ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

Towards Learning Vector Quantization in the Setting of Homomorphic Encryption

Ronny Schubert[1,3], Thomas Davies[1]*, Mandy Lange-Geisler[1], Klaus Dohmen[1], Thomas Villmann[1,2]

1- Mittweida University of Applied Sciences
Saxon Institute for Computational Intelligence and Machine Learning
Technikumplatz 17, 09648 Mittweida

2- Technische Universität Freiberg

3- NEC Laboratories Europe GmbH, Heidelberg, Germany

Abstract. With federated learning scenarios gaining popularity to outsource computational heavy tasks or to increase generalizability of machine learning models, there is also a rise of research in terms of the security and privacy of the respective data used for these tasks. While differential privacy is studied well for Learning Vector Quantization, we want to present steps towards Homomorphic Encryption. In this regard, we will show theoretically how LVQ-1 can be adapted to be compatible with the TFHE encryption scheme and present experimental results.

1 Introduction

In scenarios of scarce data decentralized tasks such as federated learning or cloud computing for machine learning are gaining popularity to improve generalizability or to outsource expensive computations [1, 2]. As a result, these methods are interesting for medical or biological data and applications. In the same breath, concerns of trust with respect to the *cloud* are apparent, which, in turn, becomes a subject of data privacy. As [3, 4, 5] showed, this privacy could rather easily be broken for neural networks and *Learning Vector Quantization* (**LVQ**). In this regard, various countermeasures have been studied with two standing out - *differential privacy* (**DP**) and *homomorphic encryption* (**HE**). Whereas DP has been extensively investigated for machine learning [2] and also LVQ [4], HE applications are still on the rise. While the main advantage of HE is, that it allows calculations on the encrypted messages, thus making it attractive for privacy-preserving machine learning, it is also what troubles bigger networks or *circuits*. The underlying security of HE schemes is based on variants of the *learning with errors* (**LWE**) problem, where the output of a function is the same for an input and its noisy variants [6]. During the evaluation of a circuit noise increases over the number and types of operations, which limits the possible circuit length to output the correct result. Although [7] introduced *bootstrapping* to obtain unlimited circuit lengths the resulting strategy slows down computations, making such schemes predominantly interesting for the inference of machine learning models [2]. However, sparse or shallow models may avoid these pitfalls by allowing also training in a feasible time. In this regard, *k-Means* has been studied in HE settings [8, 9] which will serve as a basis for our work.

*T.D. is supported by an ESF PhD-grant.
Underlined authors contributed equally.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

2 Learning Vector Quantization and TFHE

In this work we opted for LVQ-1 being the respective machine learning model and *Fast Fully Homomorphic Encryption over the Torus* (**TFHE**) [10] as the HE scheme. We will briefly describe LVQ-1 and highlight the respective modules which need to be adapted to suit the TFHE scheme. Due to space constraints, we will highlight only certain aspects of TFHE and refer the reader to [1, 10, 11, 12] for an overview and in-depth literature.

2.1 LVQ-1

Considering $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ as a data sample and $\boldsymbol{p}_j \in \mathcal{P} \subset \mathbb{R}^n$ to be a prototype and $y_{\boldsymbol{x}}, y_{\boldsymbol{p}_j} \in \mathcal{Y}$ to be their respective classes, then the update for $\boldsymbol{p}_j$ in LVQ-1 [13] can be formulated as

$$\Delta \boldsymbol{p}_j = \epsilon \cdot \psi\left(y_{\boldsymbol{x}}, y_{\boldsymbol{p}_j}\right) \cdot \left(\boldsymbol{x} - \boldsymbol{p}_j\right) \tag{1}$$

where $\epsilon > 0$ is the learning rate, the decision function

$$\psi\left(y_{\boldsymbol{x}}, y_{\boldsymbol{p}_j}\right) = \begin{cases} 1 & \text{if } y_{\boldsymbol{x}} = y_{\boldsymbol{p}_j} \text{ and } j = \arg\min_i d\left(\boldsymbol{x}, \boldsymbol{p}_i\right) \\ -1 & \text{if } y_{\boldsymbol{x}} \neq y_{\boldsymbol{p}_j} \text{ and } j = \arg\min_i d\left(\boldsymbol{x}, \boldsymbol{p}_i\right) \\ 0 & \text{else} \end{cases}$$

