Evaluation of Concrete ML for Secure Viral Strain Classification with Homomorphic Encryption

Johann Benjamin Vivas[1] and Richard Bryann Chua*[1,2]

[1] Department of Physical Sciences and Mathematics,
University of the Philippines Manila, Manila, Philippines
{jpvivas,rlchua}@up.edu.ph,
[2] Department of Computer Science,
University of the Philippines Diliman, Quezon City, Philippines

Abstract. Machine learning (ML) techniques are increasingly being used in viral strain classification. Along with this increase use, it becomes more practical to outsource these machine learning computations to the cloud. However, there are privacy issues that surround the outsourcing of viral genomic data. Hence, we can use homomorphic encryption to address these privacy issues. In our work, we used Concrete ML to perform viral strain classification with machine learning using homomorphic encryption. We evaluated the performance of the models developed with Concrete ML and shown that its performance in classification is comparable to those of normal ML libraries. However, its performance in terms of computational time is slower than normal ML libraries.

Keywords: fully homomorphic encryption, viral strain classification, logistic regression, Concrete ML

1 INTRODUCTION

During the COVID-19 pandemic last 2020, sharing of viral genome data becomes important. Although there are privacy concerns [46], sharing of viral genomic sequencing data allows researchers to better understand the SARS-CoV-2 virus. This in turn leads to development of clinical therapies and vaccines. While sharing viral genome data without contextual data was thought to be safe, various studies that analyse these data suggest that apart from SARS-CoV-2 viral sequences, some samples also contained human DNA [2, 37, 15]. This raises the concern that SARS-CoV-2 sequences obtained from patients might contain human DNA or RNA data, which increases the risk of reidentification [32, 33]. Hence, there is a need to protect these genomic data when they are outsourced to the cloud for computation. One potential solution to this privacy problem is the use of homomorphic encryption [5, 11, 14], which allows computations to be performed directly on encypted data without the need to decrypt it.

In our work, we adopted the machine learning workflow of Sim et al. used in [45] and used Concrete ML in place of the usual machine learning library.

J. Caro et al. (eds.), Proceedings of the Workshop on Computation: Theory and Practice (WCTP 2024) , Atlantis Highlights in Computer Sciences 23,
https://doi.org/10.2991/978-94-6463-684-0_12

We evaluated Concrete ML as a tool to perform a privacy-preserving viral strain classification with homomorphic encryption. In Section 2, we review some recent applications of homomorphic encryption, including how it is used in privacy-preserving viral strain classification. Homomorphic encryption and its use with machine learning, and viral strain classification are discussed in Section 3. In Section 4 we discussed how we performed training on a Concrete ML logistic regression model for viral strain classification and discuss the results in Section 5. The contribution of this paper is on the evaluation of the use Concrete ML for performing viral strain classification using homomorphic encryption.

2 RELATED WORKS

In recent years, Machine Learning (ML) has been used in the diagnosis of viral diseases. It has been used in the classification of human papillomaviruses (HPV), hepatitis B viruses (HBV), human immunodeficiency viruses type 1 (HIV-1), porcine reproductive and respiratory syndrome virus (PRRSV) sublineages [40, 28]. During the COVID-19 pandemic, ML was also used in the classification of SARS-CoV-2 viral strains [34, 16].

When these viral genome data are outsourced to the cloud for ML computation, their privacy can be protected with the use of Fully Homomorphic Encryption (FHE). FHE has already been applied on various applications like privacy-preserving genome-wide association studies [14, 30], genotype imputation [42, 29, 42], and federated analytics [19]. FHE is also used in the password monitor feature of Microsoft Edge which allows Microsoft to detect if a password is compromised without seeing the plaintext password. More recently, FHE has also been integrated into different aspects of machine learning and deep learning [7, 48, 31], and applied to specific classification problems including text classification [8], and tumor classification [21]. Such a wide range of FHE applications with ML continues to expand with the use of existing libraries such as Microsoft SEAL, IBM HElayers, TenSEAL, Pyfhel, and Concrete implemented in varying programming languages like C++ and Python [36, 35, 24, 10, 22, 23, 51].

