Slimmable Quantum Federated Learning

Won Joon Yun [1] **Jae Pyoung Kim** [1] **Soyi Jung*** [2] **Jihong Park*** [3] **Mehdi Bennis** [4] **Joongheon Kim*** [1]

Abstract

Quantum federated learning (QFL) has recently received increasing attention, where quantum neural networks (QNNs) are integrated into federated learning (FL). In contrast to the existing static QFL methods, we propose *slimmable QFL (SlimQFL)* in this article, which is a dynamic QFL framework that can cope with time-varying communication channels and computing energy limitations. This is made viable by leveraging the unique nature of a QNN where its angle parameters and pole parameters can be separately trained and dynamically exploited. Simulation results corroborate that SlimQFL achieves higher classification accuracy than Vanilla QFL, particularly under poor channel conditions on average.

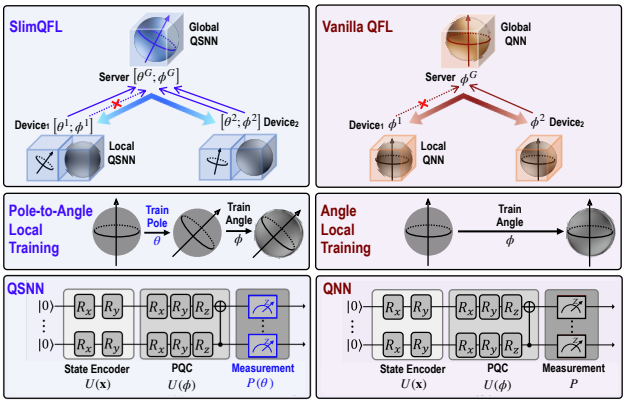

Figure 1: A schematic illustration of SlimQFL (left) wherein the pole and angle parameters of local QSNN models can be separately trained and dynamically communicated, in contrast to Vanilla QFL (right) based on QNNs wherein only angle parameters are trainable.

1. Introduction

Recent advances in noisy intermediate-scale quantum (NISQ) computing processors (Arute et al., 2019) and machine learning (ML) algorithms (Burkart & Huber, 2021) appear to be a prelude to the era of quantum ML (QML) (Schuld & Killoran, 2022). Just like the neural network (NN) of classical ML, QML is implemented by a quantum neural network (QNN) in which a parameterized quantum circuit (PQC) adjusts the input quantum qubit states, and the expected measurement determines the output for a given basis (Bharti et al., 2022). With fewer parameters, QML has achieved the level of performance comparable to classical ML in classification (Schuld, 2021; Havlíček et al., 2019), data generation (Zoufal et al., 2019), and reinforcement learning tasks (Jerbi et al., 2021). Meanwhile, by combining federated learning (FL) (Chen & Yoo, 2021; Huang et al., 2022) with QML, quantum FL (QFL) has shown its potential in utilizing distributed data and quantum computing resources (Chehimi & Saad, 2022).

[1]Korea University, Korea [2]Hallym University, Korea [3]Deakin University, Australia [4]University of Oulu, Finland. Correspondence to: S. Jung <sjung@hallym.ac.kr>, J. Park <jihong.park@deakin.edu.au>, J. Kim <joongheon@korea.ac.kr>.

DyNN workshop at the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, 2022.

While interesting, the current QFL, hereafter referred to as Vanilla QFL, is nothing more than iteratively averaging the PQC parameters. Therefore, it is difficult to cope with environmental dynamics such as time-varying communication channel conditions and energy limitations (Matsubara et al., 2022). To overcome this limitation, we propose a novel dynamic QFL framework, coined *slimmable QFL (SlimQFL)*, inspired by the resemblance between the slimmable NN (SNN) architecture (Baek et al., 2022) and a dyadic nature of the QNN architecture as elaborated next.

