Secure and non-interactive k-NN classifier using symmetric fully homomorphic encryption

Yulliwas Ameur, Rezak Aziz, Vincent Audigier, Samia Bouzefrane

Secure and non-interactive k-NN classifier using symmetric fully homomorphic encryption

Yulliwas Ameur[1][0000−0003−1435−2982], Rezak Aziz[1][0000−0002−9637−4220], Vincent Audigier[1][0000−0002−4169−7866], and Samia Bouzefrane[1][0000−0002−0979−1289]

CEDRIC Lab, Cnam, 292 rue Saint Martin, Paris, 75141, France
yulliwas.ameur@lecnam.net

Abstract. "Machine learning as a service" (MLaaS) in the cloud accelerates the adoption of machine learning techniques. Nevertheless, the externalization of data on the cloud raises a serious vulnerability issue because it requires disclosing private data to the cloud provider. This paper deals with this problem and brings a solution for the K-nearest neighbors (k-NN) algorithm with a homomorphic encryption scheme (called TFHE) by operating on end-to-end encrypted data while preserving privacy. The proposed solution addresses all stages of k-NN algorithm with fully encrypted data, including the majority vote for the class-label assignment. Unlike existing techniques, our solution does not require intermediate interactions between the server and the client when executing the classification task. Our algorithm has been assessed with quantitative variables and has demonstrated its efficiency on large and relevant real-world data sets while scaling well across different parameters on simulated data.

Keywords: k-nearest neighbors · homomorphic encryption · TFHE · data privacy · IoT · cloud computing · privacy-preserving.

1 INTRODUCTION

Cloud services have become central to data storage and data exploitation. Among these services, a machine-learning service is offered to train different models to predict for decision-making purposes. However, this raises the issue of data security because the data processed by the cloud may be sensitive and confidential while belonging to entities that do not trust the cloud provider.

The most commonly used cryptographic techniques are secret sharing, multiparty computation, and homomorphic encryption (HE) to achieve privacy-preserving for machine learning. Several techniques ensure strong privacy protection, often coming at the expense of reduced in terms of speed and communication.

homomorphic encryption is an encryption technique that allows computations directly on encrypted data. The results are encrypted and can be revealed/decrypted only by the owner of the secret key. This principle is very useful in many domains where data sharing and privacy preservation are required. For example, externalizing personal data considered as sensitive such as

medical data or banking transactions [10]. This paper considers a scenario where sensitive data are collected by IoT devices and outsourced on a resourceful cloud for Machine Learning (ML) processing. Considering that the cloud provider is not trustworthy, we propose using HE to preserve privacy.

ML over encrypted data has been particularly investigated in the domain of neural networks (NN). However, as state-of-the-art non-linear NN layers (pooling and activation) are too complex to be directly executed in the encrypted world, there is a strong need for approximating them. Much of the subsequent works have been proposed to address the limitation of implementing the non-linear activation function by a polynomial approximation using Taylor series and Chebyshev polynomials, among others [5, 7, 1, 3]. Due to the high cost of the HE systems, those methods do not scale well and are not appropriate for deep neural networks [12], which are time-consuming due to the great depth of the network.

One way to solve the computational cost problem inherent to HE is to investigate less complex supervised methods. The k-nearest neighbors (k-NN) approach presents several advantages. Indeed, for a predefined number of neighbors k, the model does not require any training step, the value of the response variable for a given individual is obtained directly from the values observed on the neighbors without needing to estimate new parameters. In addition, the method can handle continuous, categorical, and mixed data. Furthermore, as a non-parametric method, k-NN can be relevant for many data structures as long as the number of observations is sufficiently large.

The prediction for a new observation is obtained by:

- Identifying the k nearest neighbors (according to a given distance)
- Computing the majority class among them (for a classification problem) or by averaging values (for a regression problem).

HE has been recently investigated by various authors for k-NN [9, 13, 14, 16].

