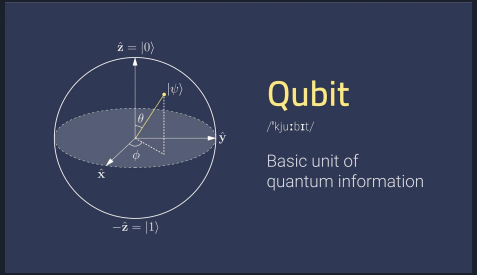
Comparing QRNN Training Methods

Erik C and Michael S

CSCE 585

Motivation and Problem



- The area of QML (Quantum Machine Learning) remains a largely unexplored research area in CS (Computer Science)
- Certain problems related to the training and deployment QRNNs[1] (Quantum Recurrent Neural Networks) are particularly of interest.
- Specifically, we aimed to develop a working framework for the implementation of the SPSA[2] (Simultaneous Perturbation Stochastic Approximation) and FDSA(Finite Difference Stochastic Approximation) algorithms for the purpose of training a QRNN.
- Implementing these training methods would allow us to benchmark the two and devise best practices for training QRNN models.

1. Connerty, E.L., Evans, E.N., Angelatos, G., Narayanan, V. (2025). Quantum Observers: A NISQ Hardware Demonstration of Chaotic State Prediction Using Quantum Echo-state Networks. arXiv preprint arXiv:2505.06799.
2. 482-492.Mitarai, K., Negoro, M., Kitagawa, M., & Fujii, K. (2018). Quantum circuit learning. Physical Review. A/Physical Review, A, 98(3). https://doi.org/10.1103/physreva.98.032309

Algorithms

SPSA

- Asymptotic runtime of O(N), where N is the number of optimization steps.

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k),$$

$$\hat{g}_{ki}(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k\Delta_k) - y(\hat{\theta}_k - c_k\Delta_k)}{2c_k\Delta_{ki}},$$

Finite Differences (FDSA)

- Asymptotic runtime of O(PN), where P is the number of parameters in the VQC, and N is the number of training steps.
- Formulation is similar, except only one parameter is perturbed at a time (Ei instead of EΔ), and the cost function must be evaluated for each parameter.

$$\frac{\partial f}{\partial \theta_i} \approx \frac{f(\theta + ce_i) - f(\theta - ce_i)}{2c}$$

Source: Optimization using SPSA | PennyLane Demos

Learning Task

- Time-series prediction is a common use case for RNNs (Recurrent Neural Networks)
- Chaotic PDEs and ODEs are often a good benchmarking tool for forecasting models due their nonlinear dynamics and dependence on initial conditions.
- As such, we chose the task for our QRNN to be a Lorenz system *cross prediction*, similar to the previous mentioned work on quantum echo-state networks.[1]
- This task is difficult not only because it requires forecasting the Lorenz system into the future, but also because it requires inferring unknown variables of the system.
- In this case, we input the *X* component of the Lorenz system while predicting both the *Y* and *Z* components one time-step into the future.

X(t) → Y(t+1), Z(t+1)

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x),$$

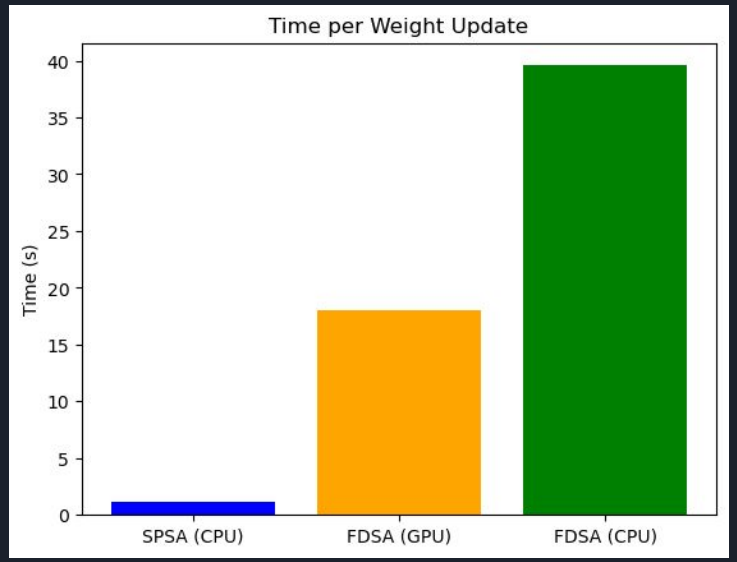
$$\frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y,$$

$$\frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z.$$

Results:
Train Time

- The first goal was to get a working implementation of both SPSA and FDSA in IBM Qiskit and PyTorch.
- This was done by implementing a custom PyTorch layer with custom “backwards” functions for each method.
- With these methods implemented successfully, we could assess the runtime complexity and real world efficiency.
- As an additional metric, the FDSA algorithm was also implemented with GPU acceleration using two NVIDIA A100 GPUs.

