Quantized Rank Reduction: A Communication-Efficient Federated Learning Scheme for Network-Critical Applications

Dimitris Kritsiolis and Constantine Kotropoulos

Department of Informatics, Aristotle University of Thessaloniki, Greece
The 2025 IARIA Annual Congress on Frontiers in Science, Technology, Services, and Applications

July 9, Venice, Italy

ARISTOTLE
UNIVERSITY OF
THESSALONIKI

Presenter's bio

Constantine Kotropoulos received the Diploma degree with honors in Electrical Engineering in 1988 and the PhD degree in Electrical & Computer Engineering in 1993, both from the Aristotle University of Thessaloniki. He is a Full Professor in the Department of Informatics at the Aristotle University of Thessaloniki. He was a visiting research scholar in the Department of Electrical and Computer Engineering at the University of Delaware, USA, during the academic year 2008-2009. He also conducted research in the Signal Processing Laboratory at Tampere University of Technology in Finland during the summer of 1993. He has co-authored 72 journal papers and 227 conference papers and contributed 9 chapters to edited books in his areas of expertise. He is co-editor of the book "Nonlinear Model-Based Image/Video Processing and Analysis" (John Wiley and Sons, 2001).
His research interests include forensics, audio, speech and language processing, signal processing, pattern recognition, multimedia information retrieval, biometrics, and forensics.
Prof. Kotropoulos was a scholar of the State Scholarship Foundation of Greece and the Bodossaki Foundation. He is a fellow of IARIA, a senior member of the IEEE, and a member of EURASIP, IAPR, and the Technical Chamber of Greece. He was a Senior Area Editor of the IEEE Signal Processing Letters (SPL). He will serve as the Editor-in-Chief of SPL, effective January 1, 2026. He has been a member of the Editorial Board of the journals Advances in Multimedia, Computer Methods in Biomechanics & Biomedical Engineering: Imaging & Visualization, Artificial Intelligence Review, MDPI Imaging, MDPI Signals, and MDPI Methods and Protocols. Prof. Kotropoulos served as Track Chair for Signal Processing in the 6th Int. Symposium on Communications, Control, and Signal Processing, Athens, 2014; Program Co-Chair of the 4th Int. Workshop on Biometrics and Forensics, Limassol, Cyprus, 2016; Technical Program Chair of the XXV European Signal Processing Conf., Kos, Greece, 2017; Technical Program Chair of the 5th IEEE Global Conf. Signal and Information Processing, Montreal, Canada, 2017; General Chair of the 2022 IEEE 14th Image, Video, and Multidimensional Signal Processing Workshop, Nafplio, Greece; Technical Program Chair of the 2023 IEEE International Conf. on Acoustics, Speech, and Signal Processing, Rhodes, Greece.

Outline

- **Federated Learning (FL)** is a machine learning approach that enables multiple devices (clients) to train a shared model cooperatively without exchanging raw data, under the coordination of a central node (server).
- A **communication overhead** is introduced in FL, since each client must communicate its local gradient to the server at each training iteration.
- **Gradient compression** and **gradient quantization** are two techniques commonly used to minimize data transmission and reduce the communication overhead in the distributed training of deep neural networks.

- **Quantized Rank Reduction (QRR):** a communication-efficient FL scheme that leverages two techniques: low-rank approximation of neural network gradients and gradient quantization.
- **Experimental results** demonstrate that QRR yields significant communication savings with a small deterioration on the model's accuracy, validating the practical effectiveness of the proposed method in network-critical applications.

- Low-rank approximation is a suitable form of compression for neural network gradient matrices and tensors, as those gradients are generally low-rank and have few dominant singular values, especially in overparameterized networks[1].

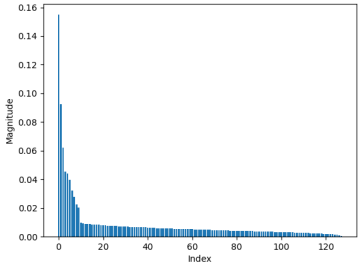


Figure 1: Magnitude of the singular values of the gradient of a fully connected layer.