Performing viral strain classification using homomorphic encryption was posted as a problem in the iDASH 2021 competition (http://www.humangenomeprivacy.org/2021/competition-tasks.html). Akavia et al. [26, 3, 43, 4] submitted a solution to this problem using Microsoft SEAL and HElayers libraries. In their solution, they first computed the DNA sequence's k-mer set and encrypted this with an FHE scheme that supports the encryption of complex numbers before it is sent to the cloud. The classification is performed based on the k-mer sets derived from the DNA sequences, and the Jaccard similarity between a particular strain in the data and various sequences representative of each of the different SARS-CoV-2 viral strains. Once Jaccard similarity scores were successfully computed homomorphically for the k-mer sets and normalized, the strain with the highest similarity score is the classification of that particular strain. Another solution to this problem is the CoVnita framework of I2R's A*FHE-2 Team [45] where they built a logistic regression model using federated learning

and homomorphic encryption where various data providers train their own local models using their data, after which a joint global model that does not require data providers to share their genomic samples, is trained. The model's FHE capabilities were powered by Microsoft SEAL. Both of these two solutions involve developing everything from scratch including the algorithm for ML training.

3 BACKGROUND

3.1 Homomorphic Encryption

The concept of homomorphic encryption was introduced in 1978 by Rivest et al. in [41]. However, it was more than 30 years later in 2009, that Craig Gentry [20] proposed the very first FHE scheme and showed that this is possible. While Gentry's FHE scheme was extremely slow, it helped kickstart the development of other FHE schemes, which can be divided into four distinct generations. The first generation is characterized by the release of Gentry's FHE scheme, which paved the way for the further development of fundamental concepts in FHE, such as the introduction of noise to a public key by DGHV (Dijk-Gentry-Halevi-Vaikutanathan) [17]. The second generation is defined by the BFV (Brakerski/Fan-Vercauteren) and BGV (Brakerski-Gentry-Vaikuntanathan) schemes [13], which introduced the LWE (Learning With Error) and RLWE (Ring Learning With Error) security models. The third generation includes the GSW (Gentry-Sahai-Waters) scheme, which avoided the computationally expensive relinearization operation when performing homomorphic multiplication, and exhibiting slower noise growth. More efficient ring variants such as the FHEW (Ducas-Micciancio — "Fastest Homomorphic Encryption in the West"), as well as the simplification and increased optimization of bootstrapping, were also introduced in this generation. The fourth generation includes the CKKS (Cheon-Kim-Kim-Song) scheme, which introduced efficient rounding operations for encrypted values, controlling noise rate increases in FHE multiplication and reducing the number of bootstraps required in a logic circuit. It also introduced PBS (programmable bootstrapping) to TFHE (torus fully homomorphic encryption) [12], reducing the number of bootstraps required in a logic circuit.

3.2 Concrete ML

Concrete ML is a machine learning library that can run all its supported ML algorithms using fully homomorphic encryption using the Concrete FHE library. With the use of Concrete ML, it allows us to work on our implementation in Python. It is also compatible with Scikit-learn modules and workflows, and has built-in support for implementation in client-server architectures. ConcreteML also features supports a variety of regression and classification models that can be compiled to FHE equivalents. These supported models are categorized into linear models, tree-based models, and neural networks for deep learning. The model produced can then be deployed in an application together with the cryptographic parameters.

3.3 Viral Strain Classification

Viral strain classification is a classification task that involves the analysis and assignment of a given viral sample sequenced from a patient to a group of known sequences with similar properties, and characteristics. This process is essential in tracking and combating the spread of viral strains through effective treatment and has the potential to enhance further studies of viruses [1]. The different approaches for viral strain classification can be grouped into three categories [40]:

1. Sequence alignment-based approaches, which are used in similarity search tools like BLAST [6],
2. Phylogeny-based approaches which involve the assignment of an unknown sequence to a known phylogenetic tree of reference sequences or the inference of a new tree. An example of a tool that uses this approach is the Pangolin COVID-19 Lineage Assigner [38].
3. Alignment-free approaches, which take advantage of nucleotide correlations and sequence composition to classify sequences, such as the approach taken by Sim et al [45].

4 VIRAL STRAIN CLASSIFICATION WITH HOMOMORPHIC ENCRYPTION

In our work, we evaluated the development and performance of privacy-preserving viral strain classification using logistic regression with Concrete ML, which is an alignment-free approach. We developed the plaintext model of logistic regression using scikit-learn. The plaintext model is used as baseline for comparison with the logistic regression models developed with Concrete ML. For the logistic regression model developed with Concrete ML, we developed two versions: quantized plaintext and FHE models. The quantized plaintext model is the result of performing quantization on the plaintext model in order to store the values in 16-bit integers [49]. For linear models such as logistic regression, quantization is performed after training [50]. The quantized plaintext model is compiled by Concrete ML to produce the FHE model.

