SAND2024-16317C

What is my quantum computer good for? Quantum capability learning with physics-aware neural networks

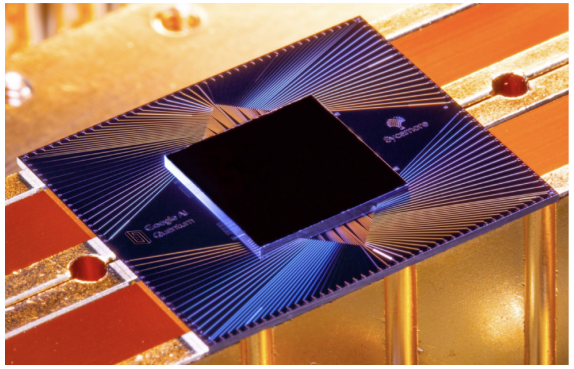
Daniel Hothem, Ashe Miller, Timothy Proctor

38th Conference on Neural Information Processing Systems (NeurIPS 2024)

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

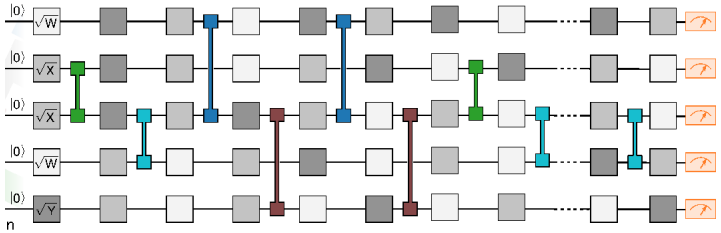
Motivation: What is my quantum computer good for?

Want to buy my quantum computer?



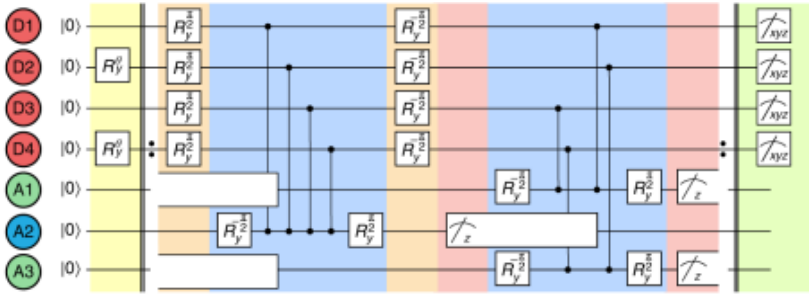
Arute et al., Nature 574, 505 (2019)

Can it run random circuit sampling?



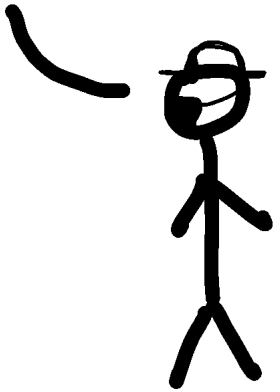
Arute et al., Nature 574, 505 (2019)

Quantum error correction?

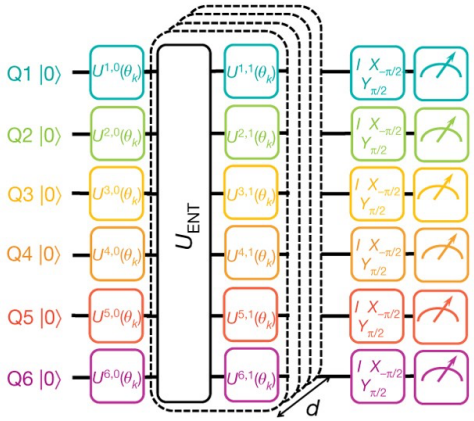


Anderson et al., Nat. Phys. 16, 875 (2020)

What can it do?



QAOA?



Kandala et al.,Nature 549, 242 (2017)

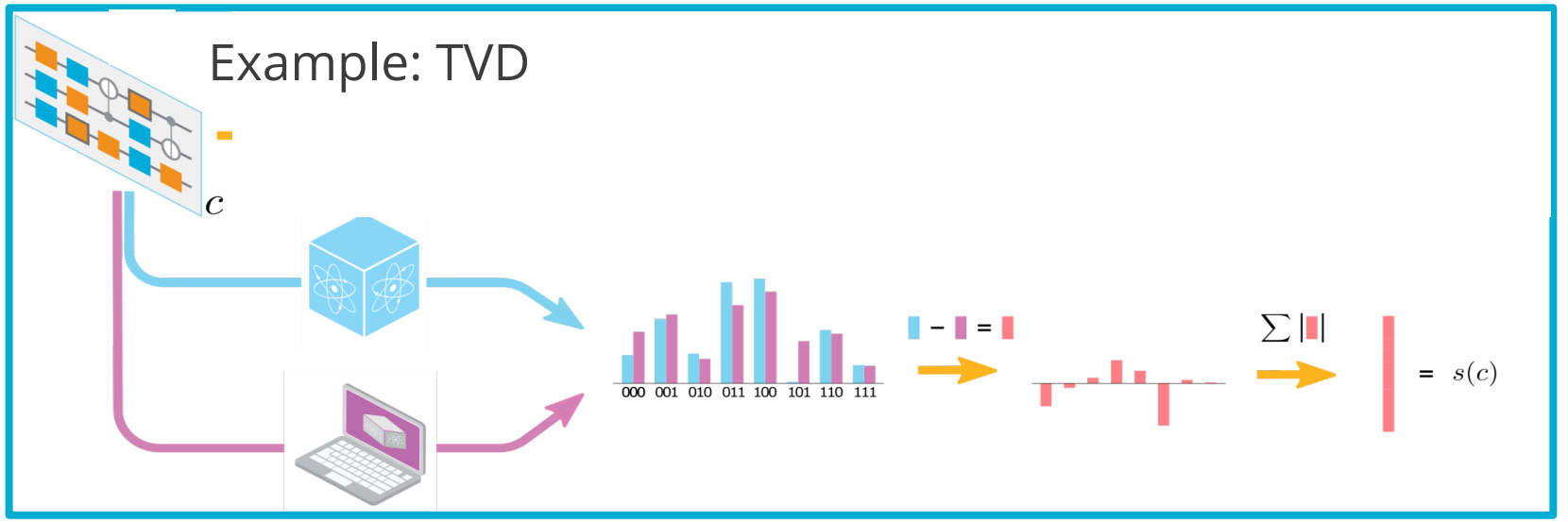
How about VQE? Phase estimation? The QFT? Yada...yada...yadda...

