log R)$ multiplications in encrypted state. Also, we point out that our method exploits $O(1)$ memory since the domain-extension processes are sequentially executed.

For the computation based on HE, the multiplicative depth is also an important factor since it determines the number of required bootstrapping during the computation. The multiplicative depth of our method is $O(\log R)$ since we uses $O(\log R)$ domain-extension processes for the interval $[-R, R]$.

Remark 2. Our algorithm can be easily extended to a function whose difference from a low degree polynomial converges to constants $\in \mathbb{R}$ at infinities. Note that such function can be represented as $f(x) + p(x)$, where $f(\cdot) \in C_{\lim}$ and $p(\cdot)$ is a lower degree polynomial.

4.1.1 Towards higher accuracy Theorem 4 states that the domain-extension methodology extends the approximation interval for function $f(\cdot)$ in $C_{\lim}$, while compromising the small maximum error bounded by a certain limit. The error growth from domain-extension process using $d(\cdot) \in \mathscr{D}(\delta, r, R, LR)$ is determined by f, δ and r. To yield higher accuracy, e.g., maximum error less than 2^{-20}, with Algorithm 1, domain-extension polynomials with small δ's are needed. We introduce optimizations that enable us to use simple domain-extension polynomials with not too small δ's for high precision polynomial approximations.

Suppose we want to approximate a function $f(\cdot) \in C_{\lim}$ on a large domain interval $[-R, R]$; by using DEP $d(\cdot)$ and a polynomial approximation on the small interval $[-r, r]$.

Our first optimization is to use the minimax approximation $Q(\cdot)$ of $f \circ d^{-1}(\cdot)$ on $[-r, r]$ instead of the minimax approximation $P(\cdot)$ of $f(\cdot)$ on $[-r, r]$. To be more precise, let $Q(x) \approx f \circ d^{-1}(x)$ where $d^{-1} \circ d(x) \approx x$, and suppose the degree of $Q(\cdot)$ and $P(\cdot)$ are equal. Then,

$$\begin{aligned}
\|f - P \circ d\|_{\infty,[-R,R]} &\geq \|f \circ d^{-1} - P \circ d \circ d^{-1}\|_{\infty,[-r,r]} \\
&\approx \|f \circ d^{-1} - P\|_{\infty,[-r,r]} \\
&\geq \|f \circ d^{-1} - Q\|_{\infty,[-r,r]} \\
&\approx \|f - Q \circ d\|_{\infty,[R,R]}.
\end{aligned}$$

To sum up, we can expect to yield a better maximum error than that in Theorem 5 by considering the minimax approximation of $f \circ d^{-1}(\cdot)$.

However, finding the minimax approximation of $f \circ d^{-1}(\cdot)$ is not easy in general. To address this issue, we suggest a second heuristic optimization. The idea is to use the polynomial approximation of d^{-1}, say $q(x)$. Then, $f \circ d^{-1}(x) \approx P \circ q(x)$. By applying domain-extension methodology to $P \circ q(\cdot)$, we get $P \circ q \circ d \approx f \circ d^{-1} \circ d \approx f$. Thus, we can yield an efficient high precision polynomial approximation on large domain intervals.

Fortunately, a low degree polynomial approximation of $d^{-1}(\cdot)$ is sufficient to achieve a high accuracy (e.g., maximum error $\leq 2^{-20}$). Algorithm 2 describes an algorithm of our method for high accuracy using the simplest domain-extension polynomial, $x - \frac{4}{27R^2}x^3$. For the optimization, we use a 5^{th} degree polynomial, $q(x) = x + \frac{4}{27}\frac{L^2(L^{2n}-1)}{L^{2n}(L^2-1)}(x^3 - x^5)$, which is a polynomial approximation of a composition of inverse of n DEPs used for the domain-extension method. We implemented Algorithm 2 to approximate the logistic function on large domain intervals within maximal error 2^{-20}. The experimental results are represented in Figure 6 and Table 2. The detailed results are reported in Section 5.

Algorithm 2 Homomorphic evaluation of $C_{\lim}$ functions on large intervals with higher accuracy

Input: $f(\cdot) \in C_{\lim}, x \in [-L^n R, L^n R]$.
Output: Approximate value of $f(x)$.
1: $P(\cdot) :=$ the minimax polynomial on $[-R, R]$
2: $y = x$
3: **for** $i \leftarrow n-1$ **to** 0 **do**
4: $\quad y = y - \frac{4}{R^2 27 L^{2i}} y^3$
5: **end for**
6: $y = y/R$
7: $y = y + \frac{4}{27} \frac{L^2(L^{2n}-1)}{L^{2n}(L^2-1)}(y^3 - y^5)$
8: $y = P(Ry)$
9: **return** y

Maximum Error ($\log_2$)					
# of extensions	0	1	2	3	4
(Size of domain)	(110)	(220)	(440)	(880)	(1760)
Algorithm 1	-21.6	-14.0	-13.7	-13.6	-13.6
Algorithm 2	-21.6	-20.3	-20.1	-20.0	-20.0

Table 2: The logarithmic values of the maximum errors of polynomial approximations of the logistic function using Algorithm 1 and 2 with various numbers of domain-extension processes. We used domain-extension polynomial $x - 4/81675x^3$, which extends the interval type II from $[-55, 55]$ to $[-110, 110]$.

4.1.2 Comparison to the previous approaches

For the privacy-preserving evaluation of non-polynomial functions, to the best of our knowledge, there are two approaches using: (a) the conversion between other HE schemes, and (b) the minimax polynomial approximation. However, both are not satisfactory for the evaluation on substantially large domain intervals.