[1] Samet Oymak, Zalan Fabian, Mingchen Li, and Mahdi Soltanolkotabi. *Generalization guarantees for neural networks via harnessing the low-rank structure of the Jacobian*. 2019. arXiv: 1906.05392 [cs.LG]. URL: https://arxiv.org/abs/1906.05392

Quantized Rank Reduction

Gradient Matrix Compression

- Consider the weights of a fully connected layer $\boldsymbol{W} \in \mathbb{R}^{D_{out} \times D_{in}}$ along with the scalar loss function $J(\cdot)$

$$J(\boldsymbol{\theta}) = \sum_{c \in \mathfrak{C}} f_c(\boldsymbol{\theta}) \tag{1}$$

  where $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{b})$.

- We approximate the gradient $\frac{\partial J}{\partial \boldsymbol{W}}$ as

$$\frac{\partial J}{\partial \boldsymbol{W}} \approx \boldsymbol{U}_v \, \boldsymbol{\Sigma}_v \, \boldsymbol{V}_v^\top, \tag{2}$$

  where $\boldsymbol{U}_v \in \mathbb{R}^{D_{out} \times v}$, $\boldsymbol{\Sigma}_v \in \mathbb{R}^{v \times v}$ and $\boldsymbol{V}_v \in \mathbb{R}^{D_{in} \times v}$ are the Singular Value Decomposition (SVD) components of $\frac{\partial J}{\partial \boldsymbol{W}}$, retaining only the v largest singular values.

- The client transmits only $\boldsymbol{U}_v$, $\boldsymbol{V_v}$ and the diagonal elements of $\boldsymbol{\Sigma}_v$.

- In a convolutional layer, the weights are represented by a 4D tensor $\mathcal{W} \in \mathbb{R}^{C_{out} \times C_{in} \times H \times W}$, where C_{out} is the number of output channels, C_{in} is the number of input channels and $H \times W$ is the size of the convolutional filter.
- We approximate the gradient $\frac{\partial J}{\partial \mathcal{W}}$ via the Tucker decomposition as

$$\frac{\partial J}{\partial \mathcal{W}} \approx \mathcal{G} \times_1 \boldsymbol{F}_1 \times_2 \boldsymbol{F}_2 \times_3 \boldsymbol{F}_3 \times_4 \boldsymbol{F}_4, \tag{3}$$

  where $\times_n$ denotes the mode-n product of a tensor and matrix, and $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3 \times r_4}$, $\boldsymbol{F}_1 \in \mathbb{R}^{C_{out} \times r_1}$, $\boldsymbol{F}_2 \in \mathbb{R}^{C_{in} \times r_2}$, $\boldsymbol{F}_3 \in \mathbb{R}^{H \times r_3}$ and $\boldsymbol{F}_4 \in \mathbb{R}^{W \times r_4}$, with r_i being the reduced ranks along each mode.
- The client transmits the core tensor $\mathcal{G}$ and the factor matrices $\boldsymbol{F}_i$, $i = 1, 2 \ldots, 4$.

- For the truncated SVD to be more communication-efficient than transmitting the full gradient, we need

$$D_{out} \cdot \nu + \nu + D_{in} \cdot \nu < D_{out} \cdot D_{in}. \tag{4}$$

- Similarly, for the Tucker decomposition to be more communication-efficient than transmitting the full gradient, we need

$$r_1 \cdot r_2 \cdot r_3 \cdot r_4 + C_{out} \cdot r_1 + C_{in} \cdot r_2 + H \cdot r_3 + W \cdot r_4 < C_{out} \cdot C_{in} \cdot H \cdot W. \tag{5}$$

- The gradient of the loss function with respect to (w.r.t.) the bias vector for each client is not compressed.