The SNN of classical ML is a dynamic architecture in the sense that not only can the entire parameters be trained, but also a fraction of the parameters can be separately trained and exploited. Slimmable FL (SlimFL) utilizes such SNNs as the local models of devices, thereby responding to the time-varying energy and communication channel conditions (Yun et al., 2022). Similarly, a QNN can be viewed as two sets of separately tunable parameters: angle parameters of the PQC and pole parameters of the measurement basis. While existing QNN architectures and training algorithms focus only on the angle parameters (Chen & Yoo, 2021), we propose a quantum SNN (QSNN) wherein both angle and pole parameters can be separately trainable for SlimQFL.

Consequently, during the local training of SlimQFL, each device first trains the pole parameters of its local QSNN, followed by the angle parameters, henceforth referred to as pole training and angle training, respectively. Then, de-

pending on the channel condition during a communication round, each device transmits either the both angle and pole parameters or only the pole parameters (7x smaller in our experiments). A server produces the averaged local QSNNs as a global QSNN that is downloaded by each device. Numerical experiments corroborate that SlimQFL achieves 11.7% higher accuracy than Vanilla QFL in MNIST classification, thanks to its successful reception of at least pole parameters even under poor channel conditions. We additionally validate that pole training without angle training can indeed improve accuracy, demonstrating the potential of QSNN as a dynamic QNN architecture.

2. Preliminaries: From Quantum Machine Learning to Quantum Federated Learning

Basic Quantum Gates. A qubit is a quantum computing unit where the quantum state is represented with two basis $|0\rangle, |1\rangle$ in Bloch sphere (Bouwmeester & Zeilinger, 2000). The quantum state is written as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha^2 + \beta^2 = 1$. Suppose there is a single qubit system, a classical data δ is encoded to quantum state with the rotation gates $R_x(\delta)$, $R_y(\delta)$, and $R_z(\delta)$, where $R_x(\delta)$, $R_y(\delta)$, and $R_z(\delta)$ represent the rotation of δ over x-, y-, and z-axes in Bloch sphere, respectively. In a multiple qubit system, qubits can be entangled with *controlled-NOT (CNOT)* gates. These basic quantum gates configure the QNNs.

QNN based QML. As shown in Fig. 1(right), the structure of a QNN is tripartite: the state encoder, PQC, and the measurement layer (Killoran et al., 2019). In the forward propagation, classical input data $\mathbf{x}$ needs to be first encoded with the state encoder via basic rotation gates, which is a unitary operation and denoted as $U(\mathbf{x})$. Then, the encoded quantum state is processed through the PQC $U(\boldsymbol{\phi})$, a multi-layered set of CNOT gates and rotation gates associated with trainable parameters $\boldsymbol{\phi}$. The output of the PQC is the entangled quantum state that can be measured after applying a projection matrix P onto the reference z-axis. The measured output $\langle O \rangle_{\mathbf{x},\boldsymbol{\phi}} \in [-1, 1]^{|y|}$ is called an *observable*, where $|y|$ denotes the output dimension. Given the observable and the ground-truth of input, the loss $\mathcal{L}(\boldsymbol{\phi})$ is calculated. Subsequently, the QNN is trained accordingly using the stochastic gradient descent algorithm:

$$\tilde{\boldsymbol{\phi}} \leftarrow \boldsymbol{\phi} - \eta \nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}) \tag{1}$$

where η is the learning rate, and the gradient $\nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi})$ is calculated using the parameter shift rule (Mitarai et al., 2018).

Vanilla QFL. `FedAvg` is the standard algorithm in FL with classical NNs (McMahan et al., 2017). In each communication round, the operations of `FedAvg` can be summarized by: (i) each device's local training of an NN; (ii) the parameter server's construction of a global NN by averaging the local NNs; and (iii) each device's replacement of its local NN with the global NN. In Vanilla QFL, (i) is implemented using (1), and the global QNN $\tilde{\boldsymbol{\phi}}^G$ in (ii) is given by the averaged PQC angle parameter, i.e.,

$$\tilde{\boldsymbol{\phi}}^G \leftarrow \frac{1}{\sum_{n=1}^{N} c_n} \sum_{n=1}^{N} c_n \cdot \tilde{\boldsymbol{\phi}}^n, \tag{2}$$

where $\tilde{\boldsymbol{\phi}}^n$ is the n-th device's local NN, and $c_n \in \{0, 1\}$ is an indicator function returning 1 if the n-th device contributes to the global model aggregation. This principle is applied for a privacy-preserving application (Li et al., 2021), binary classification (Chen & Yoo, 2021; Huang et al., 2022), and image classification tasks (Chen & Yoo, 2021).