Pegasus. Pegasus [14] proposed a conversion algorithm between TFHE and CKKS scheme to evaluate non-polynomial functions over CKKS ciphertexts. For a given CKKS ciphertext, Pegasus converts it into a number of TFHE ciphertexts, performs programmable bootstrappings of the TFHE scheme, and repacks the TFHE ciphertexts into a CKKS ciphertext.

However, to utilize Pegasus for the function evaluation on large domain intervals, a CKKS ciphertext should be converted into TFHE ciphertexts containing messages from the large domain interval. This results in the large parameters of the TFHE scheme and a substantial computational overhead incurs.

To be more precise, as mentioned in [14], when using Pegasus on $[-q/2\tilde{\epsilon}, q/2\tilde{\epsilon})$, the approximation error is about $\mathcal{L}q/(2\tilde{\epsilon}n)$ where $\mathcal{L}$ is the smoothness of the non-polynomial function, $\tilde{\epsilon}$ is a small integer (e.g., 2), and q and n are parameters for TFHE. Thus, to evaluate on a large domain interval $[-R, R]$ with a consis-

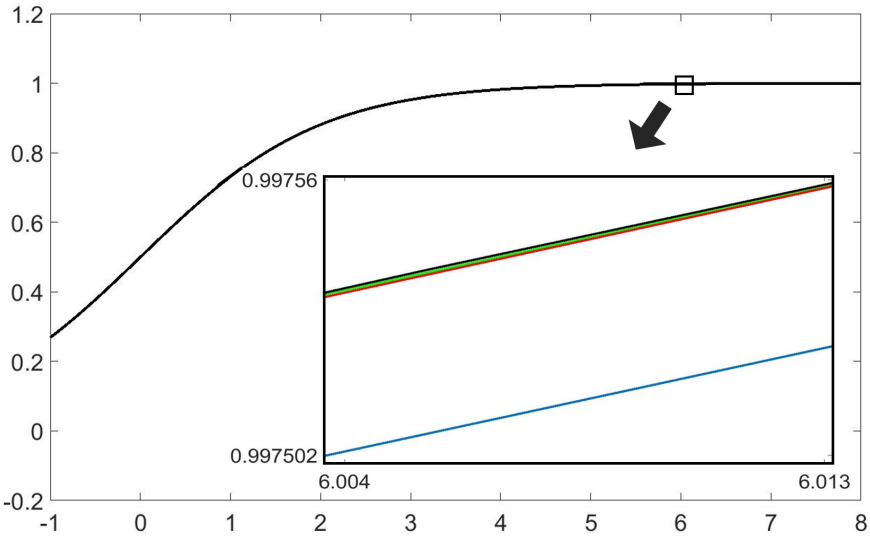

Fig. 6: The graphs of polynomial approximations of the logistic function. The black line shows the exact logistic function, the green line shows the 243^{th} order minimax approximation on $[-55, 55]$, the blue line shows the polynomial after 3 domain-extension processes (Algorithm 1), and the red line shows the polynomial after 3 domain-extension processes with optimization (Algorithm 2).

tent approximation error, the parameter RLWE dimension and the ciphertext modulus should be $\Omega(R)$. This induces both the memory and computational overhead of $\tilde{\Omega}(R)$. Pegasus mostly consider the function evaluation on narrow domain intervals, e.g., $[-8, 8]$.

It is also possible to use the small TFHE parameters for large domain intervals. Then, instead of evaluating a function $f(\cdot)$ on $[-R, R]$, one evaluates $f(\frac{R}{r}x)$ on $[-r, r]$. In that case, the smoothness $\mathcal{L}$ of the function grows, and the approximation error becomes large when we consider substantially large domain intervals. We report the implementation results in Section 5.1.

Also, Pegasus generates and manipulates a TFHE ciphertext for each slot of a given CKKS ciphertext. As a consequence, Pegasus cannot leverage the SIMD (Single Instruction Multiple Data) operations of the CKKS scheme.

Minimax polynomial approximation. For given maximum error, minimax approximation technique serves the polynomial approximation with the smallest degree, so it is prevalent to use minimax approximation for HE computations. To evaluate the minimax approximation, Paterson-Stockmeyer algorithm is being widely adopted since it uses a less number of multiplications.

However, as we illustrate in Figure 7 and prove in Theorem 5, to approximately evaluate $f(\cdot) \in C_{\text{lim}}$ on $[-R, R]$ in encrypted state with minimax approximation, we should compute a polynomial of degree $\Omega(R)$. As a consequence,

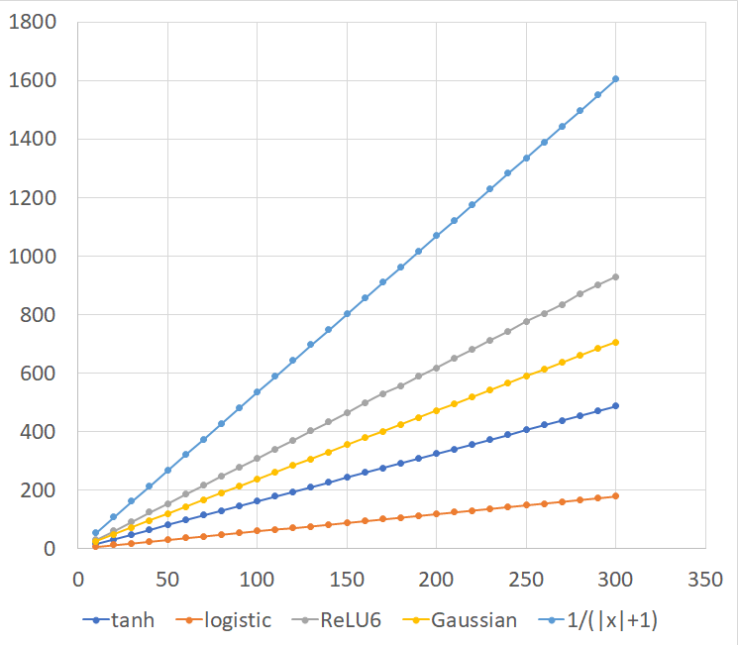

Fig. 7: The minimal degrees for the minimax approximation to approximate the several functions by maximum errors of 0.05 on various domains. The degree of minimax polynomial increases linearly with respect to the size of the approximation domain.

