Training a Quantum PointNet with Nesterov Accelerated Gradient Estimation by Projection

Ruoxi Shi **Hao Tang**

Xian-Min Jin
Center for Integrated Quantum Information Technologies (IQIT)
Shanghai Jiao Tong University
Shanghai, 200240, China
{eliphat, htang2015, xianmin.jin}@sjtu.edu.cn

Abstract

PointNet is a bedrock for deep learning methods on point clouds for 3D machine vision. However, the pointwise operations involved in PointNet is resource intensive, and the expressiveness of PointNet is limited by the size of its feature space. In this paper, we propose Quantum PointNet with a rectifed max pooling operation to achieve an exponential speedup performing the pointwise operations and meanwhile obtaining a quantum-enhanced feature space. We provide an implementation with quantum tensor networks and specify a circuit model that runs on near-term quantum computers. Meanwhile, we develop the NA-GEP (Nesterov Accelerated Gradient Estimation by Projection) optimization framework, together with a periodic batching scheme, to help train large-scale quantum networks more efficiently. We demonstrate that Quantum PointNet reaches competitive performance to its classical counterpart on a subset of the ModelNet40 dataset with 48x fewer operations required to process a point cloud. It is also shown that NA-GEP is robust under different kinds of noises. A mini Quantum PointNet is able to run on real quantum computers, achieving $\sim$100% accuracy classifying three kinds of shapes with a small number of shots.

1 Introduction

We live in a 3-dimensional world. Machines, especially robots, need to interpret 3-dimensional surroundings precisely to interact rationally with their environment. Thus, understanding 3D information is of great importance. There had already been a bunch of hand-crafted approaches to problems in 3D machine vision in the early ages of artificial intelligence (see [1, 2]). Deep-learning-based algorithms that handle different representations are developing rapidly [3–7]. They give impressive results on a variety of 3D machine vision problems.

However, the fact that the volume grows at a fast rate of $O(x^3)$ with the resolution x limits the performance of many deep learning methods based on volumetric representations such as voxels. In 2017, Qi et al. proposed an architecture, PointNet [6], which exploits the power of the symmetric operation, max pooling, to handle directly point clouds which were previously considered as an irregular data format hard to process. The PointNet classification network takes a point set as input, applies pointwise feature transformations with multi-layer perceptrons, and then aggregates point features by max pooling. Another multi-layer perceptron is employed to generate final classification scores. It reduces unnecessary volumious latents significantly.

First Workshop on Quantum Tensor Networks in Machine Learning, 34th Conference on Neural Information Processing Systems (NeurIPS 2020).

Unfortunately, since there is a large quantity of points in point clouds, the pointwise operations are resource intensive. Moreover, the architecture creates translated replicas of features to recognize the same object in different orientations, causing an exponential growth in number of features required with SO(3) (such as rotations). As a result, [6] has pointed out that the expressiveness of PointNet is sensitive to the size of the max pooling feature space. Overall performance drop is seen if the feature space is not large enough.

For this reason, we consider exploring the power of quantum to implement PointNet with a richer feature space and at a lower computation cost.

Here we interpret the pointwise operations in PointNet as generating implicit field values. In 3D machine vision, a field is an union of connected regions in the space, typically represented explicitly by meshes and point clouds. An implicit field [8] is defined by a continuous function over the space, with different values inside and outside the field area. A set of disjoint implicit fields form a non-regular grid-like structure. In this sense, max pooling is invariant not only to permutations in the input point cloud, but also to the local density of points, which gives it an advantage to other symmetric operations such as averaging. Consequently, the element setting for PointNet is quite flexible, as long as the model keeps the capability of generating such fields and uses a max-out aggregation operation.

As the size of max pooling feature space is the main issue of the expressiveness of PointNet, we propose to use a quantum implicit field learner, which can provide an exponential speedup, since quantum circuits naturally have a exponentially growing feature space. Meanwhile, the generated feature space is enhanced with quantum features. A novel rectified max pooling operation together with a regularization on sparsity is applied to add more expressiveness and robustness to the architecture. A comparison between pointwise pipelines of PointNet and Quantum PointNet is shown in Fig. 1

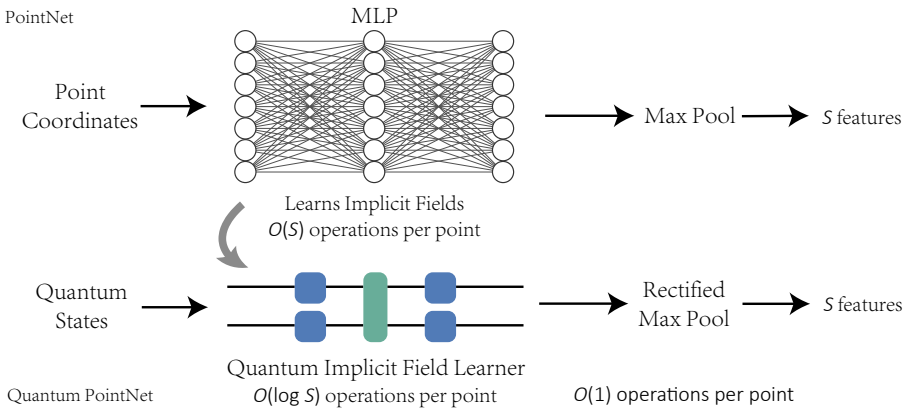

Figure 1: Comparison of pointwise pipelines between PointNet (upper) and Quantum PointNet (lower).