Algorithm 1: Pole-to-Angle Local-QSNN Train

Notation. $\mathcal{D}$: local trainset, $\mathbf{x}_i$: data of i-th batch, y_i: label of i-th batch, η_l: learning rate in l-th iterations.;
Initialization. local-QNN parameters, $\boldsymbol{\phi}$, $\boldsymbol{\theta}$;
for $l = \{1, 2, \ldots, L\}$ **do**
  for $(\mathbf{x}_i, y_i) \in \mathcal{D}$ **do**
    $\hat{y}_i \leftarrow \mathsf{QSNN}(\mathbf{x}_i; \boldsymbol{\phi}, \boldsymbol{\theta})$ // $\hat{y}_i$: logits;
    Calculate loss, $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}, (\mathbf{x}_i, y_i))$;
    Update pole, $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_l \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}, (\mathbf{x}_i, y_i))$;
$\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$;
for $l = \{1, 2, \ldots, L\}$ **do**
  for $(\mathbf{x}_i, y_i) \in \mathcal{D}$ **do**
    $\hat{y}_i \leftarrow \mathsf{QSNN}(\mathbf{x}_i; \boldsymbol{\phi}, \boldsymbol{\theta})$;
    Calculate loss, $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}, (\mathbf{x}_i, y_i))$;
    Update angle,$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \eta_l \nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}, \tilde{\boldsymbol{\theta}}, (\mathbf{x}_i, y_i))$;
$\tilde{\boldsymbol{\phi}} \leftarrow \boldsymbol{\phi}$;

3. Slimmable Quantum Federated Learning

Departing from Vanilla QFL based on QNNs, we aim to make QFL that can cope with dynamic environments such as time-varying communication channels and energy limitations. To this end, we propose a novel QSNN architecture, and develop SlimQFL involving local QSNN training and global QSNN aggregation operations, as elaborated next.

QSNN Architecture. In a QNN, its feature map is trained by adjusting the PQC angle parameters (Havlíček et al., 2019), and the corresponding qubit states can be represented on the Bloch sphere, as illustrated in Fig. 1(left). Meanwhile, the QNN output is given by the measured qubit states projected onto a hyperplane for a given basis pole of the Bloch sphere. While the standard QML focuses only on tuning the PQC angle parameters, Schuld & Killoran (2022) shows that it is possible to adjust the hyperplane by changing the pole of the measurement. Inspired by this, we propose a *QSNN* where not only can the PQC angle parameters $\boldsymbol{\phi}$ be trained but also the measurement pole parameters $\boldsymbol{\theta}$ can be trained. Therefore, the projection matrix $P_{\boldsymbol{\theta}}$ of QSNN is trainable in comparison to the fixed projection matrix P in QNN.

Pole-to-Angle Local Training. Like Vanilla QFL associated with local QNNs, each device under SlimQFL stores a local QSNN that is trained using its own local dataset. The

