Model-driven Quantum Federated Learning (QFL)

Conference or Workshop Item

Accepted Version

It is advisable to refer to the publisher's version if you intend to cite from the work. See Guidance on citing.

To link to this article DOI: http://dx.doi.org/10.1145/3594671.3594690

Publisher: ACM Digital Library

www.reading.ac.uk/centaur

CentAUR

Model-Driven Quantum Federated Learning (QFL)

Armin Moin
Technical Univ. of Munich & Univ. of
Antwerp, Germany & Belgium
armin.moin@tum.de

Atta Badii
University of Reading
United Kingdom
atta.badii@reading.ac.uk

Moharram Challenger
Univ. of Antwerp
& Flanders Make, Belgium
moharram.challenger@uantwerpen.be

ABSTRACT

Recently, several studies have proposed frameworks for Quantum Federated Learning (QFL). For instance, the Google TensorFlow Quantum (TFQ) and TensorFlow Federated (TFF) libraries have been deployed for realizing QFL. However, developers, in the main, are not as yet familiar with Quantum Computing (QC) libraries and frameworks. A Domain-Specific Modeling Language (DSML) that provides an abstraction layer over the underlying QC and Federated Learning (FL) libraries would be beneficial. This could enable practitioners to carry out software development and data science tasks efficiently while deploying the state of the art in Quantum Machine Learning (QML). In this position paper, we propose extending existing domain-specific Model-Driven Engineering (MDE) tools for Machine Learning (ML) enabled systems, such as MontiAnna, ML-Quadrat, and GreyCat, to support QFL.

CCS CONCEPTS

• **Software and its engineering** → *Development frameworks and environments*; **Domain specific languages**; • **Computing methodologies** → **Machine learning**; • **Hardware** → **Quantum computation**.

KEYWORDS

model-driven engineering, quantum computing, federated machine learning

1 INTRODUCTION AND RELATED WORK

Federated Learning (FL) enables a scalable, privacy-preserving-by-design approach to Machine Learning (ML) since raw data are not exchanged between distributed nodes running the learning algorithm for training the ML model. Hence, in each round, a server that stores the *current* version of the ML model parameters shares these parameters with the distributed nodes, called *clients*, that store the raw data required for ML model training. Each node uses its local data to run a learning algorithm, such as Stochastic Gradient Descent (SGD), and update its local model parameters accordingly. Further, each client sends its *updated* ML model parameters to the server. Finally, the server applies federated averaging [9] to calculate the new ML model parameters and shares these parameters with all of the clients [4].

However, privacy preservation is not the only benefit of this decentralized ML approach. The fact that ML model parameters are exchanged rather than raw data optimizes the network throughput and would contribute to energy efficiency gains and, thus, carbon emissions footprint. This is particularly the case where the data objects consist of multi-variate structures, such as complex, high-resolution color image data, the transmission, and processing of which is typically associated with higher energy cost and carbon emissions [14]. The capability to deliver decentralized ML becomes even more crucial when we deal with quantum data with an inherently fragile nature, which makes them difficult to transfer. Note that some advanced Quantum Computing (QC) technologies, which are adequate for Quantum Machine Learning (QML), require special conditions, such as extremely low temperatures and vibration-free environments, to store qubits and effectively maintain their quantum states [12]. This makes any exchange of quantum data between the distributed nodes running decentralized ML highly difficult and inefficient since quantum data decay gradually as they interact with the environment [4, 12]. The advantage of Quantum Federated Learning (QFL) is that we can process inherently quantum data on the distributed, federated nodes training the ML model (i.e., clients) but use only classical (i.e., non-quantum) data for the ML model parameters, which will be exchanged.

Recently, Chehimi and Saad [4] proposed a framework for QFL. They deployed the Google TensorFlow Quantum (TFQ) and TensorFlow Federated (TFF) libraries. Additionally, Yun et al. [15] proposed SlimQFL to support QFL using Slimmable Neural Networks (SNNs), thus coping with environmental dynamics, such as time-varying communication channel conditions and energy limitations. However, the majority of practitioners in Software Engineering (SE) and Data Science (DS) are yet to become familiar with the Quantum Programming (QP) paradigm and the technologies mentioned above. Therefore, as highlighted by prior work in the literature, for example, Ali and Yue [3], Delgado and Gonzalez [13], Gemeinhardt et al. [5], and Moin et al. [11], modeling languages and the Model-Driven Engineering (MDE) paradigm could be a natural fit and beneficial for QP. However, none of them have pointed out the need for supporting model-driven QFL.

2 PROPOSED APPROACH

We envision a future in the decade ahead in which hybrid quantum-classical applications will deploy QFL technologies to address the

increasingly challenging enablers for AI solutions efficiently. However, as mentioned above, few practitioners (i.e., developers and data scientists) are currently familiar with QP, let alone QFL. Hence, to facilitate the deployment of QFL technologies in ML-enabled software-intensive systems, we propose extending existing DSMLs and MDE tools as enabling technologies to support the development of such systems. This requires enhancement of state-of-the-art tools, such as MontiAnna [8], ML-Quadrat [10], and GreyCat [1, 6] to address QFL.

