Evaluation of Homomorphic Encryption for Privacy in Principal Component Analysis

David Arnold and Jafar Saniie

Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago IL, U.S.A.

Abstract– **Principal Component Analysis (PCA) is a versatile Unsupervised Learning (UL) technique for reducing the dimensionality of datasets. As a result, PCA is widely used in consumer and research applications as a preprocessing tool for identifying important features prior to further analysis. In instances where on-site personnel or developers do not have the expertise to apply UL techniques, third party processors are frequently retained. However, the release of client or proprietary data poses a substantial security risk. This risk increases the regulatory and contractual burden on analysts when interacting with sensitive or classified information. Homomorphic Encryption (HE) cryptosystems are a novel family of encryption algorithms that permit approximate addition and multiplication on encrypted data. When applied to UL models, such as PCA, experts may apply their expertise while maintaining data privacy. In order to evaluate the potential application of Homomorphic Encryption, we implemented Principal Component Analysis using the Microsoft SEAL HE libraries. The resulting implementation was applied to the MNIST Handwritten dataset for feature reduction and image reconstruction. Based on our results, HE considerably increased the time required to process the dataset. However, the HE algorithm is still viable for non-real-time applications as it had an average pixel error of near-zero for all image reconstructions.**

Keywords– Unsupervised Learning, Principal Component Analysis, Homomorphic Encryption

I. Introduction

Over the past decade, advancements in Machine Learning and Artificial Intelligence, coupled with Big Data Analytics, have resulted in tailored customer experiences. However, these conveniences took a heavy toll on user privacy, consuming personal data at an alarming rate. Similarly, companies without ML expertise began to rely on third-party experts to conduct analysis. In cases where sensitive or confidential data is the subject of analysis, transfer to third parties increases the risk of data exposure or leaks.

In response, many regulatory agencies and consumer watchdogs are pushing for stronger privacy safeguards and security standards. Within the United States, the Federal Trade Commission frequently brings enforcement actions against companies that fail to properly protect consumer privacy. While there are no general privacy or security laws in the US, the FTC applies their authority under Section 5 of the FTC Act, considering breaches of consumer privacy as deceptive or unfair commercial acts. The agency has brought over 80 general privacy lawsuits since 2002, including a \$5 billion penalty against Facebook for misrepresenting user control over personal data [1, 2]. Abroad, the General Data Protection Regulation (GDPR) includes explicit protections for consumer personal data and even extends protection to third party processors [3]. After one year of coming in effect in 2018, 91 fines had been levied from GDPR enforcement actions [4]. Of these fines, 56 dealt with the handling of personal information, ranging from improper processing, lawful processing, and secure processing.

To address these challenges, Homomorphic Encryption (HE) is a promising tool for enhancing data privacy while also maintaining access to popular ML algorithms. HE cryptosystems are a new class of encryption that permit approximate addition and multiplication operations on encrypted data. By using these simple operations, complex ML algorithms can be applied to a given dataset without compromising the privacy of the underlying data. In our previous work, we explored the potential application of HE on a Neural Network [5]. While the implementation had a large storage and computation overhead, it was still suitable for non-real-time applications. This work extends our previous work by exploring a Principal Component Analysis (PCA) implementation. PCA was selected as it is commonly used in feature reduction, which can be a powerful asset for decreasing the number of connections required in ML models. Further, PCA can be used to reconstruct the image (we will refer to this as image reconstruction) with a reduced number of features. This can be useful in highlighting defects or imperfections [6-8].

Through the remainder of this paper, we will evaluate Homomorphic Encryption for Principal Component Analysis. First, we will discuss Homomorphic Encryption and the Microsoft SEAL libraries. Next, Principal Component Analysis for feature reduction and image reconstruction will be presented. PCA will be applied to the MNIST Handwritten dataset. Finally, we will compare the time required to apply PCA on both encrypted and plaintext datasets.

