Data-driven Acceleration of Quantum Optimization and Machine Learning via Koopman Operator Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Efficient optimization methods play a crucial role for quantum optimization and machine learning on near-term quantum computers. Unlike classical computers, obtaining gradients on quantum computers is costly with sample complexity scaling with the number of parameters and measurements. In this paper, we connect the natural gradient method in quantum optimization with Koopman operator theory, which provides a powerful framework for predicting nonlinear dynamics. We propose a data-driven approach for accelerating quantum optimization and machine learning via Koopman operator learning. To predict parameter updates on quantum computers, we develop new methods including the sliding window dynamic mode decomposition (DMD) and the neural-network-based DMD. We apply our methods both on simulations and real quantum hardware. We demonstrate efficient prediction and acceleration of gradient optimization on the variational quantum eigensolver and quantum machine learning.

1 Introduction

The fields of quantum technology and quantum computation are under rapid development in recent years. Two promising applications of quantum technology, quantum optimization [34] and quantum machine learning (QML) [5], have gained increased interest. The Variational Quantum Algorithm (VQA) [7] such as the Quantum Approximate Optimization Algorithm (QAOA) [13] has been developed for solving optimization on graphs. By benchmarking against a variety of classical algorithms, a recent experiment on 289 qubits has demonstrated a powerful application of VQA for classical optimization problems [12]. The Variational Quantum Eigensolver (VQE) [37, 50] has been applied to find quantum states for understanding science in high energy physics [22, 42], condensed matter physics [52], and quantum chemistry [36]. In quantum machine learning, both theoretical advantages have been investigated [19, 29, 17, 20, 10], and experiments on real quantum computers have demonstrated encouraging progress [18, 43].

In the noisy intermediate-scale quantum (NISQ) era [39], the current quantum computers suffer from noise and the computation resources are limited. To address the issue, hybrid classical-quantum schemes have been developed for quantum optimization and quantum machine learning, which perform optimization and machine learning on parameterized quantum circuits with quantum features while optimizing the circuit's parameters through classical computers. In contrast to classical machine learning, for which the backprogation algorithm shares the same complexity as the forward evaluation, the hybrid scheme has much higher cost for obtaining gradients due to: (1) gradient calculation

Submitted to NeurIPS 2022 AI for Science Workshop.

typically has linear scaling with respect to the number of parameters; and (2) repeated measurements are required to estimate the gradient for each parameter. Despite their popularity, gradient-based methods on real quantum computers are computationally inefficient to implement, which limits their applications in practice. To develop scalable and efficient optimization methods for quantum optimization and quantum machine learning is an important open problem in the field.

Here, we propose a data-driven approach for accelerating quantum optimization and quantum machine learning via Koopman operator learning [24]. By viewing parameter optimization on quantum computers as a nonlinear dynamical process in the parameter space, we connect gradient dynamics in quantum optimization to the Koopman operator theory, which has been developed to successfully predict nonlinear dynamics through linear dynamics embedded into a higher dimensional space [6]. In particular, we find that the Koopman operator theory has a natural connection to the quantum natural gradient method. Based on the insights of the theory, we develop new Koopman operator learning algorithms, including the sliding window dynamic mode decomposition (SW-DMD) and neural-network-based DMD, for quantum optimization and quantum machine learning. Our approach is data-driven and efficiently predicts the gradient dynamics with cost that does not scale with the number of parameters. We show the effectiveness of our algorithms for the variational quantum eigensolver with the natural gradient and Adam [21] optimizers on quantum Ising model simulations and demonstrate their success on a real IBM quantum computer. Furthermore, using our methods, we accelerate quantum machine learning on the fashion-MNIST dataset.

2 Related Work

Koopman operators. Koopman operator theory [24, 51] was first proposed by Koopman and von Neumann in early 1930s to understand dynamical systems. Dynamic mode decomposition (DMD) [44] was developed to learn the Koopman operator under the linear dynamics assumption of the observed data. Later, more advanced methods such as the extended-DMD based on kernel methods [3] and dictionary learning [25] were introduced to go beyond the linear dynamics assumption, and achieved better performance. Recently, machine learning methods were integrated into Koopman operator learning where neural networks are used to learn the mapping to high dimensional space, in which the dynamics becomes linear [32, 27]. The machine learning Koopman operator methods were shown to learn nonlinear differential equation dynamics successfully. Furthermore, Koopman operator theory was applied to optimize neural network training [9, 49] and pruning [40]. These works take the perspective of viewing the optimization process of neural networks as a nonlinear dynamical evolution and uses dynamic mode decomposition to predict the parameter updates in the future. In a more empirical study, [46] trained a convolutional neural network (CNN) to predict the future weights of neural networks, trained on standard vision tasks. Recently, researchers considered Koopman operator theory for quantum control [15] and prediction of one particle quantum system evolution [23]. Since quantum mechanical systems provide a natural high dimensional Hilbert space through the wave function, the theory was considered for embedding classical equations on quantum computers for learning and solving differential equations [28, 14].

Optimization methods on quantum computers. To perform optimization on quantum computers, gradient-free methods such as SPSA [47] and COBYLA [38] are used though they may not scale well to quantum circuits with large number of parameters. For gradient methods, Adam is a common choice for better scaling though its complexity scales with the number of parameters as discussed above. There are also higher order methods such as the quantum natural gradient method [48] with faster convergence but more challenges to realize experimentally.

Our work. There are several important features of our work that distinguish it from the relevant works above. First, our goal is to accelerate quantum optimization and quantum machine learning, which compared to classical neural networks has much higher complexity of taking gradients, such that the gain of acceleration will be much more prominent. Second, instead of predicting the gradient dynamics directly, we focus on decreasing the loss function during the training and develop iterative

optimal prediction algorithms, which is much more flexible and effective than the one-time prediction in the previous work [9]. Third, the previous literature on Koopman operators for neural networks applies only the standard DMD which may not be well situated for complicated nonlinear training dynamics. Instead, we develop and investigate the sliding window DMD (SW-DMD) and several variants of neural-network-based DMD, including the multi-layer perceptron DMD (MLP-DMD), MLP-SW-DMD and CNN-DMD.