- The federated gradient descent step with quantization is

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \sum_{c \in \mathcal{C}} Q\left(\nabla f_c(\boldsymbol{\theta}^k)\right). \tag{6}$$

- We use LAQ's[2] quantization scheme, which maintains a running aggregated quantized gradient

$$\nabla^k = \nabla^{k-1} + \sum_{c \in \mathcal{C}} \delta Q_c^k. \tag{7}$$

[2] Jun Sun, Tianyi Chen, Georgios B. Giannakis, Qinmin Yang, and Zaiyue Yang. "Lazily Aggregated Quantized Gradient Innovation for Communication-Efficient Federated Learning". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.4 (2022), pp. 2031–2044. DOI: 10.1109/TPAMI.2020.3033286

- The scheme projects each gradient element onto a uniform grid that is centered at the previous quantized updated $Q_c(\boldsymbol{\theta}^{k-1})$ and has a radius of $R_c^k = ||\nabla f_c(\boldsymbol{\theta}^k) - Q_c(\boldsymbol{\theta}^{k-1})||_\infty$.
- Each element is encoded using β bits as

$$[q_c(\boldsymbol{\theta}^k)]_i = \left\lfloor \frac{[\nabla f_c(\boldsymbol{\theta}^k)]_i - [Q_c(\boldsymbol{\theta}^{k-1})]_i + R_c^k}{2\tau R_c^k} + \frac{1}{2} \right\rfloor, \tag{8}$$

  with $\tau := 1/(2^\beta - 1)$ defining the discretization interval.
- The server can recover the gradient update of client c as

$$Q_c(\boldsymbol{\theta}^k) = Q_c(\boldsymbol{\theta}^{k-1}) + \delta Q_c^k, \tag{9}$$

  with $\delta Q_c^k = Q_c(\boldsymbol{\theta}^k) - Q_c(\boldsymbol{\theta}^{k-1}) = 2\tau R_c^k \; q_c(\boldsymbol{\theta}^k) - R_c^k \mathbf{1}$, where $\mathbf{1} = [1, 1, \ldots, 1]^\top$.

- The gradient descent step becomes

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^{k} - \alpha \sum_{c \in \mathcal{C}} QRR_c\left(\boldsymbol{\theta}^{k}\right),$$
$$QRR_c\left(\boldsymbol{\theta}^{k}\right) = \mathbb{C}^{-1}\left(\mathbb{Q}\left(\mathbb{C}\left(\nabla f_c(\boldsymbol{\theta}^{k})\right), \mathbb{C}\left(\nabla f_c(\boldsymbol{\theta}^{k-1})\right)\right)\right) \tag{10}$$

- $\mathbb{Q}$ is the quantization operator, $\mathbb{C}$ is the compression operator, and $\mathbb{C}^{-1}$ is the decompression operator. Each client applies the operators $\mathbb{C}$ and $\mathbb{Q}$ to compress and quantize its gradient update, while the server receives the updates and applies $\mathbb{C}^{-1}$ to decompress them and perform gradient descent.

- For each client c and each model parameter P in the clients' gradient updates,
  - if $P = \boldsymbol{W}_c^k \in \mathbb{R}^{D_{out} \times D_{in}}$:

$$\frac{\partial J}{\partial \boldsymbol{W}_c^k} \approx Q(\boldsymbol{U}_c^k)\, Q(\boldsymbol{\Sigma}_c^k)\, Q(\boldsymbol{V}_c^k)^\top, \tag{11}$$

  - if $P = \mathcal{W}_c^k \in \mathbb{R}^{D_1 \times D_2 \times \ldots \times D_N}$:

$$\frac{\partial J}{\partial \mathcal{W}_c^k} \approx Q(\mathcal{G}_c^k) \times_1 Q((\boldsymbol{F}_1)_c^k) \times_2 Q((\boldsymbol{F}_2)_c^k) \times_3 \ldots \times_N Q((\boldsymbol{F}_N)_c^k), \tag{12}$$

