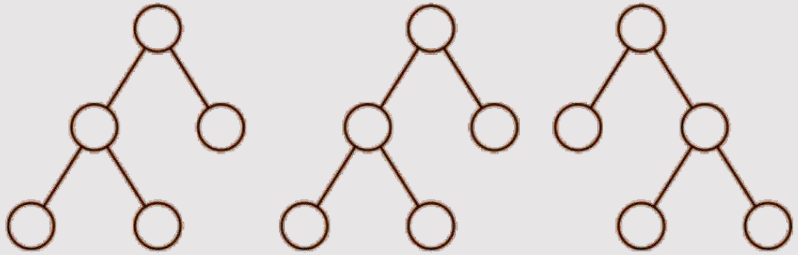
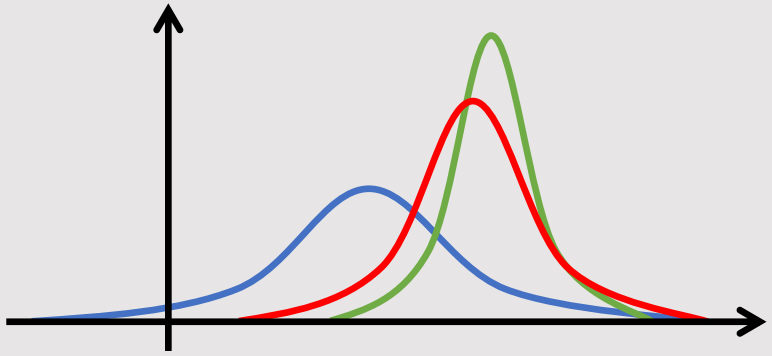
Machine learning for practical quantum error mitigation





Haoran Liao, Derek Wang, Iskandar Sitdikov, Ciro Salcedo, Alireza Seif, Zlatko Minev

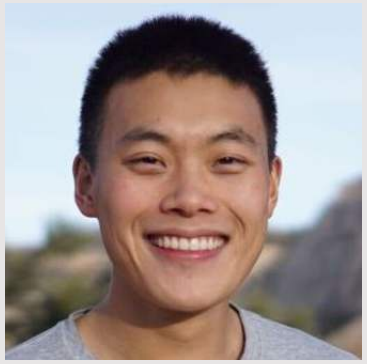
arXiv: 2309.17368

Qiskit Seminar

IBM **Quantum**

Acknowledgement

Derek Wang



Iskandar Sitdikov

Alireza Seif

Zlatko Minev

Ciro Salcedo

Brian Quanz, Patrick Rall, Oles Shtanko, Roland De Putter, Kunal Sharma, Barbara Jones, Sona Najafi, Thaddeus Pelligrini, Grace Harper, Vinay Tripathi, Antonio Mezzacapo, Christa Zoufal, Travis Scholten, Bryce Fuller, Swarnadeep Majumder, Sarah Sheldon, Youngseok Kim, Pedro Rivero, Will Bradbury, Nate Gruver, Minh Tran, Kristan Temme, and the broader IBM Quantum team

Quantum error mitigation (QEM)

QEM allows us to obtain good estimates of expectation values from noisy quantum circuits, even before the introduction of fault tolerance

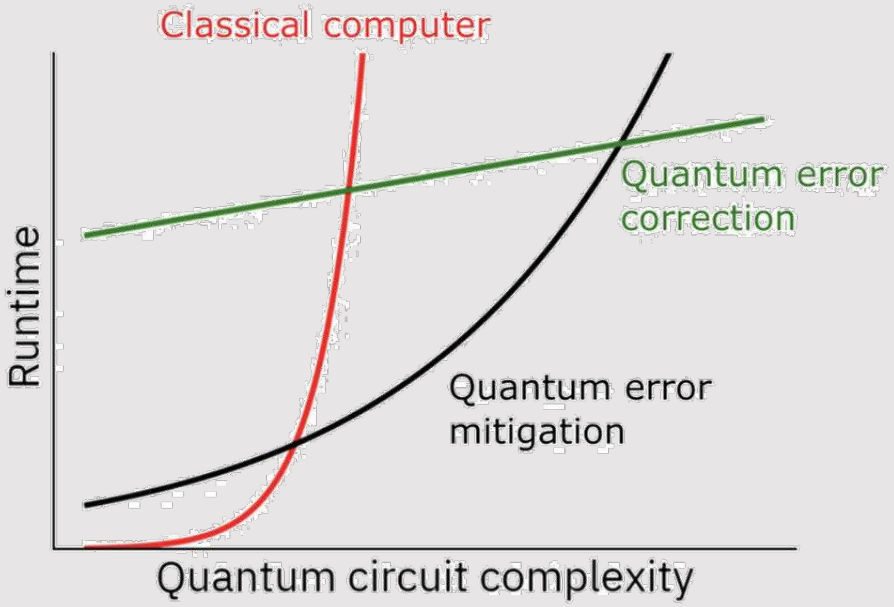


Image credit: IBM Research blog

One QEM approach is probabilistic error cancellation (PEC)