Results:
Train Time (cont)



CPU = Apple M3 Max
GPU = 2x NVIDIA A100

Train Time per epoch (hrs):

SPSA (CPU) - ~1 (*14.5x*)

FDSA (GPU) - ~8 (*1.8x*)

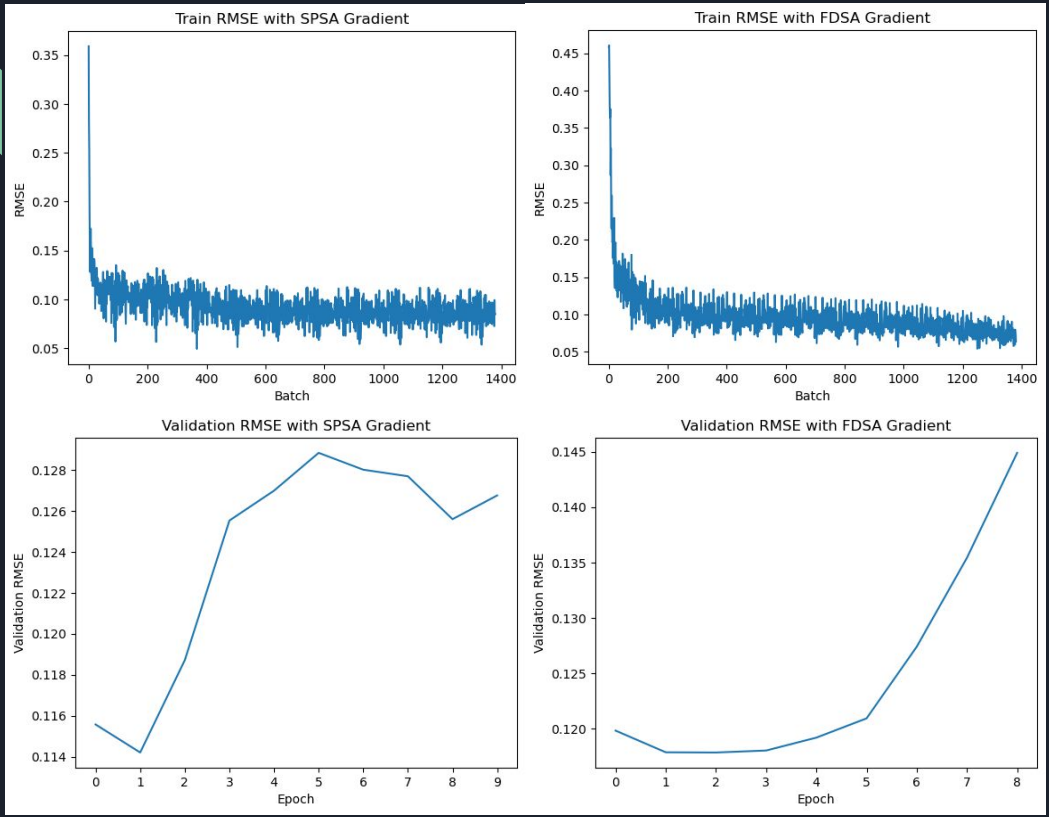
FDSA (CPU) - ~14.5 (*1x*)

Conclusion:
As expected, the SPSA algorithm is dramatically faster at training. Even when comparing to the GPU implementation of FDSA, it is an order-of-magnitude faster as predicted. When implementing FDSA on GPU vs CPU, we got roughly a 2x speedup, which maybe could have been improved if not for some bottlenecks.

Results:
Accuracy

- Training time is only one metric for this framework, with both training and validation accuracy being an important other factor.
- The initial hypothesis was that FDSA would be superior in both train and validation accuracy due to its more precise method of gradient approximation.
- To assess this, training RMSE was logged over each batch during training, and a validation accuracy loop was run and logged at the end of each epoch.

Results: Accuracy (cont.)