Algorithm 2: SlimQFL

Notation. $\tilde{\boldsymbol{\theta}}^n, \tilde{\boldsymbol{\phi}}^n$: n-th device's pole/angle parameters, $\tilde{\boldsymbol{\theta}}^G, \tilde{\boldsymbol{\phi}}^G$: pole/angle parameters of server-side QSNN;
Initialization. $\forall c_{\boldsymbol{\theta}}^n, c_{\boldsymbol{\phi}}^n \leftarrow 0$;
for $n = \{1, \ldots, N\}$ **do**
  Sample $\chi^n \sim \exp(1)$;
  if $R^n \geq u_{th}^{whole}$ **then**
    Transmit pole/angle parameters, $(\tilde{\boldsymbol{\theta}}^n, \tilde{\boldsymbol{\phi}}^n)$;
    $c_{\boldsymbol{\theta}}^n, c_{\boldsymbol{\phi}}^n \leftarrow 1$;
  else if $R^n \geq u_{th}^{pole}$ **then**
    Transmit angle parameters, $\tilde{\boldsymbol{\theta}}^n$;
    $c_{\boldsymbol{\theta}}^n \leftarrow 1$;
if $\sum_{n=1}^N c_{\boldsymbol{\theta}}^n \neq 0$ *and* $\sum_{n=1}^N c_{\boldsymbol{\phi}}^n \neq 0$ **then**
  Update with (4);

Description	Value
Number of devices (N)	{2,5,**10**,20}
Number of local iterations per epoch (L)	{2,5,**10**,20}
Epoch (E)	200
Optimizer	SGD
Learning rate (η_0), Decaying rate	$0.01, 0.001$
Observable hyperparameter (w)	1.6
Number of qubits	4
Number of parameters in SlimQFL & Vanilla QFL	40
Number of parameters in SlimQFL-Pole	4
Number of parameters in Classical FL	56
Number of data per device	64
Batch size (B)	$\{4, 8, 16, \mathbf{32}\}$
Test batch size	128
Noise (σ^2)	$\{-20, -30, -40\}$ dB

Table 1: List of simulation parameters.

new key element is that SlimQFL trains the angle and pole parameters of QSNN separately in a sequential way. Since the number of pole parameters is commonly smaller than that of angle parameters, we first train the pole parameters $\boldsymbol{\theta}$, then train the angle parameters $\boldsymbol{\phi}$. After L local iterations, the local QSNN $[\tilde{\boldsymbol{\theta}}^n; \tilde{\boldsymbol{\phi}}^n]$ of the n-th device is updated as:

$$\begin{bmatrix} \tilde{\boldsymbol{\theta}}^n \\ \tilde{\boldsymbol{\phi}}^n \end{bmatrix} \leftarrow \begin{bmatrix} \boldsymbol{\theta}^n \\ \boldsymbol{\phi}^n \end{bmatrix} - \eta_t \begin{bmatrix} \sum_{l=1}^L \nabla_{\boldsymbol{\theta}_l^n} \mathcal{L}(\boldsymbol{\phi}^n, \boldsymbol{\theta}_l^n) \\ \sum_{l=1}^L \nabla_{\boldsymbol{\phi}_l^n} \mathcal{L}(\boldsymbol{\phi}_l^n, \tilde{\boldsymbol{\theta}}^n) \end{bmatrix}, \quad (3)$$

where η_t denotes the learning rate at time t.

Dynamic Global Model Aggregation. By leveraging the QSNN architecture, at each communication round, the n-th device uploads either: (i) only the pole parameters $\tilde{\boldsymbol{\theta}}^n$ or (ii) both the pole and angle parameters $[\tilde{\boldsymbol{\theta}}^n; \tilde{\boldsymbol{\phi}}^n]$, depending on its communication channel condition, energy availability, and/or other time-varying environmental factors. The parameter server aggregates the uploaded parameters accordingly and constructs a global QSNN $[\tilde{\boldsymbol{\theta}}^G; \tilde{\boldsymbol{\phi}}^G]$:

$$\begin{bmatrix} \tilde{\boldsymbol{\theta}}^G \\ \tilde{\boldsymbol{\phi}}^G \end{bmatrix} \leftarrow \begin{bmatrix} \frac{1}{\sum_{n=1}^N c_{\boldsymbol{\theta}}^n} \sum_{n=1}^N c_{\boldsymbol{\theta}}^n \tilde{\boldsymbol{\theta}}^n \\ \frac{1}{\sum_{n=1}^N c_{\boldsymbol{\phi}}^n} \sum_{n=1}^N c_{\boldsymbol{\phi}}^n \tilde{\boldsymbol{\phi}}^n \end{bmatrix}, \quad (4)$$

where the indicator functions $c_{\boldsymbol{\theta}}^n$ and $c_{\boldsymbol{\phi}}^n$ count pole and angle uploading events respectively. Finally, each device downloads the global QSNN $[\tilde{\boldsymbol{\theta}}^G; \tilde{\boldsymbol{\phi}}^G]$, and iterates the operations mentioned above until convergence, as summarized in Algorithm 2.