even with Paterson-Stockmeyer algorithm, $\Omega(\sqrt{R})$ multiplications and $\Omega(\sqrt{R})$ memory space are needed. Also, the number of cMult (multiplication between a ciphertext and a plaintext) is $\Omega(R)$. Even though cMult is pretty faster than Mult (multiplicatoin between two ciphertexts) in HE computation, it is not negligible when R is substantially large. We report the implementation result in Section 5.

Theorem 5. *Assume that we are given a function $f(\cdot) \in C_{\lim}$ where $\lim_{x\to\infty} f(x) \neq \lim_{x\to-\infty} f(x)$. For a sufficiently small $\epsilon > 0$, the minimax polynomial approximation on $[-R, R]$ with maximum error ϵ has degree $\Omega(R)$ as $R \to \infty$. Empirically, the same result holds even if $\lim_{x\to\infty} f(x) = \lim_{x\to-\infty} f(x)$ unless f is a constant function.*

Proof. Without loss of generality, assume that $\lim_{x\to-\infty} f(x) = 0$ and $\lim_{x\to\infty} f(x) = 1$. There exists $r > 0$ such that $|f(x) - 1| < \epsilon/2$ if $x > r$, and $|f(x)| < \epsilon/2$ if $x < -r$ For sufficiently large $R \gg r$, let $P_R(\cdot)$ be the minimax polynomial approximation of $f(\cdot)$ on R by maximum error less than $\epsilon/2$. Let d be the degree

	# of Mult	# of cMult	Mult. depth	Memory
[14]	Time: $\Omega(R)$			$\Omega(R)$
Minimax	$\Omega(\sqrt{R})$	$\Omega(R)$	$\Omega(\log R)$	$\Omega(\sqrt{R})$
Ours	$O(\log R)$	$O(\log R)$	$O(\log R)$	$O(1)$

Table 3: Cost of our algorithm and previous approaches to evaluate C_{lim} functions on $[-R, R]$ under a fixed maximum error.

of P_R. Then,

$$\begin{aligned}\epsilon &\geq \sup\{|P_R(x) - sgn(x)| : x \in [-R, -r] \cup [r, R]\} \\ &= \sup\{|P_R(rx) - sgn(x)| : x \in [-\frac{R}{r}, -1] \cup [1, \frac{R}{r}]\} \\ &\geq \sqrt{\frac{2}{\pi A d}}(A-1)\Big(\frac{A-1}{A+1}\Big)^{\frac{d-1}{2}} \sim \sqrt{\frac{2}{\pi}}\Big(\sqrt{\frac{R}{dr}} - \sqrt{\frac{dr}{R}}\Big)\end{aligned}$$

where $A = R/r$. This comes from the arguments in [26].
This implies that $\sqrt{\frac{R}{dr}} < E$ for some constant $E(\epsilon)$. As a consequent, $d > CR$ for some constant $C(\epsilon)$, and $d = \Omega(R)$. The graphs in Figure 7 shows that the empirical result. □

Compared to the minimax approximation, our approach is more efficient in terms of both computational and memory costs. To approximately evaluate on $[-R, R]$ based on HE, our method uses $O(\log R)$ numbers of Mult and cMult while minimax approximation uses $\Omega(\sqrt{R})$ and $\Omega(R)$ number of Mult and cMult respectively. Both methods require $O(\log R)$ multiplicative depth. See Figure 8 and 9 for the experimental results.

Also, we point out that our method uses $O(1)$ memory during the computation while the minimax approximation uses $\Omega(\sqrt{R})$ memory for the Paterson-Stockmeyer algorithm. Table 3 summarizes the costs to approximate a function on $[-R, R]$ under a fixed uniform error by using Pegasus, minimax approximation, and our algorithms.

Another point is that our method provides a stable evaluation even for large domain intervals. This is because during the domain-extension, all we compute is DEPs of low degrees. On the other hand, when we exploit the minimax approximation with Paterson-Stockmeyer algorithm, we should precisely evaluate the Chebyshev polynomial of degree $\sqrt{d}$ where $d = \Omega(R)$. Consequently, as R grows, the error induced by HE would make it challenging to use Paterson-Stockmeyer algorithm.

As more, our algorithm is much simpler to implement. In practice, it is difficult to find the minimax polynomial on a significantly large interval. In contrast, our algorithm can be easily implemented for substantially large domain intervals, by using a simple DEP and a minimax polynomial on a small domain interval.

4.2 Accommodation of Outliers

In this section, we utilize DEPs to accommodate rare outliers distributed on a wide interval. A polynomial approximation has a weakness to the outliers from the outside of the approximation interval because its value soars rapidly at the inputs from the out of the approximation interval. This can be a serious issue since it has the potential to damage the ciphertext and ruin the entire computation. To take the training phase of neural networks in an encrypted state as an example, a single strange training datum can generate input of an activation function from the out of the approximation interval, and its polynomial evaluation might be out of the plaintext space of HE. As a result, a single outlying datum can destroy the ciphertext, and ruin the entire training phase. Moreover, all the processes are being done in an encrypted state, so we even cannot detect which datum caused the failure.