4.1 Dataset

We used data from the publicly available repository Global Initiative on Sharing Avian Influenza Data (GISAID) [27, 18, 44], specifically the dataset of Sim et al. [45] (Episet ID `EPI_SET_220924cw`) that has nearly 9000 sequences. We limited our work to four strains of interest: B.1.617.2 (Delta), C.37 (Lambda), B.1.621 (Mu), and B.1.1.529 (Omicron) lineages. Therefore from this dataset, we filtered only those samples belonging to these four strains of interest, and this resulted to a smaller, unbalanced dataset of 6805 samples. To address this imbalance, we performed an upsampling of the minority variants (B.1.1.529 and B.1.617.2) through the acquisition of a total of 2100 samples from GISAID via

the GISAIDR library [47]. The samples were compiled to a GISAID Episet with the ID `EPI_SET_230520sm` (refer to Section 7 for the link to the dataset). This resulted in a final dataset with the sample distribution shown in Table 1:

Table 1: Number of samples per lineage in the dataset.

Lineage	Number of samples
B.1.1.529	2066
B.1.617.2	2044
B.1.621	2305
C.37	2478
Total	**8893**

4.2 Preprocessing and Feature Selection

To improve preprocessing and model training speeds, we followed Sim et al's framework and truncated the first 20kB of each genome sequence in the dataset. The truncated regions correspond to the regions of the sequences that preceded the S gene, which is the key driver of biological mutations between the SARS-CoV-2 strains [45]. Since viral strains are typically defined based on their phenotypic characteristics instead of simple sequence similarity, we utilized dashing, an alignment-free preprocessing method [9] for transforming genomic sequences into feacture vectors. We used the dashing tool from https://github.com/dnbaker/dashing with the following parameters: $k = 31$, number of threads is 13, and sketch size is 9. To reduce the high number of features, we used the k-best features algorithm with $k = 20$ which reduces the 512 features to the 20 highest-scoring features.

4.3 Logistic Regression Model

Logistic regression is a type of statistical approach that is used extensively when performing binary or multi-class classification [25, 39]. It performs well when used with linearly separable classes and classification [39, 52]. Zeller et al. used logistic regression in viral strain classification since the genetic patterns in sequences, which come about as a result of genetic divergence over time, are linearly separable across aligned amino acid positions [52].

We trained three logistic regression models: plaintext, quantized plaintext and FHE models. The plaintext model was trained using scikit-learn. This is the model where the performance of the FHE model is compared. Concrete ML was used to develop the quantized plaintext and FHE models. A standard train-test split of 80% training data and 20% testing data was utilized.

5 RESULTS

5.1 Model Accuracy and Error Analysis

Table 2: Model performance in terms of accuracy and AUROC score (One vs Rest)

Logistic Regression Model	Accuracy	ROC AUC Score
Plaintext	99.28%	0.999879
Quantized Plaintext	99.25%	0.999877
FHE	99.25%	0.999877

Our test set consists of 1779 samples, which is 20% of the total samples. Table 2 shows that all three models: plaintext, quantized plaintext and FHE models were able to achieve relatively high performance but there are small errors in the output of FHE computation due to quantization. The performance lost is below 0.01% which means this performance lost is tolerable.

We conducted an error analysis on one run and created confusion matrices shown in Figures 1a, 1b, and 1c to look at this performance lost due to quantization. Quantization causes the classifications of the quantized plaintext and FHE models to deviate from that of the plaintext model. We can see from these confusion matrices that this deviation causes either an error in the quantized plaintext and FHE model classifications compare to the plaintext model or a more "accurate" quantized plaintext and FHE model classifications compared to the plaintext model. In both cases, the difference between the models is at most 1, which is an acceptable level of error.