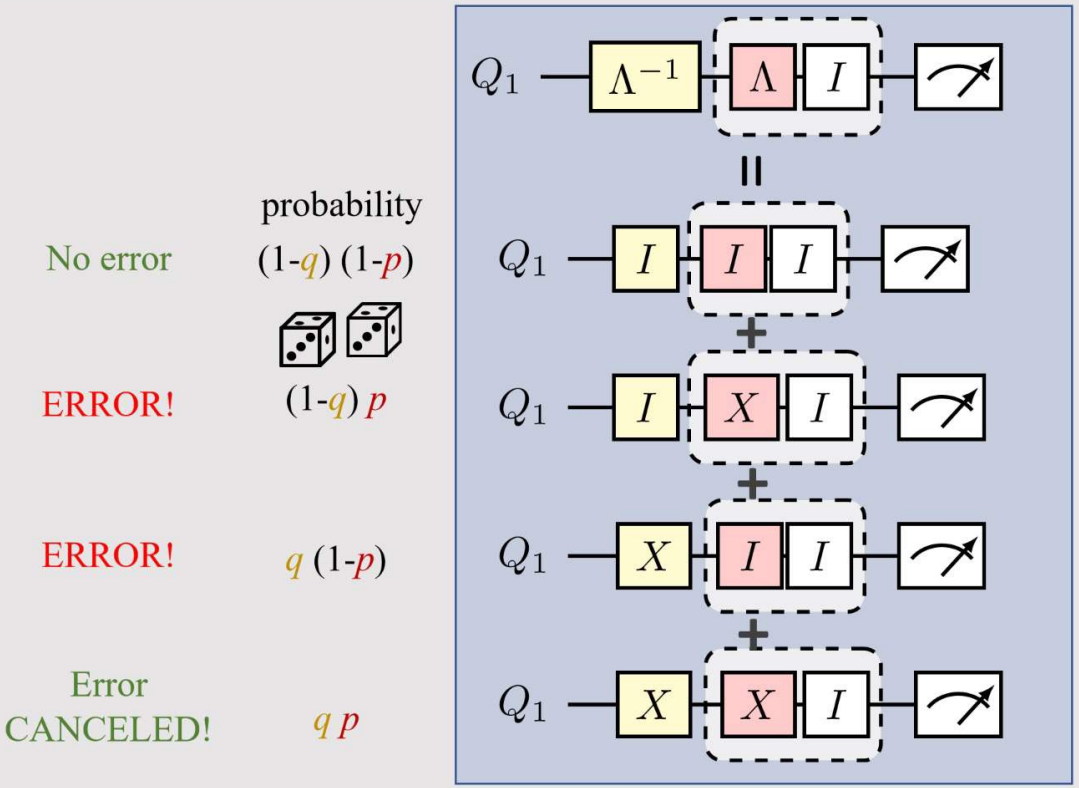


Image credit: Zlatko Minev's blog

To learn more about PEC, check out Zlatko Minev's nice tutorial: zlatko-minev.com/blog/pec-talk

For a review on QEM, see Z. Cai, R. Babbush et al., arXiv: 2210.00921

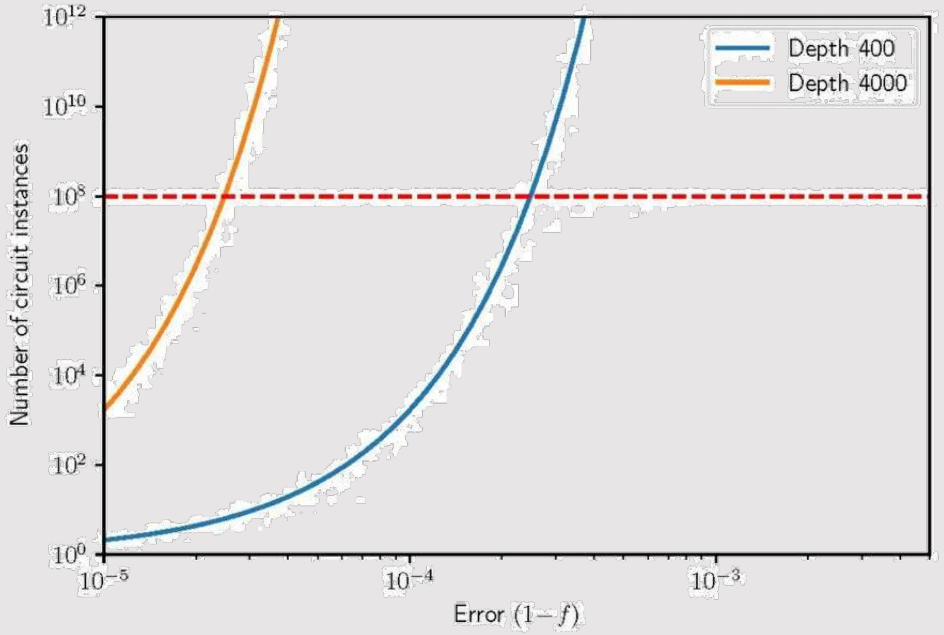
K. Temme, S. Bravyi, and J. M. Gambetta. Phys. Rev. Lett. 119, 180509 (2017)
E. Berg, Z. K. Minev, A. Kandala, and K. Temme. Nat. Phys. (2023)

No free lunch with any error mitigation method

Balance scalability, generality, and cost-effectiveness

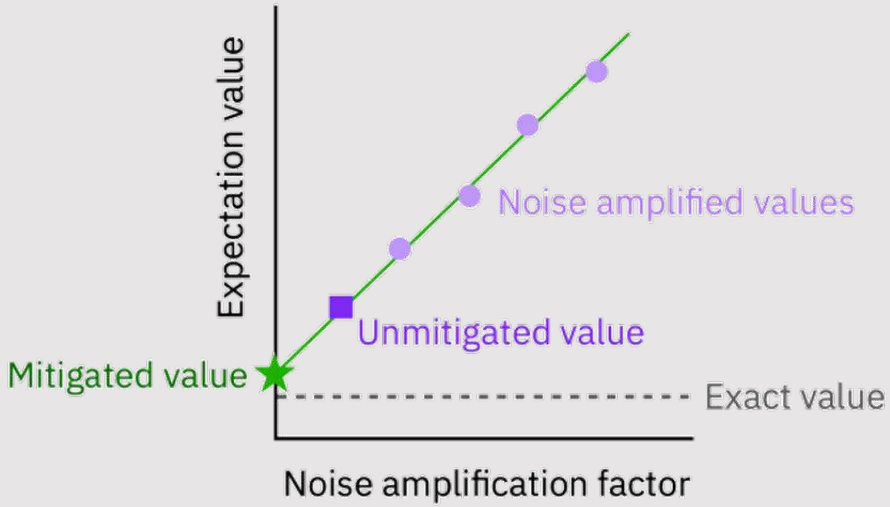
PEC

Pros: Unbiased expectation values
Cons: Known exponentially growing overhead with noise to quench variance, learning noise is practically expensive



ZNE

Pros: Relatively low overhead, no training, more straightforward to implement
Cons: Biased expectation values



Our ML approach

Pros: Arguably lowest runtime overhead, ...
Cons: Biased expectation values

Image credit: IBM Research blog

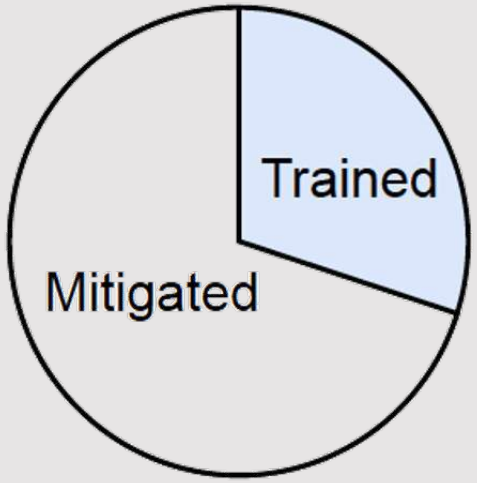
K. Temme, S. Bravyi, and J. M. Gambetta. Phys. Rev. Lett. 119, 180509 (2017)
E. Berg, Z. K. Minev, A. Kandala, and K. Temme. Nat. Phys. (2023)

arXiv: 2309.17368

Machine learning methods accelerates QEM at runtime

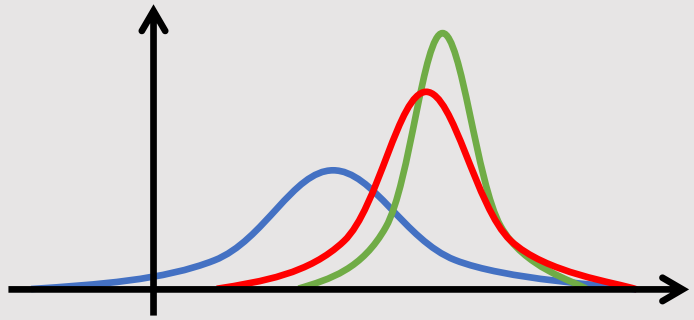
ML for quantum error mitigation (ML-QEM) offers significant advantages in runtime efficiency by front-loading training for on-the-fly error mitigation