To address this issue, there should be a clever way to manage such outlying inputs from wide intervals. However, it is inefficient to use a polynomial approximation on huge approximation intervals to manage all such outliers. We suggest, instead, considering a polynomial approximation that is accurate on a relatively small interval (interval type I) and *bounded on a huge interval* (interval type II) at the same time. The rare outliers from the huge interval may not produce meaningful results; albeit, they do not damage either the ciphertext or other parts of the algorithm. For example, in the case of the neural network training over encrypted data, we now can prevent abnormal data from contaminating the entire process.

For the formal description, we consider $\mathscr{F}$, a class of approximate functions of $f(\cdot)$, as followings. We note that this is a class of approximate polynomials that is accurate on a small interval $[-r, r]$, and bounded by ρ on the large interval $[-R, R]$.

Definition 2. *For a given function $f(\cdot)$ on $[-R, R]$ and given $R > \rho > r > 0$ with small $\epsilon > 0$, we define $\mathscr{F}(f; \epsilon, r, \rho, R)$ to be a class of function of $p(x)$ satisfying:*
(a) $|p(x) - f(x)| < \epsilon \ \forall x \in [-r, r]$
(b) $|p(x)| < \rho \ \forall x \in [-R, R]$.

$\mathscr{F}(f; \epsilon, r, \rho, R)$ is a class of functions that is accurately approximates $f(\cdot)$ on $[-r, r]$ and is bounded by ρ on $[-R, R]$. Thus, the HE evaluation on $[-r, r]$ would be valuable, and that on $[-R, R]$ would be stable (i.e., each function value on $[-R, R]$ belongs to the plaintext space of HE).

In this case, our domain-extension methodology can be applied; as we regard $[-r, r]$ as the interval type I and $[-R, R]$ as the interval type II. Theorem 3 explains how domain-extension methodology enables us to efficiently increase the stable interval. Note that Theorem 6 is an immediate outcome of Theorem 3.

Theorem 6. *Assume that we are given a function $f(\cdot)$ and its approximation $p(\cdot) \in \mathscr{F}(f; \epsilon, r, \rho, R)$; also, suppose we are given a DEF $d(\cdot) \in \mathscr{D}(\delta, r, R, LR)$*

where $LR > R > \rho > r > 0$ *and* $\delta, \epsilon > 0$ *are small. Then,*

$$p \circ d(\cdot) \in \mathscr{F}\Big(f; \epsilon + Mr^3\delta, r, \rho, LR\Big)$$

where $M = sup\{|p'(x)| : -r < x < r\}$*. Moreover, if we let*

$$B_n(x) := L^n d\Big(\frac{x}{L^n}\Big)$$

for each non-negative integer n*, then*

$$F \circ B_0 \circ \cdots \circ B_{n-1}(\cdot) \in \mathscr{F}\Big(f; \epsilon + Mr^3\frac{L^2}{L^2-1}\delta, r, \rho, L^n R\Big).$$

To put it all together, when we use a polynomial approximation in $\mathscr{F}$, we can manage the outliers from a wide interval $[-R, R]$ by using additional $O(\log R)$ number of operations.

5 Experiments

In this section, we implement our domain-extension methodology and apply it to the privacy-preserving logistic regression based on HE. Note that the training and inference of a logistic regression classifier can be done by the computation of linear operations and the logistic function.

The logistic function is in C_{lim}, so domain-extension methodology provides an efficient uniform approximation of the logistic function on large intervals as described in Section 4.1. We implement our domain-extension methodology for the logistic function, and by using it, we perform the logistic regression based on HE. We adopt CKKS scheme [16, 17] among many HE schemes since it supports floating-point operation on real numbers.

We introduce two experiments: (1) homomorphic evaluation of the logistic function on large domain intervals by using our methods, and (2) privacy-preserving training of logistic regression by using our method.

We approximately evaluate the logistic function on various domain intervals by using each method. Our experiments include approximation for both moderate (i.e., maximum error≈ 0.045) and high (i.e., maximum error $\leq 2^{-20}$) accuracy.

For the privacy-preserving logistic regression, we used two datasets: MNIST and Swarm Behavior datasets. For the MNIST dataset, all data belong to a bounded interval, $[0, 255]^{28\times 28}$, and we approximate the logistic function on a sufficiently large input domain that contains all the possible inputs. The Swarm Behavior dataset, on the other hand, contains the data of values without an explicit bound. We approximate the logistic function on a sufficiently large domain interval, $[-7683, 7683]$. We stress that for the Swarm Behavior dataset, as we observed the intermediate values during the training in an unencrypted state, the approximation interval of the logistic function should be large enough, i.e., larger than $[-1333, 1333]$.

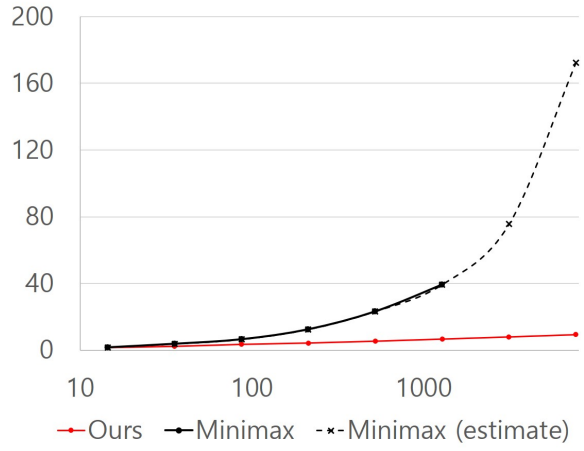

Fig. 8: Runtime(s) for the evaluation of the logistic function on various domain intervals using two different methods: minimax approximation and Algorithm 1. The x-axis is logarithmic scaled.