5.2 Model Training and Classification Speed

Table 3: Average training time in milliseconds for each model, with respective time increase

Model Type	Training time	Training time increase
Plaintext	95.99	
Quantized Plaintext	115.68	20.52%

We measured the training and prediction times of our models in a machine with an Intel Core i7-11370H processor and 16GB RAM running Windows 10 and

Windows Subsystem for Linux using Ubuntu 22.04. Table 3 shows the training times of the plaintext and quantized plaintext models. We didn't measure the training time for the FHE model since it is produced by compiling the quantized plaintext model. From Table 3, we can see that there is a 20.52% increase in the training time of quantized plaintext model compare to the plaintext model. We can attribute this to the quantization step performed after training.

Table 4: Average prediction time per sample in nanoseconds of each model

Model Type	Avg. Prediction Time	Increase in Prediction Time
Plaintext	150.11	
Quantized Plaintext	460.55	206.80%
FHE	537.06	257.77%

We also evaluated model performance in terms of prediction time of a single sample. The results of the prediction time comparison are shown in Table 4. The quantized plaintext model prediction has a dramatic increase of almost 210%. On the other hand, the FHE model prediction time also has a dramatic increase of almost 260% due to the quantization and the expected increase in running time of FHE computations.

6 CONCLUSION

We have shown that Concrete ML can be used in enabling fully homomorphic encryption for viral strain classification using logistic regression. We have also shown that Concrete ML can be used in a way that is similar to a normal ML workflow. There is no need for parameter setting and choosing the appropriate encoding scheme for the data, which is needed when we are using FHE libraries. With Concrete ML, it would be easier to implement machine learning models and deploy them in a privacy-preserving manner for genomic data. Its performance on classification tasks is comparable to that of scikit-learn. However, a major roadblock remains before it can be widely adopted due to the high computational time required. This high computational time is a major problem in all FHE libraries which hinders its adoption in practice. We might need to wait for the next generation of FHE libraries to be developed before FHE can be widely adopted in practice.

7 DATA AVAILABILITY

The datasets used in this paper (described in subsection 4.1) can be accessed via GISAID's EPISET ID `EPI_SET_230520sm` and is accessible via https://doi.org/10.55876/gis8.230520sm.