3 Connecting Koopman Operator Theory and Quantum Optimization

3.1 Koopman Operator Theory

Consider a dynamical system with a collection of state variables $\{x(t) \in \mathbb{R}^n\}$ with a transition function T such that $x(t+1) = T(x(t))$. According to the Koopman operator theory [24], there exists a linear operator $\mathcal{K}$ and a function g such that

$$\mathcal{K}g(x(t)) = g(T(x(t))) = g(x(t+1)) \tag{1}$$

where $\mathcal{K}$ is the Koopman operator. While the Koopman operator generally can act on an infinite-dimensional space, it can be represented as a Koopman matrix $K \in \mathbb{R}^{m\times m}$ when it is constrained to a finite dimensional invariant subspace with $g : \mathbb{R}^n \to \mathbb{R}^m$. An important question in Koopman operator theory is to search for the function g, and a number of methods have been developed to tackle the problem [32, 27, 44, 3, 25].

3.2 Variational Quantum Eigensolver (VQE)

Consider a Hamiltonian $\mathcal{H}$ describing interactions in a physical system with N qubits. $\mathcal{H}$ is a Hermitian operator acting on the 2^N-dimesional Hilbert space for wave functions, which are l_2-normalized complex-valued vectors. The energy of the system from a wave function ψ is given by $\mathcal{L}(\psi) = \langle \psi, \mathcal{H}\psi\rangle$. VQE encodes the wave function as $\psi_{\boldsymbol{\theta}}$ by a set of parameters $\boldsymbol{\theta} \in \mathbb{R}^{n_{\text{params}}}$ via a parameterized quantum circuit as the top left part of Figure 1 shows. The goal of VQE is to minimize the loss by minimizing for $\boldsymbol{\theta}$ in the following objective $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \langle \psi_{\boldsymbol{\theta}}, \mathcal{H}\psi_{\boldsymbol{\theta}}\rangle$.

Quantum machine learning has a similar setup that targets at minimizing $\mathcal{L}(\boldsymbol{\theta})$, and parameterized quantum circuits in QML are called quantum neural networks. To compute the gradient for the quantum circuit, it requires to evaluate the loss with a perturbation in each direction i, for example by using the parameter-shift rule [33, 45] $(\mathcal{L}(\theta_i + \pi/2) - \mathcal{L}(\theta_i - \pi/2))/2$, which leads to an $O(n_{\text{params}})$ computational cost per iteration. This is much more expensive than $O(1)$ cost of backpropagation per iteration in classical machine learning. Hence, the classical computational resources in the Koopman operator learning, even including training neural-network-based algorithms in the following sections, typically are much cheaper than the quantum cost.

3.3 Quantum Fisher Information and Quantum Natural Gradient

The quantum natural gradient method [48] generalizes the classical natural gradient method [1] to the context of wave function optimization. The following nonlinear differential equation describes the natural gradient update for parameter θ under $\arg\min_{\boldsymbol{\theta}} \langle \psi_{\boldsymbol{\theta}}, \mathcal{H}\psi_{\boldsymbol{\theta}}\rangle$

$$\frac{d}{dt}\boldsymbol{\theta}(t) = -\eta F^{-1} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}(t)), \tag{2}$$

where η is the learning rate, and F is the quantum Fisher Information matrix given by $F_{ij} = \langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i}, \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j}\rangle - \langle \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_i}, \psi_{\boldsymbol{\theta}}\rangle\langle \psi_{\boldsymbol{\theta}}, \frac{\partial \psi_{\boldsymbol{\theta}}}{\partial \theta_j}\rangle$. The above equation for $\boldsymbol{\theta}$ can be shown equivalent to the dynamical equation of $\psi_{\boldsymbol{\theta}}(t)$ with the projection operator $\mathbb{P}_{\psi_\theta}$ on the manifold given by the parameterized quantum circuit [16]

$$\frac{d\psi_{\boldsymbol{\theta}}(t)}{dt} = -\mathbb{P}_{\psi_{\boldsymbol{\theta}}} \mathcal{H}\psi_{\boldsymbol{\theta}}(t). \tag{3}$$

Since $\mathcal{H}$ is a linear operator, Eq. 3 is close to a linear differential equation when the parametrized quantum circuit is sufficiently expressive such that the projection is within the manifold. Viewing

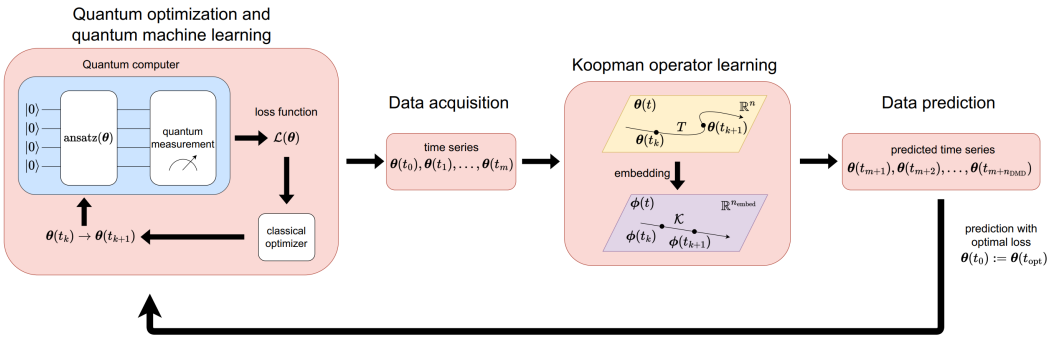


Figure 1: Koopman operator learning for quantum optimization and quantum machine learning under a hybrid quantum-classical scheme.

the parameters $\boldsymbol{\theta}(t)$ as the state variable $x(t)$ in the Koopman theory, the wave function $\psi_{\boldsymbol{\theta}}$ naturally generated by the quantum circuit plays the role of g in Eq. 1, which makes the embedded dynamics close to linear in the Hilbert space. The existence of $\psi_{\boldsymbol{\theta}}$ and its approximate linear dynamics builds the theoretical foundation for Koopman operator learning algorithms in optimization and the application of parameter update acceleration.