Realizing the proposed approach will help increase the productivity of practitioners, shorten the time-to-market for new software-based products and services, and increase the quality of software systems. Domain-specific modeling has already been applied to other domains and has resulted in a productivity leap of 500-1000% [7]. Moreover, the possibility of achieving full automation for program synthesis (i.e., generation of the source code and other artifacts, such as ML models) makes turnarounds for developing software-based products and services shorter and increases software quality, for example, by reducing the number of software defects.
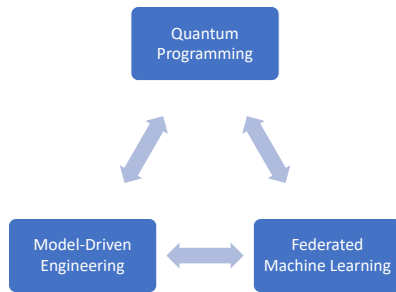

Figure 1: The main pillars of the proposed approach

The DSML and modeling tool for delivering the above enabling technologies should support the user (i.e., the practitioner) to specify the number of distributed nodes (i.e., the clients and the server) participating in FL at the design time. Moreover, for each node, it should be specified whether this node - on the physical layer - possesses a quantum processor or a classical one. Further, the specific quantum hardware technology (e.g., superconducting NISQ vs. trapped-ions) and model of computation (e.g., quantum circuits vs. quantum annealing) may be provided. This should help optimize the generated code for the particular quantum and classical processors on which the QFL application will be deployed.

Additionally, it should be possible to choose the preferred learning algorithm for each client participating in FL, the desired federated averaging algorithm (e.g., appropriate aggregation of model gradients), and the ML model architecture. In each case, the system should be able to revert to a best-suited *default* choice to be selected automatically should the user (i.e., practitioner) not provide the respective information explicitly. For instance, the default choices for the learning algorithm, federated averaging algorithm, and ML model architecture could be SGD, the federated averaging algorithm of [9], and Quantum Convolutional Neural Networks (QCNN) [4], respectively.

In particular, we plan to integrate the open-source TFQ and TFF libraries based on Cirq [2], thus building on prior work by Chehimi and Saad [4]. Furthermore, we intend to extend the open-source ML-Quadrat project [10], built based on the Eclipse Modeling Framework (EMF) and the Xtext framework. Currently, ML-Quadrat supports neither QP nor FL. Thus, an approach integrating both of the requisite capabilities (namely, QP and FL, thus QFL) will be required. Finally, the DSML and tool can be enhanced with more sophisticated QFL solutions, such as the SNN-based approach proposed by Yun et al. [15]. The DSML and tool should be modular and extensible to support the above requirements. Figure 1 illustrates the main pillars of the proposed approach.

3 CONCLUSION AND FUTURE WORK

In this paper, we have proposed our approach to delivering model-driven QFL. We have argued that existing DSMLs and modeling tools should be extended to handle advanced ML needs, specifically regarding QFL. In the future, we plan to realize the proposed solution and conduct experiments to validate the prototype. To this aim, we will build on existing QFL and MDE solutions.

REFERENCES

[1] 2019. GreyCat: Efficient what-if analytics for data in motion at scale. *Information Systems* 83 (2019), 101–117.

[2] n/a. Cirq. https://github.com/quantumlib/Cirq, accessed: 2021-06-24.

[3] Shaukat Ali and Tao Yue. 2020. Modeling Quantum Programs: Challenges, Initial Results, and Research Directions *(APEQS 2020)*. ACM, New York, NY, USA, 14–21.

[4] Mahdi Chehimi and Walid Saad. 2022. Quantum Federated Learning with Quantum Data. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 8617–8621.

[5] Felix Gemeinhardt, Antonio Garmendia, and Manuel Wimmer. 2021. Towards Model-Driven Quantum Software Engineering. In *Second Int. Workshop on Quantum Software Eng. (ICSEW: Proceedings of the IEEE/ACM International Conference on Software Engineering Workshops)*.

[6] T. Hartmann, A. Moawad, F. Fouquet, and Y. Le Traon. 2019. The next evolution of MDE: a seamless integration of machine learning into domain modeling. *Software and System Modeling (SoSyM)* 18 (May 2019), 1285–1304.

[7] Steven Kelly and Juha-Pekka Tolvanen. 2008. *Domain-Specific Modeling: Enabling Full Code Generation* (1st ed.). Wiley.

[8] Evgeny Kusmenko, Sebastian Nickels, Svetlana Pavlitskaya, Bernhard Rumpe, and Thomas Timmermanns. 2019. Modeling and Training of Neural Processing Systems. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*. 283–293.

[9] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proc. of the 20th Int. Conf. on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, 1273–1282.

[10] Armin Moin, Moharram Challenger, Atta Badii, and Stephan Günnemann. 2022. A model-driven approach to machine learning and software modeling for the IoT. *Software and System Modeling (SoSyM)* 21 (2022), 987–1014.

[11] Armin Moin, Moharram Challenger, Atta Badii, and Stephan Günnemann. 2022. Towards Model-Driven Engineering for Quantum AI. In *INFORMATIK 2022*, Daniel Demmler, Daniel Krupka, and Hannes Federrath (Eds.). 1121–1131.

[12] Michael A. Nielsen and Isaac L. Chuang. 2010. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press. https://doi.org/10.1017/CBO9780511976667

[13] Carlos A. Pérez-Delgado and Hector G. Perez-Gonzalez. 2020. Towards a Quantum Software Modeling Language. In *Proc. of the IEEE/ACM 42nd Int. Conf. on Software Engineering Workshops* (Seoul, Republic of Korea) *(ICSEW'20)*. ACM, New York, NY, USA, 442–444.

[14] Rajkumar K. Raval and Atta Badii. 2019. Investigating the Impact of Image Content on the Energy Efficiency of Hardware-Accelerated Digital Spatial Filters. 24, 5 (2019).

[15] Won Joon Yun, Jae Pyoung Kim, Soyi Jung, Jihong Park, Mehdi Bennis, and Joongheon Kim. 2022. Slimmable Quantum Federated Learning. In *DyNN (Dynamic Neural Networks) workshop at the 39th International Conference on Machine Learning (ICML), Baltimore, Maryland, USA*.