[9] suggested a Homomorphic additive encryption scheme [11]. They investigated the privacy preservation in an outsourced k-NN system with various data owners. The untrusted entity securely computes the computations of distances by using HE. However, the comparison and classification phases require interactions. Given that the computational and communication difficulties scale linearly, they admit that the method may not be practical for massive data volumes. The cost of communications between the entities is also a limitation in the deployment of this work [13].

[14], used an "asymmetric scalar-product-preserving encryption" (ASPE). However, the client has the ciphertext, and the server can decrypt it. The proposed solution is vulnerable to Chosen Plaintext Attacks as stated by [15].

Recently, [16] proposed a secure k-NN algorithm in quadratic complexity concerning the size of the database completely non-interactively by using a fully homomorphic encryption [6]. However, they assume that the majority vote is done on a clear-text domain, which is a significant security flaw that we will address here. Doing a majority vote on a clear-text domain imposes interaction between entities, which causes information leakage.

Unlike other existing works, in this paper, we propose a new methodology to apply k-NN on encrypted data by using fully homomorphic encryption avoiding interaction between entities.

We consider a client/service provider architecture that supports scenarios described here. The service provider is a company that owns a labeled training dataset D composed of sensitive data allowing predict for novel observation by k-NN. This model is offered as a service.

We assume a context that is concerned by some privacy issues as in the following:

- Because the training data are sensitive, they cannot be shared with a third party such as a client.
- The model is an intellectual property of the service provider. Hence, the service provider does not want to share the used model with his clients.
- A client who needs to perform classification on the service-provider platform does not trust the service provider.

This work aims to do a classification using k-NN algorithm on sensitive data using HE. Our solution assumes that the service provider, which is the dataset owner, has all the necessary resources to perform the data classification and storage. This assumption ensures that encrypting the training dataset is not necessary since these data are kept with the data owner. Only the client will need to encrypt his query that includes his data by using a private key and by sending it to the data owner for classification. The goal is to protect the training dataset, the query, and the model parameter k. Our solution meets the following privacy requirement as in the following:

- The contents of D are known only by the data owner since they are not sent to other parties.
- The client's query is not revealed to the data owner.
- The client knows only the predicted class.
- The index of the k nearest neighbors is unknown from the data owner or the client.

Our solution has a greater added value than the existing literature solutions. First, it guarantees that no information leakage occurs during the process: the only things known by the data owner are the dataset and the model used. The only things that the client knows are the query and the class. All intermediate results are encrypted. In addition, our solution is fully non-interactive since prediction is performed by the data owner and do not need any decryption during the process. Finally, it supports multi-label classification.

The rest of this paper is organized as follows: Section 2 presents the background of HE before highlighting the principle of Fast Fully homomorphic encryption over the Torus (TFHE). Our proposed solution is then described in Section 3. A simulation study is presented in Section 4 to assess our methodology based on real datasets. Finally, Section 5 concludes the paper.

2 BACKGROUND

2.1 homomorphic encryption

HE allows a third party (a service provider in our scenario) to compute functions on ciphertexts to preserve the confidentiality of the data. An encryption scheme is called homomorphic over an operation $*$ if it supports the following property:

$$E(m_1) * E(m_2) = E(m_1 * m_2)$$

where E is the encryption algorithm and m_1, m_2 belong to M the set of all possible messages. An additional algorithm is needed for homomorphic encryption schemes, called the *Eval* algorithm. This algorithm is defined as follows:

$$Eval(f, C_1, C_2) = f(m_1, m_2)$$

where $Dec(C_1) = m_1$ and $Dec(C_2) = m_2$ and f is a function that determines the type of the HE scheme. In case f supports the evaluation of arbitrary functions for an unlimited number of times, the HE is called *Fully* homomorphic encryption (FHE).