4 Koopman Operator Learning Algorithms

4.1 Our framework: data-driven approach for accelerating optimization

Our approach is shown in Figure 1. We use a quantum computer to measure $\mathcal{L}$ at the initial parameter $\boldsymbol{\theta}(t_0)$ of the quantum circuit. Then, we use an optimizer on a classical computer, such as Adam to update the parameter $\boldsymbol{\theta}$, thus incrementing the optimization iteration by 1. If the optimizer is gradient-based, for each iteration, $O(n_{\text{params}})$ additional quantum measurements are needed to evaluate the gradient. Next, we input the updated parameters into the quantum circuit and repeat this procedure for m steps. We acquire a time sequence $\boldsymbol{\theta}(t_0), \boldsymbol{\theta}(t_1), ..., \boldsymbol{\theta}(t_m)$. We then the apply Koopman operator learning algorithms to the acquired data and predict the time series for n_{DMD} steps to obtain $\boldsymbol{\theta}(t_{m+1}), \boldsymbol{\theta}(t_{m+2}), ..., \boldsymbol{\theta}(t_{m+n_{\text{DMD}}})$. The loss at these predicted parameters can also be evaluated on the quantum computer, but no gradient measurement is needed. From these n_{DMD} loss function values, we find the lowest loss and the corresponding time t_{opt}. The optimal Koopman-predicted parameter $\boldsymbol{\theta}(t_{\text{opt}})$ is then used as the initial point for the next quantum optimization and quantum machine learning. We repeat this cycle to accelerate the loss decrease with less resources.

4.2 Sliding Window DMD (SW-DMD)

Dynamic mode decomposition [6] uses a linear fit for the dynamics in the original space for the column vector $\boldsymbol{\theta} \in \mathbb{R}^n$ as follows $\boldsymbol{\theta}(t_{k+1}) = K\boldsymbol{\theta}(t_k)$. We concatenate $\boldsymbol{\theta}$ at successive times to obtain two data matrices

$$\boldsymbol{\Theta}(t_0) = [\boldsymbol{\theta}(t_0) \quad \boldsymbol{\theta}(t_1) \quad \cdots \quad \boldsymbol{\theta}(t_m)], \quad \boldsymbol{\Theta}(t_1) = [\boldsymbol{\theta}(t_1) \quad \boldsymbol{\theta}(t_2) \quad \cdots \quad \boldsymbol{\theta}(t_{m+1})], \tag{4}$$

where $\boldsymbol{\Theta}(t_1)$ is the one-step time evolution of $\boldsymbol{\Theta}(t_0)$. In the case of approximate linear dynamics, the matrix K is the same for all times t_k, and then the DMD fit becomes $\boldsymbol{\Theta}(t_1) \approx K\boldsymbol{\Theta}(t_0)$, where $K \in \mathbb{R}^{n\times n}$. The best fit is at the minimum of the Frobenius loss

$$K = \arg\min_K \|\boldsymbol{\Theta}(t_1) - K\boldsymbol{\Theta}(t_0)\|_F = \boldsymbol{\Theta}(t_1)\boldsymbol{\Theta}(t_0)^+, \tag{5}$$

where $^+$ is the pseudo-inverse.

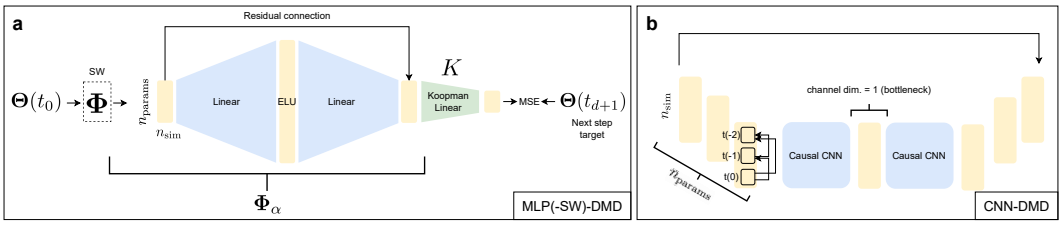


Figure 2: Neural network architectures for our neural-network-based DMD approaches. (a) MLP bottleneck architecture with MSE loss for training. (b) CNN bottleneck architecture that operates on simulations as temporal dimension and parameters as channel dimension.

When the dynamics of $\boldsymbol{\theta}$ is not linear, we can instead consider a time-delayed embedding with a sliding window and concatenate the steps to form an extended data matrix [11]

$$\boldsymbol{\Phi}(\boldsymbol{\Theta}(t_0)) = [\boldsymbol{\phi}(t_0) \quad \boldsymbol{\phi}(t_1) \quad \cdots \quad \boldsymbol{\phi}(t_m)] = \begin{bmatrix} \boldsymbol{\theta}(t_0) & \boldsymbol{\theta}(t_1) & \cdots & \boldsymbol{\theta}(t_m) \\ \boldsymbol{\theta}(t_1) & \boldsymbol{\theta}(t_2) & \cdots & \boldsymbol{\theta}(t_{m+1}) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\theta}(t_d) & \boldsymbol{\theta}(t_{d+1}) & \cdots & \boldsymbol{\theta}(t_{m+d}) \end{bmatrix}. \tag{6}$$