determines the direction of the vector shift in (1), and $d\left(\boldsymbol{x}, \boldsymbol{p}_j\right)$ is the squared Euclidean distance. As shown in [14], we are able to decompose (1) into mappings which give rise to respective functions in TFHE. Moreover, we note, that $d\left(\boldsymbol{x}, \boldsymbol{p}_j\right)$ can be considered as an inner product, helping us to define the appropriate distance function in (2).

2.2 TFHE

TFHE uses multiple *ciphertext* variants to provide a rich set of homomorphic operations [10, 11]. In this regard, a message m is considered to be $m \in \mathcal{R}_p = \mathbb{Z}_p[Z]/\left(\phi_a(Z)\right)$ with p the modulo usually chosen as a power of 2 and $\phi_a(Z)$ the a-th cyclotomic polynomial of $\mathbb{Z}_p$, where a is commonly chosen to be a power of 2, reducing $\phi_a(Z)$ to a polynomial of the form $(Z^N + 1)$. Moreover, TFHE operates over the set $\{0, 1\}$, i.e., considers the binary representation of messages and ciphertexts and circuits operating on these objects to be *boolean circuits*. At this point we want to emphasize, that a respective integer encoding of the usual floating-point data $\mathcal{X} \subseteq \mathbb{R}^n$ is needed. Furthermore, the samples $\boldsymbol{x} \in \mathcal{X}$ are often in vectorial representation while the above representation of m is the representation of *a single integer*. Thus, $\hat{\boldsymbol{x}} \in \mathcal{R}_p^n = \mathbb{Z}_p^n[Z]/\left(Z^N + 1\right)$ to consider $\boldsymbol{x} \in \mathbb{R}^n$ encoded as a n-dimensional vector of plaintext messages. To finally represent our circuit, i.e., a full training step, we will use *generalized*-LWE (**GLWE**) ciphertexts. Nevertheless, we emphasize, that certain operations like homomorphic multiplication are not natively supported by GLWE ciphertexts, which is circumvented by the conversion to a supporting ciphertext variant (in this case GSW) for which *key-switching* is used [10, 11]. However, using GLWE to illustrate our circuit, ciphertexts are represented $c_m \in \mathcal{R}_q^k = \mathbb{Z}_q^k[Z]/\left(Z^N + 1\right)$ with $0 \leq p \leq q$ as the k-dimensional *encryption* of m - a single integer and $\boldsymbol{c}_{\boldsymbol{x}} \in \mathcal{R}_q^{n \times k} = \mathbb{Z}_q^{n \times k}[Z]/\left(Z^N + 1\right)$ of the n-dimensional $\hat{\boldsymbol{x}}$ vector.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

3 Training of LVQ-1 as TFHE Circuit

Based on [8, 9] we will adapt the training step of LVQ-1 to a TFHE circuit. Let $\boldsymbol{c_x}, \boldsymbol{c_{p_j}} \in \mathcal{R}_q^{n\times k} = \mathbb{Z}_q^{n\times k}[Z]/\left(Z^N+1\right)$ be GLWE ciphertexts of data sample $\boldsymbol{x}$ and prototype $\boldsymbol{p}_j$ as defined above and $\oplus, \ominus, \odot$ operations in the encryption scheme realizing the analoga of $+, -, \cdot$ in the plaintext space, we define

$$\delta\left(\boldsymbol{c_x}, \boldsymbol{c_{p_j}}\right) = \bigoplus_{i=1}^{n}\left(\left[\boldsymbol{c_x}\right]_i \odot \left[\boldsymbol{c_x}\right]_i\right) \ominus \left(\hat{2} \odot \left[\boldsymbol{c_x}\right]_i \odot \left[\boldsymbol{c_{p_j}}\right]_i\right) \oplus \left(\left[\boldsymbol{c_{p_j}}\right]_i \odot \left[\boldsymbol{c_{p_j}}\right]_i\right) \tag{2}$$

for the translation of the squared euclidean distance into a TFHE compatible function $\delta \in \mathcal{R}_q^k$ with $[\cdot]_i$ selecting the i-th component of a ciphertext and $\hat{2}$ being the representation of the constant 2 in $\mathcal{R}_q^k$. Moreover, (2) can be considered as an analogon to the euclidean inner product but on the ciphertexts and, further, can be used to also translate $\psi(\cdot)$ into a suitable TFHE function. Moreover, as [8] pointed out, it is possible to replace the euclidean distance by the manhattan distance to speed-up operations.