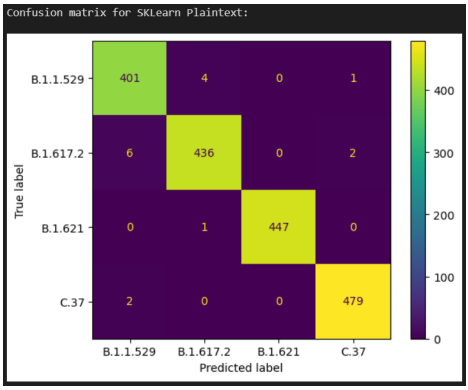
References

1. (2021). URL http://www.humangenomeprivacy.org/2021/competition-tasks.html
2. SAGO statement on newly released sars-cov-2 metagenomics data from china CDC on GISAID (2023). URL https://www.who.int/news/item/18-03-2023-sago-statement-on-newly-released-sars-cov-2-metagenomics-data-from-china-cdc-on-gisaid
3. Akavia, A., Galili, B., Shaul, H., Weiss, M., Yakhini, Z.: Efficient privacy-preserving viral strain classification via k-mer signatures and fhe. In: 2023 IEEE 36th Computer Security Foundations Symposium (CSF), pp. 489–504. IEEE Computer Society, Los Alamitos, CA, USA (2023). DOI 10.1109/CSF57540.2023.00012
4. Akavia, A., Galili, B., Shaul, H., Weiss, M., Yakhini, Z.: Efficient privacy-preserving viral strain classification via k-mer signatures and fhe. In: 2023 IEEE 36th Computer Security Foundations Symposium (CSF), pp. 489–504. IEEE Computer Society, Los Alamitos, CA, USA (2023). DOI 10.1109/CSF57540.2023.00012
5. Aloufi, A., Hu, P., Song, Y., Lauter, K.: Computing blindfolded on data homomorphically encrypted under multiple keys: A survey. ACM Computing Surveys **54**(9), 1–37 (2022). DOI 10.1145/3477139
6. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research **25**(17), 3389–3402 (1997). DOI 10.1093/nar/25.17.3389
7. Arnold, D., Saniie, J., Heifetz, A.: Homomorphic encryption for machine learning and artificial intelligence applications (2022). DOI 10.2172/1886256. URL https://www.osti.gov/biblio/1886256
8. Badawi, A.A., Hoang, L., Mun, C.F., Laine, K., Aung, K.M.M.: PrivFT: Private and fast text classification with homomorphic encryption. IEEE Access **8**, 226,544–226,556 (2020). DOI 10.1109/access.2020.3045465
9. Baker, D.N., Langmead, B.: Dashing: Fast and accurate genomic distances with hyperloglog. Genome Biology **20**(1) (2019). DOI 10.1186/s13059-019-1875-0
10. Benaissa, A., Retiat, B., Cebere, B., Belfedhal, A.E.: TenSEAL: A library for encrypted tensor operations using homomorphic encryption (2021)
11. Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Hoffstein, J., Lauter, K., Lokam, S., Moody, D., Morrison, T., Sahai, A., Vaikuntanathan, V.: Security of homomorphic encryption (2018). URL https://www.microsoft.com/en-us/research/wp-content/uploads/2018/01/security_homomorphic_encryption_white_paper.pdf
12. Chillotti, I., Joye, M., Paillier, P.: Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. Cryptology ePrint Archive, Paper 2021/091 (2021). DOI 10.1007/978-3-030-78086-9_1. URL https://eprint.iacr.org/2021/091. https://eprint.iacr.org/2021/091
13. Creeger, M., Inc., C.: The rise of fully homomorphic encryption. ACM Queue (2022). URL https://queue.acm.org/detail.cfm?id=3561800
14. Cruz, J.A., Chua, R.: Secure remote genome-wide association studies using fully homomorphic encryption. Theory and Practice of Computation Proceedings of the Workshop on Computation: Theory and Practice (WCTP 2019) (2020). DOI 10.1201/9780367814656
15. Cullinan, K.: High drama as scientists who may have found covid 'animal x' are kicked off data-sharing platform (2023). URL https://healthpolicy-watch.news/drama-in-quest-for-covid-origins/

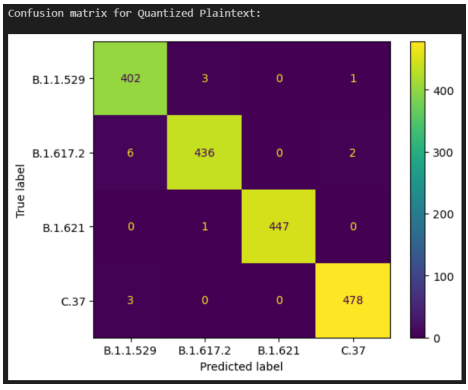
16. Câmara, G.B., Coutinho, M.G., Silva, L.M., Gadelha, W.V., Torquato, M.F., Barbosa, R.d., Fernandes, M.A.: Convolutional neural network applied to SARS-COV-2 sequence classification. Sensors **22**(15), 5730 (2022). DOI 10.3390/s22155730
17. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: H. Gilbert (ed.) Advances in Cryptology – EUROCRYPT 2010, pp. 24–43. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
18. Elbe, S., Buckland-Merrett, G.: Data, disease and diplomacy: GISAID's innovative contribution to global health. Global Challenges **1**(1), 33–46 (2017). DOI 10.1002/gch2.1018
19. Froelicher, D., Troncoso-Pastoriza, J.R., Raisaro, J.L., Cuendet, M.A., Sousa, J.S., Cho, H., Berger, B., Fellay, J., Hubaux, J.P.: Truly privacy-preserving federated analytics for precision medicine with multiparty homomorphic encryption. Nature Communications **12**(1) (2021). DOI 10.1038/s41467-021-25972-y
20. Gentry, C.: Fully homomorphic encryption using ideal lattices. Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC '09 (2009). DOI 10.1145/1536414.1536440. URL https://www.cs.cmu.edu/~odonnell/hits09/gentry-homomorphic-encryption.pdf
21. Hong, S., Park, J.H., Cho, W., Choe, H., Cheon, J.H.: Secure tumor classification by shallow neural network using homomorphic encryption. BMC Genomics **23**(1) (2022). DOI 10.1186/s12864-022-08469-w
22. Ibarrond: Ibarrond/pyfhel: Python for homomorphic encryption libraries, perform encrypted computations such as sum, mult, scalar product or matrix multiplication in python, with numpy compatibility. uses seal/palisade as backends, implemented using cython. GitHub (2022). URL https://github.com/ibarrond/Pyfhel. Available at https://github.com/ibarrond/Pyfhel
23. Ibarrondo, A., Viand, A.: Pyfhel: Python for homomorphic encryption libraries. In: ACM (ed.) WAHC 2021, 9th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Associated with the ACM CCS 2021 conference, 15 November 2021, Seoul, South Korea. Seoul (2021)
24. IBM: HElayers. URL https://github.com/IBM/helayers
25. IBM: What is logistic regression? IBM URL https://www.ibm.com/topics/logistic-regression
26. IBM: Efficient Privacy-Preserving Viral Strain Classification via k-mer Signatures and FHE (Poster) (2022). URL https://drive.google.com/file/d/1R1HZSUwFBMf4cotQEEVhPtObGlvRoas_/view?usp=sharing. Available at https://drive.google.com/file/d/1hAcn_DZPQ5KA837JVoMkMwg__W4FtESA/view
27. Khare, S., Gurry, C., Freitas, L., Schultz, M.B., Bach, G., Diallo, A., Akite, N., Ho, J., TC Lee, R., Yeo, W., et al.: GISAID's role in pandemic response. China CDC Weekly **3**(49), 1049–1051 (2021). DOI 10.46234/ccdcw2021.255
28. Kim, J., Lee, K., Rupasinghe, R., Rezaei, S., Martínez-López, B., Liu, X.: Applications of machine learning for the classification of porcine reproductive and respiratory syndrome virus sublineages using amino acid scores of ORF5 gene. Frontiers in Veterinary Science **8** (2021). DOI 10.3389/fvets.2021.683134
29. Kim, J., Lee, K., Rupasinghe, R., Rezaei, S., Martínez-López, B., Liu, X.: Applications of machine learning for the classification of porcine reproductive and respiratory syndrome virus sublineages using amino acid scores of ORF5 gene. Frontiers in Veterinary Science **8** (2021). DOI 10.3389/fvets.2021.683134
30. Ladisla, A.K., Chua, R.B.: Secure computation outsourcing of genome-wide association studies using homomorphic encryption. Theory and Practice of Computation Proceedings of the Workshop on Computation: Theory and Practice (WCTP 2018) (2019). DOI 10.1201/9780429261350