Capability learning: a crash course

We can formalize a quantum computer's ability to run circuits (i.e., programs) with a *capability function:*

$$s(c) = \varepsilon(\text{ideal implementation of } c, \text{actual implementation of } c),$$

where $\varepsilon(\cdot,\cdot)$ is some quality metric (e.g., total variational distance, process fidelity, diamond distance).



arXiv:2305.08796, arXiv:2304.10650

Capability learning: a crash course

Learning $s(c)$ directly is really hard...

- learning the all of the errors in your quantum computer
- and how they interact with each circuit c.

Instead, we try to learn approximate *capability models* from data.

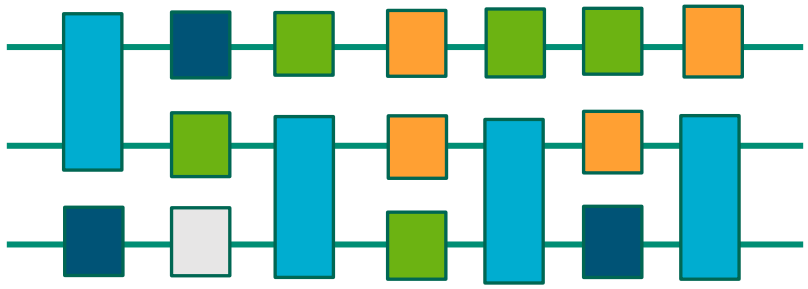
Characteristics of good capability models

- Accurate
- Scalable
- Interpretable

We present a new quantum physics-aware neural network architecture for building capability models

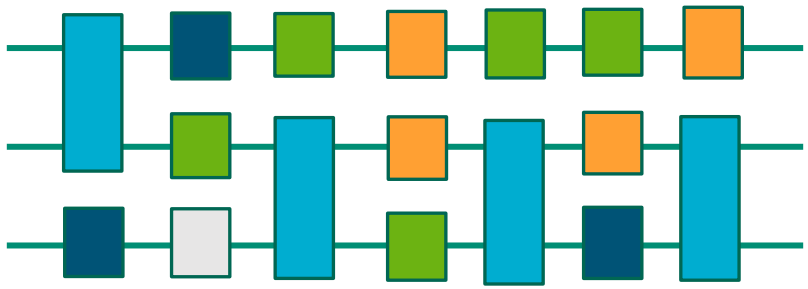
qpa-NNs: How do they work?

$C =$

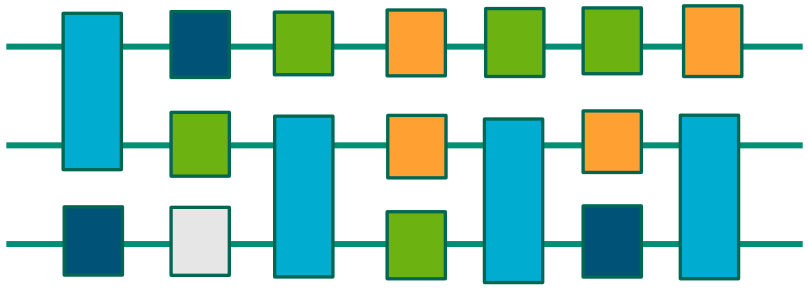


qpa-NNs: How do they work?

$$C = L_1\ L_2\ L_3\ L_4\ L_5\ L_6\ L_7$$

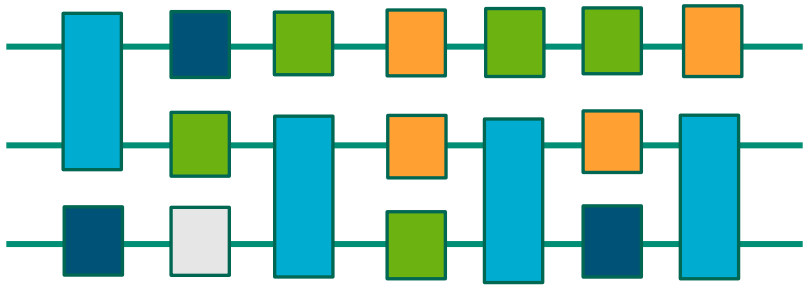


qpa-NNs: How do they work?