Not be confused with the step of “learning device noise” in quasi-probability-based QEM approaches, e.g., PEC, probabilistic error amplification (PEA)*



Its practical utility remains uncertain

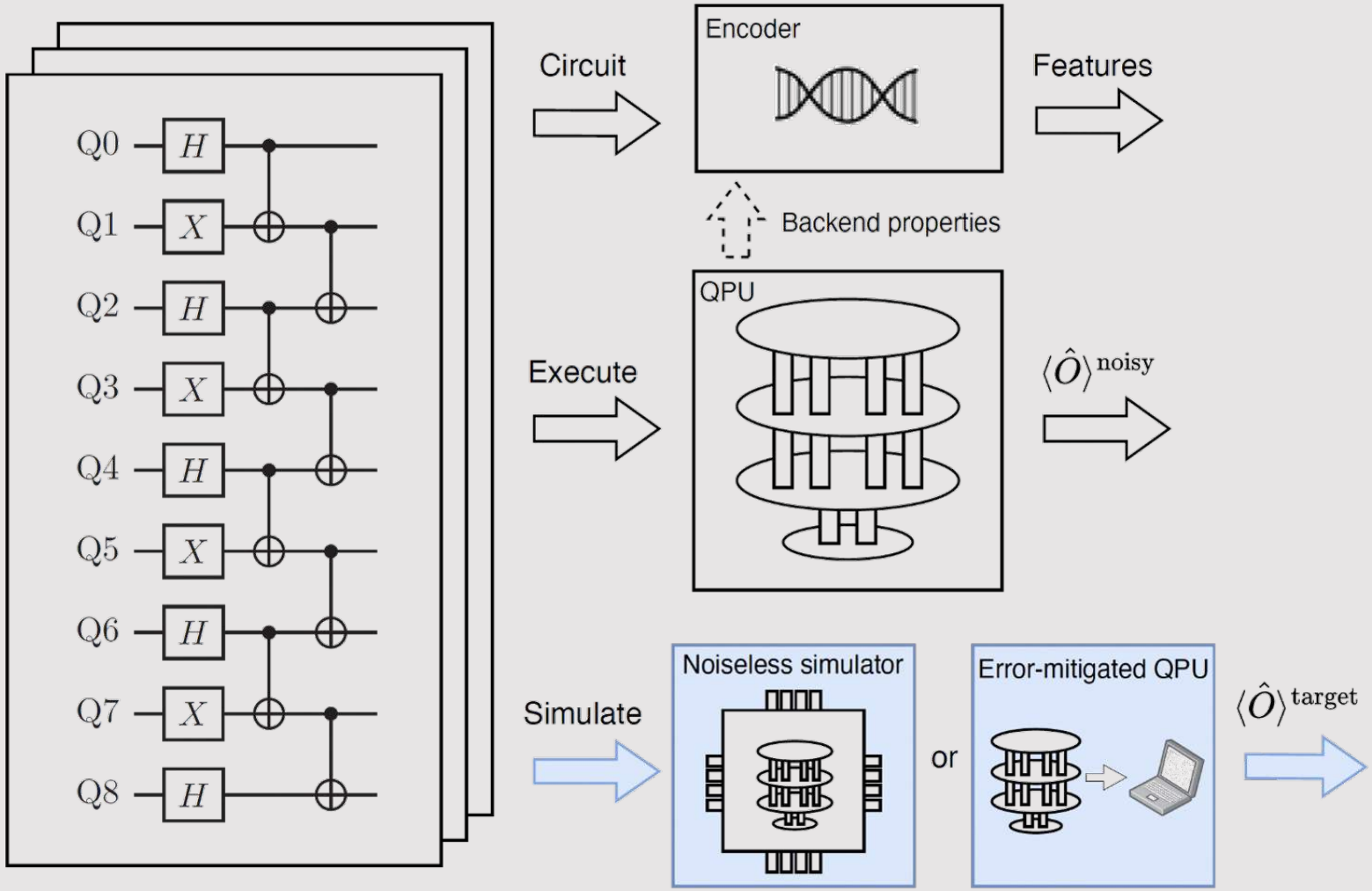
- Performance
- Efficiency
- Applications
- Scalability



*Y. Kim, A. Eddins, S. Anand et al., Nature 618, 500–505 (2023)