This paper uses "TFHE: Fast Fully homomorphic encryption over the Torus" as an RLWE-based scheme in his fully homomorphic setting, especially in gate bootstrapping mode. The bootstrapping procedure is the homomorphic evaluation of a decryption circuit on the encryption of a secret key.

2.2 Functional Bootstrap in TFHE

TFHE defines three types of ciphertexts, TLWE Sample, TRLWE Sample and TRGSW Sample.

There are two bootstraps algorithms in TFHE. Gate Bootstrap was introduced to implement logic gates, and the Circuit Bootstrap, which converts TLWE samples to TRGSW samples. In our work, we use Functional Bootstrap. By "Functional" we mean that the bootstrap can evaluate functions using a BlindRotate algorithm to perform a lookup table (Lut) evaluation. LUTs are a simple, efficient way of evaluating discretized functions. For instance the sign function was used in [2] and [8].

3 OUR CONTRIBUTION

3.1 The System Model

Our system uses the client-server architecture (see Figure 1). The client is the querier, and the server is the data owner.

1. The data owner: owns the data and can do heavy calculations. For example, it receives the query in an encrypted way, performs an encrypted k-NN algorithm then sends the result to the querier for decryption.
2. The querier: generates the keys, encrypts the query that contains its data and sends it to the data owner for computations before decrypting the result. The querier can be an ordinary computer or any IoT device that collects data.

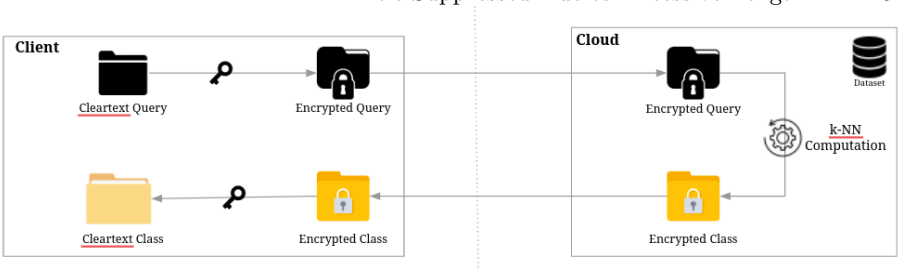

Fig. 1: The system model: the client is the querier, and the server is the data owner: the data owner receives the query in an encrypted way, performs an encrypted k-NN algorithm then sends the result to the querier for decryption.

3.2 Encrypted k-NN Challenges

In order to propose an encrypted version of k-NN, we should substitute the challenging operations used in the standard k-NN with equivalent operations in encrypted domains. As seen before, k-NN is composed of three parts: the distance calculation, the distance sorting, the selection of the k nearest neighbors, and the majority vote.

This subsection will introduce the equivalent operations as integrated with our solution.

Distance Calculation The euclidean distance calculation between the dataset entries x_i as well as the query q are necessary in order to find the k nearest neighbors to the query. We can use the standard formula of the distance as in (1).

$$d^2(x_i, q) = \sum_{j=0}^{p} x_{ij}^2 + \sum_{j=0}^{p} q_j^2 - 2 * \sum_{j=0}^{p} x_{ij} q_j \tag{1}$$

What is relevant in our case is the difference between two distances to compare them. So, we get the formula (2) as follows:

$$d^2(x_i, q) - d^2(x_{i'}, q) = \sum_{j=0}^{p} (x_{ij}^2 - x_{i'j}^2) - 2 * \sum_{j=0}^{p} (x_{i'j} - x_{ij}) q_j \tag{2}$$

Since the dataset is a clear text, we can easily calculate formula (2) using the TFHE scheme. However, we need to adapt it. Using TFHE, the difference between the distances should be in the range of $[-\frac{1}{2}, \frac{1}{2}]$. Another constraint is that the multiplication is done between a clear-text integer and a ciphertext. Two rescaling values are required to resolve these constraints. Let v be the first one. It is used to have values of the differences between $[-\frac{1}{2}, \frac{1}{2}]$. Let p be the second one. It indicates the precision of the differences. Each attribute of the dataset as well as the query are rescaled using v. p is used when calculating the product $(x_{i'j} - x_{ij})q_j$.