  - if $P = \boldsymbol{b}_c^k \in \mathbb{R}^{D_{out} \times 1}$:

$$\frac{\partial J}{\partial \boldsymbol{b}_c^k} \approx Q(\frac{\partial J}{\partial \boldsymbol{b}_c^k}). \tag{13}$$

Quantized Rank Reduction

QRR Algorithm

Algorithm 1 QRR

Input parameters: α, p, β
Initialize: $\boldsymbol{\theta}^0$, and $\{\mathbb{Q}(\mathbb{C}(\nabla f_c(\boldsymbol{\theta}^0))\}_{c\in\mathcal{C}}$ to $\mathbf{0}$
for $k = 1$ to K **do**
    Server transmits $\boldsymbol{\theta}^k$ to all clients
    for $c = 1$ to C **do**
        Client c computes $\nabla f_c(\boldsymbol{\theta}^k)$
        for each parameter P in $\nabla f_c(\boldsymbol{\theta}^k)$ **do**
            if $P \in \mathbb{R}^{D_{out}\times D_{in}}$ **then**
                Approximate P according to (11)
            else if $P \in \mathbb{R}^{D_1\times\ldots\times D_N}$ **then**
                Approximate P according to (12)
            else if $P \in \mathbb{R}^{D_{out}}$ **then**
                Approximate P according to (13)
            end if
        end for
        Client c transmits the compressed and quantized gradients to the server
        Server updates global model using (10)
    end for
end for

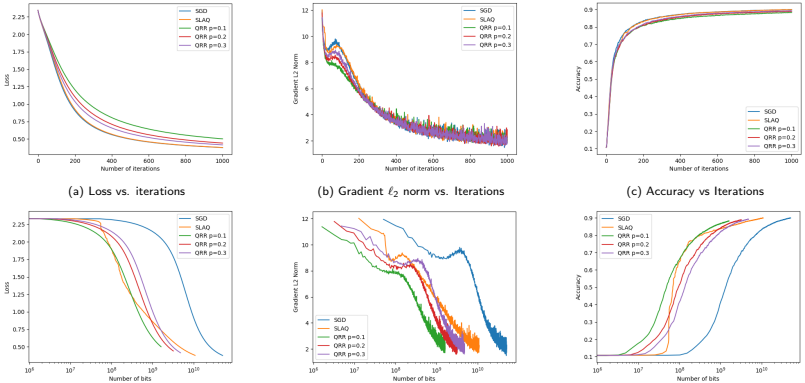
Table 1: Results of QRR compared to SLAQ and SGD for an MLP applied to the MNIST dataset.

Algorithm	# Iterations	# Bits	# Communications	Loss	Accuracy	Gradient ℓ_2 norm
SGD	1000	5.088×10^{10}	10000	0.376	89.92%	2.297
SLAQ	1000	1.089×10^{10}	8559	0.378	89.89%	2.026
QRR($p = 0.3$)	1000	4.798×10^{9}	10000	0.415	89.20%	1.945
QRR($p = 0.2$)	1000	3.205×10^{9}	10000	0.441	88.93%	2.846
QRR($p = 0.1$)	1000	1.612×10^{9}	10000	0.501	88.22%	1.866

- QRR achieves an accuracy of around 1-2% lower than SGD and SLAQ.
- QRR transmits 3.16-9.43% of the bits transmitted by SGD and 14.8-44.05% of the bits transmitted by SLAQ, depending on the choice of the parameter p.

Experimental Results

Experiment 1 - MLP and MNIST



(a) Loss vs. iterations

(b) Gradient ℓ_2 norm vs. Iterations

(c) Accuracy vs Iterations

(d) Loss vs. Bits

(e) Gradient ℓ_2 norm vs. Bits

(f) Accuracy vs. Bits

Figure 2: Loss, gradient ℓ_2 norm, and accuracy plotted against the number of iterations and bits for the MLP network and the MNIST dataset.