5.1 Homomorphic Evaluation of the Logistic Function

In this experiment, we demonstrate the efficiency of our methods to homomorphically evaluate the logistic function on large domain intervals.

For a given fixed maximum error, we evaluate the logistic function based on HE on large intervals by using (1) our methods and (2) minimax approximation with Paterson-Stockmeyer algorithm. We compare the methods on various sizes of domain intervals in terms of accuracy and runtime.

For the minimax approximation, we use the minimax polynomial with the smallest degree among the minimax polynomials with the maximum norm error less than 0.045, and we evaluate the minimax polynomial with Paterson-Stockmeyer algorithm with Chebyshev basis [8]. To find the minimax polynomials, we used chebfun library [21]. For significantly large domain intervals, it fails to find appropriate minimax polynomials, so we estimate the cost by (1) estimating the degree of the minimax polynomial for the given large domain interval, and (2) measuring the cost for the evaluation of a random odd polynomial of the estimated degree.

For the CKKS parameters for our methods and the minimax approach, we took $N = 2^{16}$ with initial ciphertext modulus $q_L = 2^{1081}$. Hamming weight of the secret polynomial is set to 128. Note that these parameters achieve 128-bits security [27–29].

The experiments were performed on Intel Xeon Silver 4114 CPU at 2.20GHz processor. We used a single thread for the experiments.

5.1.1 Moderate accuracy In this experiment, we compare Algorithm 1 to the minimax approximation. For both methods, we measure the runtime for the

Size of domain	Minimax Max error	Minimax Runtime (sec)	Algorithm 1 Max error	Algorithm 1 Runtime (sec)
29	0.04416	1.57	0.04416	1.57
71	0.04161	3.72	0.04441	2.30
174	0.04389	6.42	0.04445	3.51
426	0.04376	12.43	0.04446	4.33
1045	0.04477	23.18	0.04446	5.44
2560	0.04479	39.28	0.04446	6.72
6272	-	(75.54)	0.04446	8.04
15366	-	(172.46)	0.04446	9.43

Table 4: The maximum errors and runtime of our method and minimax approximation for the homomorphic evaluation of the logistic function on various domain intervals with a moderate accuracy.

homomorphic evaluation of the logistic function with the maximum norm error of less than 0.045.

We measured the evaluation time of each method under various sizes of the domain intervals, from $[-14.5, 14.5]$ to $[-7683, 7683]$, increasing 2.45 times each.

For Algorithm 1, we used a DEP $B(x) = x - \frac{4}{27}\frac{1}{14.5^2}x^3$ with extension factor 2.45. We extend the domain interval of $P(x)$, the degree 9 minimax polynomial of the logistic function on $[-14.5, 14.5]$. Note that the maximum errors of extended polynomial approximations are all less than 0.045.

We also compared our result to the Pegasus [14, 30]. To practically evaluate a function $f(\cdot)$ on a large domain interval $[-R, R]$ using Pegasus, we evaluated the function $f(Rx/8)$ on $[-8, 8]$. We used the default parameters in the open source: for CKKS, $N = 2^{16}$ with ciphertext modulus 2^{599}, and for programmable TFHE, $N_{lwe} = 1024$ and $N_{lut} = 4096$. We used 2 CKKS ciphertexts with 256 slots, and measure the amortized runtime for a single input. We point out that, in contrast to [14], we measured *the maximum error* (which will be captured near 0) rather than the average error.

Results. We report the experimental result in Figure 8, Table 4 and 5. Figure 8 shows the runtime in seconds of each method to evaluate the logistic function on interval $[-x, x]$. Note that The x-axis is logarithmic scaled.

Our method is more efficient than the minimax approach. To uniformly approximate the logistic function on $[-1279, 1279]$, ours consumed 6.72 seconds, and it is $5.85\times$ faster than the minimax approach that consumed 39.28 seconds. For the larger domain interval, $[-7683, 7683]$, ours is $18.29\times$ faster than the estimated time of minimax approach.

Also, the time difference between our method and the minimax approach becomes larger as the size of the domain interval grows. In Figure 8, we can observe that the runtime for the minimax approach increases rapidly as the size of the domain interval grows. On the other hand, the runtime for our method

Size of domain	Pegasus [14]		Algorithm 1	
	Max error	Amortized time (ms)	Max error	Amortized time (ms)
29	0.01470	837.01	0.04416	0.048
71	0.02617		0.04441	0.070
174	0.04968		0.04445	0.107
426	0.12189		0.04446	0.132
1045	0.33802		0.04446	0.166
2560	0.73180		0.04446	0.205
6272	0.96092		0.04446	0.245
15366	0.97499		0.04446	0.288

Table 5: The maximum errors and amortized runtime of our method and [14] for the evaluation of the logistic function at a real number from various sizes of domain intervals.

increases linearly with respect to $\log R$. This matches to our expectation in Section 4.1.

Our method uses a bit larger multiplicative depth than the minimax approach; for example, in the case of the experiment on $[-1279, 1279]$, the multiplicative depth of our method is 14 while that for the minimax approach is 11. However, the multiplicative depths for both methods are asymptotically the same (see Section 4.1).

Our method provides an easy implementation with reasonable precision bits. In contrast, the minimax polynomial has the coefficients with high precision, and in our implementation, it fails to work for the domain interval larger than $[-1279, 1279]$.

The accuracy of Pegasus decreases as the size of domain increases, and Pegasus is not accurate for large domain intervals. We remark that the logistic function values belong to $[0, 1]$. Also, since Pegasus cannot utilize the SIMD operations of CKKS scheme, the amortized runtime is slower than ours. We note that our method evaluates the logistic function at all slots of a given CKKS ciphertext (16384 real number values) by using SIMD operation of CKKS scheme.