Sorting Sorting computed distances is a crucial step in k-NN. The standard algorithm for sorting, like the bubble sort, can be used while considering encrypted data. However, these algorithms are time-consuming in an encrypted world because the worst case is computed every time. The authors [4] propose two methods to sort an array of values. The method of the direct sort is used in [16]. It is based on a matrix of comparison called delta matrix:

$$\begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,n} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,n} \end{pmatrix}$$

with

$$m_{i,j} = \bar{sign}(X_i - X_j) = \begin{cases} 1 \text{ if } X_i < X_j \\ 0 \text{ else.} \end{cases}$$

When summing columns of this matrix, we will have a sorting index of the distances.

Majority Vote The majority vote can cause a problem because the operation requires comparison to detect the class. To determine the predicted class, we need to know k nearest neighbors' classes. This step is challenging in two ways: first, we need to avoid information leakage, unlike the solutions in the literature. Second, the majority vote requires comparison in order to predict the class.

To the best of our knowledge, no solution in the literature studied this point in an encrypted way without information leakage. Therefore, in the following subsection, we will demonstrate a solution to process k-NN with the majority vote in an encrypted way while supporting a multi-label classification.

3.3 Our proposed k-NN algorithm

Our proposed algorithm, called "HE-kNN-V", is composed of three steps: the construction of the delta matrix, the selection of k-nearest neighbors, and the majority vote. The two first steps are similar to the solution of [16] even if we adapt the existing formulas in order to eliminate unnecessary calculations. [16] use polynomials to define the formulas, while what interests us is just one term of those polynomials to eliminate un-necessary calculations. The majority vote is our added value and is specific to our solution. We will discuss in this subsection the design of our solution, including each building block.

Building the delta matrix To build the delta matrix, we need to know the sign of the differences between the distances to sort. Since we defined a method to calculate the differences in the last subsection, the sign can easily be achieved using the standard bootstrapping sign function in TFHE. However, the standard bootstrapping function returns +1 if the phase is greater than 0 and -1 if the

phase is lower than 0. Therefore, since we need to have 0 or 1 in the matrix, we need to adapt the bootstrapping operation to return $\frac{1}{2}$ and $-\frac{1}{2}$ then by adding $\frac{1}{2}$ to the result we will have 0 or 1.

Even if building this matrix is time-consuming, it is highly parallelizable.

Selecting the *k*-nearest neighbors To select the k–nearest neighbors, we use the scoring operation proposed by Zuber [16]. By using the delta matrix, the principle is as follows:

1. Sum m values in each column with m the number of possible operations without bootstrapping.
2. If there are still values to sum: do a bootstrapping operation using the modified sign bootstrapping function (See Algorithm 1 in [16]) and go to Step 1.
3. Otherwise, execute the modified sign bootstrapping and return the last sign returned by this operation.

Finally, we obtain an encrypted vector where the position i equals the cipher of 1 if the individual with index i is among the k-nearest neighbors, the cipher of 0 otherwise. We call this vector the "mask" (See Figure 2 for more clarity).

Majority vote The majority vote is the most important added value in our work. We propose to do the majority vote without any leakage of information, unlike existing works like that of [16] in which the majority value is done in clear text or by using other alternative solutions proposed in the literature.

First, we illustrate the issue with the method of [16]. We consider the scenario where the querier does the calculations. The majority vote is done in clear text, but we need to decrypt the vector of indexes of the nearest neighbor. The data owner does the decryption. Significant information leakage occurs if the data owner knows the vector of indexes. Then, he will know the classification of the query, and by doing some triangulation, he can approximate the query. In addition, the solution will be interactive. If we consider the scenario where the data owner does the calculation, the decryption of the vector is done by the querier. However, to do the classification, the querier should know the labels of the dataset, which is also a critical leakage of information. In addition, the querier will know the size of the dataset and the k parameter of nearest neighbors considered. This information is considered as internal information of the model used, and it should be protected.