II. Homomorphic Encryption

Currently, the common use-case for HE is for computations in a cloud infrastructure, as presented in Figure 1. During operation, a client encrypts their data and sends it to a remote server. In our case, we will be encrypting images from the MNIST Handwritten Dataset. Next, the server will apply the desired algorithm on the encrypted dataset in HE. After receiving the result and decrypting the data, the client receives the desired output. For our application, we will be applying Principal Component Analysis for both feature reduction and image reconstruction.

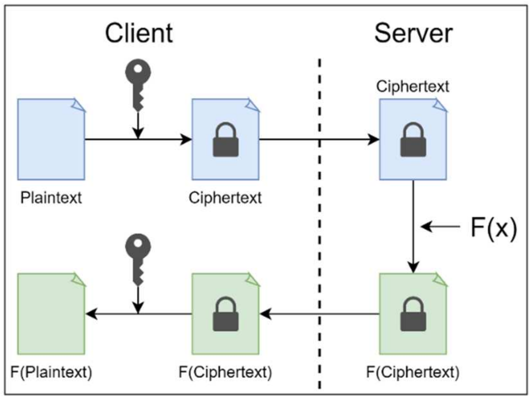

Fig. 1. Potential application of Homomorphic Encryption using Cloud infrastructure.

For our research, we have selected the Microsoft SEAL libraries to implement Principal Component Analysis (PCA) in Homomorphic Encryption (HE) [9]. The SEAL libraries are a popular tool for software developers to quickly apply HE without needing to implement the complex mathematics. Currently, C++ and python are supported, however since the libraries are written in C++, we selected C++. Three encryption schemes are provided with the toolkit, namely BFV, BGV, and CKKS. BFV and BGV permit modular arithmetic operations with encrypted integers whereas CKKS can be used for approximate additions and multiplications on encrypted real or complex numbers [10]. Since we will be standardizing our dataset for PCA training, we used the CKKS scheme as it allowed us to use double values.

Through our previous work, we explored the potential application of HE on a simple Neural Network (NN) and observed potential challenges in adoption [5]. Among these challenges included limitations regarding the number of computations that can be completed, the size of the encrypted data, and the computation time. The root cause of these limitations occurs during initialization of the CKKS algorithm, in which the ciphertext space for operations is set by a poly-modulus degree. During a multiplication operation, the size of ciphertext needs to grow linearly and must be rescaled and re-linearized before completing the next operation. While this rescaling process limits the potential error size, it also sets a hard limit on the number of operations we can completed. In the case of our Neural Network implementation, we needed to complete 4 multiplications, which required a poly-modulus degree of 16,384. For our PCA implementation, we only need to complete 1 matrix multiplication for feature reduction and 2 matrix multiplications for image reconstruction. This translated to a poly-modulus degree of 8192. This should positively impact the storage requirements for our planned implementation.

In addition to challenges presented by the poly-modulus degree, the complexity of operations is also limited to additions and multiplications. As a result, operations such as comparisons and divisions, common among activation functions, need to be approximated. This was required in our NN approach, in which we used polynomial approximation to model the ReLU function. However, since PCA only requires multiplication and transposition, we did not need to approximate any functions this time around. Further, this makes training of our models infeasible in the Homomorphic Encryption domain. As a result, we will complete training via plaintext operations while the application of the trained model will be completed through HE.

Recently, there has been an increase in similar works applying Homomorphic Encryption to Machine Learning Applications. Applications include Decision Trees, Neural Networks, and Convolutional Neural Networks [11-18]. Additionally, similar work has been undertaken regarding Principal Component Analysis [19-20]. Our contribution expands the literature on this subject by completing a full comparison between the computation time of the HE and plaintext implementations, analyzing the average error in the results when using the CKKS scheme, and reconstructing the images using a reduced feature space.

III. Principal Component Analysis

Principal Component Analysis (PCA) is a popular unsupervised learning technique for feature reduction and image reconstruction. During operation, PCA attempts to transform the data into a coordinate system where the data is better described with fewer dimensions. This is accomplished by maximizing the variance between each feature. Since the training process is far too complex for our Homomorphic Encryption toolkit to handle, we will only briefly describe the steps that we took.