Conclusion:

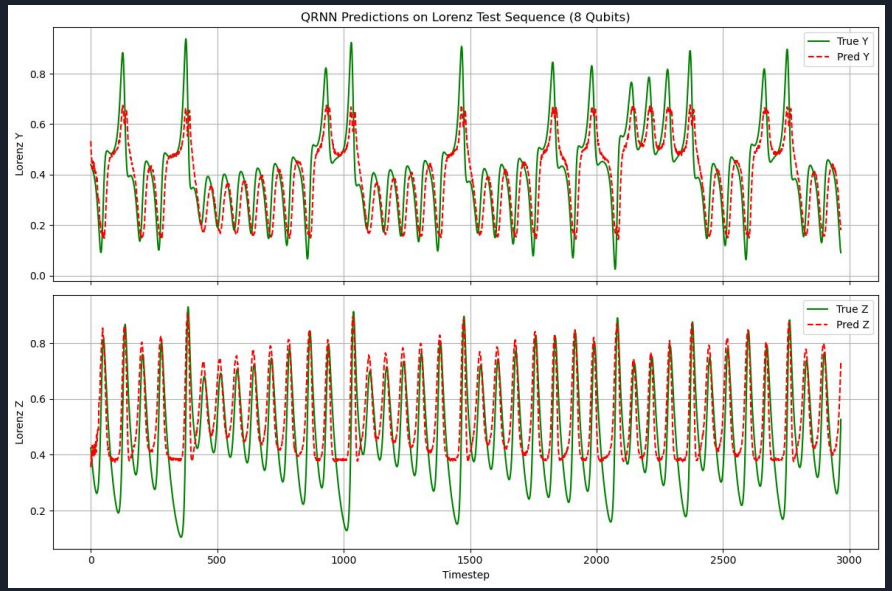
The train RMSE for both methods appears to be similar for both methods throughout training when the learn rate is low enough.

Validation accuracy decreases increases after just 1 epoch with SPSA, and after ~3 with FDSA, making the FDSA algorithm slightly more stable over several epochs.

Overall, there does not appear to be any reason to use FDSA over SPSA given it's slower runtime and comparable accuracy.

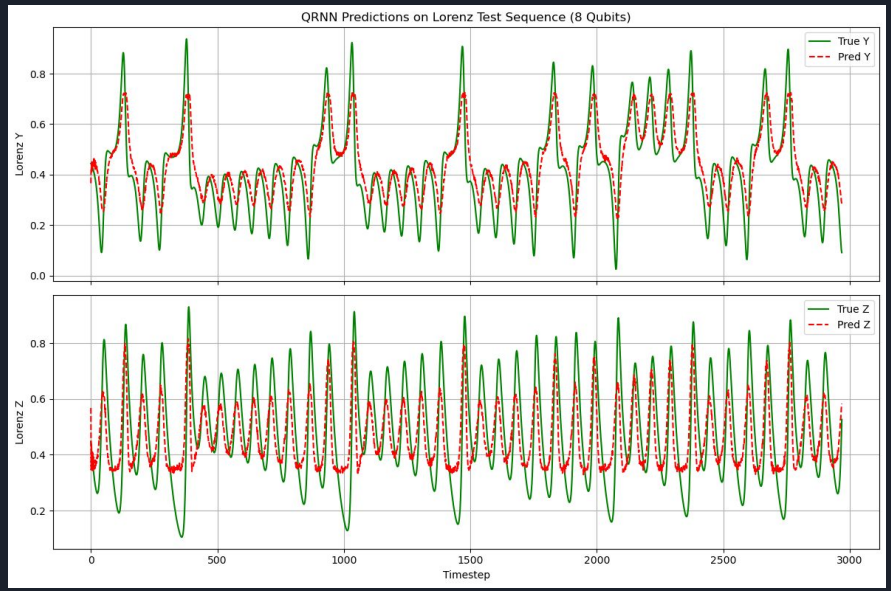
Results: Predictions

SPSA



Lowest RMSE Observed: .09

FDSA



Lowest RMSE Observed: .11

Demo

Conclusion

- We implemented a custom PyTorch layer to train a QRNN model that used 8 qubits
- We benchmarked two well known methods for differentiating QNNs (Quantum Neural Networks), SPSA and FDSA, and determined what their strengths and weaknesses may be.
- Overall, we found that even though FDSA was a slightly more stable training algorithm, SPSA was an order-of-magnitude faster and found better parameters than FDSA in less epochs.
- This project helps informs future researchers of the pitfalls and best practices for training QRNN models for forecasting.