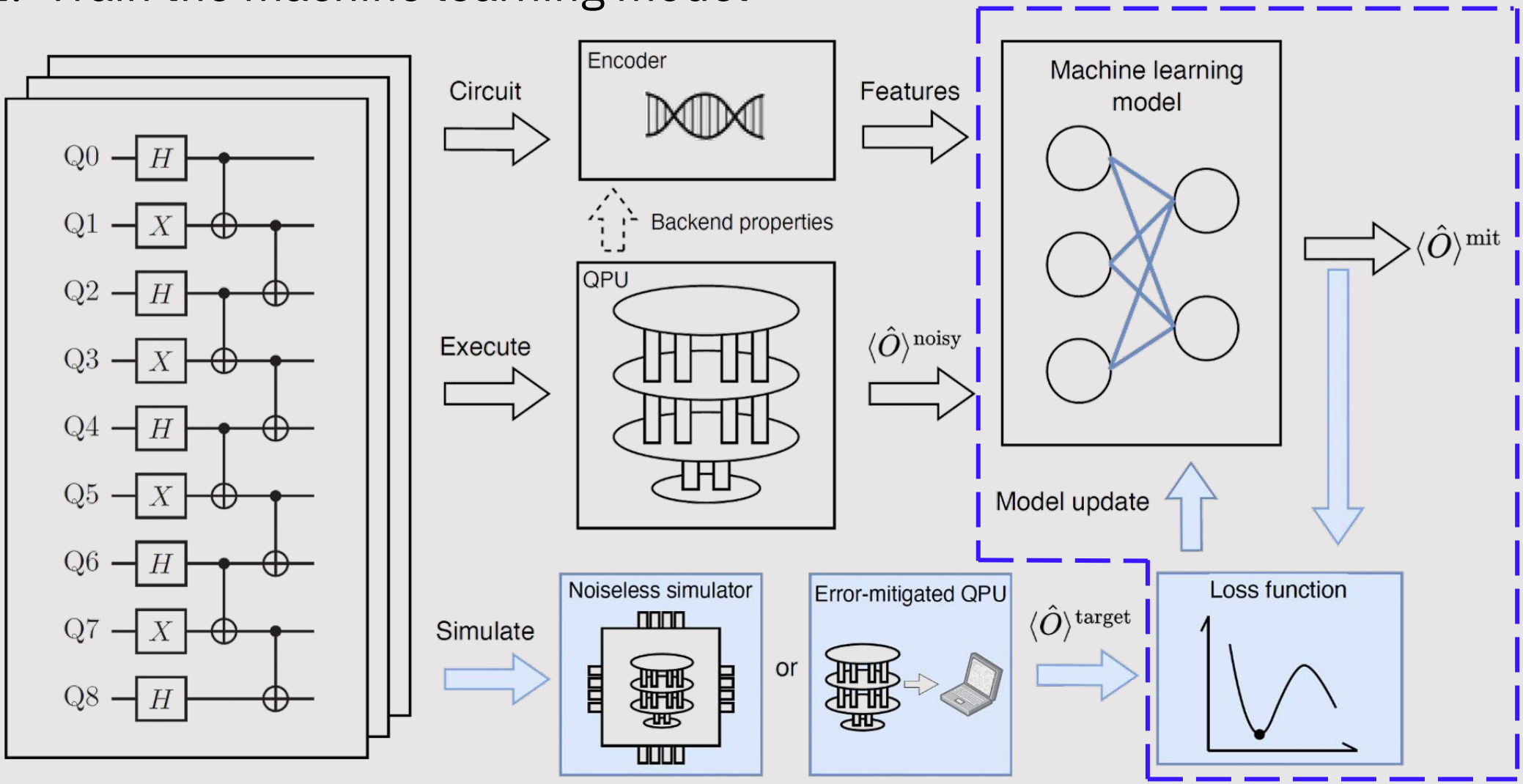
Workflow of ML-QEM

1. Generate training data



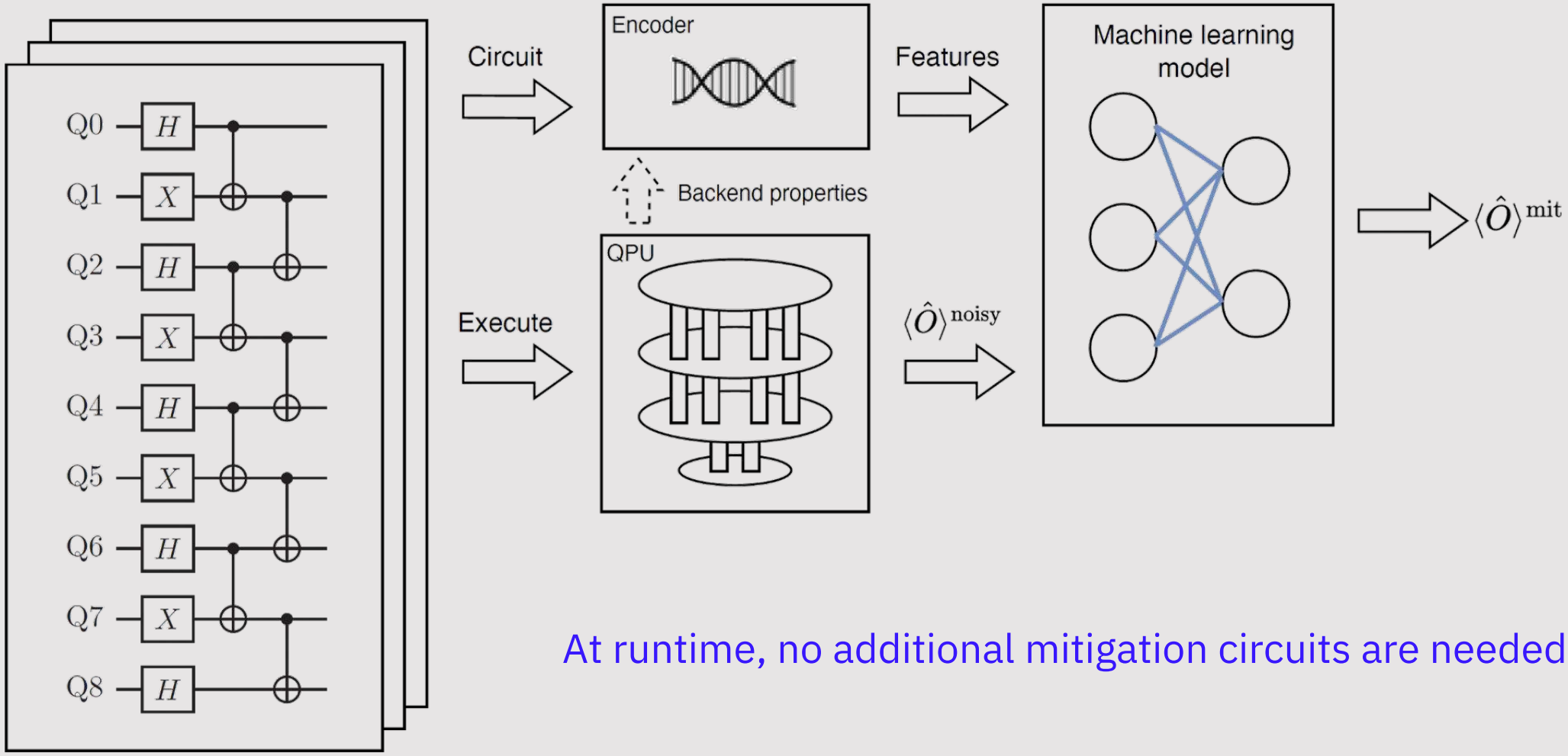
Workflow of ML-QEM

2. Train the machine learning model



Workflow of ML-QEM

3. Mitigate noisy expectation values at runtime

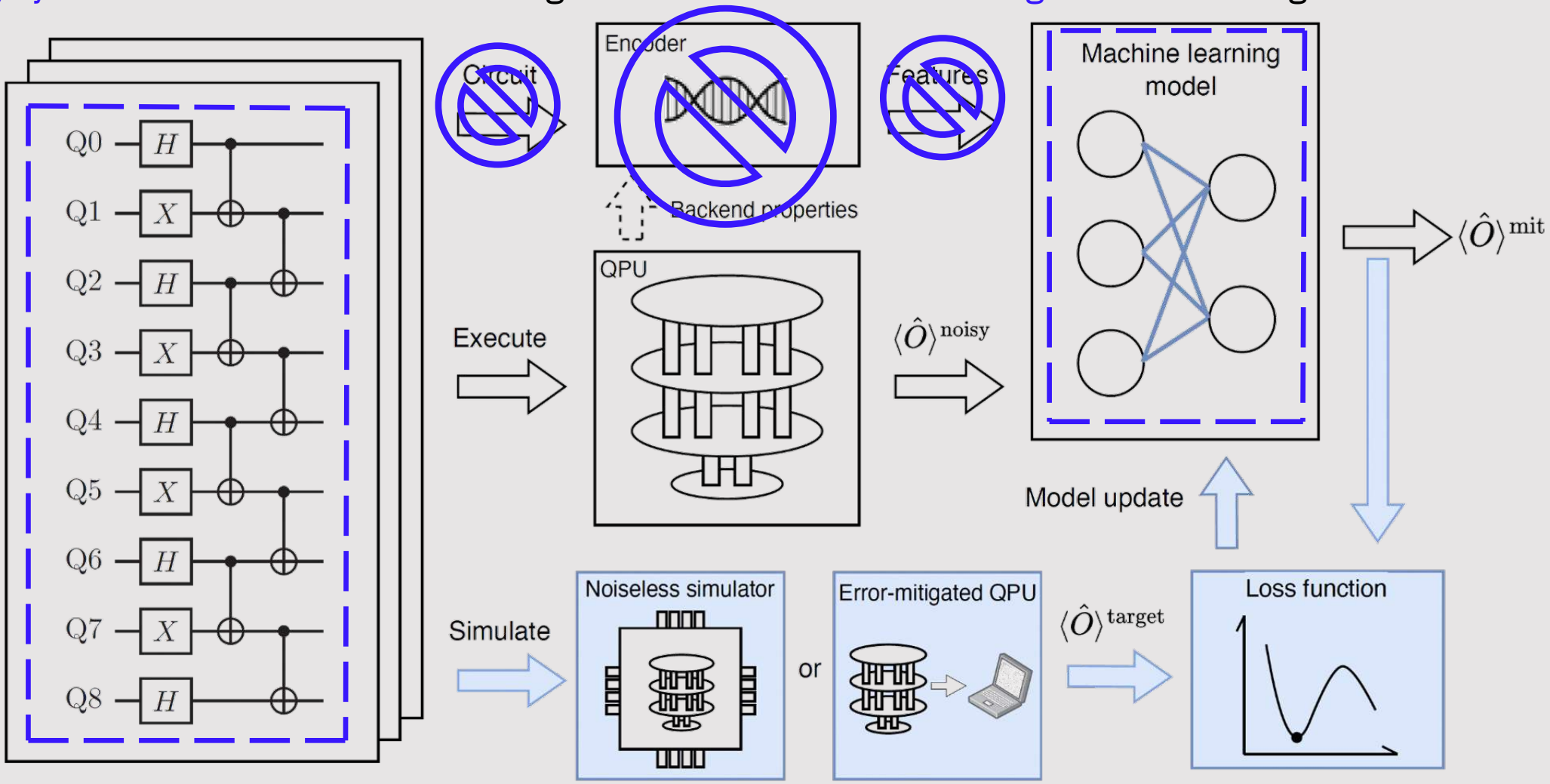


At runtime, no additional mitigation circuits are needed

Compared to previous ML methods

Clifford data regression (CDR)* collects largely-Cliffordized circuits for training

CDR uses only linear regression for mitigation



*P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, Quantum 5, 592 (2021)