$$U(C) = U(L_m) \cdots U(L_2)U(L_1)$$

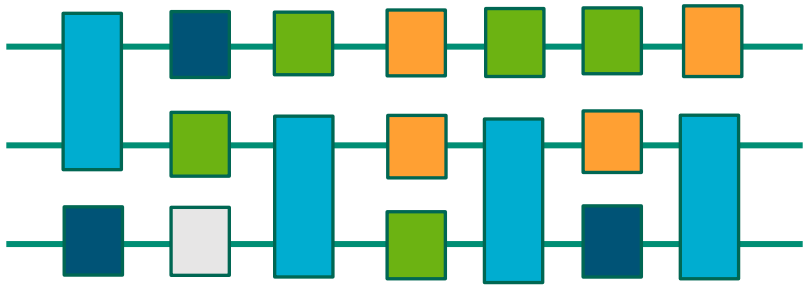
qpa-NNs: How do they work?



$$U(C) = U(L_m) \cdots U(L_2)U(L_1)$$

$$\tilde{U}(C) = \Lambda_m \circ U(L_m) \circ \cdots \circ \Lambda_2 \circ U(L_2) \circ \Lambda_1 \circ U(L_1)$$

qpa-NNs: How do they work?

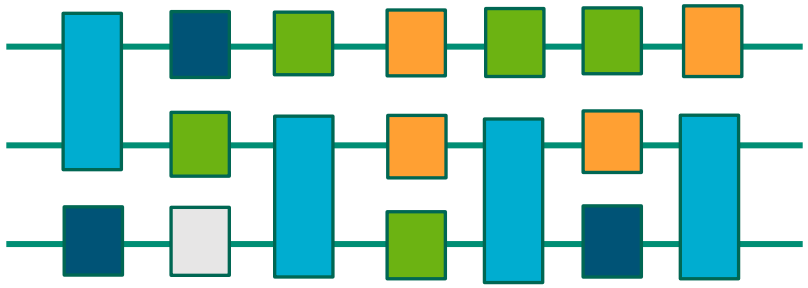


$$U(C) = U(L_m) \cdots U(L_2)U(L_1)$$

$$\tilde{U}(C) = \Lambda_m \circ U(L_m) \circ \cdots \circ \Lambda_2 \circ U(L_2) \circ \Lambda_1 \circ U(L_1)$$

$$\tilde{U}(C) = \Lambda(C) \circ U(C)$$

qpa-NNs: How do they work?



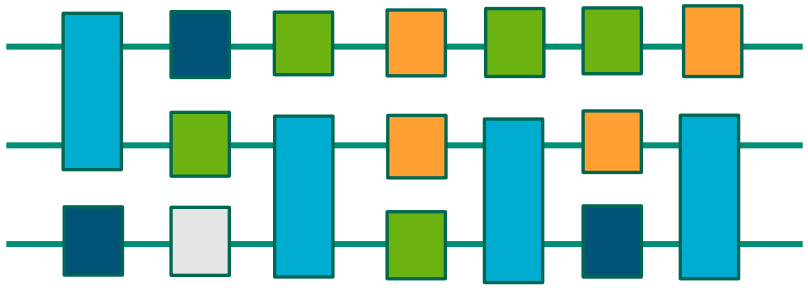
$$U(C) = U(L_m) \cdots U(L_2)U(L_1)$$

$$\tilde{U}(C) = \Lambda_m \circ U(L_m) \circ \cdots \circ \Lambda_2 \circ U(L_2) \circ \Lambda_1 \circ U(L_1)$$

$$\tilde{U}(C) = \Lambda(C) \circ U(C)$$

Many capability metrics can be calculated or approximated by knowing $\Lambda(C)$!

qpa-NNs: How do they work?



Many capability metrics can be calculated or approximated by knowing $\Lambda(C)$!

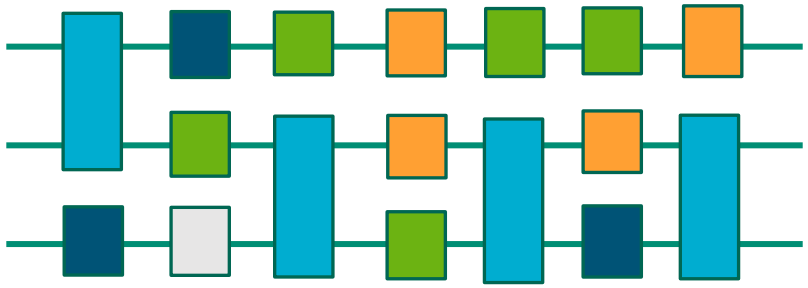
$$U(C) = U(L_m) \cdots U(L_2)U(L_1)$$

$$\tilde{U}(C) = \Lambda_m \circ U(L_m) \circ \cdots \circ \Lambda_2 \circ U(L_2) \circ \Lambda_1 \circ U(L_1)$$

$$\tilde{U}(C) = \Lambda(C) \circ U(C)$$

Key insight: For many circuits learning Λ_i is the hard part

qpa-NNs: How do they work?