For our application, we conducted our training using python, which streamlined our development process through the numpy and scipy libraries. To begin, we standardized our training data to a mean of 0 and a standard deviation of 1, which will need to be applied to our testing data prior to encryption. Next, we calculated the covariance matrix between the features, which describes the correlation of the features. Ideally, we want to find a plane that describes the most features, which can be accomplished by calculating the eigenvalues and eigenvectors of the covariance matrix. Each eigenvector describes a spread of data, and the eigenvalue describes the magnitude of that spread. The eigenvalue with the greatest magnitude represents the greatest spread in the dataset, which is what we are looking for. At this stage, we may select an arbitrary number of eigenvectors (which translates to the number of features we want) or we may set a cumulative contribution that we are aiming for. Contribution is calculated by dividing each eigenvalue by the sum of all eigenvalues. By taking certain eigenvalues and eigenvectors, we can determine a cumulative contribution towards the overall dataset. For instance, our results indicated that for images in the MNIST Handwritten Dataset, 28% of the information was retained within 10 features. The trained model was then composed of the selected eigenvectors.

After completing training, we then explored feature reduction and image reconstruction. In order to reduce the number of features, we multiplied our principal component vector (E) into the transpose of our standardized input vector (X). This process is described by Equation 1. This was a simple matrix multiplication and could be transferred to the Homomorphic Encryption Domain. Image reconstruction was completed by taking the reduced feature matrix and multiplying it by the transpose of the principal component vector. This is described by Equation 2, with Y being the result from Equation 1. An example of image reconstruction with different numbers of features is presented in Figure 2. As we increase the number of features that are considered, the image starts to get clearer. In the future, we anticipate applying image reconstruction to highlight defects in Nondestructive Evaluation of Additive Manufacturing.

$$Y = EX^T \tag{1}$$
$$Z = YE^T \tag{2}$$

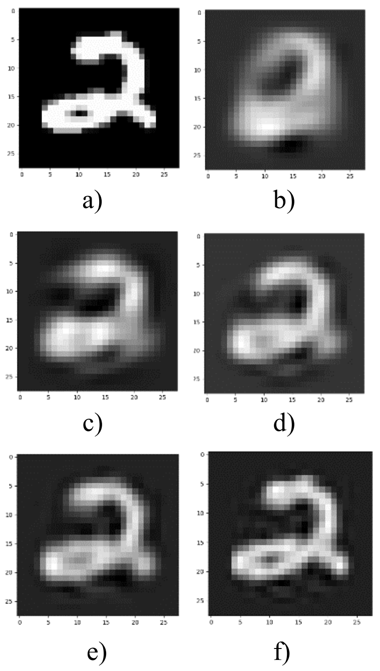

Fig. 2. Image reconstruction of a handwritten 2 from the MNIST Handwritten Dataset. a) Initial Image b) 10 Features c) 25 Features d) 75 Features e) 150 Features f) 300 Features.

IV. Results and Analysis

We selected the MNIST Dataset as it is widely available and simple to work with. The dataset is composed of 42,000 training images and 28,000 testing images. Each image was a 28x28 grayscale image of a handwritten digit, between 0 and 9. We took each pixel as a feature, bringing our initial features to 784 features. Since we were applying PCA as an unsupervised learning model, we opted to train our dataset on the entirety of the 42,000 images of the training dataset. However, when it came to applying the trained model, we only used 2000 images from the testing dataset. This was done as the HE implementation required a considerable amount of time compared to the plaintext implementation. Our implementation was tested on an 8-core Intel Xeon E-2234 processor that operated at 3.60 GHz and had 16 GB of available RAM. For a fair comparison of our results, both the HE and plaintext implementations were written in C++.

During testing, our PCA model was applied to the images and computation times were recorded for encryption, feature reduction, reconstruction, and decryption. Each of these values were calculated as an average on a per-image basis as this would be expected during run-time. Encryption and decryption remained consistent across all calculations with an average encryption time of 2.89 seconds and an average decryption time of 1.74 seconds per image. This was expected as the input size and output size of all images were going to remain constant at 784 pixels.