Linear regression is insufficient in general

When the noise channels in the circuit are successive depolarizing channels, we show that*

(d: depth, p: depolarizing rate)

$$\beta_1(d,p)\langle\hat{O}\rangle^{\text{noisy}} + \beta_0(d,p) = \langle\hat{O}\rangle^{\text{ideal}}$$

This motivates the use of linear regression in CDR
– However, in general, non-linear model should produce better mitigation (caveat: overfitting)

A spherical cow (GIF credit: Wikipedia)

*Similar results shown in P. Czarnik et al., Quantum 5, 592 (2021)

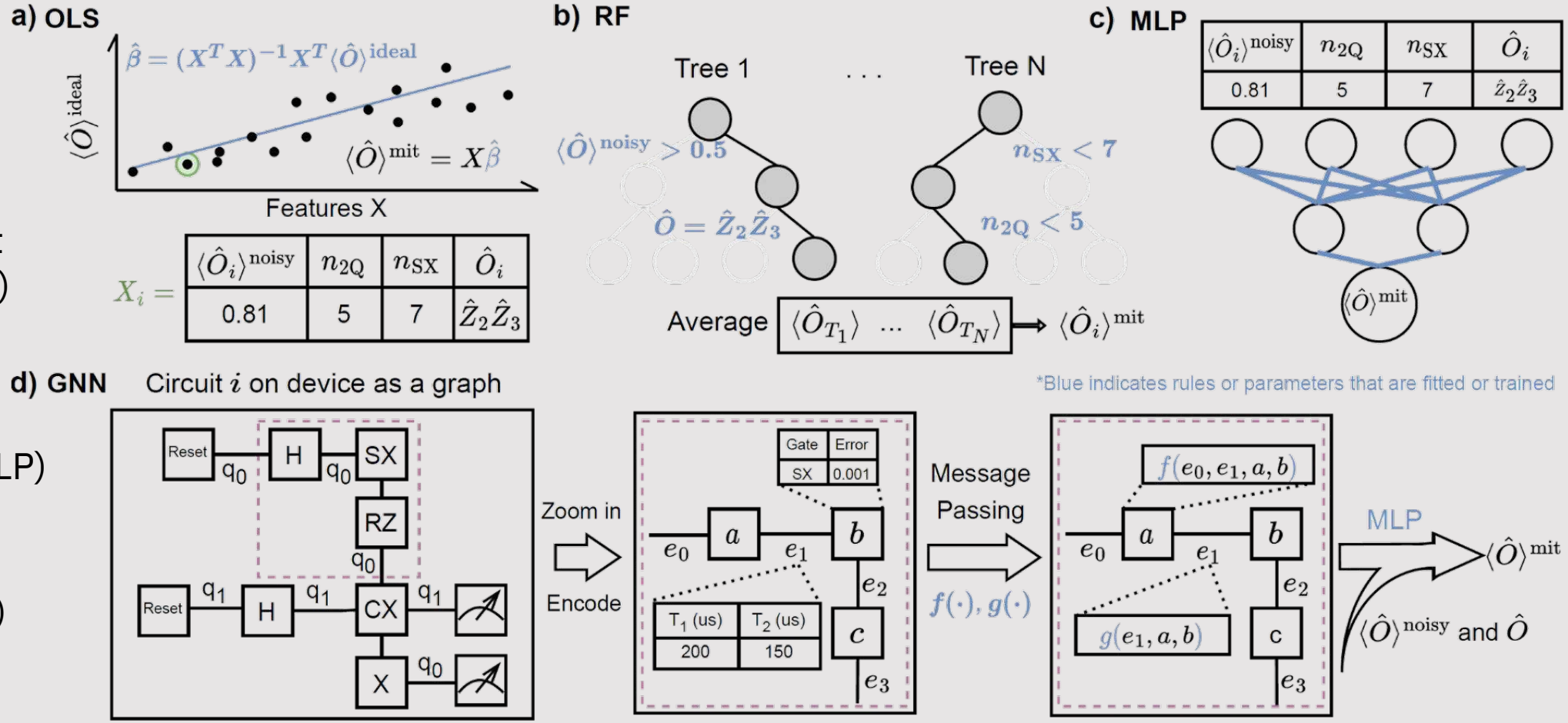
We compare performance of different ML models