31. Lee, J.W., Kang, H., Lee, Y., Choi, W., Eom, J., Deryabin, M., Lee, E., Lee, J., Yoo, D., Kim, Y.S., No, J.S.: Privacy-preserving machine learning with fully homomorphic encryption for deep neural network (2021)
32. LEHRER, S., RHEINSTEIN, P.H.: Human gene sequences in sars-cov-2 and other viruses. In Vivo **34**(3 suppl), 1633–1636 (2020). DOI 10.21873/invivo.11954
33. Li, H., Hong, X., Ding, L., Meng, S., Liao, R., Jiang, Z., Liu, D.: Sequence similarity of sars-cov-2 and humans: Implications for sars-cov-2 detection. Frontiers in Genetics **13** (2022). DOI 10.3389/fgene.2022.946359
34. Lopez-Rincon, A., Tonda, A., Mendoza-Maldonado, L., Mulders, D.G., Molenkamp, R., Perez-Romero, C.A., Claassen, E., Garssen, J., Kraneveld, A.D.: Classification and specific primer design for accurate detection of SARS-COV-2 using Deep Learning. Scientific Reports **11**(1) (2021). DOI 10.1038/s41598-020-80363-5
35. Microsoft: Eva - compiler for microsoft seal. URL https://github.com/microsoft/EVA
36. Microsoft: Microsoft/seal: Microsoft seal is an easy-to-use and powerful homomorphic encryption library. URL https://github.com/microsoft/SEAL
37. Mole, B.: Genetic data links sars-cov-2 to raccoon dogs in china market, scientists say (2023). URL https://arstechnica.com/science/2023/03/genetic-data-links-sars-cov-2-to-raccoon-dogs-in-china-market-scientists-say/
38. O'Toole, A., Scher, E., Underwood, A., Jackson, B., Hill, V., McCrone, J.T., Colquhoun, R., Ruis, C., Abu-Dahab, K., Taylor, B., Yeats, C., du Plessis, L., Maloney, D., Medd, N., Attwood, S.W., Aanensen, D.M., Holmes, E.C., Pybus, O.G., Rambaut, A.: Assignment of epidemiological lineages in an emerging pandemic using the pangolin tool. Virus Evolution **7**(2), veab064 (2021). DOI 10.1093/ve/veab064
39. Pradhan, A., Prabhu, S., Chadaga, K., Sengupta, S., Nath, G.: Supervised learning models for the preliminary detection of COVID-19 in patients using demographic and epidemiological parameters. MDPI (2022). URL https://doi.org/10.3390/info13070330
40. Remita, M.A., Halioui, A., Malick Diouara, A.A., Daigle, B., Kiani, G., Diallo, A.B.: A machine learning approach for viral genome classification. BMC Bioinformatics **18**(1) (2017). DOI 10.1186/s12859-017-1602-3
41. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Foundations of secure computation **4**(11), 169–180 (1978). URL https://luca-giuzzi.unibs.it/corsi/Support/papers-cryptography/RAD78.pdf
42. Sarkar, E., Chielle, E., Gursoy, G., Mazonka, O., Gerstein, M., Maniatakos, M.: Fast and scalable private genotype imputation using machine learning and partially homomorphic encryption. IEEE Access **9**, 93,097–93,110 (2021). DOI 10.1109/access.2021.3093005
43. Shaul, H., Galili, B., Akavia, A., Weiss, M., Yakhini, Z.: IBM homomorphic encryption: A DASHing solution for healthcare data privacy. IBM Developer (2022). URL https://developer.ibm.com/blogs/ibm-homomorphic-encryption-a-dashing-solution-for-healthcare-dataprivacy/
44. Shu, Y., McCauley, J.: GISAID: Global initiative on sharing all influenza data – from vision to reality. Eurosurveillance **22**(13) (2017). DOI 10.2807/1560-7917.es.2017.22.13.30494
45. Sim, J.J., Zhou, W., Chan, F.M., Annamalai, M.S.M.S., Deng, X., Tan, B.H.M., Aung, K.M.M.: CoVnita, an end-to-end privacy-preserving framework for SARS-CoV-2 classification. Scientific Reports **3** (2023). DOI 10.1038/s41598-023-34535-8