5.1.2 High Accuracy In this experiment, we compare Algorithm 1, 2 and the minimax approximation for the high precision evaluations. For Algorithm 2 and the minimax approach, we measure the runtime for the HE evaluation of the polynomial approximations with the maximum norm error less than 2^{-20}. Also, we compare Algorithm 1 and 2, and observe how Algorithm 2 effectively reduces the approximation error.

For Algorithm 1 and 2, we used a DEP $B(x) = x - \frac{4}{27}\frac{1}{55^2}x^3$ with extension factor 2. We extend the domain interval of $P(x)$, the degree 243 minimax polynomial of the logistic function on $[-55, 55]$. Note that the maximum errors of extended polynomial approximations are all less or equal to 2^{-20}.

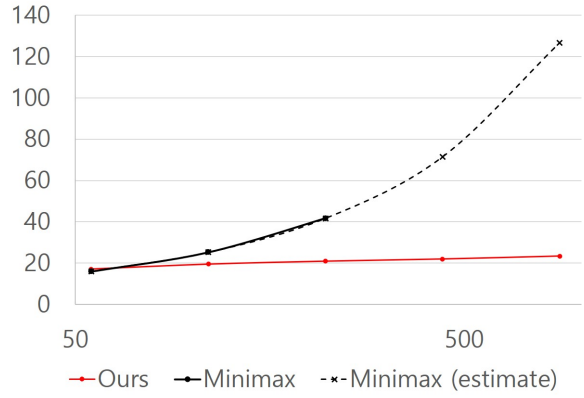

Fig. 9: Runtime(s) for the evaluation of the logistic function on various domain intervals using two different methods: minimax approximation and Algorithm 2. The x-axis is logarithmic scaled.

	Minimax		Algorithm 1		Algorithm 2	
Size of Domain	Error ($\log_2$)	Time (sec)	Error ($\log_2$)	Time (sec)	Error ($\log_2$)	Time (sec)
110	-20.0	16.0	-21.6	17.1	**-21.6**	17.2
220	-20.1	25.2	-14.0	17.7	**-20.3**	19.5
440	-20.0	41.7	-13.7	18.8	**-20.1**	20.9
880	-	(71.4)	-13.6	20.0	**-20.0**	22.0
1760	-	(126.7)	-13.6	21.3	**-20.0**	23.4

Table 6: The precision bits and runtime of our methods and minimax approximation for the homomorphic evaluation of the logistic function on various domain intervals with a high accuracy.

Results. We report the experimental result in Figure 9 and Table 6. Figure 9 shows the runtime in seconds of each method to evaluate the logistic function on interval $[-x, x]$. Note that The x-axis is logarithmic scaled.

As in the experiments for the moderate accuracy, our method is more efficient than the minimax approach. To uniformly approximate the logistic function on $[-440, 440]$, ours consumed 20.9 seconds, which is $2\times$ faster than the minimax approach that consumed 41.7 seconds. For the larger domain interval, $[-880, 880]$, ours is $5.42\times$ faster than the estimated time of minimax approach.

Also, Algorithm 2 outperformed Algorithm 1 in terms of the accuracy. While Algorithm 1 did not provide a high precision approximation (i.e. $\approx 2^{-20}$), Algorithm 2 did. Even though Algorithm 1 is a bit faster, the cost is asymptotically the same, and the time difference is relatively small.

5.2 Logistic Regression on MNIST Dataset

By using our method (Algorithm 1), we train the logistic regression classifier based on HE.

In these experiments, we use Algorithm 1 with the same setting in Section 5.1.1. More precisely, we use the DEP $B(x) = x - \frac{4}{27}x^3$ of extension factor 2.45, and we extend the domain interval of $P(\cdot)$, the degree 15 minimax polynomial of the logistic function on $[-14.5, 14.5]$. We utilize at most 7 times of domain-extension processes, which extend the approximation interval to about $[-7683, 7683]$.

For CKKS parameters, we took $N = 2^{17}$ with initial ciphertext modulus $q_L = 2^{2150}$. Hamming weight of the secret polynomial is set to 128. Note that these parameters achieve 128-bits security [27–29].

All experiments were performed on Intel Xeon CPU E5-2620 v4 at 2.10GHz processor. Also, 8 threads were used for the experiments.

5.2.1 MNIST MNIST is a dataset of handwritten digits, which contains 60000 training samples and 10000 test samples [31]. In this experiment, we selected the samples with two labels, 3 and 8, to perform the binary classification using logistic regression. For the training set, the first 9600 training samples with a label of either 3 or 8 were selected. For the test set, all test samples with a label of either 3 or 8 were selected.

By using our method, we train logistic regression classifier based on the CKKS scheme. We used a mini-batch stochastic gradient descent with mini-batch size 320. We iterated 30 times, which is 1 epoch. We used two learning rates, 0.1 and 1.0, and compared two results.

Selection of approximation intervals. All samples of MNIST belong to a bounded interval (i.e., $[0, 1]^{28\times 28}$). Thus, we can calculate the upper bound of the size of the norm of the trained weight by logistic regression for each iteration. The polynomial approximations on such domain intervals guarantee the success of privacy-preserving logistic regression training without exceeding the domain of the polynomial approximation.