We've isolated results for feature reduction into Table I and reconstruction in Table II since they are used for different applications. In general, we found that the computation time increased linearly as additional features were reintroduced. This was expected as each additional feature would only increase the number of operations by one multiplication and one addition. We've also included the cumulative contribution based on the number of features that were used. This is beneficial as the cumulative contribution did not grow linearly and selecting a lower set of features requires less computation time.

TABLE I. PRINCIPAL COMPONENT ANALYSIS FEATURE REDUCTION IN HOMOMORPHIC ENCRYPTION AND PLAINTEXT

Features	**Homomorphic Encryption**	**Plaintext**	**Cumulative Contribution**
	Reduction Time (s)	Reduction Time (s)	
1	0.758	0.000426	0.0575
5	3.002	0.000489	0.1914
10	5.797	0.000575	0.2808
25	14.248	0.000879	0.4256
50	28.236	0.001259	0.5590
75	42.226	0.001710	0.6471
150	84.213	0.003083	0.8092
300	168.632	0.005701	0.9423
500	281.167	0.009953	0.9867
700	393.37	0.014022	0.9999

Reviewing our results for Feature Reduction, it's clear that Homomorphic Encryption greatly increases the computation required to apply the PCA model. At the low end, the HE implementation was 2,000 times slower (at 1 feature) than the plaintext version and the high end, it was 60,000 times slower (at 600 features). As a result, it is recommended that HE be applied to non-real-time applications. To combat the large computation times, an operator could select a lower number of features as the cumulative contribution did not follow a linear relationship. Following this logic, we've highlighted three rows pertaining to 10, 50, and 150 features and correspond to 28%, 56%, and 81%. Despite having a low number of features compared to the overall 784, they offer far better computation time compared to higher numbers of features that offer only incremental increases in cumulative contributions. Additionally, we've included a visual comparison between the Reduction Time and Cumulative Contribution compared to the Number of Features. As is presented in Figure 3, the Reduction Time is linearly related to the Number of Features while the Cumulative Contribution is logarithmic.

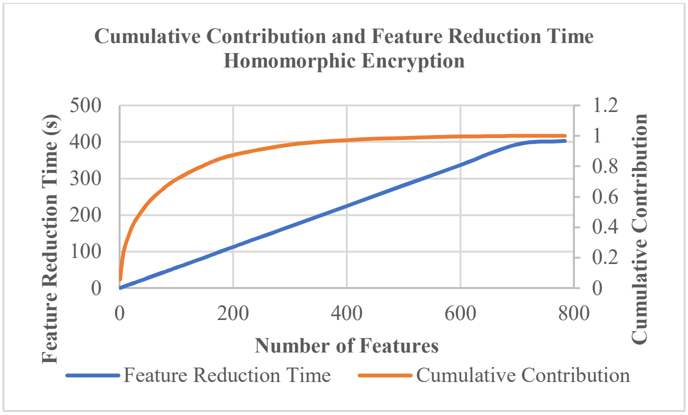

Fig. 3. Cumulative Contribution and Feature Reduction Time (in Homomorphic Encryption) based on the Number of Features used. Based on the graph, the Feature Reduction Time was linearly related to the Number of Features while the Cumulative Contribution had a Logarithmic relationship.

In addition to comparing the computation time for our feature reduction, we also examined our image reconstruction times (presented in Table II). This yielded a very similar trend, with the Homomorphic Encryption implementation being associated with a far greater time consumption. In a similar manner, we've highlighted 10, 50, and 150 features as they are good benchmarks for operators and engineers looking to strike a balance between performance and time consumption. Since the CKKS scheme of HE applies approximate addition and multiplication, we also wanted to compare the average pixel error between our HE and plaintext implementations. Based on our results, the HE implementation had near zero error across all 784 pixels with the greatest average pixel error at 2.57E-5. We've included a visual comparison between the Image Reconstruction and Cumulative Contribution compared to the Number of Features. As is presented in Figure 4, the Image Reconstruction is linearly related to the Number of Features while the Cumulative Contribution is logarithmic.