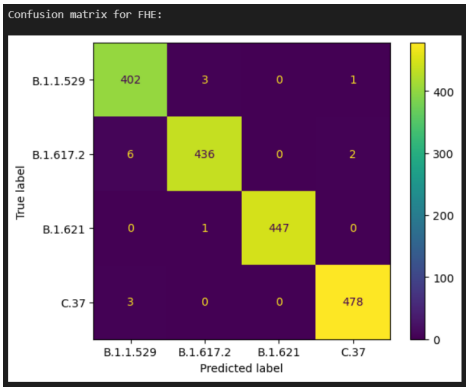
46. Song, L., Liu, H., Brinkman, F.S.L., Gill, E., Griffiths, E.J., Hsiao, W.W.L., Savić-Kallesøe, S., Moreira, S., Van Domselaar, G., Zawati, M.H., Joly, Y.: Addressing privacy concerns in sharing viral sequences and minimum contextual data in a public repository during the covid-19 pandemic. Frontiers in Genetics **12** (2022). DOI 10.3389/fgene.2021.716541
47. Wirth, W., Duchene, S.: WYTAMMA/GISAIDR: Programmatically interact with the GISAID database. (2022). URL https://doi.org/10.5281/zenodo.6474693
48. Wood, A., Najarian, K., Kahrobaei, D.: Homomorphic encryption for machine learning in medicine and bioinformatics. ACM Comput. Surv. **53**(4) (2020). DOI 10.1145/3394658
49. Zama-AI: Key concepts. Concrete ML URL https://docs.zama.ai/concrete-ml/getting-started/concepts
50. Zama-AI: Quantization. Concrete ML URL https://docs.zama.ai/concrete-ml/advanced-topics/quantization
51. Zama-Ai: ZAMA ConcreteML. URL https://github.com/zama-ai/concrete-ml
52. Zeller, M.A., Arendsee, Z.W., Smith, G.J., Anderson, T.K.: Classlog: Logistic regression for the classification of genetic sequences. bioRxiv (2022). URL https://doi.org/10.1101/2022.08.15.503907



(a) The confusion matrix for the plaintext model



(b) The confusion matrix for the quantized plaintext model



(c) The confusion matrix for the FHE model

Fig. 1: The confusion matrices for the different models