Let $w_t \in \mathbb{R}^D$ be trained weight vector(including bias component) by logistic regression, after t iterations. Also, α be the learning rate, n be the mini-batch size, $D = d + 1$, d be the number of attributes of the sample, and $x \in \mathbb{R}^D$ be an arbitrary single sample that satisfies that every component belongs to $[0, 1]$. Note that the last component of

Results. We report the results in Table 7. Each row describes the state of logistic regression model after the given number of iterations. The second and fourth columns, (accuracy(0.1), accuracy(1.0)), indicate the accuracy of the (encrypted) logistic regression model with learning rate 0.1 and 1.0 respectively. The third and fifth columns, (maxinput (0.1), maxinput (1.0)), show the maximal size of inputs of the logistic function during each training iteration. The result shows us two points as follows.

# of iterations	Accuracy (0.1)	Max input (0.1)	Accuracy (1.0)	Max input (1.0)
3	91.12%	1.22	50.90%	38.3
6	89.16%	2.06	85.13%	25.2
9	92.13%	2.37	93.85%	16.3
12	92.13%	3.00	94.35%	17.0
15	92.33%	3.35	94.55%	16.1
18	93.29%	3.60	94.70%	16.3
21	93.80%	3.39	96.06%	12.1
24	94.15%	4.02	96.11%	14.8
27	94.15%	3.69	95.61%	12.5
30	94.65%	4.17	96.11%	13.4

Table 7: The result of MNIST dataset training with learning rate 0.1 and 1.0.

- The learning rate of 1.0 shows better performance, but the input values of the logistic functions during training become larger. Thus, too narrow approximation intervals (e.g., $[-8, 8]$ or less as in the previous works) would restrict us from selecting the optimal hyperparameters.
- We cannot ensure that the input value for the logistic function varies gently. As seen in our experiment with learning rate 1.0, maxinput value varies largely at the early stage of the training, even though maxinput value became about $10 - 15$ at the later iterations. This shows us that just moderately extending the interval of approximation (such as $[-8, 8]$ to $[-16, 16]$) cannot be a fundamental solution.

5.2.2 Swarm Behavior Dataset Swarm Behavior Dataset is a dataset for determining 3 behaviors of each swarm sample. The dataset contains 24016 swarm samples, which have 3 binary behavior labels that indicate whether the swarm is (1) aligned, (2) flocking, and (3) grouped. Each swarm comprises 200 individuals, and each individual is characterized by 12 features such as position and velocity; in other words, each sample swarm has 2400 real-valued features.

In this experiment, we solve the binary classification of 'aligned' label by using logistic regression. For the training set, we randomly pick 3840 samples among 24016 samples, and the other samples are used for the test set.

We used the mini-batch stochastic gradient descent with mini-batch size 240. We iterated 16 times, which is 1 epoch. We set 10^{-6} for the learning rate. Also, we approximate all the logistic functions on $[-7683, 7683]$, by using DEP $B(x) = x - \frac{4}{27}x^3$ with extension factor 2.45. We extend the domain interval of $P(x)$, the degree 15 minimax polynomial of the logistic function on $[-14.5, 14.5]$.

We train the logistic regression classifier in encrypted and unencrypted states, and compare the trained model. To compare two logistic regression models, we measure the relative error. More precisely, for to logistic regression classifiers w_p and w_c, we measure the difference between them by $\|w_p - w_c\|/\|w_p\|$ in L^2-norm.

# of iterations	Accuracy (ciphertext)	Accuracy (plaintext)	Relative error	Max input
1	78.60%	78.60%	0.05%	0.0
2	82.90%	83.05%	1.32%	10.0
3	85.88%	85.88%	1.83%	72.0
4	88.52%	88.53%	1.81%	6.3
5	89.60%	89.72%	4.47%	86.2
6	90.38%	90.39%	3.50%	40.6
7	92.62%	92.65%	5.65%	1358.5
8	93.11%	93.15%	4.69%	154.3
9	93.27%	93.27%	6.54%	353.2
10	94.01%	94.02%	6.37%	33.4
11	94.67%	94.61%	6.25%	129.6
12	94.52%	94.51%	6.13%	284.6
13	94.69%	94.71%	5.67%	176.8
14	95.15%	95.14%	8.32%	1221.2
15	95.55%	95.51%	7.89%	1843.3
16	95.78%	95.72%	7.52%	26.6

Table 8: The result of Swarm Behavior dataset training.

Results. Table 8 shows the experimental results of logistic regression training on the Swarm Behavior dataset. Each row describes the trained model after the given number of iterations. The second and third columns show the accuracy of the model trained in encrypted and unencrypted states respectively. The fourth column, relative error, indicates the difference between two models that are trained in encrypted and unencrypted states. Finally, the last column, max input, is the maximal size of inputs of the logistic function during each training iteration. We point out two points as follows.

- Our approximation of the logistic function on a large interval behaves similarly to the exact logistic function. In particular, the accuracy of the HE-based trained model is almost the same as the plaintext-based trained model. Also, we can observe that the relative error between the models is small.
- The polynomial approximation on a substantially large approximation interval was necessary. There are several large inputs of the logistic function during the training, so the polynomial approximation of the logistic function on a large interval (e.g., $[-7000, 7000]$) had a key role.

6 Conclusion

In this work, we propose a new efficient method for homomorphic evaluation on large intervals. For the stable computation based on HE, an efficient method for HE evaluation on large intervals is crucial. For instance, to train a logistic

regression model based on HE, it is necessary to approximate the logistic function on a large domain interval; e.g, larger than $[-1843, 1843]$ for the Swarm Behavior dataset.

Our method evaluates on a large domain interval $[-R, R]$ a function that converges at infinities, with $O(\log R)$ multiplications and $O(1)$ memory space. This is both asymptotically and practically more efficient than the previous approach, the minimax approximation and Paterson-Stockmeyer algorithm, which uses $O(\sqrt{R})$ multiplications and $O(\sqrt{R})$ space. Also, our algorithm can be easily extended to functions whose differences from a low-degree polynomial converge to real-valued constants at infinities. We leave an efficient method for the homomorphic evaluation of arbitrary functions on large intervals as an open problem.