$\boldsymbol{\Phi}$ is generated by a sliding window of size $d+1$ at $m+1$ consecutive time steps. Each column of $\boldsymbol{\Phi}$ is a time-delayed embedding for $\boldsymbol{\Theta}$, and the different columns $\boldsymbol{\phi}$ in $\boldsymbol{\Phi}$ are embeddings at different starting times. The time-delayed embedding captures some nonlinearity in the dynamics of $\boldsymbol{\theta}$, with

$$\boldsymbol{\Theta}(t_{d+1}) \approx K\boldsymbol{\Phi}(\boldsymbol{\Theta}(t_0)), \tag{7}$$

where $K \in \mathbb{R}^{n \times n(d+1)}$. The best fit is given by

$$K = \arg\min_K \|\boldsymbol{\Theta}(t_{d+1}) - K\boldsymbol{\Phi}(\boldsymbol{\Theta}(t_0))\|_F = \boldsymbol{\Theta}(t_{d+1})\boldsymbol{\Phi}(\boldsymbol{\Theta}(t_0))^+. \tag{8}$$

The used data from the acquired time series with the largest time in the above equation is $\boldsymbol{\theta}(t_{m+d+1})$. When making prediction, we start with $\boldsymbol{\theta}(t_{m+d+2}) = K\boldsymbol{\phi}(t_{m+1})$. Then we update from $\boldsymbol{\phi}(t_{m+1})$ to $\boldsymbol{\phi}(t_{m+2})$ by removing the oldest data $\boldsymbol{\theta}(t_{m+1})$ and adding the newly predicted data $\boldsymbol{\theta}(t_{m+d+2})$. We repeat this prediction using $\boldsymbol{\theta}(t_{k+d+1}) = K\boldsymbol{\phi}(t_k)$ iteratively. Our approach is different from the approach used by [11], as we do not perform an additional singular value decomposition before doing DMD and our matrix K is non-square.

We denote DMD performed this way as sliding window DMD (SW-DMD). The standard DMD is a special case of SW-DMD when the sliding window size is 1 (*i.e.*, $d = 0$).

4.3 Neural DMD

General formulation. In order to improve DMD by making it more amenable to nonlinear dynamics, we investigate whether $\boldsymbol{\Phi}$ can be a neural network. By simply writing $\boldsymbol{\Phi}$ in Eq. 8 as a neural network, we obtain

$$\arg\min_{K,\alpha} \|\boldsymbol{\Theta}(t_{d+1}) - K\boldsymbol{\Phi}_\alpha(\boldsymbol{\Theta}(t_0))\|_F, \tag{9}$$

where $K \in \mathbb{R}^{N_{in} \times N_{out}}$ is a linear Koopman operator, and $\boldsymbol{\Phi}_\alpha(\boldsymbol{\Theta}(t_0))$ is a nonlinear neural embedding by a neural network $\boldsymbol{\Phi}_\alpha$ with parameters α. More specifically, $\boldsymbol{\Phi}_\alpha := \mathrm{NN}_\alpha \circ \boldsymbol{\Phi}$ is a composition of the neural network architecture NN_α and the sliding window embedding $\boldsymbol{\Phi}$ from the previous section. The parameters K and α are trained jointly on Eq. 9 from scratch with every new batch of optimization history by using the Adam optimizer. See hyperparameter details in Appendix C. We use the MSE loss, which minimizes the Frobenius norm.

MLP-DMD, CNN-DMD and MLP-SW-DMD methods. We consider three natural choices for the architecture of the neural network, which we list below. Figure 2 (a) demonstrates that our first choice is a simple MLP. The result is *MLP-DMD*. The architecture is two linear layers with an ELU [8]

activation on the hidden layer and a residual connection with expansion ration 1 (higher ratios are more prone to overfitting to the noise in the history of optimization that is used as training data to the neural network). MLP-DMD does not use mixing of information between simulation steps, as in our SW-DMD method. Thus, we also consider a 1D CNN encoder in Figure 2 (b) to form the *CNN-DMD* method. In CNN-DMD, the simulation steps form the temporal dimension, and the parameters form the channel dimension of the CNN. We use causal masking of the CNN kernels on the encoder in order to avoid look-ahead bias. During the inference, we look at the history of simulated steps, and look at the prediction of the last step. Then we recurrently feed the last step and resume the predictions. The architecture is two 1D-CNN layers with bottleneck middle channel number of 1 (to avoid overfitting) and ELU activation in between. Because DMD is a special case of SW-DMD, we also naturally define an *MLP-SW-DMD* method, *i.e.*, we simply consider our MLP-DMD method for input obtained by window size larger than 1. We build our neural networks with Pytorch [35]. See Appendix C for additional details.

5 Experiments

5.1 Quantum Optimization

For quantum optimization, we perform experiments for the 1D Ising model with the periodic boundary condition and the transverse field $h = 0.8$ to minimize the energy and find the ground state wave function. We perform simulations of VQE using Qiskit [2], a python framework for quantum computation. The experimental details are in Appendices A and B.

5.1.1 Noiseless Quantum Simulations

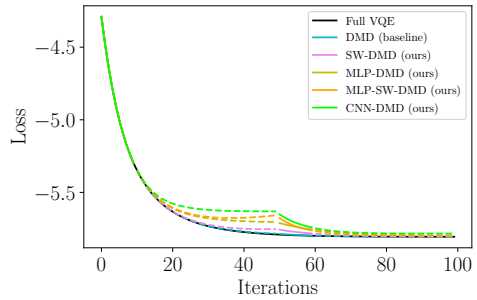
Noiseless quantum Natural gradient simulations. In Figure 3a, when using the natural gradient, we compare our DMD methods and the baseline standard DMD method on a 5-qubit Ising model with the circular-entanglement RealAmplitudes ansatz and reps=1 (2 layers, 10 parameters). Solid parts of the lines are from VQE, and dashed parts are from DMD. The learning rate is 0.01. We perform $n_{\text{total}} = 100$ full VQE iterations. Since with the natural gradient, $\boldsymbol{\theta}$ has the quantum wave function as a natural embedding with approximately linear dynamics, Koopman operator learning can be hypothesized to yield good prediction of dynamics. All the DMD algorithms are able to achieve the full VQE optimal loss. DMD and SW-DMD both match the full VQE. Our MLP-DMD, MLP-SW-DMD, and CNN-DMD begin to deviate from the full VQE loss when the DMD iterations increase. This could be related to over-fitting of the neural networks for simple parameters dynamics. In summary, all DMD methods can be used to accelerate the optimization.