Many capability metrics can be calculated or approximated by knowing $\Lambda(C)$!

$$U(C) = U(L_m) \cdots U(L_2)U(L_1)$$

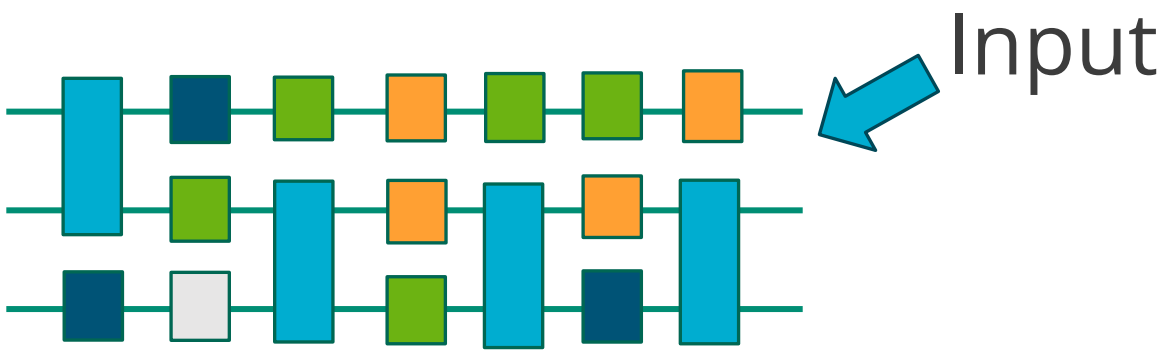
$$\tilde{U}(C) = \Lambda_m \circ U(L_m) \circ \cdots \circ \Lambda_2 \circ U(L_2) \circ \Lambda_1 \circ U(L_1)$$

$$\tilde{U}(C) = \Lambda(C) \circ U(C)$$

Key insight: For many circuits learning Λ_i is the hard part.

Solution: Use neural networks!

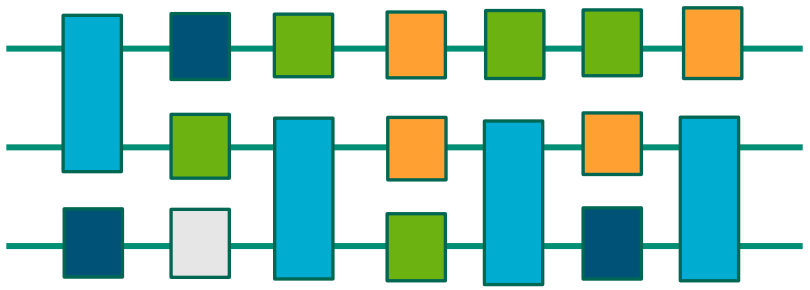
qpa-NNs: How do they work?



Output

$F_{predict}(c)$

qpa-NNs: How do they work?



Input

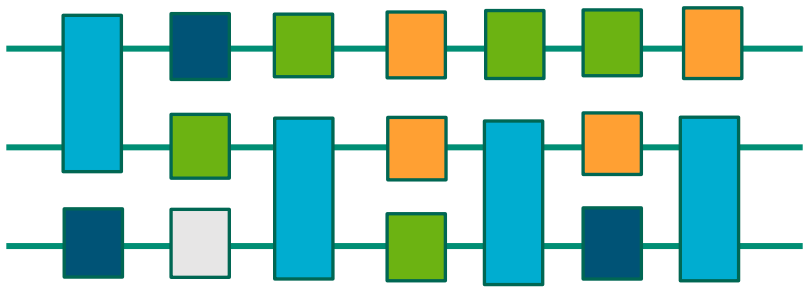
We can express an error channel as: $\Lambda_i = \exp(\sum_P r_{H,P} H_P + \sum_P r_{S,P} S_P)$

$$\vec{r}_i = (r_{H,P1}, \cdots, r_{S,P1}, \cdots)$$

Output

$F_{predict}(c)$

qpa-NNs: How do they work?



Input

We can express an error channel as: $\Lambda_i = \exp(\sum_P r_{H,P} H_P + \sum_P r_{S,P} S_P)$

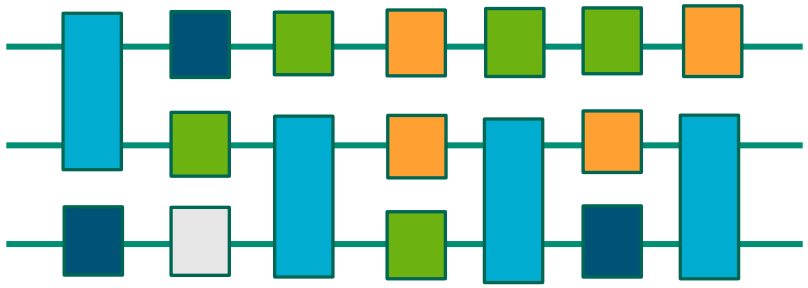
$\vec{r}_i = (r_{H,P1}, \cdots, r_{S,P1}, \cdots)$

We use multilayered perceptrons
to predict each of these error rates.

Output

$F_{predict}(c)$

qpa-NNs: How do they work?



