ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

About Vector Quantization and its Privacy in Federated Learning

Ronny Schubert*and Thomas Villmann

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

Abstract. In this work, we will consider how privacy for vector quantization models can be broken in a federated learning environment. We show how a potential attacker can expose data from the prototype updates without needing to know about the specific model used by exploiting the transparency of vector quantization. Finally, a 1-user environment example based on GLVQ will be shown.

1 Introduction

Federated Learning (**FL**) has gained attraction for settings in which data privacy has to be ensured [1]. The collaborative nature of FL allows to train a machine learning model across various *Users* by only sharing the respective updates with a *server* which accumulates the updates and provides the updated parameters of the model to the users [1, 2]. Nevertheless, *Vector Quantization* (**VQ**) models have been studied in similar high-stakes domains, such as medicine [3], since VQ models offer rich interpretability options due to their transparent nature and a respective consideration of VQ in FL settings is found in [2]. However, the authors of [1] show how neural networks violate the privacy concern for FL settings. Our contribution therefore aims to transfer the approach of [1] to FL scenarios based on VQ models yielding a scheme like

$$\text{User}_i \underset{\mathcal{P}(t+1)}{\overset{\mathcal{U}_i(t)}{\rightleftharpoons}} \text{Server} \tag{1}$$

in which a user i sends the calculated update set $\mathcal{U}_i(t) \subset \mathbb{R}^n$ for time step t to the server, which accumulates the updates of the users and sends back the updated *prototype set* $\mathcal{P}(t+1) \subset \mathbb{R}^n$. Thereby, the set $\mathcal{U}_i(t)$ can be comprehended as the calculated gradients or in general as the update vectors provided by user i and, ultimately, the privacy violation is based on reconstructing the data used to create $\mathcal{U}_i(t)$.

1.1 Setting

The threat model we are considering in this work is based on an *honest-but-curious* server in agreement with [1]: An attacker is allowed to record the communication between the users and the server and store respective send contents,

*R. S. is supported by grants of the Agriculture Ministry of Germany (28DK112A20).

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

but any interference with and modifications to the learning scheme are excluded. Additionally, an attacker may only know that VQ is used, but not which specific variant. Moreover, we shall consider in this work a 1-user case with the remark, that the here shown approach can be extend for a multi-user scenario, since an attacker could simply limit its investigation to some specific user. Therefore, we shall drop in the following the index notation for the user. The goal of the attacker is then to reconstruct the data by only considering the (stored and recorded) prototypes and update vectors.

2 Abusing Transparency of Vector Quantization Models

Due to the lacking knowledge of the actual VQ variant in use, there is also no knowledge about the learning scheme and the attacker can only guess how $\mathcal{U}(t)$ is constructed on the user-side and how the updates are accumulated on the Server-side. Nevertheless, [4] proposed a *generic learning rule* for VQ models which is formulated based on the underlying decision making and thus directly linked to the transparency and interpretability of VQ: Given a data sample $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ the rule yields for the update vector $\boldsymbol{u}(t) = g(\boldsymbol{x}, \boldsymbol{p}(t)) \cdot \boldsymbol{s}(\boldsymbol{x}, \boldsymbol{p}(t)) \in \mathcal{U}(t)$ of a particular prototype

$$\Delta \boldsymbol{p}(t) = \boldsymbol{p}(t+1) - \boldsymbol{p}(t) = \sigma \cdot \epsilon \cdot \boldsymbol{u}(t) \tag{2}$$

where $\sigma \in \{-1, 1\}$ is a sign, ϵ is a learning rate, $g(\boldsymbol{x}, \boldsymbol{p}(t))$ determines a gain and $\boldsymbol{s}(\boldsymbol{x}, \boldsymbol{p}(t))$ constitutes an abstract vector shift. Using (2) the attacker may assumes: $g(\boldsymbol{x}, \boldsymbol{p}(t))$ can be considered as an unknown scalar $\beta \in \mathbb{R}$ and $\boldsymbol{s}(\boldsymbol{x}, \boldsymbol{p}(t))$ being the most crucial part, since the real architecture is not known. Yet, a common realization considers $\boldsymbol{s}(\boldsymbol{x}, \boldsymbol{p}(t))$ as a Euclidean vector shift [3] giving the ansatz $\tilde{\boldsymbol{u}}(t) = \beta \cdot (\boldsymbol{x} - \boldsymbol{p}(t))$. Thus, an attacker can assume