Noiseless Adam simulations. Instead of the natural gradient, we show another case using the Adam optimizer based on the vanilla gradient obtained from the parameter-shift rule. We perform experiments on the 10-qubit Ising model with the circular-entanglement RealAmplitudes ansatz and reps=1 (2 layers, 20 parameters). In Figure 3b, we use $n_{\text{total}} = 400$, $n_{\text{sim}} = 10$, $n_{\text{DMD}} = 90$. For SW-DMD and MLP-SW-DMD, the sliding window size is $n_{\text{SW}} = 6$. The Adam learning rate is 0.01. The increasing part of dashed parts is not a problem, because we always start follow-up VQE from the best DMD prediction. While DMD has the worst performance as its lowest loss is still far from the full VQE final loss, whereas all our other methods can optimize the loss efficiently.

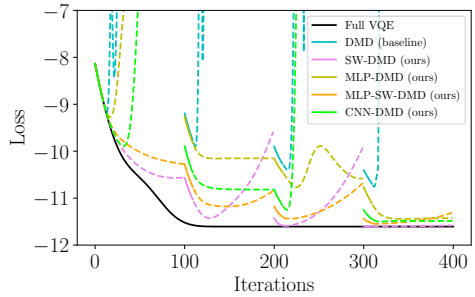
5.1.2 Quantum Optimization on a Real Quantum Computer

We implement VQE for the 3-qubit Ising model at $h = 0.8$ on a real IBM quantum computer Lima using the SPSA optimizer (learning rate 0.04 and perturbation 0.1) and 10k quantum shots. We have applied measurement error mitigation to reduce the effect of noise on the real hardware. The ansatz is RealAmplitudes with 2 layers (6 parameters) with linear entanglement.

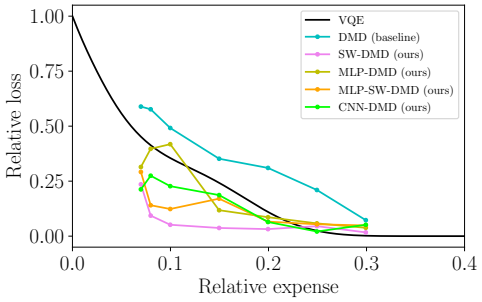
With the SPSA optimizer, every iteration of update contains 1 center measurement and 2 gradient measurements in only one random direction in the parameter space, so each iteration takes 3 measurements. The smaller number of gradient measurements makes it easier to implement on near-term real



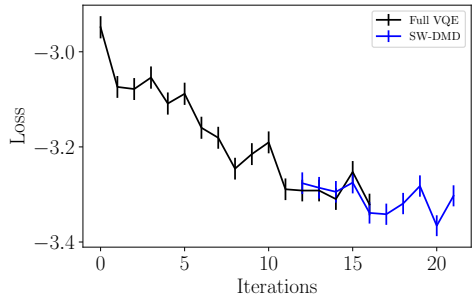
(a) Natural gradient 5-qubit results with $n_{\text{sim}} = 10$, $n_{\text{DMD}} = 40$, and $n_{\text{SW}} = 6$ for SW-DMD and MLP-SW-DMD. For the various DMD methods, solid parts are VQE runs, and the dashed parts are DMD predictions.



(b) Adam 10-qubit results with $n_{\text{sim}} = 10$, $n_{\text{DMD}} = 90$, $n_{\text{SW}} = 6$ for SW-DMD and MLP-SW-DMD. For the various DMD methods, solid parts are VQE runs, and the dashed parts are DMD predictions.



(c) Adam 10-qubit results for relative loss versus relative cost with $n_{\text{sim}} = 7, 8, 10, 15, 20, 25, 30$, $n_{\text{DMD}} = 90$, and $n_{\text{SW}} = 6$ for SW-DMD and MLP-SW-DMD.



(d) Experimental results from the real quantum computer IBM Lima.

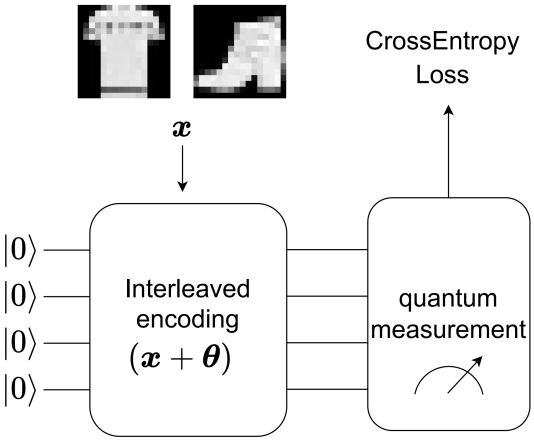
Figure 3: Experimental results.

quantum computers, compared to Adam. On the other hand, using SPSA may also require a higher number of iterations. DMD always only takes 1 center measurement without gradient measurements.

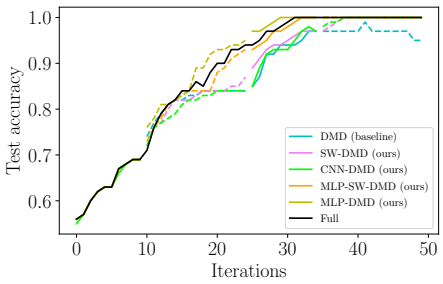
As shown in Figure 3d, on IBM Lima, we first perform a full pure VQE for 17 iterations. To mitigate the effect of the random direction in gradient measurements, we use the first 12 VQE iterations for Koopman training and apply SW-DMD with $n_{\text{SW}} = 8$ to make prediction for 10 iterations. Early in prediction, SW-DMD agrees with the full VQE result, and then further decreases the loss. The total expense of the first 12 iterations of VQE and 10 iterations of SW-DMD is still smaller than 17 iterations of full VQE. It is a promising validation of our approach that SW-DMD can help reduce the quantum resource needed in quantum optimization on a real quantum computer. Note that IBM Lima can have a relatively big uncertainty over different experimental times, and the final difference between the full VQE and SW-DMD could be within the experimental uncertainty in quantum lab.