In our solution, the majority vote is done by the data owner in an encrypted way. First, the data owner encodes the labels using one hot encoding. Having the mask and the matrix of labels in one hot form, it is easy to do an AND operation between the mask and each column of the labels, as in Figure 2. We get a matrix A (for affectation) with A_{ij} equal to 1 if the individual i is among the k-nearest neighbors and its class is j. Using this matrix, it is possible to sum the columns and obtain the probability of each class. We can now return only the class and guarantee no information leakage and no interactivity.

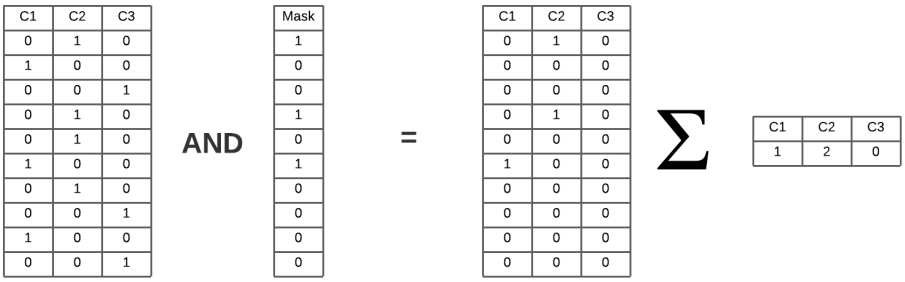

Fig. 2: Majority Vote illustration by using the mask

4 PERFORMANCE EVALUATION

In this section, we discuss the experiments of our solution. First, we describe the technical and the setup of the environment. Then, we will evaluate the performances of our solution according to different criteria: execution time, accuracy, bandwidth consumption.

4.1 Test Environment

Setup Our solution is implemented using the TFHE scheme in C/C++ and Python for training k-NN in clear text and for tests. To test the effect of parallelism, we used OpenMP to do some parallelization. The source code is available in the following github "https://github.com/Yulliwas/HE-kNN-V". Our solution is tested on Linux Ubuntu 64-bit machine with i7-8700 CPU 3.20GHz.

Table 1 shows the parameters used to setup TFHE scheme.

λ	**N**	σ
110	1024	10^{-9}

Table 1: TFHE Parameters: λ for the overall security, N for the size of the polynomials,σ for the Gaussian noise parameter.

m	v	p	b
64	4	1000	4*m-4

Table 2: HE-kNN Parameters: the number of operations m without needing a bootstrapping, the bootstrapping base b, and the rescaling factors v and p

Datasets To test our solution, we choose to use 6 datasets: Iris, Breast Cancer, Wine, Heart, Glass and MNIST as in Table 3. The goal is to test the performances of our algorithm in different distributions of data, so that to confirm that our solution works with any dataset and that has performances that are equivalent to those of clear-text domains.

Dataset	n	d	classes
Iris	150	4	3
Wine	178	13	3
Heart	303	75	5
Breast Cancer	699	10	2
Glass	214	10	2
MNIST	1797	10	3

Table 3: Datasets:
number of individuals(n), the size of the model (d) and number of classes

Simulation procedure First, we preprocess the data by rescaling each attributes to a value between 0 and 1. Our dataset and the query should be rescaled by a factor of v as seen above. We must also multiply the dataset vectors by the precision factor τ and then rounded. In the other hand, the query vector is divided by this same factor. To obtain the classification rate, first we need to divide our dataset to a training set and a test set. We choose to use 20% of our dataset as a test set and the rest as a training set. Among the training set, we select a certain number of points that represent as well as possible our dataset. The process for choosing the best points that represent our training set is as follows:

1. choose n individuals randomly ;
2. calculate the classification rate ;
3. Repeat the previous Step 1 and Step 2 a certain amount of time and keep the best accuracy and the best individuals.