a) Ordinary least square (OLS)

b) Random forest regression (RF) [novel]

c) Multi-layer perceptron (MLP)

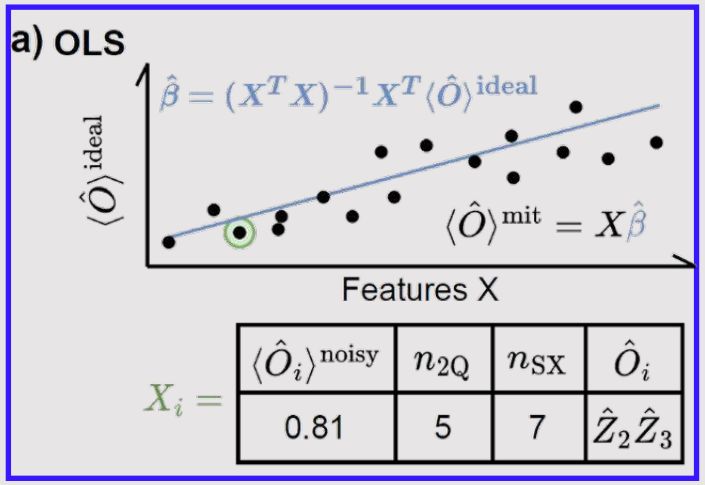
d) Graph neural network (GNN) [novel]



We compare performance of different ML models

a) Ordinary least square (OLS)

Linear

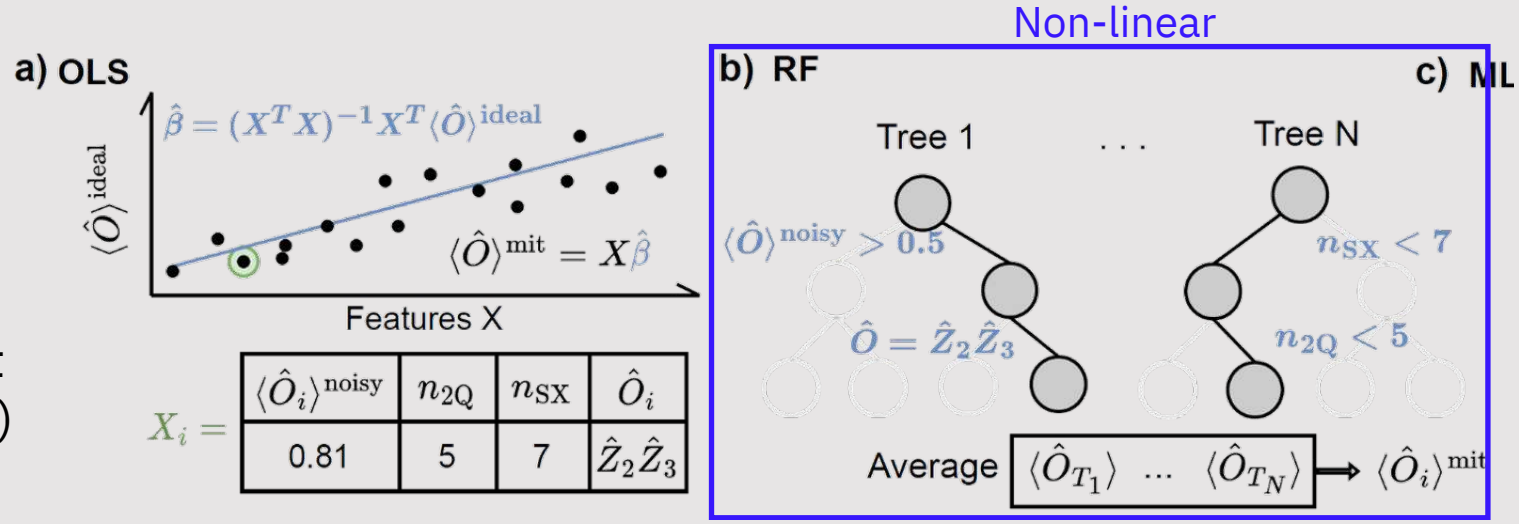


We compare performance of different ML models

a) Ordinary least square (OLS)

b) Random forest regression (RF) [novel]

Non-linear



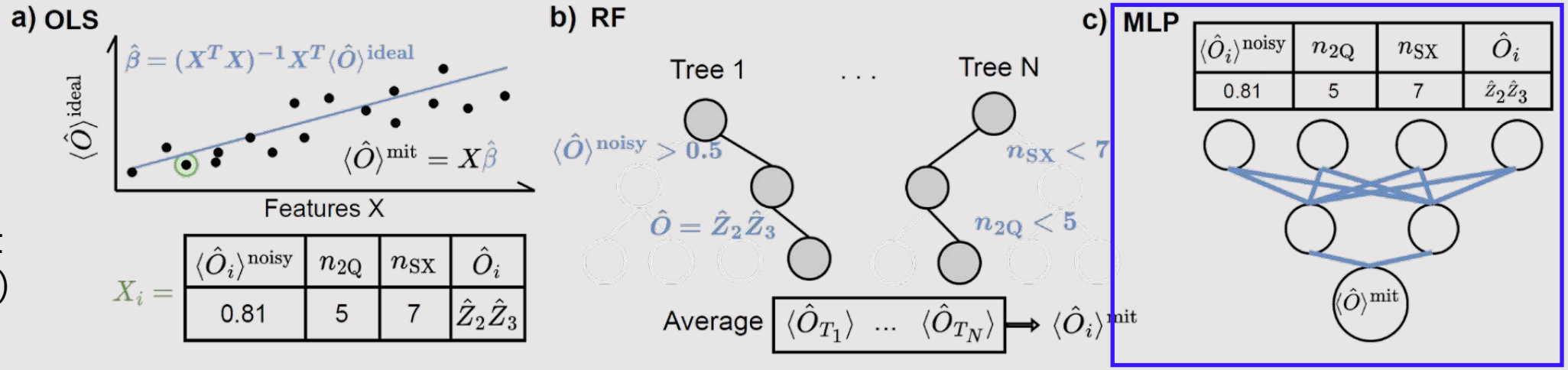
We compare performance of different ML models

a) Ordinary least square (OLS)

b) Random forest regression (RF)

c) Multi-layer perceptron (MLP)