Meanwhile, we take a new approach towards training quantum networks, since this is a very general and severe challenge so far. We propose the Nesterov Accelerated Gradient Estimation by Projection (NA-GEP) optimization framework, which can adapt between a precise gradient evaluation and a rough estimation from pertubation to perform efficient optimization steps without the need of analytically evaluating the gradients. Moreover, a batching scheme to use in company with NA-GEP is introduced, which eliminates the need for iterating through the full training set at each step, and is essential for efficient training on large datasets.

Therefore, we combine the innovative quantum PointNet arcitecture and the improved optimization framework to achieve a highly practical alternative PointNet. It uses 48x fewer operations for processing a point cloud on a subset of the ModelNet40 dataset, and a mini Quantum Pointnet achieves $\sim$100% accuracy classifying three kinds of shapes using a real quantum computer. Our optimization method demonstrates robustness under various types of noises and shows good capability in training large-scale networks.

2 Related works

Quantum Machine Learning As quantum computing is entering an NISQ (Noisy Intermediate Scale Quantum) [9] era, many quantum-based methods are achieving supremacy over their classical counterparts, including quantum machine learning [10]. Even before real quantum computers are manufactured, there is already early implementations of QPCA (Quantum Principle Component Analysis) [11] and QSVM (Quantum Support Vector Machine) [12], suggesting a quantum speedup in the learning process. The concept of quantum deep learning was suggested in [13]. More proposals of quantum learning algorithms on real devices, such as Quantum versions of CNN (Convolutional Neural Networks) [14] and quantum enhanced kernel methods [15, 16] have been raised, showing that quantum methods can efficiently learn patterns that are considered difficult to recognize with classical models.

Quantum circuit optimization methods In the celebrated VQE (Variational Quantum Eigensolver) [17] hybrid optimization algorithm, gradient-free optimization methods such as the Nelder-Mead method [18] are suggested to apply. But the cost of these methods grow quickly to the number of parameters and thus do not perform well for machine learning tasks where typically thousands or millions of parameters are trainable. Heuristic searching methods have been used in a hybrid approach for training quantum circuits [19], but these methods do not have a guarantee of convergence on larger models. [20, 21] suggested using the parameter shift rule to evaluate the gradients of parametric quantum gates for optimization. But extra data evaluations scale linearly to the number of parameters and soon become intractable for large-scale networks, and the application of their method to arbitrary circuits rely on the decomposition of gates, which causes another overhead.

Quantum Tensor Networks Recently, Quantum Tensor Networks (see [22]) as a universal model for quantum computation is rapidly gaining attention, especially for the construction of machine learning architectures [23–27]. [1] By decomposing large tensors into a network of smaller ones, modular hardware designs and efficient computations are allowed, which are shown to be important in both machine learning and quantum mechanics. Theoretically, [28] proves that two types of quantum feature maps, namely the *parallel scenario* and the *sequential scenario*, have universal approximation properties, which means any continuous function from input Hilbert spaces to measurements can be approximated by deep enough quantum tensor networks. This sheds light upon innovation for more quantum-tensor-network-inspired machine learning protocols.

3 Methods

3.1 Quantum PointNet

The data pipeline of Quantum PointNet is illustrated in Fig. 2a. The architecture is composed of a feature map, a quantum implicit field learner (QIFL), a rectified max pooling operation, and a classifier.

The feature map converts the raw point coordinates in the point cloud to a quantum state vector. In order to put aside operations for a powerful QIFL, only single-qubit operations are used to form the feature map:

$$F(x, y, z) = Z \circ Rx(x)|0\rangle \otimes Z \circ Rx(y)|0\rangle \otimes Z \circ Rx(z)|0\rangle \otimes |00\ldots0\rangle, \tag{1}$$

where Z is Pauli Z flip and Rx is a parametrized rotation around x. Point coordinates are mapped into amplitudes of the quantum states in this way. Some other candidates can be found in [16, III. A.].

QIFL, the quantum implicit field learner, is able to generate implicit field function values from the input state vector. There are many possible implementations of this learner module. In our implementation (Fig. 2b) , the learner is decomposed into a tensor network with layers of parametric one-qubit operations and fixed entangler maps. More specifically, in the circuit model a fully parameterized U gate is used for the one-qubit operation, and CNOT gates are stacked to enable full entanglement between all qubits. (For more details see Appendix A)

[1] The widely-adopted circuit model for universal quantum computers is a special case in Quantum Tensor Networks.