To select the k parameter, we use the same procedure as in the clear domain. In our case, we tested different values of k and we keep the best k value that gives the best results.

4.2 Performance results

To position our approach according to existing works, and especially regarding the voting step that is performed without information leakage, we compare in Table 4 our solution with Zuber's solution and with a clear-text version based on the Iris dataset and a fixed k=3. The comparison is done in terms of complexity (C), Information Leakage (L), accuracy (A), interactivity (I) and execution time (T). The accuracy and the prediction time are indicated only when it is possible.

Empirical study

Classification rate To evaluate the classification rate, we have chosen the accuracy instead of other metrics like: recall or F1-score. We studied the accuracy according to two parameters: the number of data sampled from the dataset and the number k of neighbors. The goal is to choose the best points that represent the datasets and the best k parameters for each dataset.

Work	C	L	I	A	T
HE-kNN-V	$O(n^2)$	N	N	0.97	1.72s
HE-kNN-VP	$O(n^2)$	N	N	0.97	0.46s
Zuber	$O(n^2)$	Y	Y	0.98	1.74s
Clear k-NN	$O(n)$	Y	N	0.95	1.8ms

Table 4: Comparison between solutions for Iris Dataset: complexity (C), Information Leakage (L), accuracy (A), interactivity (I) and execution time (T).

We chose real-world datasets in order to see the evolution of the accuracy and compared it to clear-text accuracy.

In one hand, we know that the accuracy depends on the k parameter and we can confirm it easily in the graphs. On the other hand, the assumption that the accuracy depends on the number of data used is not complete. For the dataset where the data is well separated (like Iris), having a lot of data is not necessary, the best accuracy can be achieved using only few data. But, in the case where data is not well separated (like in Heart dataset), the accuracy seems to depend on the number of data.

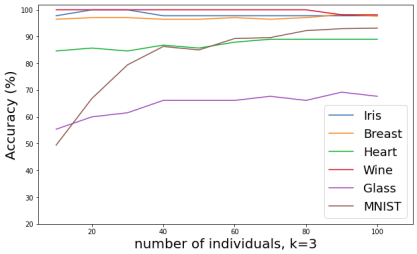

Fig. 3: Encrypted Accuracy vs number of individuals

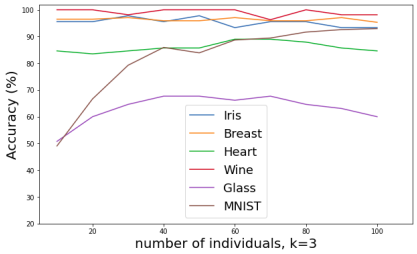

Fig. 4: Clear-text Accuracy vs number of attributes

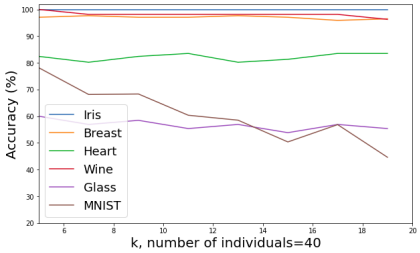

Fig. 5: Encrypted Accuracy vs k-parameter

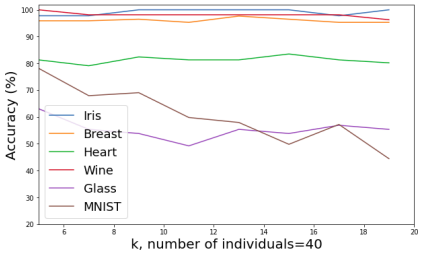

Fig. 6: Clear-text Accuracy vs k-parameter

According to our different simulations illustrated in Figure 3 and Figure 4, we do not lose accuracy when we apply our HE-kNN-V method on the encrypted data compared to the application of the kNN on the plain data. This is possible by varying the number of individuals and by fixing k to 3.