TABLE II. PRINCIPAL COMPONENT ANALYSIS IMAGE RECONSTRUCTION IN HOMOMORPHIC ENCRYPTION AND PLAINTEXT

Features	**Homomorphic Encryption**	**Plaintext**	**Average Pixel Error**	**Cumulative Contribution**
	Reconstruction Time (s)	Reconstruction Time (s)		
1	1.234	0.000426	2.74E-5	0.0575
5	4.927	0.000489	1.18E-5	0.1914
10	9.532	0.000575	8.72E-6	0.2808
25	23.471	0.000879	2.43E-5	0.4256
50	46.587	0.001259	1.87E-5	0.5590
75	69.824	0.001710	2.17E-5	0.6471
150	139.375	0.003083	2.22E-5	0.8092
300	278.913	0.005701	1.55E-5	0.9423
500	465.632	0.009953	2.42E-5	0.9867
700	651.675	0.014022	2.57E-5	0.9999

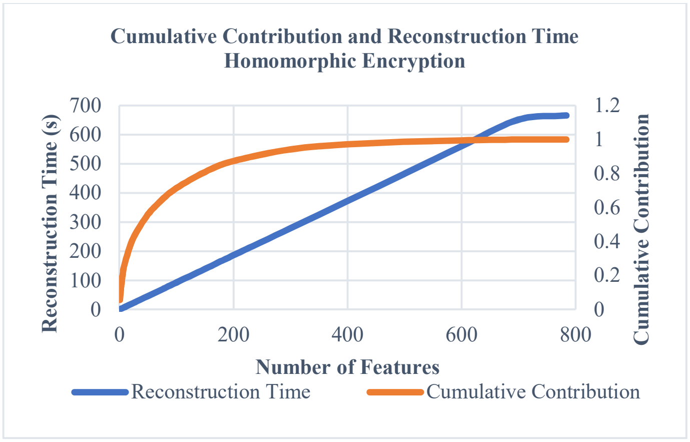

Fig. 4. Cumulative Contribution and Image Reconstruction Time (in Homomorphic Encryption) based on the Number of Features used. Based on the graph, the Image Reconstruction Time was linearly related to the Number of Features while the Cumulative Contribution had a Logarithmic relationship.

V. CONCLUSION

Overall, we were successful in applying a Principal Component Analysis model in Homomorphic Encryption. While the overall computation times for the HE implementation were drastically greater than their plaintext models, they exhibited near-zero error in their computations. For non-real-time applications, operators may balance performance and computation time by selecting a lower number of features, capturing 50% of the data in as little as 50 features (out of 784). Additional time savings can be made by exploring additional features of the Microsoft SEAL libraries, as different vector rotation options may permit faster matrix multiplication. In the future, we will pair our PCA implementation with a Neural Network for classification purposes. Additionally, we are interested in exploring HE applications in the space of Nondestructive Evaluation (NDE) as the data is sensitive in nature and frequently involves third party experts.

REFERENCES

[1] Federal Trade Commission (FTC) , "Federal Trade Commission 2020 Privacy and Data Security Update," 2020.

[2] Federal Trade Commission (FTC), "FTC Imposes $5 Billion Penalty and Sweeping New Privacy Restrictions on Facebook," FTC, 24 July 2019. [Online]. Available: https://www.ftc.gov/news-events/news/press-releases/2019/07/ftc-imposes-5-billion-penalty-sweeping-new-privacy-restrictions-facebook.

[3] B. Wolford, "What is GDPR, the EU's New Data Protection Law?," Proton Technologies , 2023. [Online]. Available: https://gdpr.eu/what-is-gdpr/.

[4] C. Barrett, "Emerging Trends from the First Year of EU GDPR Enforcement," American Bar Association, 28 February 2020. [Online]. Available: https://www.americanbar.org/groups/science_technology/publications/scitech_lawyer/2020/spring/emerging-trends-the-first-year-eu-gdpr-enforcement/.