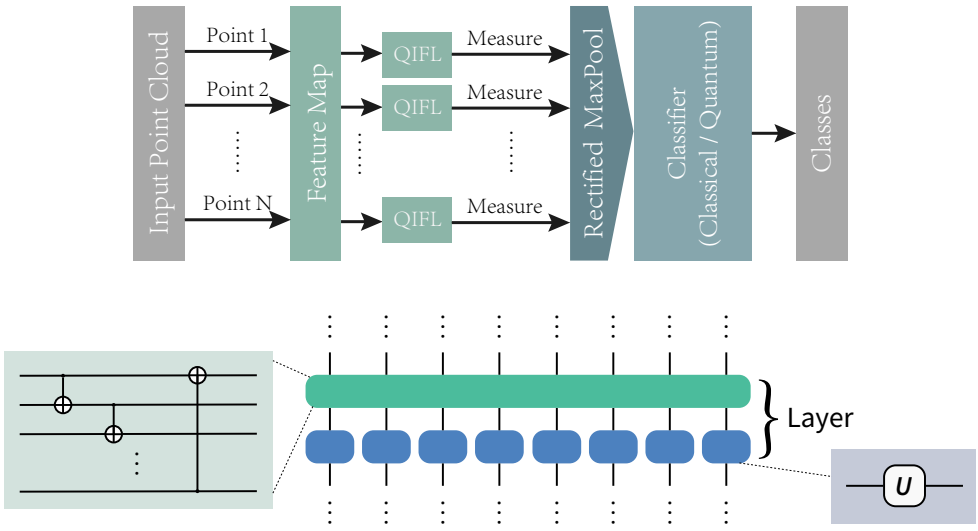

Figure 2: Architecture of our model. ***Upper:*** The classification pipeline of Quantum PointNet. The feature map and the quantum implicit field learner (QIFL) runs on a quantum computer, the rectified max pooling works on a classical computer, and the classifier can be either quantum or classical. ***Lower:*** The layered Quantum Tensor Network structure for QIFL and its circuit implementation.

The rectified max pooling operation is of key importance in Quantum PointNet. First, in order to mitigate noise present in current quantum computers, we impose sparsity on the generated values for each point by adding a regularization term to be maximized:

$$\mathcal{L}_{Sp} = \lambda \sum_i \texttt{std}(QIFL(|\psi_i\rangle)), \tag{2}$$

where $|\psi_i\rangle$ denotes the quantum state generated by the feature map and `std` is the standard deviation function. To provide the network back with capabilities of generating different amplitudes and meanwhile further reduce noise effects, a rectifying operation is applied after max pooling:

$$Rec(p) = \begin{cases} p^2 & p \leq Th \\ p & otherwise \end{cases}, \tag{3}$$

where Th defaults to 0.15. Note that this may cause gradients to vanish in the beginning as sparsity has not yet been learned. We suggest employing a huber function if $Th \leq 10/n$ where n is the size of the feature space, denoted as Rec_H:

$$Rec_H(p) = \begin{cases} p^2/Th & p \leq Th \\ p & otherwise \end{cases}, \tag{4}$$

and in other cases a normalized version of Rec, denoted as Rec_N:

$$Rec_N(p) = (Rec(p) - \texttt{mean}(Rec(p)))/\texttt{std}(Rec(p)). \tag{5}$$

Mean values and standard deviations are computed on the samples' basis.

Finally, the classifier generates the final results of classification from the max-pooled global implicit field values. The implementation is arbitrary, as long as linear combinations of features can be represented. A traditional softmax cross-entropy loss function is then applied to guide the training process.

3.2 Optimization method

In this section, we present the proposed NA-GEP (Nesterov Accelerated Gradient Estimation by Projection) optimization framework, and introduce a batching scheme for efficient training on large datasets. This enables us to train large-scale quantum networks, and thus we can apply quantum computing techniques more flexibly to deep learning.

3.2.1 NA-GEP

The starting point is a one-sided finite-difference estimation of the gradient, which is a direct consequence of projecting gradient onto a random vector $\Delta\vec{x}$:

$$\hat{\nabla} f(\vec{x})_i = \frac{f(\vec{x} + \Delta\vec{x}) - f(\vec{x})}{||\Delta\vec{x}||} \Delta\vec{x}_i. \tag{6}$$

If we repeat the operation above on a set of random but orthogonalized vectors, we will get a more accurate estimation of the gradient. In addition, we introduce Nesterov's Accelerated Gradient [29] into the procedure. Under this setting, NAG is able to average over the past few steps of estimated gradients to reduce variance of estimation. Furthermore, we found that the Nesterov's momentum term is much better a starting direction for the projection process in Eq. (6), as it combines effectively with NAG's lookahead property for a brake towards the direction of the momentum term. Meantime, it can contribute more in the descending rate since it is probably in a direction closer to the current gradient than a random vector.

Hyper-parameters are adapted on plateau to improve local convergence. For the detection of plateau, we use the *function scheme* proposed by [30] for a clean target f without stochastic properties or noises. For noisy fs we simply track the moving average and set a threshold.

A regularization of momentum clipping is employed which empirically may help improve generalization capabilities. It also helps NAG to brake more timely.

Combining all of these we arrive at a gradient-free optimization framework with good performance. The whole optimization process is shown in Algorithm 1. η is the momentum coefficient. P is the order of projection estimation. μ is finite difference delta. δ is the step size, or learning rate. c, e_δ, e_η are clipping and damping coefficients. See Appendix B for a detailed gradient projection routine.

Algorithm 1 NA-GEP optimization process

Input: f, θ_0; **Output**: θ
$\theta \leftarrow \theta_0, m \leftarrow \vec{0}$
while not converged **do**
  $G \leftarrow \texttt{grad\_proj}(f, \theta + \eta m, P, \mu, \eta m)$ ▷ NAG
  $m \leftarrow \eta m - \alpha G$
  $m \leftarrow \texttt{clip}(m, -c, c)$ ▷ Momentum clipping
  $\theta \leftarrow \theta + m$ ▷ Updating parameters
  Compute and record $f(\theta)$
  if on plateau **then**
    $\eta \leftarrow e_\eta \eta$, $P \leftarrow P + 1$, $\delta \leftarrow e_\alpha \alpha$
  end if
end while

3.2.2 Batching scheme

Traditional mini-batch gradient descent attains a set of random samples from the training set at each iteration. However, this works poorly with stochastic optimization methods.