Acknowledgements This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-00840, Development and Library Implementation of Fully Homomorphic Machine Learning Algorithms supporting Neural Network Learning over Encrypted Data)

References

1. J. H. Cheon, D. Kim, Y. Kim, and Y. Song, "Ensemble method for privacy-preserving logistic regression based on homomorphic encryption," *IEEE Access*, vol. 6, pp. 46 938–46 948, 2018.
2. A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon, "Logistic regression model training based on the approximate homomorphic encryption," *BMC medical genomics*, vol. 11, no. 4, p. 83, 2018.
3. H. Chen, R. Gilad-Bachrach, K. Han, Z. Huang, A. Jalali, K. Laine, and K. Lauter, "Logistic regression over encrypted data from fully homomorphic encryption," *BMC medical genomics*, vol. 11, no. 4, p. 81, 2018.
4. K. Han, S. Hong, J. H. Cheon, and D. Park, "Logistic regression on homomorphic encrypted data at scale," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 9466–9471.
5. R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, 2016, pp. 201–210.
6. E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *arXiv preprint arXiv:1711.05189*, 2017.
7. M. S. Paterson and L. J. Stockmeyer, "On the number of nonscalar multiplications necessary to evaluate polynomials," *SIAM Journal on Computing*, vol. 2, no. 1, pp. 60–66, 1973.
8. H. Chen, I. Chillotti, and Y. Song, "Improved bootstrapping for approximate homomorphic encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2019, pp. 34–54.
9. J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No, "High-precision bootstrapping of rns-ckks homomorphic encryption using optimal minimax polynomial approxiamtion and inverse sine function," *IACR Cryptol. ePrint Arch*, vol. 2020, 2020.

10. J.-P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Hubaux, "Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys," *IACR Cryptol. ePrint Arch*, vol. 2020, p. 1203, 2020.
11. J. H. Cheon, D. Kim, and D. Kim, "Efficient homomorphic comparison methods with optimal complexity." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1234, 2019.
12. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Tfhe: fast fully homomorphic encryption over the torus," *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, 2020.
13. I. Chillotti, M. Joye, and P. Paillier, "Programmable bootstrapping enables efficient homomorphic inference of deep neural networks," in *International Symposium on Cyber Security Cryptography and Machine Learning*. Springer, 2021, pp. 1–19.
14. W.-j. Lu, Z. Huang, C. Hong, Y. Ma, and H. Qu, "Pegasus: Bridging polynomial and non-polynomial evaluations in homomorphic encryption," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1057–1073.
15. C. Boura, N. Gama, M. Georgieva, and D. Jetchev, "Chimera: Combining ring-lwe-based fully homomorphic encryption schemes," *Journal of Mathematical Cryptology*, vol. 14, no. 1, pp. 316–338, 2020.
16. J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 409–437.
17. J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 360–384.
18. J. H. Cheon, K. Han, S.-M. Hong, H. J. Kim, J. Kim, S. Kim, H. Seo, H. Shim, and Y. Song, "Toward a secure drone system: Flying with real-time homomorphic authenticated encryption," *IEEE access*, vol. 6, pp. 24 325–24 339, 2018.
19. J. H. Cheon, D. Kim, and J. H. Park, "Towards a practical cluster analysis over encrypted data," in *International Conference on Selected Areas in Cryptography*. Springer, 2019, pp. 227–249.
20. R. Pachón and L. N. Trefethen, "Barycentric-remez algorithms for best polynomial approximation in the chebfun system," *BIT Numerical Mathematics*, vol. 49, no. 4, p. 721, 2009.
21. T. A. Driscoll, N. Hale, and L. N. Trefethen. (2014) Chebfun guide. Oxford.
22. C. S. Burrus, J. W. Fox, G. A. Sitton, and S. Treitel, "Horner's method for evaluating and deflating polynomials," *DSP Software Notes, Rice University, Nov*, vol. 26, 2003.
23. M. Fasi, "Optimality of the paterson–stockmeyer method for evaluating matrix polynomials and rational matrix functions," *Linear Algebra and its Applications*, vol. 574, pp. 182–200, 2019.
24. M. Kim, Y. Song, S. Wang, Y. Xia, X. Jiang *et al.*, "Secure logistic regression based on homomorphic encryption: Design and evaluation," *JMIR medical informatics*, vol. 6, no. 2, p. e8805, 2018.
25. A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10," *Unpublished manuscript*, vol. 40, no. 7, pp. 1–9, 2010.
26. A. Eremenko and P. Yuditskii, "Polynomials of the best uniform approximation to sgn (x) on two intervals," *Journal d'Analyse Mathématique*, vol. 114, no. 1, p. 285, 2011.
27. J. H. Cheon, Y. Son, and D. Yhee, "Practical fhe parameters against lattice attacks," Cryptology ePrint Archive, Report 2021/039, 2021, https://eprint.iacr.org/2021/039.

28. M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *Journal of Mathematical Cryptology*, vol. 9, no. 3, pp. 169–203, 2015.
29. M. R. Albrecht. (2017) A Sage Module for estimating the concrete security of Learning with Errors instances. https://bitbucket.org/malb/lwe-estimator.
30. "Openpegasus," https://github.com/Alibaba-Gemini-Lab/OpenPEGASUS, accessed: 2022-05-01.
31. Y. LeCun and C. Cortes, "MNIST handwritten digit database," http://yann.lecun.com/exdb/mnist/, 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/