$$\Delta \boldsymbol{p} \approx \sigma \cdot \epsilon \cdot \beta \cdot (\boldsymbol{x} - \boldsymbol{p}(t)) = \sigma \cdot \epsilon \cdot \tilde{\boldsymbol{u}}(t) \tag{3}$$

as a valid approximation of the unknown update vector $\boldsymbol{u}(t)$. In this way, an attacker already has a guess of the learning scheme structure constituted by the user and the server, while the approximative behavior is caused by the unknown realization of $\boldsymbol{s}(\boldsymbol{x}, \boldsymbol{p}(t))$ which could also be realized in terms of a linear mapping as for example in [5, 6]. The attacker then proceeds by using $\tilde{\boldsymbol{u}}(t)$ to attempt the reconstruction of the data necessary to create $\boldsymbol{u}(t)$. Note, that the summary (3) yields that σ and ϵ are added on the server-side. However, as in [4] considered, these parameters could also be incorporated into the design of $g(\boldsymbol{x}, \boldsymbol{p}(t))$ on the user-side and thus also in β, as for example in LVQ [7]. Nevertheless, the reconstruction approach shown in 2.1 can be considered to be independent of these parameters.

2.1 Reconstructing Data Samples and Guessing the Update Scheme

Using the approximation (3) and the respective assumed expression $\tilde{\boldsymbol{u}}(t)$ for the given update vectors $\boldsymbol{u}(t) \in \mathcal{U}(t)$, the attacker can now attempt the reconstruction of the used data sample $\boldsymbol{x}$ to generate $\boldsymbol{u}(t)$: Given $\boldsymbol{p}(t) \in \mathcal{P}(t)$, such that for the respective update vector $\boldsymbol{u}(t) \neq \mathbf{0}$ holds, we consider a numerical reconstruction approach similar to [1]

$$\beta^*, \boldsymbol{x}^* = \underset{\beta\in\mathbb{R},\bar{\boldsymbol{x}}\in\mathbb{R}^n}{\arg\min} \|\boldsymbol{u}(t) - \tilde{\boldsymbol{u}}(t)\|^2 = \underset{\beta\in\mathbb{R},\bar{\boldsymbol{x}}\in\mathbb{R}^n}{\arg\min} \|\boldsymbol{u}(t) - \beta \cdot (\bar{\boldsymbol{x}} - \boldsymbol{p}(t))\|^2 \tag{4}$$

which can be realized as a simple gradient based optimization problem with $\|\cdot\|^2$ being the squared L^2 norm, such that $\tilde{\boldsymbol{u}}(t) = \beta^* \cdot (\boldsymbol{x}^* - \boldsymbol{p}(t))$ is the approximation of $\boldsymbol{u}(t)$ with $\boldsymbol{x}^*$ being the reconstructed data sample and β^* the estimated gain $g(\boldsymbol{x}, \boldsymbol{p}(t))$. Yet, it is likely that the attacker obtains multiple reconstructions, since it is common, that multiple prototypes receive an update during a training step [3, 4]. Nevertheless, with the considerations in section 2.2 it becomes possible to verify the obtained results.

2.2 Exposing the Update Scheme and Verifying Reconstructions

The attacker may expose extended informations due to the transparent nature of (2). Assuming, the attacker calculates for some stored $\boldsymbol{p}(t) \in \mathcal{P}(t)$ the difference vector $\delta\boldsymbol{p}(t) = \boldsymbol{p}(t+1) - \boldsymbol{p}(t)$ together with its updated version $\boldsymbol{p}(t+1) \in \mathcal{P}(t+1)$, such that for the corresponding update vector $\boldsymbol{u}(t) \neq \mathbf{0}$ holds. The sign information can then be recovered by considering

$$\min\left(\|\boldsymbol{n}^-\|^2, \|\boldsymbol{n}^+\|^2\right) \tag{5}$$

with $\boldsymbol{n}^\pm = \delta\boldsymbol{p}(t) \pm \boldsymbol{u}(t)$ and $\|\cdot\|^2$ as in (4). A matching of the results of (5) across *all* $\boldsymbol{p}_i(t) \in \mathcal{P}(t)$, for which $\boldsymbol{u}_i(t) \neq \mathbf{0}$ is valid, then indicates which update scheme is used. In the case of $\boldsymbol{n}^+$ being the regarding result, the update scheme is likely to correspond to gradient descent with $\mathcal{U}(t)$ containing the respective gradients, while $\boldsymbol{n}^-$ yields gradient ascent. Nevertheless, assuming the result of (5) differs across the prototypes, the sign σ is likely to be adjusted on the User-side by $g(\boldsymbol{x}, \boldsymbol{p}(t))$ [4].