3.1 Decision Function in TFHE

To translate $\psi(\cdot)$ into a compatible circuit function we need to consider its structure in a 2-fold way, i.e., we are splitting the winner determination and the verification of the classes. For the winner determination we can use an approach similar to the *FindMin* function of [8] in which the minimum distance is determined and finally yields 1 for the respective index and 0 else, but for our purpose amended for only one sample with respect to all prototypes. The class of the winner is then determined via *look-up tables* in TFHE with respect to the given index. Moreover, [10] showed, that comparisons between ciphertexts in TFHE can be realized through *deterministic weighted finite automata* which is converted to a circuit of *controlled* - MUX (**CMux**) gates. We denote as $fm(\cdot)$ the amended $FindMin$ function yielding the winner index, P the number of prototypes and the equality function $B_i\left(\hat{y}_{\boldsymbol{x}}\right)$ as a circuit of CMux gates to verify a class (mis-) matching

$$B_i\left(\hat{y}_{\boldsymbol{x}}\right) = \begin{cases} 1 & \text{if } \hat{y}_{\boldsymbol{x}} = \hat{y}_{\boldsymbol{p}_i} \\ 0 & \text{else} \end{cases} \tag{3}$$

such that we define for the encrypted $\hat{\psi}(\cdot)$ on the encrypted classes $\hat{y}_{\boldsymbol{x}}, \hat{y}_{\boldsymbol{p}_j}$ with δ_i the ciphertext distance between $\boldsymbol{c_x}$ and $\boldsymbol{c_{p_i}}$

$$\begin{aligned} j &= fm\left(\delta_1, \ldots, \delta_P\right) \\ \hat{\psi}\left(\hat{y}_{\boldsymbol{x}}, \hat{y}_{\boldsymbol{p}_j}\right) &= B_j\left(\hat{y}_{\boldsymbol{x}}\right) \ominus \overline{B_j\left(\hat{y}_{\boldsymbol{x}}\right)} \end{aligned} \tag{4}$$

with $\overline{B_i\left(\hat{y}_{\boldsymbol{x}}\right)}$ being the negation of $B_j\left(\hat{y}_{\boldsymbol{x}}\right)$, such that (4) yields $\hat{1} \in \mathcal{R}_q^k$ or $-\hat{1} \in \mathcal{R}_q^k$ to resemble the attraction-repelling for the winner index j. Ultimately, this allows us to define the prototype update.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

3.2 Prototype Update in TFHE

With the above adaptations (2) - (4) we are able to define the update for an encrypted prototype in TFHE. Hence, let $\hat{\epsilon} \in \mathcal{R}_q^k$ be an encrypted learning rate and $\boldsymbol{c}_{\boldsymbol{p}_j}$ the encrypted winner prototype

$$\boldsymbol{c}_{\boldsymbol{p}_j}(t+1) = \boldsymbol{c}_{\boldsymbol{p}_j}(t) \oplus \hat{\epsilon}^{-1} \odot \hat{\psi}\left(\hat{y}_{\boldsymbol{x}}, \hat{y}_{\boldsymbol{p}_j}\right) \odot \left(\boldsymbol{c}_{\boldsymbol{x}} \ominus \boldsymbol{c}_{\boldsymbol{p}_j}(t)\right) \tag{5}$$

As a remark, the learning rate $\hat{\epsilon}$ cannot be considered like the common learning rate which would be in $[0, 1]$ but instead $\hat{\epsilon}$ constitutes an integer in $\mathcal{R}_q^k$ and to emulate the scheduling of conventional learning, i.e., to regularize learning by decreasing the learning rate to 0, we instead consider the multiplication with $\hat{\epsilon}^{-1}$ to be the integer division by $\hat{\epsilon}$ and increase it over the training time t.