Input

We can express an error channel as: $\Lambda_i = \exp(\sum_P r_{H,P} H_P + \sum_P r_{S,P} S_P)$

$\vec{r}_i = (r_{H,P1}, \cdots, r_{S,P1}, \cdots)$

We use multilayered perceptrons to predict each of these error rates.

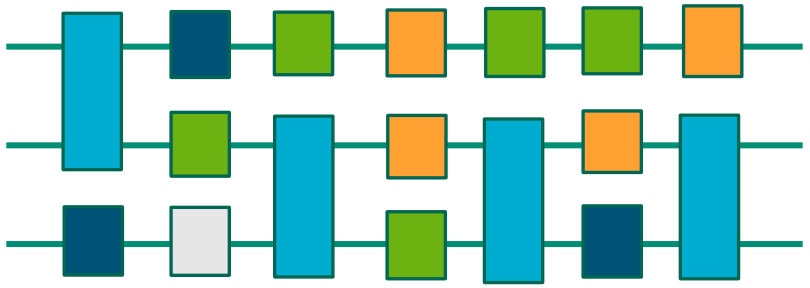
Map circuit to an error rates vector for each layer

$\vec{r}_1 \quad \vec{r}_2 \quad \vec{r}_3 \quad \cdots \quad \vec{r}_d$

Output

$F_{predict}(c)$

qpa-NNs: How do they work?



Input

We can express an error channel as: $\Lambda_i = \exp(\sum_P r_{H,P} H_P + \sum_P r_{S,P} S_P)$

$\vec{r}_i = (r_{H,P1}, \cdots, r_{S,P1}, \cdots)$

We use multilayered perceptrons to predict each of these error rates.

Map circuit to an error rates vector for each layer

$\vec{r}_1 \quad \vec{r}_2 \quad \vec{r}_3 \quad \cdots \quad \vec{r}_d$

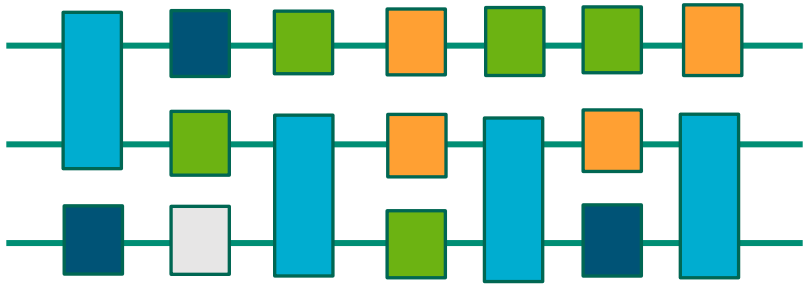
Use circuit-dependent signed permutation matrices

$\vec{r'}_1 \quad \vec{r'}_2 \quad \vec{r'}_3 \quad \cdots \quad \vec{r'}_d$

Output

$F_{predict}(c)$

qpa-NNs: How do they work?



Input

We can express an error channel as: $\Lambda_i = \exp(\sum_P r_{H,P} H_P + \sum_P r_{S,P} S_P)$

$\vec{r}_i = (r_{H,P1}, \cdots, r_{S,P1}, \cdots)$

We use multilayered perceptrons to predict each of these error rates.

Map circuit to an error rates vector for each layer

$\vec{r}_1 \quad \vec{r}_2 \quad \vec{r}_3 \quad \cdots \quad \vec{r}_d$

Use circuit-dependent signed permutation matrices

$\vec{r'}_1 \quad \vec{r'}_2 \quad \vec{r'}_3 \quad \cdots \quad \vec{r'}_d$

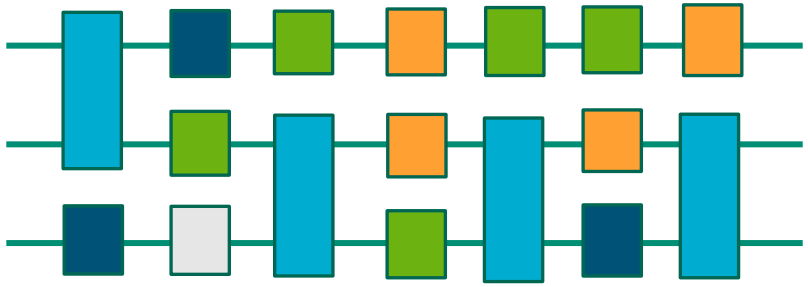
We take a 1st order approximation to combine all error maps.

$\vec{r}_{total}(c)$

Output

$F_{predict}(c)$

qpa-NNs: How do they work?



Input

We can express an error channel as: $\Lambda_i = \exp(\sum_P r_{H,P} H_P + \sum_P r_{S,P} S_P)$

$\vec{r}_i = (r_{H,P1}, \cdots, r_{S,P1}, \cdots)$

We use multilayered perceptrons to predict each of these error rates.

Map circuit to an error rates vector for each layer

$\vec{r}_1 \quad \vec{r}_2 \quad \vec{r}_3 \quad \cdots \quad \vec{r}_d$

Use circuit-dependent signed permutation matrices

$\vec{r}'_1 \quad \vec{r}'_2 \quad \vec{r}'_3 \quad \cdots \quad \vec{r}'_d$

We take a 1st order approximation to combine all error maps.