We also notice that by setting the number of individuals to 40 and varying k, (see Figure 5 and Figure 6) the accuracy behaves in the same way between the application of the kNN on the plain data and the application of our method HE-kNN-V on the encrypted data.

Execution time In our solution, the execution time is independent of the content of the dataset, it does not depend on the values, but does depend on the content, since it depends on the number of tuples . We can use either simulated dataset or real world dataset. To visualize the evolution of the execution time according to k, n and d, we choose to use the Breast Cancer dataset instead of simulating a new dataset. We change n, k, d and we see the evolution of the execution time.

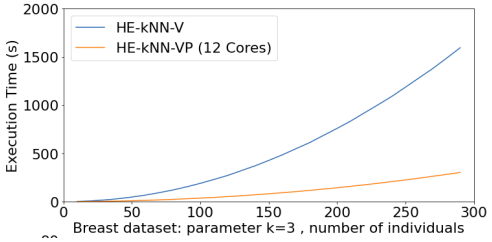

Fig. 7: Execution time vs number of individuals

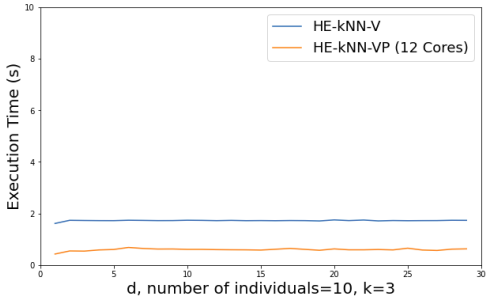

Fig. 8: Execution time vs number of attributes

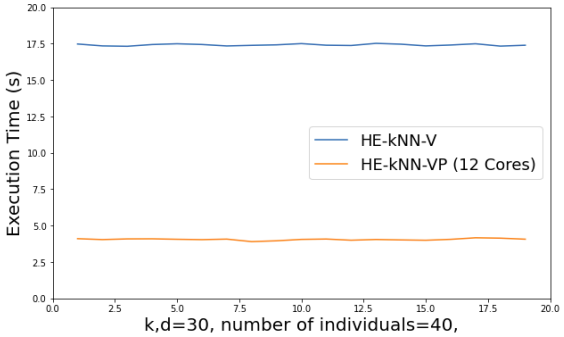

Fig. 9: Execution time vs k-parameter

Our simulations, as depicted in Figure 7, illustrate that HE-kNN-V is parallelizable, and also that the number of individuals strongly impacts the execution time unlike the two simulations of Figure 8 and Figure 9 where the variation of respectively d the number of attributes and k does not impact the execution time.

Bandwidth In our solution, the only thing that is communicated is the query in the ciphertext and the response in the ciphertext. The size of the query is proportional to the number of attributes d. Each attribute is a TLWE Sample with the size of 4 KB and the size of the response (number of classes)*4 KB. The bandwidth according to each dataset is illustrated in Table 5.

Table 5: Bandwidth

Dataset	**Bandwidth (KB)**
Iris	28
Wine	64
Heart	64
Breast Cancer	128
Glass	60
MNIST	296

Discussion According to our experiments, we can say that the accuracy in our case depends on three factors: the number of individuals, the representativity of these individuals and the k parameter. To have a better model that fits our dataset, we must select the individuals that are more representative of our dataset and the best k parameter. We also should take care of the number of individuals because most of the execution time depends on that number.