Taking into account the above discussion, the attacker may proceed to use these informations by verifying the obtained results of the reconstruction. Exemplary, suppose gradient descent is used as determined above and the estimated gains β_j^* from the reconstruction (4) differ in sign. In this case, the most important reconstructions $\boldsymbol{x}_i^*$ correspond to the estimated gains for which $\beta_i^* < 0$ is valid. Due to the transparent nature of VQ and (2), a negative gain in an gradient descent scheme corresponds to attraction of the prototype [3, 4, 8]. Consequently, for this hypothetical scenario, the attacker could leak class information, while also gaining a measure of the goodness and importance of all reconstructions obtained. Note, that the same argumentation is also valid for User-adjusted signs in which the gain is responsible for the *attraction-repelling*. To, however, show this also empirically, we considered in section 3 a prototype-based classifier which makes use of the *attraction-repelling* scheme, i.e. using gains which differ in sign.

3 Experiments

For our case example, we considered the prototype-based classifier GLVQ [8] with the cost-function

$$C = \sum_i sgd_\theta \left(\frac{d(\boldsymbol{x}_i, \boldsymbol{p}^+) - d(\boldsymbol{x}_i, \boldsymbol{p}^-)}{d(\boldsymbol{x}_i, \boldsymbol{p}^+) + d(\boldsymbol{x}_i, \boldsymbol{p}^-)} \right) \tag{6}$$

with $sgd_\theta(\cdot)$ being the sigmoid-function with parameter θ (here $\theta = 0.25$), $d(\cdot)$ the squared euclidean distance and $\boldsymbol{p}^+$ ($\boldsymbol{p}^-$) being the nearest prototype with matching (not-matching) class and the respective updates $\boldsymbol{u}^\pm$ being derived as the gradients of C. The classifier was trained on a subset of the MNIST[1] dataset (1000 samples) for 100 epochs in a *stochastic gradient descent* (**SGD**) manner and with one prototype per class for a classification task. We used no feature extraction, but the samples were normalized into the range $[0, 1]$.

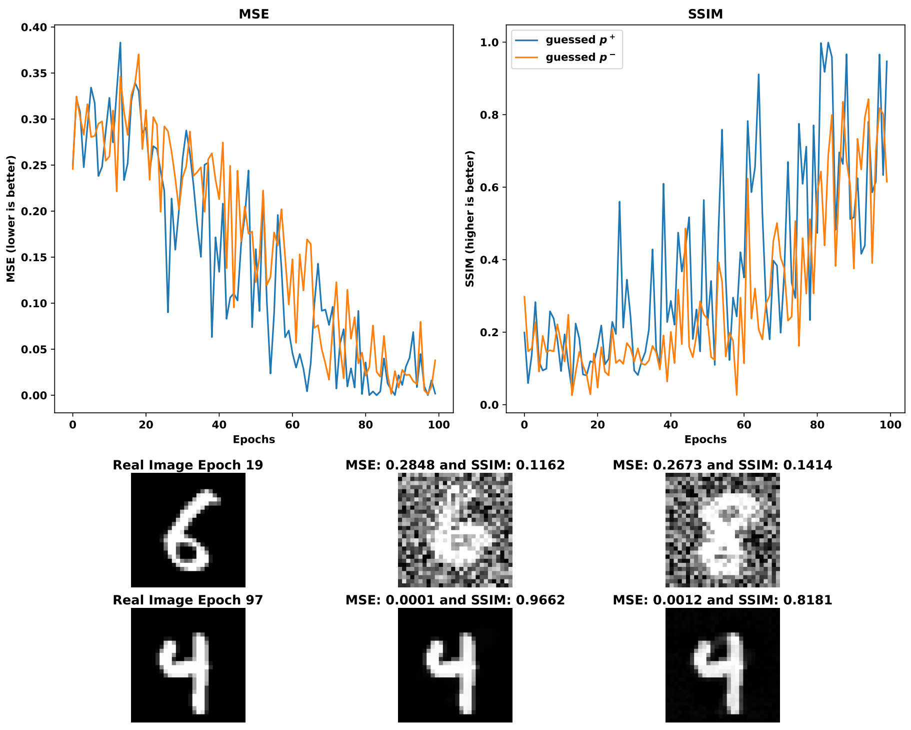

Fig. 1: Top-Left: Performance in terms of MSE and determined prototypes $\boldsymbol{p}^\pm$; Top-Right: Performance in terms of SSIM; Bottom: Examples of the reconstruction for an early (19) and later (97) epoch in model training

The reconstruction in (4) was executed within each epoch of the models training

[1] https://www.openml.org/d/554 - version 1

for a randomly chosen image by using ADAM with the default parameters [9] as the optimization method for 10^5 steps[2]. The performance is evaluated by recording the *mean squared error* (**MSE**) and *structural similarity index measure* (**SSIM**) between the reconstructed image and the true image.