4 Experiments

In our experiments[1], we used the *concrete* library from Zama [12] along with the Iris and Wine datasets provided by scikit-learn[2]. Our circuit constitutes a training step of LVQ with the above realizations of the respective functions that updates the set of prototypes as in (5) by using a single, randomly chosen encrypted data sample $\boldsymbol{c}_{\boldsymbol{x}}$ to emulate the stochastic approximation of LVQ. These prototypes can then be reinserted into the circuit to obtain the training routine. Moreover, the circuit can be executed for encrypted data, but also for integer data. As previously discussed, operations on ciphertexts increase noise which limit stable numerical values with respect to the circuit's computational depth. To manage this, we experimentally determined the maximum threshold values achievable with our chosen distance measure and dataset. We precomputed the learning rates to reduce the need for computation within the ciphertext space. Finally, because the Euclidean squared distance can yield large values and is expected to generate a lot of noise due to squaring polynomials [10], we also conducted experiments using the *Manhattan distance*, which has previously been applied in k-Means training [8]. The resulting experiment configuration is provided in Table 1.

Configuration	**Bounds**	**Epochs**	**Learning Rates**
Iris + Manhattan	$[-39, 39] \subset \mathbb{Z}$	5	$(5, 11, 17, 23, 29)$
Iris + Euclidean	$[-11, 11] \subset \mathbb{Z}$	5	$(5, 6, 7, 8, 9)$
Wine + Manhattan	$[-19, 19] \subset \mathbb{Z}$	5	$(5, 7, 9, 11, 13)$
Wine + Euclidean	$[-8, 8] \subset \mathbb{Z}$	5	$(4, 5, 6, 7, 8)$

Table 1: Experimental Settings containing the configuration, estimated bounds, the number of epochs and the pre-computed learning rates.

To transform the dataset to the specified bounds, we first normalized the data across the dataset, applied *feature scaling* into the range of the determined bounds and then rounding, adjusting them to the maximum bound to preserve as much information as possible during learning. We conducted the experiments for the circuit with encrypted data and plaintext integer data with the respective feature scaling (see 1). The results for seven different random seeds can be found

[1]Source code to experiments can be found at `https://github.com/lvlanson/LVQ_TFHE`

[2]These datasets can be found in the Python package scikit-learn.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

in table 2. Presented are the mean test accuracies (μ) with a test data ratio of 0.2 and the respective standard deviations (σ) for each variant to be evaluated for FHE numerical stability and vanilla LVQ-1 serving as our baseline. Additionally, memory consumption and training time for the encrypted version are presented.

Configuration	Accuracy (μ, σ) **Encrypted**	Accuracy (μ, σ) **Integer**	Accuracy (μ, σ) **Vanilla LVQ-1**	**Memory Demand** (in GByte)	**Training Time** (in hours)
Iris + Manhattan	0.900, 0.06	0.914, 0.05	0.910, 0.04	45.01	42.2
Iris + Euclidean	0.910, 0.05	0.905, 0.05	0.914, 0.05	39.30	65.7
Wine + Manhattan	0.813, 0.08	0.821, 0.09	[2]0.488, 0.09	31.30	73.5
Wine + Euclidean	0.397, 0.06	0.877, 0.06	[2]0.480, 0.08	39.35	76.6

Table 2: Recorded mean test accuracies and deviations and additionally the memory demand in GByte and training time of the encrypted version.[3]

First, we observe that both the encrypted and integer representations generally perform well, except when using the Euclidean distance with the Wine dataset. We suspect that the combination of low numerical bounds and the squaring operation in the Euclidean distance leads to significant noise overflow, making computations unreliable, with accuracies ranging from 0.30 to 0.52. The time measurements included the circuit compilation, key generation, encryption/decryption, and evaluation during training, highlighting the current computational complexity impact of FHE on these tasks[4]. Another observation is the accuracy deviation between the encrypted and plaintext (Integer) versions when evaluating the same circuit. To investigate this, we analyzed the prototypes computed by both methods. Table 3 shows the deviation of prototypes from encrypted training compared to plaintext integer training across all experiments. The deviation is represented by the absolute difference of prototype components, with the maximum deviation indicating the largest observed difference and the average deviation showing the mean. The relative average accounts for the bound, calculated as average/bound. We observe that the Manhattan distance provides greater numerical stability compared to the squared Euclidean distance.