In view of this problem, we introduce a batching scheme to apply together with NA-GAE: The *periodic* scheme. A mini-batch is generated and kept for several iterations before another mini-batch is sampled. The scheme aims at reducing the variance within the gradient to be estimated while keeping unbiased among the dataset to achieve higher descending rates. With a moderately large batch size the effect of this scheme is dramatic under NA-GAE.

4 Results

4.1 Point cloud classification

4.1.1 Classifying shapes on a quantum computer

This experiment is carried out on a mini-net that runs on a real quantum device available at the time. The QIFL is composed of 5 qubits and 2 layers. 3 types of point clouds, namely spheres, cubes and cylinders, are generated and a Quantum PointNet is trained to classify them. 1536 point clouds are generated for the training set and 512 point clouds are generated for the test set. After 80 epochs 100% accuracy is reached on the test set. The test is run on a device at 5-qubit IBMQ Valencia.

We further study the effect of noise on our model. Classification accuracy is tested for different numbers of shots with and without noise. For each case, extra random samples are generated and evaluated until the half Wilson Interval [31, 32] with 3σ confidence (99.87%) is shorter than 1%. The noise model of IBMQ device is obtained and simulated with Qiskit[33].

The results are presented in Fig. 3. It is shown that QIFL, though shallow, has a good capability of learning relatively complicated implicit fields (Fig. 3a). Moreover, only an extremely small number of shots (within 20, see Fig. 3b) is required to get good classification accuracies, even under the noise of currently available devices in the NISQ era.

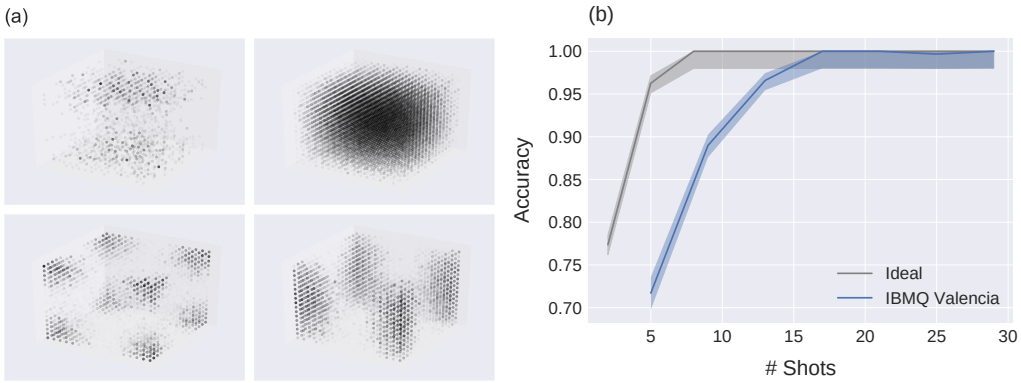

Figure 3: (a) Visualization of randomly chosen implicit fields learned by QIFL. 512 points distributed uniformly within the cube $[-1, 1]^3$ are sent to the trained QIFL to get activation values of the implicit fields. (b) Wilson's interval of 3 sigma confidence for classification accuracy to number of shots, on a simulated ideal device and under noise of 5-qubit IBMQ Valencia.

4.1.2 A real-world classification problem

To further illustrate the power of Quantum PointNet and our optimization framework, we train a Quantum PointNet with 8 qubits and 5 layers for a real-world task: to distinguish point clouds of planes, cars and vases. We randomly pick these three classes of point clouds from ModelNet40 [34] to form a subset, ModelNet3, for classification benchmarking. 256 points are uniformly sampled from each model. Under this setting, there are 1989 point clouds in the training set and 300 in the test set.

Quantum PointNet achieves a high accuracy of 99.0% on the test set, matching its classical counterpart, with way fewer operations required, as listed in Table 1. Parameters for the classical PointNet is adopted from [6] and scaled to a feature space size of 256, resulting in 4 pointwise dense layers of size 64, 64, 256, 256. Number of operations for Quantum PointNet is calculated with number of shots set to 20. A *classical operation* means an arithmetic operation between two numbers. It is an estimated value, and may change slightly with different implementations. (Refer to Appendix A for details about the circuit)

The training process is shown in Fig. 4. For both loss (Fig. 4a) and accuracies (Fig. 4b), it is shown that our optimization method enjoys a fast descending rate in the early stage of optimization, and still keeps steady towards the local convergence.

Table 1: Comparison between classical and quantum PointNets on ModelNet3.