Non-linear



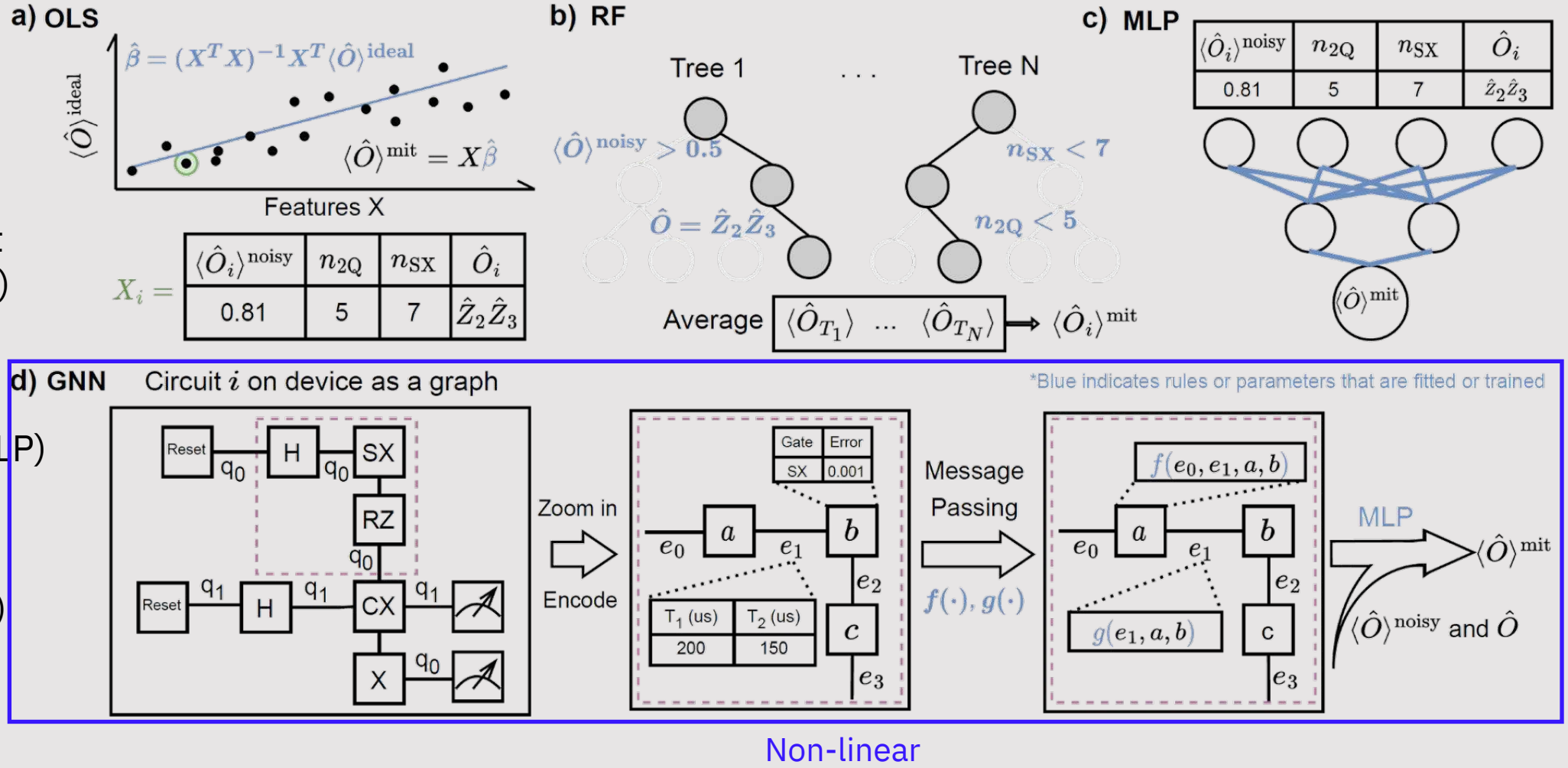
We compare performance of different ML models

a) Ordinary least square (OLS)

b) Random forest regression (RF)

c) Multi-layer perceptron (MLP)

d) Graph neural network (GNN) [novel]



Key results

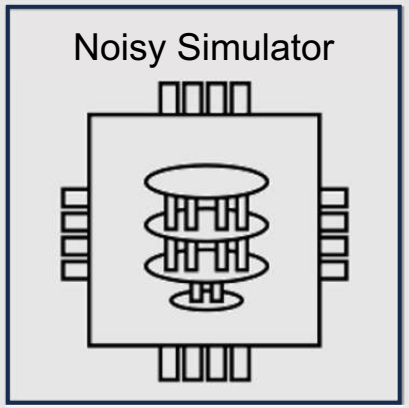
1. Using a general framework for ML for QEM approach, we enhance and survey a broad spectrum of simple to complex machine learning models (linear regression, random forest, multi-layer perceptron, and graph neural network)

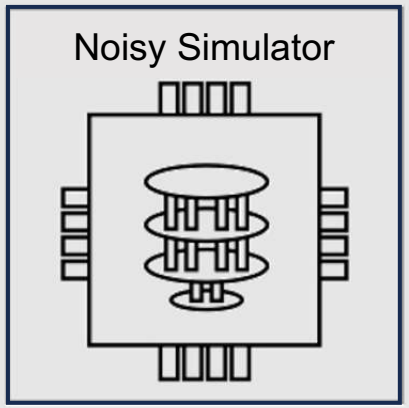
2. Random forest (RF) model consistently outperforms other models

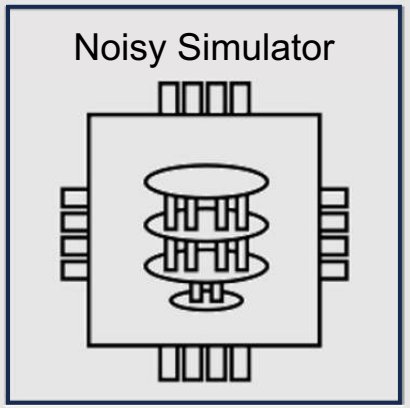
3. On small-scale circuits, ML-QEM significantly and consistently outperforms digital ZNE with lower overhead over a diverse set of tasks, in both noisy classical simulations and hardware experiment