4. Numerical Experiments

4.1. Simulation Settings

To show the effectiveness of SlimQFL with both pole training and angle training of QSNNs, we consider the following baselines: SlimQFL-Pole with only pole training of QSNNs, Vanilla QFL with QNNs (Chehimi & Saad, 2022), and Classical FL with classical NNs (McMahan et al., 2017). For a fair comparison, we consider that QSNNs, QNNs, and classical NNs have almost the same number of trainable parameters. The performance is measured using the top-1 accuracy in an MNIST classification task. Since quantum computing suffers from the lack of input qubits, we consider a *mini-version of the MNIST (mini-MNIST)* task, where the images are interpolated into 7x smaller sizes via inter-area interpolation. Compared to MNIST, mini-MNIST consists of 4 labels (*i.e.*, $\{0, 1, 2, 3\}$). We conduct all experiments on mini-MNIST with *independent and identically distributed (IID)* data. Other important simulation parameters are summarized in Table 1. The bold-faced parameters imply the values used for Fig. 2–Fig. 4.

To showcase the dynamic characteristics of SlimQFL, we consider time-varying channel conditions in the uplink communications from each device to the server, while the downlink communications are assumed to be perfect. In the uplink, the throughput R^n of the n-th device is given as $R^n = \log_2(1 + g^n/\sigma^2)$ (bits/sec), where σ^2 is a constant noise power and $g^n \sim \exp(1)$ is a random gain under Rayleigh fading (Tse & Viswanath, 2005). When the transmitter encodes an input with a code rate u, its receiver successfully obtains the encoded data if $R > u$. Then, we consider the standard opportunistic transmission by assuming the channel condition is known at the transmitter. Namely, in a good channel condition, the device transmits both pole and angle parameters $[\tilde{\boldsymbol{\theta}}^n; \tilde{\boldsymbol{\phi}}^n]$ to the server, while in a poor channel condition, the device transmits only pole parameters $\tilde{\boldsymbol{\theta}}^n$. Consequently, the server constructs a global QSNN by aggregating the receptions, while counting the pole and angle parameter receptions using $c_{\boldsymbol{\theta}}^n = \mathbb{1}(R^n \geq u_{th}^{pole})$ and $c_{\boldsymbol{\phi}}^n = \mathbb{1}(R^n \geq u_{th}^{whole})$, respectively.

4.2. Results

Effectiveness of QSNN. As mentioned above, the proposed scheme changes the parameters being transmitted according to the channel conditions. To examine the robustness of SlimQFL in various channel conditions, we consider three different conditions (i.e., good, moderate, and poor). The proposed scheme is expected to produce high accuracy with good channel condition and maintains that result even as the channel condition deteriorates, thanks to the advantage of transmission opportunity. Our prediction is verified in

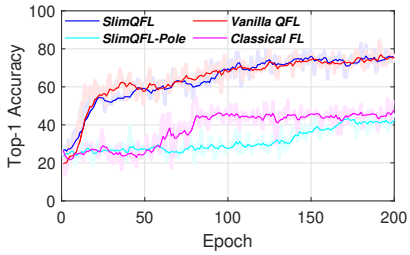

Figure 2: Good channel ($\sigma^2 = -40$dB).

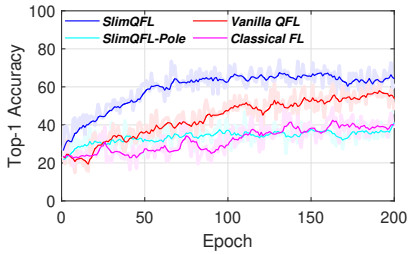

Figure 3: Moderate channel ($\sigma^2 = -30$dB).