5.2 Quantum Machine Learning

We use an interleaved block-encoding scheme [26, 41] for QML with its quantum neural network architecture shown in Figure 4a and apply Koopman operator learning. We consider the task of binary classification on a filtered Fashion-MNIST dataset with samples labeled by "T-shirt" and "ankel boot" and implement it in Yao [31], a framework for quantum algorithms in Julia [4]. Each is downsampled to 16×16 pixels, and is then as an input $\boldsymbol{x} \in [0, 1]^{256}$ fed into interleaved encoding quantum gate. The parameters $\boldsymbol{\theta}$ are also encoded in the interleaved encoding quantum gate. Then the quantum measurements are used as the output for computing the cross-entropy loss. The details of the QML



(a) Quantum machine learning architecture.



(b) Test accuracy of binary classification for full QML simulation and DMD methods. For the various DMD methods, solid parts are QML runs, and the dashed parts are DMD predictions.

Figure 4: Quantum machine learning architecture and results.

data and architecture are in Appendix F. The full QML training has $n_{\text{total}} = 50$ iterations. We choose $n_{\text{sim}} = 10$, $n_{\text{DMD}} = 15$, and $n_{\text{SW}} = 6$ for SW-DMD and MLP-SW-DMD. To reduce the number of parameters in the neural networks used in DMD including MLP-DMD, MLP-SW-DMD, CNN-DMD, we adopt a layer-wise partition in $\boldsymbol{\theta}$ with details in Appendix F. Figure 4b shows that DMD methods can achieve good accuracy while saving quantum resources compared to full QML.

5.3 Ablation

We show an ablation analysis for all DMD methods associated by varying n_{sim}, with $n_{\text{DMD}} = 90$, and $n_{\text{SW}} = 6$ for SW-DMD and MLP-SW-DMD. We define the relative expense spent on the quantum computer measured by the relative number of VQE iterations compared to the full VQE. n_{DMD} is chosen large enough so that it does not affect the minimum loss much. The relative loss is defined as $(L_{\text{min, VQE + DMD}} - L_{\text{min, full VQE}})/(L_{\text{initial, full VQE}} - L_{\text{min, full VQE}})$. A smaller relative loss means better performance. At the same relative loss, a smaller relative expense implies more acceleration of quantum optimization (*e.g.*, a relative expense at 0.1 indicates 10 times speedup). Figure 3c shows the relative loss versus the relative expense. The performance of DMD methods should be compared to pure VQE (black line). When the relative expense is about 30% (about 3 times speedup), all of the DMD methods yield good performance within a tolerance of 10% relative loss, but this is also comparable to the relative loss achieved by the pure VQE. DMD algorithms start to have better performance than pure VQE only when we further decrease the relative expense by decreasing n_{sim}. The baseline DMD does not have advantage over pure VQE, but all our DMD methods perform better than pure VQE and significantly save the used quantum resources when the relative expense is small.

6 Conclusion

Quantum optimization and quantum machine learning are promising for solving certain problems beyond the capability of classical computation. However, in the NISQ era, gradient-based optimization on quantum computers is costly since it takes $O(n_{\text{params}})$ gradient measurements for each iteration while the current quantum computing resources are limited. Built on the insight from Koopman operator theory, we have developed data-driven sliding-window and neural-network based DMD for accelerating quantum optimization and quantum machine learning. We demonstrated the effectiveness of our methods on both simulations and real quantum computers. Our work bridges the fields of machine learning and quantum optimization and opens up opportunities for further exploration of optimization problems through Koopman operator theory.

References

[1] S.-I. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[2] M. S. ANIS, Abby-Mitchell, H. Abrahamand, et al. Qiskit: An open-source framework for quantum computing, 2021.

[3] P. J. Baddoo, B. Herrmann, B. J. McKeon, and S. L. Brunton. Kernel learning for robust dynamic mode decomposition: linear and nonlinear disambiguation optimization. *Proceedings of the Royal Society A*, 478(2260):20210830, 2022.

[4] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[5] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[6] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz. Modern koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.

[7] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

[8] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[9] A. S. Dogra and W. Redman. Optimizing neural networks via koopman operator theory. *Advances in Neural Information Processing Systems*, 33:2087–2097, 2020.

[10] D. Dong, C. Chen, H. Li, and T.-J. Tarn. Quantum reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(5):1207–1220, 2008.

[11] D. Dylewsky, E. Kaiser, S. L. Brunton, and J. N. Kutz. Principal component trajectories for modeling spectrally continuous dynamics as forced linear systems. *Physical Review E*, 105(1):015312, 2022.

[12] S. Ebadi, A. Keesling, M. Cain, T. T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar, X.-Z. Luo, B. Nash, X. Gao, B. Barak, E. Farhi, S. Sachdev, N. Gemelke, L. Zhou, S. Choi, H. Pichler, S.-T. Wang, M. Greiner, V. Vuletić , and M. D. Lukin. Quantum optimization of maximum independent set using rydberg atom arrays. *Science*, 376(6598):1209–1215, jun 2022.

[13] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[14] D. Giannakis, A. Ourmazd, P. Pfeffer, J. Schumacher, and J. Slawinska. Embedding classical dynamics in a quantum computer. *Phys. Rev. A*, 105:052404, May 2022.

[15] A. Goldschmidt, E. Kaiser, J. L. DuBois, S. L. Brunton, and J. N. Kutz. Bilinear dynamic mode decomposition for quantum control. *New Journal of Physics*, 23(3):033035, mar 2021.