	# Classical operations	Qubit operations	Test accuracy
PointNet	∼22M	/	99.0%
Quantum PointNet	∼59K	∼399K	99.0%

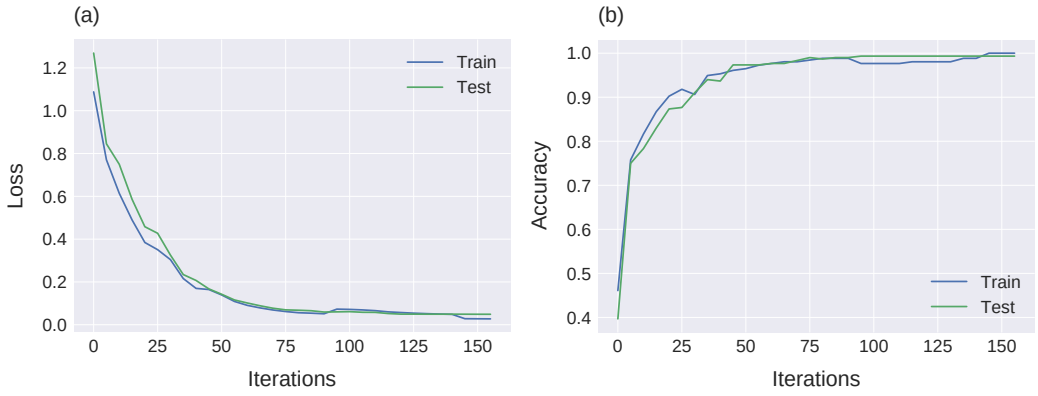

Figure 4: Convergence of (a) training and test loss and (b) overall accuracy during the training process of Quantum PointNet on ModelNet3, with our proposed optimization method.

4.2 Performance of NA-GEP

In this section, experiments on our optimization framework, NA-GEP, are carried out. A target function of the form

$$f(x) = ||\max(Ax, 0) \cdot Y||^2 \tag{7}$$

is generated and optimized under different settings. A and Y are matrices sampled from the Glorot Normal [35] distribution and x is a vector of $n = 512$ dimensions. In addition, our method is compared to another method that also does not require full evaluations of the gradient of target function, SPSA [36].

The results are shown in Fig. 5. With a reasonable choice of parameters (e.g., $P = 2, 5$), NA-GEP achieves better performance than SPSA (Fig. 5a). It is shown that NA-GEP is robust under both typical additive noise and a stronger noise where a random value may be returned from the target function each time it gets evaluated (Fig. 5b). Also, it can be seen that the projection method enjoys a rate much faster than the Monte-carlo search, and perfectly matches the ground-truth at full dimensions, 512. (Fig. 5c).

See Appendix C for a study on the batching schemes.

5 Discussion and outlook

NA-GEP, SGD and the parameter-shift rule We notice that SGD [37] (Stochastic Gradient Descent, with an optional NAG setup) algorithm is a special case under our optimization framework. It is the case where f involves a stochastic mini-batch sampling and P is set to full dimensions, n. The parameter shift rule [20, 21] focus on reducing the approximation error of discarding higher-order terms by utilizing properties of rotations on gates. The experiments show that this kind of error is small, without the need of tuning hyper-parameters heavily (see Fig. 5c, the estimated rate reaches the ground truth at 512 evaluations with almost no error). Nevertheless, utilizing these properties may further improve performance of NA-GEP, which may be explored in the future.

Extension to Quantum PointNet We now discuss some possible extensions to Quantum PointNet so that it can perform other 3D Machine Vision tasks. We consider merging global information with local information, and more specifically here, merging the implicit field values with the point

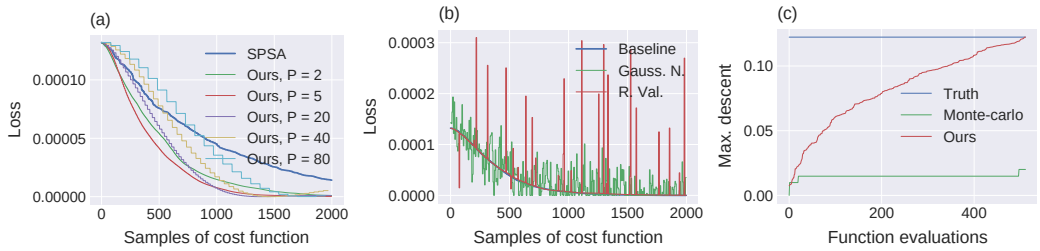

Figure 5: Results of NA-GEP on an optimization problem with 512 tunable parameters. (a) Effect of different P values on the optimization process is studied and compared with SPSA. $\eta = 0.95$. Other parameters are set to default values. (b) The robustness of our method to noise. NA-GEP with $P = 5$ and $\eta = 0.95$ is applied to minimize the generated f with and without noise. The values are those recorded by the algorithm. Two types of noises are studied: 1) Additive Gaussian noise. (Gauss. N.) 2) Random value noise where the observed value is replaced with a uniformly random value with 10% probability. (R. Val.) (c) Maximum rate of descent achieved with number of observations, compared with random sampling (Monte-carlo method) and the ground truth (the rate in the direction of gradient). The SPSA method has large variance in its performance estimating the rate of descent, and is thus not compared.

coordinates, which is helpful for segmentation and other tasks, and can further increase the capability of learning non-linear features.

The basic consideration is that there may not be too many classical operations in the implicit field learner for the exponential quantum speedup to take place, which in general limits the complexity of the merging process. Hopefully, a Quantum Convolutional Neural Network [14] combined with Autoencoder [38] technologies can be employed to compress the information into a few classical values, after which the feature mapping process can be applied without much computational penalty.

In all, there is still much more to explore in the field of quantum speedups or quantum-enhanced methods for 3D Machine Vision.

Deep learning on quantum circuits in the NISQ era Our work, together with several previous works [14–16], shows that quantum models have interesting capabities of pattern recognition under some specific settings. More generally speaking, computational power is the basis of all deep learning tasks, while universal approximation theories predict the flexiblility of building blocks in deep learning models. Thus the exponential speedup provided by quantum circuits can be fully utilized in this area, bringing possible merits due to the extra computational power brought by quantum computing.