5 CONCLUSION

We proposed HE-kNN-V a method for performing k-NN on encrypted data that includes a majority vote for class-label assignment. The proposed solution addresses all stages of k-NN algorithm with fully encrypted data. It guarantees that no information leakage occurs during the process. Unlike other techniques, our solution eliminates the need for intermediate interactions between the server and the client when performing classification tasks. Our algorithm has been evaluated using quantitative variables and demonstrated its efficiency on large and relevant real-world data sets. As a perspective, it would be interesting to see how a hardware acceleration of the TFHE scheme could improve the computation time of our proposed solution HE-kNN-V.

References

1. Ahmad Al Badawi, Chao Jin, Jie Lin, Chan Fook Mun, Sim Jun Jie, Benjamin Hong Meng Tan, Xiao Nan, Khin Mi Mi Aung, and Vijay Ramaseshan Chandrasekhar. Towards the alexnet moment for homomorphic encryption: Hcnn, the first homomorphic CNN on encrypted data with gpus. *IEEE Trans. Emerg. Top. Comput.*, 9(3):1330–1343, 2021.
2. Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. *Fast Homomorphic Evaluation of Deep Discretized Neural Networks: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part III*, pages 483–512. 01 2018.
3. Alon Brutzkus, Oren Elisha, and Ran Gilad-Bachrach. Low latency privacy preserving inference. *ArXiv*, abs/1812.10659, 2019.
4. Gizem S Çetin, Yarkın Doröz, Berk Sunar, and Erkay Savaş. Depth optimized efficient homomorphic sorting. In *International Conference on Cryptology and Information Security in Latin America*, pages 61–80. Springer, 2015.
5. H. Chabanne, Amaury de Wargny, J. Milgram, Constance Morel, and E. Prouff. Privacy-preserving classification on deep neural network. *IACR Cryptol. ePrint Arch.*, 2017:35, 2017.
6. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption library, August 2016. https://tfhe.github.io/tfhe/.
7. Ehsan Hesamifard, H. Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *ArXiv*, abs/1711.05189, 2017.
8. Malika Izabachène, Renaud Sirdey, and Martin Zuber. *Practical Fully Homomorphic Encryption for Fully Masked Neural Networks*, pages 24–36. 10 2019.
9. Frank Li, Richard Shin, and Vern Paxson. Exploring privacy preservation in outsourced k-nearest neighbors with multiple data owners. In *Proceedings of the 2015 ACM Workshop on Cloud Computing Security Workshop*, CCSW '15, page 53–64, New York, NY, USA, 2015. Association for Computing Machinery.
10. Oliver Masters, Hamish Hunt, Enrico Steffinlongo, Jack Crawford, and F. Bergamaschi. Towards a homomorphic machine learning big data pipeline for the financial services sector. *IACR Cryptol. ePrint Arch.*, 2019:1113, 2019.
11. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. volume 5, pages 223–238, 05 1999.
12. Luis Pulido-Gaytan, Andrei Tchernykh, Jorge Cortés-Mendoza, Mikhail Babenko, Gleb Radchenko, Arutyun Avetisyan, and Alexander Drozdov. Privacy-preserving neural networks with homomorphic encryption: Challenges and opportunities. *Peer-to-Peer Networking and Applications*, 14, 05 2021.
13. Bharath K. Samanthula, Yousef Elmehdwi, and Wei Jiang. k-nearest neighbor classification over semantically secure encrypted relational data. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1261–1273, 2015.
14. Wai Kit Wong, David Wai-lok Cheung, Ben Kao, and Nikos Mamoulis. Secure knn computation on encrypted databases. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, page 139–152, New York, NY, USA, 2009. Association for Computing Machinery.
15. Xiaokui Xiao, Feifei Li, and Bin Yao. Secure nearest neighbor revisited. In *Proceedings of the 2013 IEEE International Conference on Data Engineering (ICDE 2013)*, ICDE '13, page 733–744, USA, 2013. IEEE Computer Society.
16. Martin Zuber and R. Sirdey. Efficient homomorphic evaluation of k-nn classifiers. *Proceedings on Privacy Enhancing Technologies*, 2021:111 – 129, 2021.