As in Figure 1 depicted, the measurements of MSE and SSIM are 2-fold. Following the above discussed approach in section 2.2, we first determined the sign via (5) and evaluated the result against the signs of the estimated gains. When the determined sign was negative across all prototypes $\boldsymbol{p}_j$ for which $\boldsymbol{u}_i \neq \boldsymbol{0}$ was valid, we fixed $\boldsymbol{p}_i = \boldsymbol{p}^+$ whenever $\beta_i^* < 0$ and vice versa for $\boldsymbol{p}^-$. Anyway, the results indicate, that the performance increases as the adaptation of the prototypes to the data is improved. This is a somewhat expected behavior, since the prototypes in the assumed shape $\tilde{\boldsymbol{u}}(t)$ of the update vectors already leak partial information for the search space of the reconstruction. Consequently, salting the prototypes by adding random noise would only increase the time needed for a *good* reconstruction, but sophisticated evaluations of counter measures are left for future works and the reader is referred to [10, 11] for related discussions. Nevertheless, although, the results of the guessed $\boldsymbol{p}^+$ often outperform the ones for $\boldsymbol{p}^-$, deviations in the later stages of the models training may be caused by interactions of: the limited number of steps to optimize (4), adversarial pertubations [1] and the robustness of GLVQ [12]. In other words, a correct classified data sample may be faster reconstructed from $\boldsymbol{p}^+$, while an insecure (w.r.t. the model itself) or misclassfied sample may be faster reconstructed from $\boldsymbol{p}^-$. However, a profound evaluation and discussion of such effects are out of the scope of this work and left for future considerations.

4 Summary and Remarks

The results for the 1-User scenario indicate that the updates for VQ models carry significant data information which, due to VQ's transparent nature, could rather easily be exposed. Moreover, a potential attacker may be free of assuming a specific model, as we showed here for GLVQ and the attack being based on (3). Nevertheless, future research may also take into account various abstractions of the shift or understand VQ in terms of neural networks as in [13, 14] to be in agreement with [1].

Acknowledgement

The authors want to thank Alexander Engelsberger and Thomas Davies from UAS Mittweida for the fruitful and stimulating discussions.

References

[1] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in feder-

[2]The experiments are available at `https://github.com/rmschubert/data_privacy_VQ`

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

ated learning? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16937–16947. Curran Associates, Inc., 2020.

[2] Johannes Brinkrolf and Barbara Hammer. Federated learning vector quantization. In *Proceedings of the ESANN, 29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2021.

[3] David Nova and Pablo A. Estévez. A review of learning vector quantization classifiers. *Neural Computing and Applications*, 25:511 – 524, 2013.

[4] Ronny Schubert and Thomas Villmann. About interpretable learning rules for vector quantizers - a methodological approach. In *International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM+)*, 2024.

[5] Petra Schneider, Michael Biehl, and Barbara Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21(12):3532–3561, 2009.

[6] Kerstin Bunte, Petra Schneider, Barbara Hammer, Frank-Michael Schleif, Thomas Villmann, and Michael Biehl. Limited rank matrix learning, discriminative dimension reduction and visualization. *Neural Networks*, 26:159–173, 2012.

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

[8] Atsushi Sato and Keiji Yamada. Generalized learning vector quantization. In *Neural Information Processing Systems*, 1995.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[10] Johannes Brinkrolf, Kolja Berger, and Barbara Hammer. Differential private relevance learning. In *The European Symposium on Artificial Neural Networks*, 2018.

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

[12] Sascha Saralajew, Lars Holdijk, and Thomas Villmann. Fast adversarial robustness certification of nearest prototype classifiers for arbitrary seminorms. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13635–13650. Curran Associates, Inc., 2020.

[13] Sascha Saralajew, Lars Holdijk, Maike Rees, and Thomas Villmann. Prototype-based neural network layers: Incorporating vector quantization. *CoRR*, abs/1812.01214, 2018.

[14] Thomas Villmann, John Ravichandran, Andrea Villmann, David Nebel, and Marika Kaden. Activation functions for generalized learning vector quantization - a performance comparison. *ArXiv*, abs/1901.05995, 2019.