The proposed rectifying operation with a regularization on sparsity can be extended to a range of other models that exhibit similar properties. The requirement for fewer shots and the stronger resistance towards noise are appealing as they lead to practical use of quantum circuit-based computers in the NISQ era for deep learning.

Broader Impact

We would like to state that this work not only benefits the field of 3D machine vision, but also, broadly speaking, stimulates the multidisciplinary research for the emerging quantum machine learning science and techniques. Our work with detailed explanation on the method and open access for relevant data is beneficial for creating equal learning opportunaties among all learners. On the other hand, it's also worth noting the good performance by quantum speedups is highly related to the hardware quality of quantum computers. This would cause a concern that the users with access to better quantum computer hardware would enjoy more benefit from this work.

References

[1] Mamun A. *Development of situation recognition, environment monitoring and patient condition monitoring service modules for hospital robots*. PhD thesis, Dublin City University, 2012.

[2] O' Mahony N., Campbell S., Krpalkova L. *et al.* Computer vision for 3d perception a review. In *Proceedings of the 2018 Intelligent Systems Conference (IntelliSys)*, pages 788–804, 2018.

[3] Maturana D. and Scherer S. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[4] Kanezaki A., Matsushita Y., and Nishida Y. Rotationnet: Joint learning of object classification and viewpoint estimation using unaligned 3d object dataset. *arXiv preprint*, arXiv:1603.06208, 2016.

[5] Kumawat S. and Raman S. Lp-3dcnn: Unveiling local phase in 3d convolutional neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4898–4907, 2019.

[6] Qi C.R., Su H., Mo K.C. *et al.* PointNet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[7] Zhang K., Hao M., Wang J. *et al.* Linked dynamic graph cnn: Learning on point cloud via linking hierarchical features. *arXiv preprint*, arXiv:1904.10014, 2019.

[8] Chen Z. Q. and Zhang H. Learning implicit fields for generative shape modeling. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5932–5941, 2019.

[9] Preskill J. Quantum computing in the nisq era and beyond. *Bulletin of the American Physical Society*, 2: 79, 2018.

[10] Biamonte J., Wittek P., Pancotti N. *et al.* Quantum machine learning. *Nature*, 549:195–202, 2017.

[11] Lloyd S., Mohseni M., and Rebentrost P. Quantum principal component analysis. *Nature Physics*, 10: 631–633, 2014.

[12] Anguita D., Ridella S., Rivieccio F. *et al.* Quantum optimization for training support vector machines. *Neural Networks*, 16:763–770, 2003.

[13] Wiebe N., Kapoor A., and Svore K. Quantum deep learning. *Quantum Information and Computation*, 16: 541–587, 2016.

[14] Cong I., Choi S., and Lukin M. D. Quantum convolutional neural networks. *Nature Physics*, 15:1273–1278, 2019.

[15] Havlícek V., Córcoles A., Temme K. *et al.* Supervised learning with quantum-enhanced feature spaces. *Nature*, 567:209–212, 2019.

[16] Schuld M. and Killoran N. Quantum machine learning in feature hilbert spaces. *Physical Review Letters*, 122:040504, 2019.

[17] Peruzzo A., McClean J., Shadbolt P. *et al.* A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5:4213, 2014.

[18] Nelder J. A. and Mead R. A simplex method for function minimization. *The computer journal*, 7:308–313, 1965.

[19] Benedetti M., Garcia-Pintos D., Perdomo O. *et al*. A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quantum Information*, 5:45, 2019.

[20] Mitarai K., Negoro M., Kitagawa M. *et al.* Quantum circuit learning. *Bulletin of the American Physical Society*, 2018, 2018.

[21] Schuld M., Bergholm V., Gogolin C. *et al.* Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99:032331, 2019.

[22] Biamonte J. and Bergholm V. Tensor networks in a nutshell. *arXiv preprint*, arXiv:1708.00006, 2017.

[23] Orús R. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1:538–550, 2019.

[24] Huggins W., Patil P., Mitchell B. *et al.* Towards quantum machine learning with tensor networks. *Quantum Science and Technology*, 4:024001, 2019.

[25] Clark S. R. Unifying neural-network quantum states and correlator product states via tensor networks. *Journal of Physics A: Mathematical and Theoretical*, 51:135301, 2018.

[26] Liu D., Ran S. J., Wittek P. *et al.* Machine learning by two-dimensional hierarchical tensor networks: A quantum information theoretic perspective on deep architectures. *arXiv preprint*, arXiv:1710.04833v2, 2018.

[27] Stoudenmire E. M. and Schwab D. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, page 6211, 2016.

[28] Goto T., Tran Q. T., and Nakajima K. Universal approximation property of quantum feature map. *arXiv preprint*, arXiv:2009.00298, 2020.

[29] Nesterov Y. E. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

[30] O'Donoghue B. and Candès E. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15:715–732, 2015.

[31] Edwin B. W. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22:209–212, 1927.

[32] Siegrist K. Estimation in the bernoulli model, 2019. URL `http://www.randomservices.org/random/interval/Bernoulli.html`.

[33] Abraham H., Agarwal R., Akhalwaya I. Y. *et al.* Qiskit: An open-source framework for quantum computing, 2019. URL `https://zenodo.org/record/2562111`.