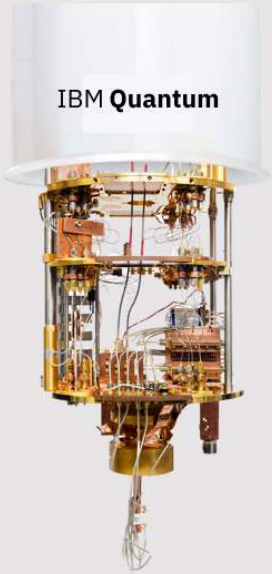
4. On large-scale circuits, we demonstrate in hardware experiment how ML-QEM can significantly improve efficiency of existing QEM methods, e.g., digital ZNE

Noisy simulations and experiments









Random circuits

OLS, RF, MLP, GNN

Structured circuits

Incoherent/coherent noise, readout error

Use cases

Mitigating unseen Pauli observables, variational algorithms (e.g., VQE)

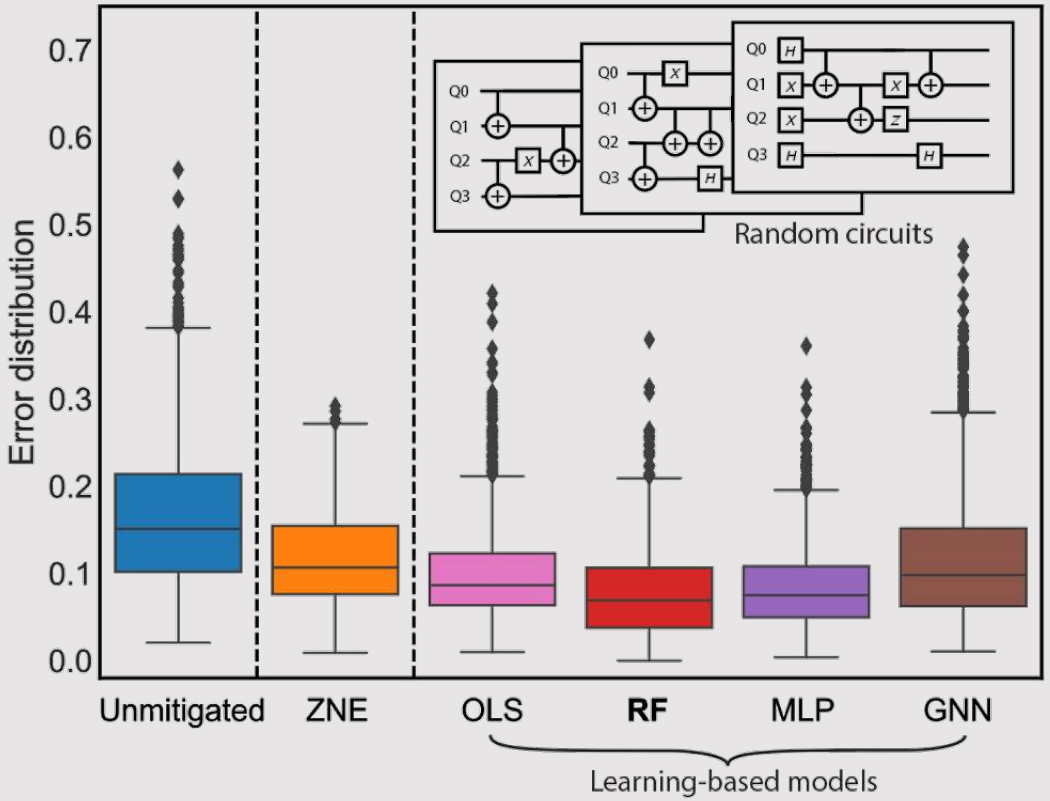
Scalability

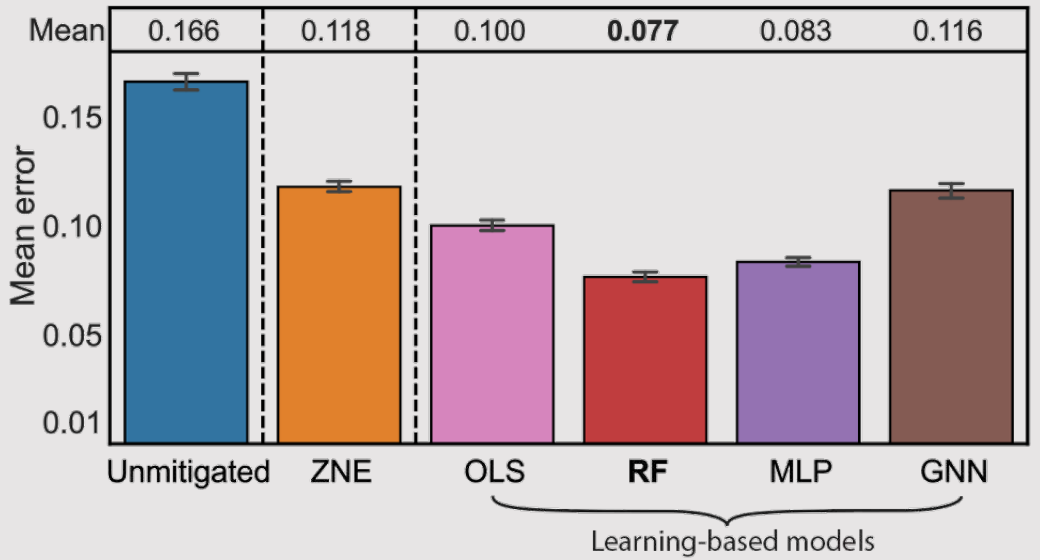
Mimicry of ZNE on 100Q experiment

Performance on random circuits

Train on 5,000 and test on 2,000 random circuits with 4 qubits and 2-qubit gate depth varying from 0 to 18; measure all single-Z observables

Noisy simulation with incoherent noise + SPAM error





Random forest (RF) performs the best

ML-QEM outperforms digital ZNE

Performance on Trotterized circuits

- First-order Trotterized 1d TFIM with 4 qubits
- *J*: coupling strength, *h*: transverse-field strength
- Train on 14 Trotter steps; 300 circuits (different *J*/*h*) per step
- Test on 28 Trotter steps; 300 circuits (different *J*/*h*) per step
- Each circuit is measured in a random Pauli basis with all weight-one observables
- Noise model varies in simulation

1d Transverse-field Ising model (TFIM)

$$\hat{H} = -J \sum_j \hat{Z}_j \hat{Z}_{j+1} + h \sum_j \hat{X}_j = -J \hat{H}_{ZZ} + h \hat{H}_X$$

One Trotter step (first order)