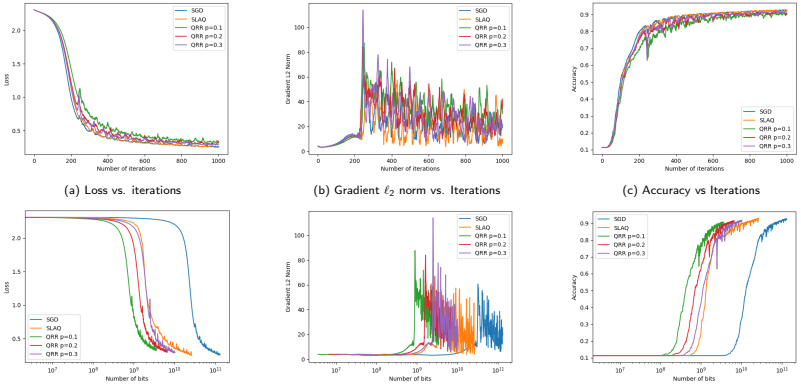
Table 2: Results of QRR compared to SLAQ and SGD for a CNN applied to the MNIST dataset.

Algorithm	# Iterations	# Bits	# Communications	Loss	Accuracy	Gradient ℓ_2 norm
SGD	1000	1.302×10^{11}	10000	0.263	92.56%	21.154
SLAQ	1000	2.653×10^{10}	8151	0.251	92.70%	9.769
QRR($p = 0.3$)	1000	1.022×10^{10}	10000	0.291	91.49%	19.287
QRR($p = 0.2$)	1000	6.650×10^{9}	10000	0.335	89.91%	42.026
QRR($p = 0.1$)	1000	3.588×10^{9}	10000	0.330	90.48%	30.455

- QRR achieves an accuracy of around 1-3% lower than SGD and SLAQ.
- QRR transmits 2.75-7.84% of the bits transmitted by SGD and 13.52-38.52% of the bits transmitted by SLAQ, depending on the choice of the parameter p.

Experimental Results

Experiment 2 - CNN and MNIST



(a) Loss vs. iterations

(b) Gradient ℓ_2 norm vs. Iterations

(c) Accuracy vs Iterations

(d) Loss vs. Bits

(e) Gradient ℓ_2 norm vs. Bits

(f) Accuracy vs. Bits

Figure 3: Loss, gradient ℓ_2 norm, and accuracy plotted against the number of iterations and bits for the CNN and the MNIST dataset.

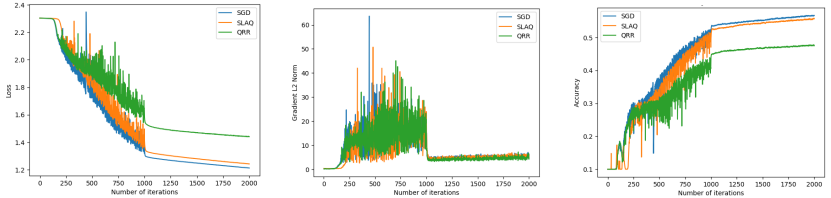
Table 3: Results of QRR compared to SLAQ and SGD for a VGG-like CNN applied to the CIFAR-10 dataset.

Algorithm	**# Iterations**	**# Bits**	**# Communications**	**Loss**	**Accuracy**	**Gradient ℓ_2 norm**
SGD	2000	3.52×10^{11}	20000	1.213	56.72%	6.246
SLAQ	2000	7.72×10^{10}	17548	1.242	55.73%	5.493
QRR	2000	1.17×10^{10}	20000	1.441	47.57%	5.088

- Evenly spaced values in $[0.1, 0.3]$ were assigned to the p parameter of each client to demonstrate that p can be chosen based on the client's connection speed and the amount of data we want transmitted from that client.
- QRR achieves an accuracy of around 8–9% lower than SGD and SLAQ.
- QRR transmits 3.34% of the bits transmitted by SGD and 15.26% of the bits transmitted by SLAQ.