[34] Wu Z.R., Song S.R., Khosla A. *et al.* A deep representation for volumetric shape modeling. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[35] Xavier G. and Yoshua B. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[36] Spall J. An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins Apl Technical Digest*, 19:482–492, 1998.

[37] Kiefer J. and Wolfowitz J. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23:462–466, 1952.

[38] Ballard D. H. Modular learning in neural networks. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, page 279, 1987.

[39] Summary of quantum operations. URL `https://qiskit.org/documentation/tutorials/circuits/3_summary_of_quantum_operations.html`.

[40] Sutskever I., Martens J., Dahl G. *et al.* On the importance of initialization and momentum in deep learning. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1139–1147, 2013.

[41] Lan G. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133: 365–397, 2012.

[42] Chen Y.X. Stochastic gradient methods, 2019. URL `http://www.princeton.edu/~yc5/ele522_optimization/lectures/stochastic_gradient.pdf`.

Appendix A. Implementation details of Quantum PointNet

Adaptive entangler map An advantage of composing the network with one-qubit gates and fixed entangler maps is that there is lower dependence on the expressiveness of entangler map. Any sequence and combination of CNOT gates can be employed, as long as full entanglement is fulfilled. This allows hardware-efficient implementations for free by adaptively selecting strongly connected qubits to apply CNOT on.

For the IBMQ Valencia device, the qubit connectivity topology (obtained with Qiskit [33]) is shown in Fig. A1a. Correspondingly, in the entangler map (Fig. A1b) CNOTs gates are applied according to the connectivity.

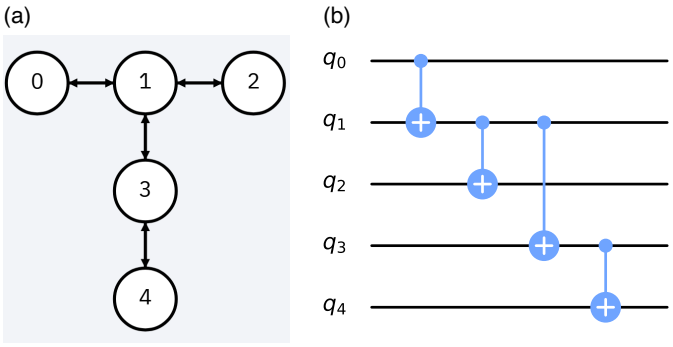

Figure A1: (a) Qubit connectivity topology of 5-qubit IBMQ Valencia. (b) The entangler map adapted to IBMQ Valencia device.

QIFL implementation for ModelNet3 In the circuit implementation of QIFL for the experiment of classification on ModelNet3, 4 layers composed of parametric U3 gates and an entangler map are stacked, with a final layer composed of parametric U1 gates. The structure of first 4 layers is shown in Fig. A2a (repeated 4 times). The entangler map is made of cycling CNOT gates (Fig. A2b). The final layer is shown in Fig. A2c. This results in a total 72 gate applications required for each point in the point cloud.

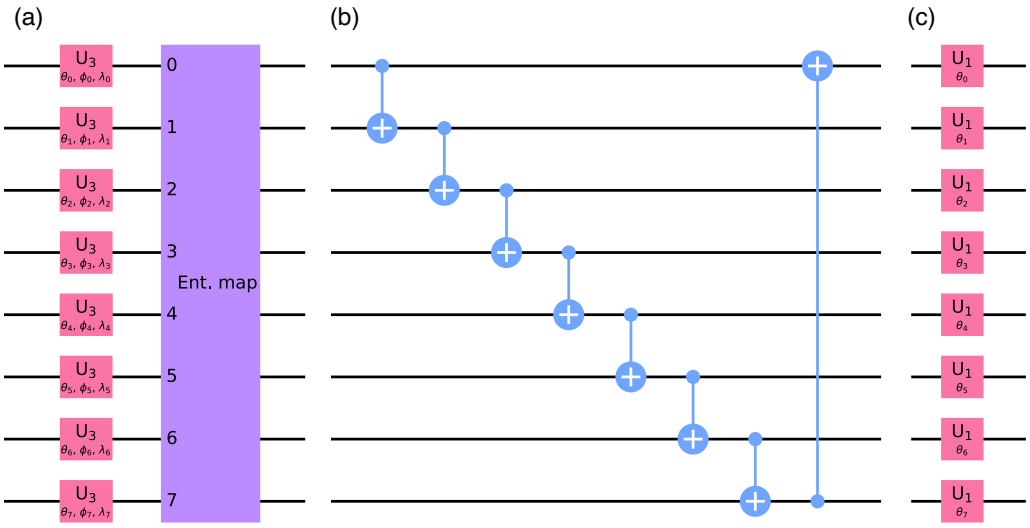

Figure A2: (a) Layer structure of 8-qubit QIFL composed of parametric U3 gates and an entangler map. (b) Fixed entangler map in each layer of 8-qubit QIFL. (c) The final layer composed of parametric U1 gates in 8-qubit QIFL. For detailed explanations about gates see [39].