[16] L. Hackl, T. Guaita, T. Shi, J. Haegeman, E. Demler, and I. Cirac. Geometry of variational methods: dynamics of closed quantum systems. *SciPost Physics*, 9(4), oct 2020.

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

[18] H.-Y. Huang, M. Broughton, J. Cotler, S. Chen, J. Li, M. Mohseni, H. Neven, R. Babbush, R. Kueng, J. Preskill, and J. R. McClean. Quantum advantage in learning from experiments. *Science*, 376(6598):1182–1186, jun 2022.

[19] H.-Y. Huang, R. Kueng, G. Torlai, V. V. Albert, and J. Preskill. Provably efficient machine learning for quantum many-body problems. *Science*, 377(6613), sep 2022.

[20] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash. q-means: A quantum algorithm for unsupervised machine learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] N. Klco, E. F. Dumitrescu, A. J. McCaskey, T. D. Morris, R. C. Pooser, M. Sanz, E. Solano, P. Lougovski, and M. J. Savage. Quantum-classical computation of schwinger model dynamics using quantum computers. *Phys. Rev. A*, 98:032331, Sep 2018.

[23] S. Klus, F. Nüske, and S. Peitz. Koopman analysis of quantum systems$*$. $Journal of Physics A : Mathematical and Theoretical, 55(31) : 314002, jul 2022$.

[24] B. O. Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[25] Q. Li, F. Dietrich, E. M. Bollt, and I. G. Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10):103111, 2017.

[26] W. Li, Z. Lu, and D.-L. Deng. Quantum Neural Network Classifiers: A Tutorial. *SciPost Phys. Lect. Notes*, 61, 2022.

[27] Y. Li, H. He, J. Wu, D. Katabi, and A. Torralba. Learning compositional koopman operators for model-based control. *arXiv preprint arXiv:1910.08264*, 2019.

[28] Y. T. Lin, R. B. Lowrie, D. Aslangil, Y. Subaşı, and A. T. Sornborger. Koopman von neumann mechanics and the koopman representation: A perspective on solving nonlinear dynamical systems with quantum computers, 2022.

[29] Y. Liu, S. Arunachalam, and K. Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17(9):1013–1017, jul 2021.

[30] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[31] X.-Z. Luo, J.-G. Liu, P. Zhang, and L. Wang. Yao.jl: Extensible, Efficient Framework for Quantum Algorithm Design. *Quantum*, 4:341, 2020.

[32] B. Lusch, J. N. Kutz, and S. L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1), nov 2018.

[33] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Phys. Rev. A*, 98:032309, Sep 2018.

[34] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3(3):030503, jun 2018.

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[36] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1), jul 2014.

[37] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):1–7, 2014.

[38] M. J. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*, pages 51–67. Springer, 1994.

[39] J. Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, Aug. 2018.

[40] W. T. Redman, M. Fonoberova, R. Mohr, Y. Kevrekidis, and I. Mezic. An operator theoretic view on pruning deep neural networks. In *International Conference on Learning Representations*, 2021.

[41] W. Ren, W. Li, S. Xu, K. Wang, W. Jiang, F. Jin, X. Zhu, J. Chen, Z. Song, P. Zhang, H. Dong, X. Zhang, J. Deng, Y. Gao, C. Zhang, Y. Wu, B. Zhang, Q. Guo, H. Li, Z. Wang, J. Biamonte, C. Song, D.-L. Deng, and H. Wang. Experimental quantum adversarial learning with programmable superconducting qubits. 2022.

[42] E. Rinaldi, X. Han, M. Hassan, Y. Feng, F. Nori, M. McGuigan, and M. Hanada. Matrix-model simulations using quantum computing, deep learning, and lattice monte carlo. *PRX Quantum*, 3:010324, Feb 2022.

[43] M. S. Rudolph, N. B. Toussaint, A. Katabarwa, S. Johri, B. Peropadre, and A. Perdomo-Ortiz. Generation of high-resolution handwritten digits with an ion-trap quantum computer. *Phys. Rev. X*, 12:031010, Jul 2022.

[44] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[45] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran. Evaluating analytic gradients on quantum hardware. *Phys. Rev. A*, 99:032331, Mar 2019.

[46] A. Sinha, M. Sarkar, A. Mukherjee, and B. Krishnamurthy. Introspection: Accelerating neural network training by learning weight evolution. *arXiv preprint arXiv:1704.04959*, 2017.

[47] J. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

[48] J. Stokes, J. Izaac, N. Killoran, and G. Carleo. Quantum Natural Gradient. *Quantum*, 4:269, May 2020.

[49] M. E. Tano, G. D. Portwood, and J. C. Ragusa. Accelerating training in artificial neural networks with dynamic mode decomposition, 2020.

[50] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth, et al. The variational quantum eigensolver: a review of methods and best practices. *Physics Reports*, 986:1–128, 2022.

[51] J. v. Neumann. Zur operatorenmethode in der klassischen mechanik. *Annals of Mathematics*, 33(3):587–642, 1932.

[52] D. Wecker, M. B. Hastings, and M. Troyer. Progress towards practical quantum variational algorithms. *Physical Review A*, 92(4), oct 2015.

Supplementary Materials

A Setup of VQE+DMD runs

In quantum optimization of the 1D Ising model, we run VQE for n_{sim} iterations starting from a random initial point in the parameter space. Then, we apply DMD methods to the history of parameters from VQE, predict the future trajectory for n_{DMD} iterations, and evaluate the loss on the predicted trajectory. From the n_{DMD} evaluations, we find the minimum loss and the corresponding point in the parameter space, and use this optimal point as the starting point of the next n_{sim} iterations of VQE. We then repeat the alternating runs of VQE and DMD. n_{sim} and n_{DMD} are hyperparameters in our algorithm. In SW-DMD, there is an additional hyperparameter, the sliding window size n_{SW}, with the constraint $n_{\mathrm{SW}} < n_{\mathrm{sim}}$. Note that DMD is a special case of SW-DMD with $n_{\mathrm{SW}} = 1$. We benchmark the baseline DMD, and our SW, MLP, MLP-SW and CNN DMDs with $n_{\mathrm{sim}} = 10$. We choose the sliding window size $n_{\mathrm{SW}} = 6$ for SW-DMD and MLP-SW-DMD. In parallel, we also perform a pure VQE run for n_{total} iterations, starting from the same initial point as the alternating VQE+DMD runs.