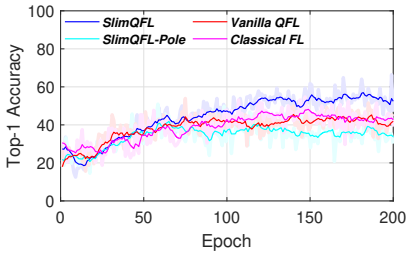

Figure 4: Poor channel ($\sigma^2 = -20$dB).

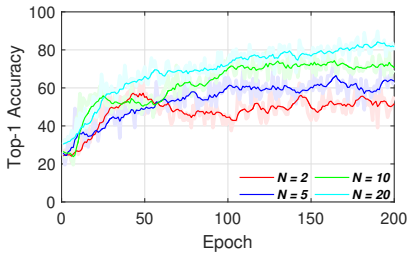

Figure 5: Number of devices.

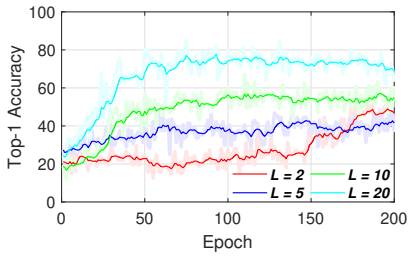

Figure 6: Number of local iterations.

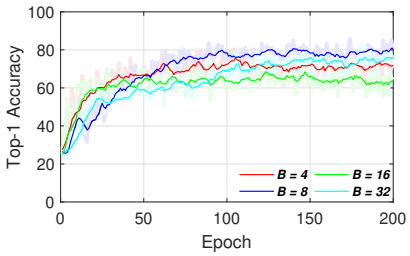

Figure 7: Batch size.

Fig. 2–4 where we confirm that in good channel conditions, all four schemes yield high accuracy. Despite SlimQFL-Pole having small parameters and low performance, its learning curve shows that SlimQFL-Pole is trained. However, as the channel condition deteriorates, the performance degradation of Vanilla QFL and Classical FL is observed while SlimQFL-Pole and Proposed maintain similar levels of accuracy. Among them, Proposed shows the highest accuracy. Therefore, we conclude that our proposed scheme is superior to the other algorithms.

Effectiveness of Quantum Computing. The efficacy of quantum computing is compared with four different schemes. FL using QNNs (i.e., Proposed, Vanilla QFL) is predicted to show superior performance to its competitors due to the utilization of quantum computing. This is proven true by the results of Fig. 2 which shows the accuracy results of each algorithm in a good channel condition. Proposed and Vanilla QFL outperforms Classical FL in terms of top-1 accuracy. However, SlimQFL-Pole shows low performance because it consists of only pole parameters. Thus, we conclude that FL utilizing QNNs has outperformed the classical FL in terms of top-1 accuracy level due to the property of quantum computing.

Number of Devices. The number of devices, N, affects the model's performance. According to (Li et al., 2020), the more participating devices, the higher the performance of the model. In Fig. 5, we see that SlimQFL with 20 devices shows significantly improved results compared to the other baselines which verify the statement above.

Number of Local Iterations. Similarly, the number of local iterations, L, has a notable impact on the performance. According to (Li et al., 2020), `FedAvg` shows a higher performance when the number of local iterations is small but when the number of local iterations exceeds a certain threshold, the performance declines (*e.g.,* 20 (Wang et al., 2020)). It is possible that this trend may not apply to our paper because of QSNN. Fig. 6 corroborates that the larger number of local iterations, the top-1 accuracy also increases. Our results matched the research trends in FL.

Batch Size. We investigate the impact of the batch size used for training by carrying out simulations using 4, 8, 16, and 32 as batch sizes. Fig. 7 shows the simulation results. The results in the figure show that the simulation with a batch size of 8 produces the highest accuracy while the simulation with a batch size of 16 has the lowest accuracy. By observing the accuracy of the other two simulations, it is deduced that there is no coherent relationship between model performance and the batch size used in training.