Configuration	**Maximum Deviation**	**Average Deviation**	**Relative Average Deviation in %**
Iris + Manhattan	5	0.59	1.52
Iris + Euclidean	2	0.26	2.38
Wine + Manhattan	3	0.17	0.93
Wine + Euclidean	25	8.62	107.83

Table 3: Prototype deviations between the decrypted prototypes and the integer prototypes in terms of the maximum, average and relative (with respect ot the bound) deviation.

5 Conclusion

The relatively lightweight nature of LVQ algorithms enabled a successful implementation within an FHE scheme and our considerations may stimulate new directions in privacy preserving prototype-based machine learning. Future investigations could focus on the numerical stability of a TFHE implementation over the parameter space outlined in the experiment section. During our experiments, we observed an optimum in the choice of bounds: beyond a certain threshold,

[3]The rather poor results for Vanilla LVQ-1 on the Wine dataset are likely due to a unfavorable prototype initialization. However, provably, neither the *discrete* nor the encrypted versions did suffer as dramatic under the used initialization.

[4]The experiments were executed on an AMD Epyc 7713 clocking at 2.1GHz.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

increases in bound size led to diminishing returns in accuracy. With further implementation improvements, it may be possible both to reduce the computational time needed to determine optimal bounds for the learning task and to expand the range of feasible bounds for the data space. More complex LVQ algorithms are challenging to implement with current TFHE implementations due to limitations in numerical stability and computational complexity. Therefore, alternative FHE schemes could be considered to apply more sophisticated LVQ algorithms on encrypted data.

Acknowledgement
We want to thank the developers and maintainers of the library *concrete* [12] (`https://github.com/zama-ai/concrete`) for the helpful advices on the implementation.

References

[1] Alexander Wood, Kayvan Najarian, and Delaram Kahrobaei. Homomorphic encryption for machine learning in medicine and bioinformatics. *ACM Comput. Surv.*, 53(4), August 2020.

[2] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 201–210, New York, New York, USA, 20–22 Jun 2016. PMLR.

[3] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020.

[4] Johannes Brinkrolf, Christina Göpfert, and Barbara Hammer. Differential privacy for learning vector quantization. *Neurocomputing*, 342:125–136, 2019.

[5] Ronny Schubert and Thomas Villmann. About vector quantization and its privacy in federated learning. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN*, pages 81–86, 01 2024.

[6] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), September 2009.

[7] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. Cryptology ePrint Archive, Paper 2011/566, 2011.

[8] Angela Jäschke and Frederik Armknecht. Unsupervised machine learning on encrypted data. Cryptology ePrint Archive, Paper 2018/411, 2018.

[9] Georgios Sakellariou and Anastasios Gounaris. Homomorphically encrypted k-means on cloud-hosted servers with low client-side load. *Computing*, 101, 2019.

[10] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. Cryptology ePrint Archive, Paper 2018/421, 2018.

[11] Sara Logsdon. Fully homomorphic encryption: A mathematical introduction. Cryptology ePrint Archive, Paper 2023/1402, 2023.

[12] Ilaria Chillotti, Marc Joye, Damien Ligier, and Jean-Baptiste Orfila. Concrete: Concrete operates on ciphertexts rapidly by extending tfhe. 2020.

[13] Teuvo Kohonen. *Self-Organizing Maps*. Springer Berlin Heidelberg, 1995.

[14] Ronny Schubert and Thomas Villmann. About interpretable learning rules for vector quantizers - a methodological approach. In *Advances in Self-Organizing Maps, Learning Vector Quantization, Interpretable Machine Learning, and Beyond - WSOM+ 2024*, pages 152–162, 2024.