B Quantum Ising Model with Transverse Field

The 1D quantum Ising model with the periodic boundary condition and transverse field h is the Hamiltonian

$$\mathcal{H} = -\sum_{i=1}^{N} Z_i \otimes Z_{i+1} - h \sum_{i=1}^{N} X_i, \tag{10}$$

where

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{11}$$

are Pauli matrices, and the subscripts on them denote which qubit they act on. The periodic boundary condition defines the qubits at $i = 1$ and $i = N + 1$ to be identical.

C Further details on neural DMD

Optimization hyperparameter The neural network is trained from scratch by using Adam for 30k steps with 9k steps of linear warmup from 0 to 0.001 and then cosine decay back to 0 at step 30k.

Activation Besides ELU, we also explored cosine, ReLU and tanh activations. We found ELU to be the best which could be because: tanh suffers from vanishing gradients, ReLU biases to positive numbers (while input phases quantum circuit parameters $\boldsymbol{\theta}$ are unconstrained) and cosine is periodic.

The importance of learning rate scheduler. We train the neural network $\mathbf{\Phi}_\alpha$ from scratch every time we obtain parameters optimization history as training data. A stable learning with well converging neural network at every single Koopman operator fitting stage is important. It is found vital to use a cosine-decay scheduler with a linear warmup, which is typically useful in the computer vision literature [30]. We linearly scale the learning rate from 0 to 0.001 for the first 9k steps of the neural network optimization, and then use a cosine-decay from 0.001 to 0 until the final step at 30k.

The importance of residual connections. In our work we use a residual connection so that DMD becomes as a special case of the neural DMD parameterization. The residual connection is indeed very useful, as it is driven by the Koopman operator learning formulation and makes it possible for the encoder to learn the identity. For identity encoder, MLP(-SW)-DMD or CNN-DMD become vanilla (SW-)DMD.

D RealAmplitudes Ansatz

The RealAmplitudes ansatz from Qiskit always produces a real-valued wave function $\psi_{\boldsymbol{\theta}}$ with no imaginary part. The minimum loss of the Ising-model can always be achieved by a real-valued $\psi_{\boldsymbol{\theta}}$, so this ansatz can help to reduce the redundancy in the functional form of $\psi_{\boldsymbol{\theta}}$ and therefore, is beneficial for our use.

With the RealAmplitudes ansatz, $\psi_{\boldsymbol{\theta}}$ as a function of $\boldsymbol{\theta}$ is a nonlinear function consisting of alternating layers of the rotational gates for the Pauli matrix

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{12}$$

with rotation angles being the parameters $\boldsymbol{\theta}$, and controlled-X gates with no parameter. Each layer of the rotational gates for Y has N (the number of qubits) parameters, so N times the number of rotational layers gives the total number of parameters.

E Comparison between Real Lima and FakeLima

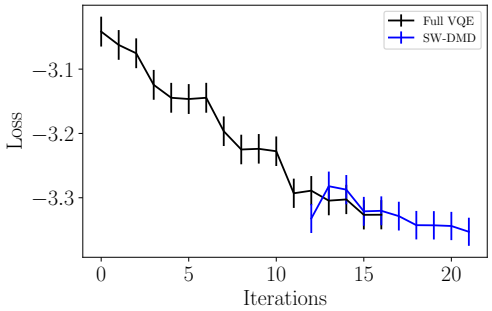


Figure A1: FakeLima results.

We perform simulation using FakeLima provided in Qiskit, a noise model that mimics the real hardware Lima, on the same 3-qubit Ising model at $h = 0.8$ as we do on the real Lima. We use the same setup as the setup on the real Lima with $n_{\text{total}} = 17$, $n_{\text{sim}} = 12$, $n_{\text{SW}} = 8$ with SW-DMD. The FakeLima results are shown in Figure A1, and are qualitatively consistent with Figure 3d with less fluctuation in the loss evaluation.

F QML Data and Architecture Details

In our QML example, the quantum computer has $N = 10$ qubits. The interleaved encoding gate consists of 9 layers. Each layer has a rotational layer and a linear entanglement layer. On the linear entanglement layer, each qubit has three rotational gates for X, Z, X in a sequence. Therefore, each layer has 30 rotational angles, and the whole QML architecture has 270 rotational angles. Since each input example $\boldsymbol{x}$ is 256-dimensional, we only use the first 256 rotational angles to encode the input data. The parameters $\boldsymbol{\theta}$ are also encoded in the rotational angles such that the angles are $\boldsymbol{x} + \boldsymbol{\theta}$. All the examples share the same $\boldsymbol{\theta}$.

In each quantum measurement, each qubit is in the 0-state or the 1-state, and the probability for the qubit to be in the 0-state is between 0 and 1. We regard this probability as the probablity for the image to be a "T-shirt". We only use the probability on the 5th qubit ($i = 5$) as the output and compute the cross-entropy loss with the labels.

We use 500 training examples and 100 test examples. During training, we use the stochastic gradient descent optimizer with the batch size 50 and learning rate 0.1.

In neural DMD including MLP-DMD, MLP-SW-DMD, CNN-DMD, we group $\boldsymbol{\theta}$ by layers in the encoding gate. There are 9 groups with group size 30. We perform neural DMD for each group, so that the number of parameters in the neural networks for DMD is not too large. After predicting each group separately, we combine all groups of $\boldsymbol{\theta}$ to evaluate the loss and accuracy.