5. Conclusion

In this paper, we propose SlimQFL, a novel QFL framework with QSNN and a training algorithm to achieve adaptability to a dynamic communication environment. The numerical results corroborate that SlimQFL is robust to poor channel conditions compared to QFL or classical FL. Furthermore, we verify that there is a certain advantage in utilizing quantum computing. Based on these remarkable results, it could be an interesting topic to analyze the convergence of SlimQFL under various communication channel conditions as well as global data distributions that would be non-IID.

Acknowledgement. This research was supported by NRF-Korea (2021R1A4A1030775, 2022R1A2C2004869, 2019M3E4A1080391) and also by MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2022-2017-0-01637) by IITP (Institute for Information & Communications Technology Planning & Evaluation).

References

Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., Biswas, R., Boixo, S., Brandao, F. G., Buell, D. A., et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

Baek, H., Yun, W. J., Kwak, Y., Jung, S., Ji, M., Bennis, M., Park, J., and Kim, J. Joint superposition coding and training for federated learning over multi-width neural networks. In *Proc. of INFOCOM*, Virtual, May 2022.

Bharti, K., Cervera-Lierta, A., Kyaw, T. H., Haug, T., Alperin-Lea, S., Anand, A., Degroote, M., Heimonen, H., Kottmann, J. S., Menke, T., et al. Noisy intermediate-scale quantum algorithms. *Reviews of Modern Physics*, 94(1):015004, 2022.

Bouwmeester, D. and Zeilinger, A. The physics of quantum information: basic concepts. In *the Physics of Quantum Information*, pp. 1–14. 2000.

Burkart, N. and Huber, M. F. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

Chehimi, M. and Saad, W. Quantum federated learning with quantum data. In *Proc. of ICASSP*, pp. 8617–8621, Singapore, China, May 2022.

Chen, S. Y.-C. and Yoo, S. Federated quantum machine learning. *Entropy*, 23(4):460, 2021.

Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., and Gambetta, J. M. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

Huang, R., Tan, X., and Xu, Q. Quantum federated learning with decentralized data. *IEEE Journal of Selected Topics in Quantum Electronics*, 2022.

Jerbi, S., Gyurik, C., Marshall, S., Briegel, H. J., and Dunjko, V. Variational quantum policies for reinforcement learning. In *Proc. of NeurIPS*, Virtual, December 2021.

Killoran, N., Bromley, T. R., Arrazola, J. M., Schuld, M., Quesada, N., and Lloyd, S. Continuous-variable quantum neural networks. *Physical Review Research*, 1(3):033063, 2019.

Li, W., Lu, S., and Deng, D.-L. Quantum federated learning through blind quantum computing. *Science China Physics, Mechanics & Astronomy*, 64(10):1–8, 2021.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. In *Proc. of ICLR*, Addis Ababa, Ethiopia, April 2020.

Matsubara, Y., Callegaro, D., Singh, S., Levorato, M., and Restuccia, F. Bottlefit: Learning compressed representations in deep neural networks for effective and efficient split computing. *arXiv preprint arXiv:2201.02693*, 2022.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Proc. of AISTATS*, pp. 1273–1282, 2017.

Mitarai, K., Negoro, M., Kitagawa, M., and Fujii, K. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

Schuld, M. Supervised quantum machine learning models are kernel methods. *CoRR*, abs:2101.11020, January 2021.

Schuld, M. and Killoran, N. Is quantum advantage the right goal for quantum machine learning? *arXiv preprint arXiv:2203.01340*, 2022.

Tse, D. N. C. and Viswanath, P. *Fundamentals of Wireless Communications*. 2005.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., and Khazaeni, Y. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

Yun, W. J., Kwak, Y., Baek, H., Jung, S., Ji, M., Bennis, M., Park, J., and Kim, J. SlimFL: Federated learning with superposition coding over slimmable neural networks. *CoRR*, abs/2203.14094, 2022.

Zoufal, C., Lucchi, A., and Woerner, S. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):1–9, 2019.