Experimental Results

Experiment 3 - Small VGG-Like CNN and CIFAR-10



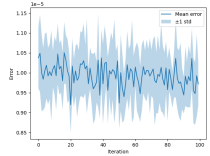
(a) Loss vs. iterations (b) Gradient ℓ_2 norm vs. Iterations (c) Accuracy vs Iterations

(d) Loss vs. Bits (e) Gradient ℓ_2 norm vs. Bits (f) Accuracy vs. Bits

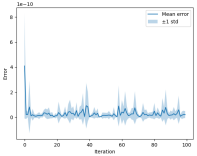
Figure 4: Loss, gradient ℓ_2 norm, and accuracy plotted against the number of iterations and bits for the VGG-like CNN and the CIFAR-10 dataset.

Experimental Results

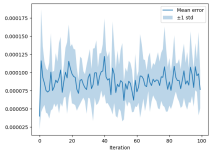
Error Analysis



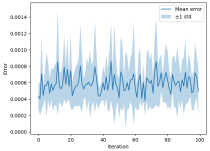
(a) Low-rank approximation error.



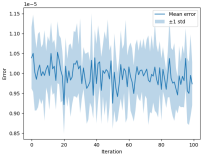
(b) Quantization error of the diagonal of $\boldsymbol{\Sigma}$.



(c) Quantization error of $\boldsymbol{U}$.



(d) Quantization error of $\boldsymbol{V}$.

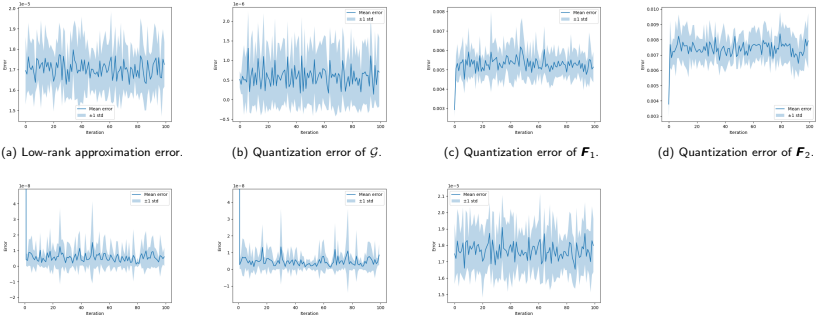


(e) Final error.

Figure 6: Low-rank approximation and quantization errors of the gradient of a fully connected layer and its SVD.

Experimental Results

Error Analysis



(a) Low-rank approximation error. (b) Quantization error of $\mathcal{G}$. (c) Quantization error of $\boldsymbol{F}_1$. (d) Quantization error of $\boldsymbol{F}_2$.

(e) Quantization error of $\boldsymbol{F}_3$. (f) Quantization error of $\boldsymbol{F}_4$. (g) Final error.

Figure 8: Low-rank approximation and quantization errors of the gradient of a convolutional layer and its Tucker decomposition.

Conclusions

Conclusions

1. Combining quantization with compression greatly reduces communication costs in federated learning.
2. SVD's robustness allows low-rank gradient compression paired with quantization, with final error mainly from low-rank approximation.
3. Low-rank approximation error depends on retained rank (p), dataset complexity, and network overparameterization.
4. QRR drastically cuts bits needed for gradient updates versus SGD and LAQ, while maintaining comparable model performance.
5. QRR's convergence is provable under biased gradients, converging near the optimum depending on compression-induced bias.

GitHub Code

The code for the proposed framework can be found at:

`https://github.com/Kritsos/QRR-code`

or by scanning the QR code:

Figure 9: QR code to GitHub.

Thank You!

Thank you very much for your attention.
Q & A?
Email costas@csd.auth.gr