$\vec{r}_{total}(c)$

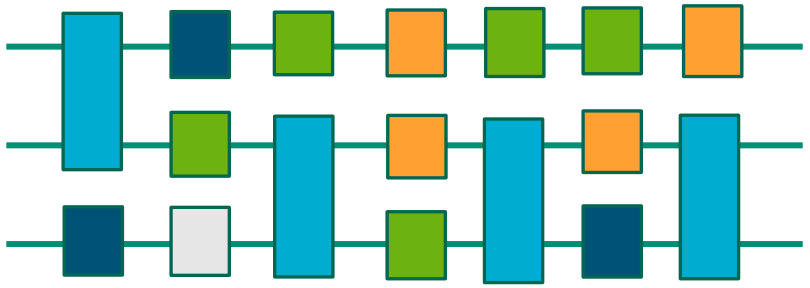
We predict s(c) from $\vec{r}_{total}(c)$. For process fidelity:

$\sum r_{S,P} + r_{H,P}^2$

Output

$F_{predict}(c)$

qpa-NNs: How do they work?



Input

Key missing details:
1. What information do we give each MLP?
2. Which error rates should we track?
3. How do we combine the error maps at the end?

Map circuit to an error rates vector for each layer

$$\vec{r}_1 \quad \vec{r}_2 \quad \vec{r}_3 \quad \cdots \quad \vec{r}_d$$

Use circuit-dependent signed permutation matrices

$$\vec{r}'_1 \quad \vec{r}'_2 \quad \vec{r}'_3 \quad \cdots \quad \vec{r}'_d$$

We take a 1st order approximation to combine all error maps.

$$\vec{r}_{total}(c)$$

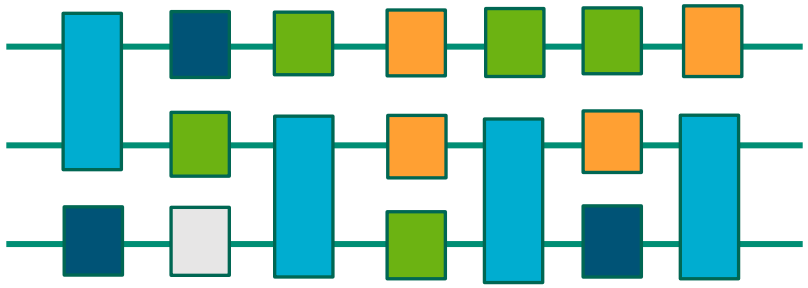
We predict s(c) from $\vec{r}_{total}(c)$. For process fidelity:

$$\sum r_{S,P} + r_{H,P}^2$$

Output

$F_{predict}(c)$

qpa-NNs: How do they work?



Input

Key missing details:

1. What information do we give each MLP?
   Geometrically localized information based on the device's topology.
2. Which error rates should we track?
   Polynomially few, physically relevant low-weight errors.
3. How do we combine the error maps at the end?
   Baker-Campbell-Hausdorff Approximation.

Map circuit to an error rates vector for each layer

$$\vec{r}_1 \quad \vec{r}_2 \quad \vec{r}_3 \quad \cdots \quad \vec{r}_d$$

Use circuit-dependent signed permutation matrices

$$\vec{r}'_1 \quad \vec{r}'_2 \quad \vec{r}'_3 \quad \cdots \quad \vec{r}'_d$$

We take a 1st order approximation to combine all error maps.

$$\vec{r}_{total}(c)$$

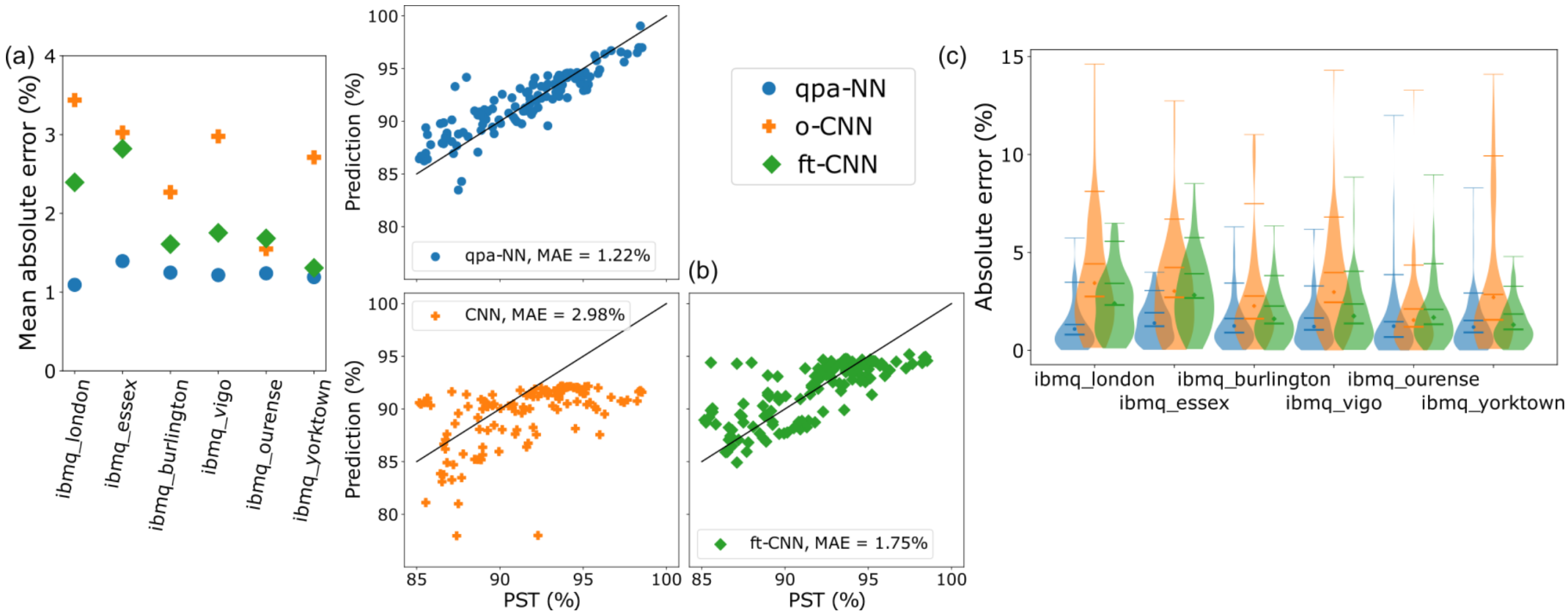
We predict s(c) from $\vec{r}_{total}(c)$. For process fidelity:

$$\sum r_{S,P} + r_{H,P}^2$$

Output

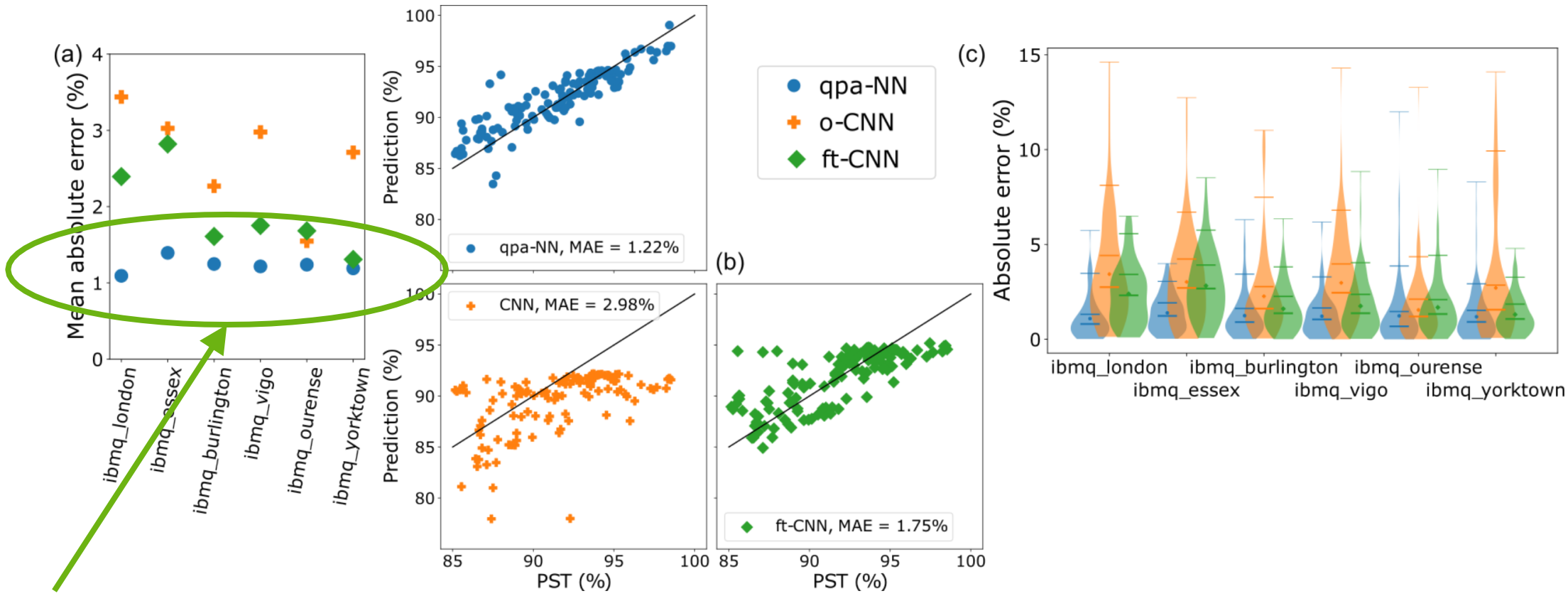
$F_{predict}(c)$

Experimental Results



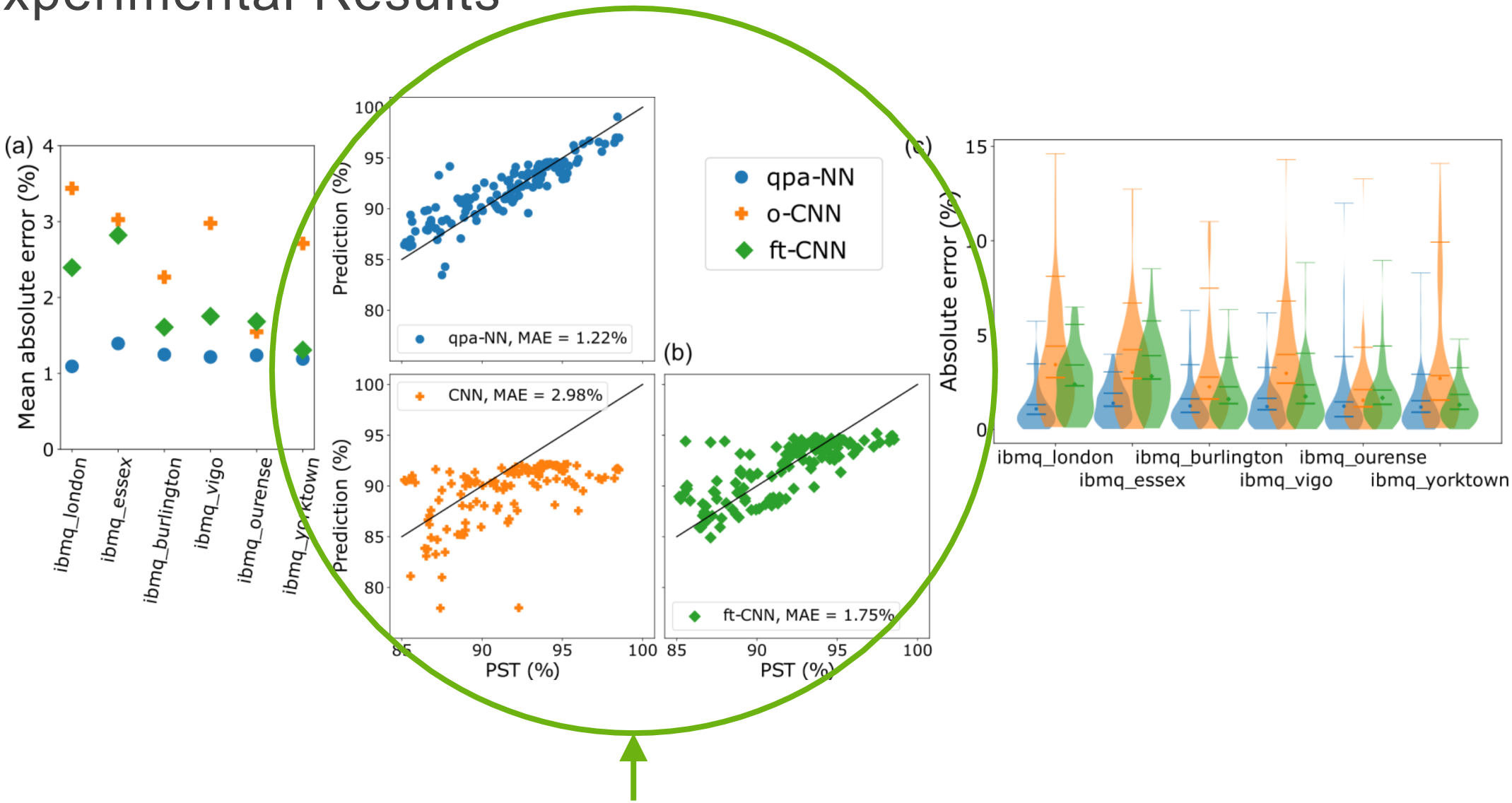
Prediction accuracy on real quantum computers. (a) The mean absolute error of our qpa-NNs (•), the CNNs from Hothem et al. [2023c] (o-CNN, +), and fine-tuned CNNs (ft-CNN, ◆) on the test data. **(b)** The predictions of the three models for ibmq_vigo on the test data, and **(c)** the distribution of each model's absolute error on the test data, including the 50th, 75th, 95th and 100th percentiles (lines) and the means (points)

Experimental Results



qpa-NNs consistently outperform CNN-based capability models.

Experimental Results



Qualitatively superior predictions

References and acknowledgements

Defining the capability learning problem and CNN capability models

Daniel Hothem, Kevin Young, Tommie Catanach, and Timothy Proctor, *Learning a quantum computer's capability*. IEEE Transactions on Quantum Engineering. arXiv preprint: 2304.10650 (April 2023).

Efficient and scalable circuit fidelity estimation

Timothy Proctor, Stefan Seritan, Erik Nielsen, Kenneth Rudinger, Kevin Young, Robin Blume-Kohout, and Mohan Sarovar, *Establishing trust in quantum computations.* arXiv preprint:2204.07568 (April 2022).

Error rates models & ResNet50 transfer learning

Daniel Hothem, Jordan Hines, Karthik Nataraj, Robin Blume-Kohout, and Timothy Proctor, *Predictive models from quantum computer benchmarks*. Proceedings of QCE23. arXiv preprint: 2305.08796 (May 2023).

Quantum Performance Laboratory

ICME

GNNs for capability learning

Raymond Iacobacci, Daniel Hothem, Misbah Ali, Leon Kloker, and Timothy Proctor. *In preparation.*

Berkeley UNIVERSITY OF CALIFORNIA

This work!!!!

Daniel Hothem, Ashe Miller, Timothy Proctor. arXiv preprint: 2406.05636 (May 2024)

UCDAVIS UNIVERSITY OF CALIFORNIA

This work was funded in part by Sandia's Laboratory Directed Research and Development program. This material was funded in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research Quantum Testbed Pathfinder Program. All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of the U.S. Government. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.