Combining with the feature map which involves 2 gates on each of 3 coordinates, 78 qubit operations are required for each shot and for each point in the point cloud, which results in $78 \times 20 \times 256 = 399,360$ operations required to process each point cloud at 20 shots, under the experiment setting. Classical operations in Quantum Pointnet involve measurement, input conversion, rectified max pooling and the linear classifier. $11 \times 20 \times 256 + 256 \times 9 = 58,624$ classical operations are required in our implementation for each point cloud. Compared with $256 \times (3 \times 64 + 64 \times 64 + 64 \times 256 + 256 \times 256) = 22,069,248$ operations required for its classical counterpart, there is a significant reduction in the number of operations required. The difference will be even larger if more qubits are employed, since there is an exponential speedup from classical to quantum under the setting.

Also note that the mean value of the QIFL outputs for each point are fixed to $1/2^n$ where n is number of qubits, due to the normalization property of measurement probability. Thus the `std` function involved in Eq. (2) can be computed on a shots' basis, without the need of dealing with a dense vector of size 2^n classically.

Appendix B. Details and analysis on NA-GEP

Convergence of NA-GEP The optimization framework we proposed falls into the category of stochastic gradient optimization, which has a systematic convergence theory. [40] and [41] prove that the convergence rate of these methods on smooth convex functions is given by $O(L/T + \sigma/\sqrt{T})$, where L is a Lipshitz constant of ∇f, σ is the variance of gradient estimation and T is number of iterations of optimization. With NAG the convergence rate is improved to $O(L/T^2 + \sigma/\sqrt{T})$, which gives an advantage of our proposed method to SPSA in the early stages of the optimization when the term L/T dominates the rate.

In addition, hyper-parameters are adjusted on plateau, which forms an adaptive warm restart operation as defined in [30]. [42] shows that this could reach $O(\sigma/T)$ local convergence rate instead of $O(\sigma/\sqrt{T})$ for strongly convex functions.

Intuitively, the NAG reduces the variance of the estimated gradient by averaging, leading to fast learning in the early stage of training. When the optimization reaches a neighbourhood of the critical point, NAG will oscillate, preventing the algorithm from convergence. In this case momentum decay is increased and learning rate is reduced on plateau.

The Gradient Estimation by Projection procedure and relation to SPSA The procedure is shown in Algorithm 2. ϵ is a small number to prevent zero-length projection bases and is set to 0.001 in the experiments.

Note that if η is set to 0 and `random_vector` samples from $\{-1, 1\}^n$ the procedure is reduced to one-sided SPSA.

Algorithm 2 Procedure of Gradient Estimation by Projection

Input: f, θ, P, μ, β_0; **Output**: $\hat{G}$
$\hat{G} \leftarrow \vec{0}$
$\beta_0 \leftarrow \beta_0 + \texttt{random\_vector}() \cdot \epsilon$ ▷ Prevent zero-length β_0
for $i = 1$ to $P - 1$ **do**
  $\beta_i \leftarrow \texttt{random\_vector}()$
end for
$r \leftarrow f(\theta)$
for $i = 0$ to $P - 1$ **do**
  $\beta_i \leftarrow \texttt{gram\_schmidt}(\beta_i, [\beta_0, \beta_1, \ldots, \beta_{i-1}])$
  $\hat{G} \leftarrow \hat{G} + (f(\theta + \mu\beta_i) - r) \cdot \beta_i / \mu$ ▷ Projection
end for

Appendix C. Study on the batching schemes

In this section, experiments are carried out on the proposed batching scheme. We consider a least squares linear regression problem with 512 variables:

$$\min_{\vec{w}} \mathcal{L}(\vec{x}, y) = \min_{\vec{w}} \sum_i (\vec{w} \cdot \vec{x}_i - y_i)^2 \tag{C1}$$

SPSA and NA-GAE are applied to optimize the function on a dataset (training and test) randomly generated by computing values with a ground-truth $\vec{w}$ added by a gaussian noise. *Random* denotes the classical batching scheme. *Periodic* denotes the proposed approach. Under the *Iterative* scheme, for each iteration a sample within the mini-batch is replaced with a new random sample from the training set. The results are shown in Fig. C1.

The *Iterative* scheme may enjoy a higher rate of descent under some situations, but it suffers from stability issues and is not employed.

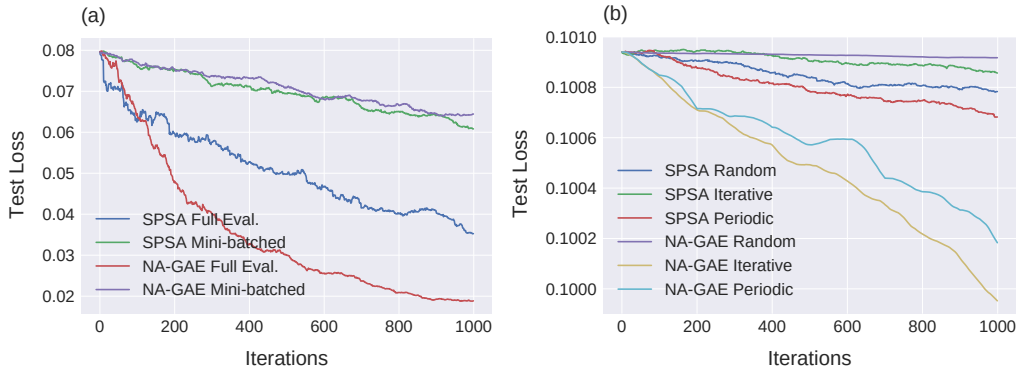

Figure C1: (a) Comparison of the optimization process based on full and mini-batched evaluations. (b) Comparison between different batching schemes for a fixed batch size of 128.