[5] D. Arnold, J. Saniie and A. Heifetz, "Homomorphic Encryption for Machine Learning and Artificial Intelligence Applications," Argonne National Lab, 2022.

[6] X. Zhang, J. Saniie, S. Bakhtiari and A. Heifetz, "Compression of Pulsed Infrared Thermography Data with Unsupervised Learning for Nondestructive Evaluation of Additively Manufactured Metals," *IEEE Access,* vol. 10, pp. 9094-9107, 2022.

[7] X. Zhang, J. Saniie and A. Heifetz, "Detection of Defects in Additively Manufactured Stainless Steel 316L with Compact Infrared Camera and Machine Learning Algorithms," *JOM,* vol. 72, no. 12, pp. 4244-4253, 2020.

[8] X. Zhang, J. Saniie, W. Cleary and A. Heifetz, "Quality Control of Additively Manufactured Metallic Structures with Machine Learning of Thermography Images," *JOM,* vol. 72, no. 12, pp. 4682-4694, 2020.

[9] Microsoft Research, *Microsoft SEAL,* Redmond, Washington, 2022.

[10] 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*, 2017.

[11] E. Hesamifard, H. Takabi and M. Ghasemi, "CryptoDL: Deep Neural Networks over Encrypted Data," 2017. [Online]. Available: https://arxiv.org/abs/1711.05189v1.

[12] S. Meftah, B. H. M. Tan, C. F. Mun, K. M. M. Augn, B. Veeravalli and V. Chandrasekhar, "DOReN: Toward Efficient Deep Convolutional Neural Networks with Fully Homomorphic Encryption," *IEEE Transactions on Information Forensics and Security,* vol. 16, pp. 3740-3752, 2021.

[13] E. Lee, J.-W. Lee, J. Lee, Y.-S. Kim, Y. Kim, J.-S. No and W. Choi, "Low-Complexity Deep Convolutional Neural Network on Fully Homomorphic Encryption Using Multiplexed Parallel Convolutions," in *International Conference on Machine Learning*, 2022.

[14] B. Pulido-Gaytan, A. Tchernykh, J. Cortes-Mendoza, M. Babenko, G. Radchenko, A. Avetisyan and A. Y. Drozdov, "Privacy-Preserving Neural Networks with Homomorphic Encryption: Challenges and Opportunities," *Peer-to-Peer Networking and Applications,* vol. 14, no. 3, pp. 1666-1691, 2021.

[15] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim and J.-S. No, "Privacy-Preserving Machine Learning with Fully Homomorphic Encryption for Deep Neural Network," *IEEE Access,* vol. 10, pp. 30039-30054, 2022.

[16] X. Sun, P. Zhang, J. K. Liu, J. Yu and W. Xie, "Private Machine Learning Classification Based on Fully Homomorphic Encryption," *IEEE Transactions on Emerging Topics in Computing,* vol. 8, no. 2, pp. 352-364, 2018.

[17] K. Cong, D. Das, J. Park and H. V. L. Pereira, "SortingHat; Efficient Private Decision Tree Evaluation via Homomorphic Encryption and Transcribing," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[18] R. Hamza, A. Hassan, A. Ali, M. B. Bashir, S. M. Alqhtani, T. M. Tawfeeg and A. Yousif, "Towards Secure Big Data Analysis via Fully Homomorphic Encryption Algorithms," *Entropy,* vol. 24, no. 519, pp. 1-17, 2022.

[19] S. Panda, "Principal Component Analysis using CKKS Homomorphic Scheme," in *Cyber Security Cryptography and Machine Learning: 5th International Symposium, CSCML 2021*, 2021.

[20] W.-j. Lu, S. Kawasaki and J. Sakuma, "Using Fully Homomorphic Encryption for Statistical Analysis of Categorical, Ordinal and Numerical Data," IACR Cryptography ePrint Archive, 2016.