Quantum Reservoir Computing for Sustainable Time-Series Forecasting in High-Energy Physics

Srinjoy, Shalini, Pranay, Krishna, et al.

QIntern Summer Project

May 16, 2025

Abstract

We propose a modular research project to apply **Quantum Reservoir Computing (QRC)** to time-series forecasting in high-energy physics. Focusing on publicly-available physics datasets (e.g. Fermilab's BOOSTR accelerator data or cosmic-ray flux records), we will forecast interpretable targets such as beam energies or detector count rates. Our implementation will use Python (Qiskit/PennyLane for quantum simulation, PyTorch and scikit-learn for data processing and models). In QRC, a fixed quantum dynamical system maps input time series into high-dimensional observables, and only a simple linear readout is trained. This leads to *sustainable* models with greatly reduced training cost and memory compared to deep networks. We will benchmark QRC against classical baselines (e.g. LSTM and Echo State Networks), evaluating forecast accuracy and resource usage. The project deliverables will include an open GitHub repository with code and data, and a paper draft targeting a high-impact venue (e.g. *Scientific Reports* or *NeurIPS Datasets & Benchmarks*). Key novelties are the application of QRC to real physics data and emphasis on model sustainability (low training time and footprint) along with interpretability of forecasting.

1 Background

Reservoir Computing (RC) is an efficient machine-learning framework for temporal data, where a fixed nonlinear dynamical system (the *reservoir*) projects inputs into a high-dimensional space, and only the output layer is trained (typically via linear regression). Unlike deep recurrent networks, RC works well with small training sets and linear optimization, greatly reducing computational resources. This simplicity also aids interpretability, since the trained linear output weights can be analyzed to understand feature contributions. RC avoids difficult training issues in RNNs (e.g. exploding gradients), enabling fast learning and power-efficient inference.

Quantum Reservoir Computing (QRC) extends this concept by using a quantum system as the reservoir. A quantum reservoir (e.g. a network of qubits with fixed couplings) naturally provides an exponentially large feature space and intrinsic memory of past inputs. Crucially, only measurements of observables (e.g. qubit spin values) are used as features for a *linear* output model. QRC thus inherits RC's training efficiency while potentially gaining power from quantum dynamics. Indeed, QRC avoids many variational training challenges of quantum neural networks (such as barren plateaus) by not training the reservoir itself. Recent studies demonstrate that QRC can outperform classical reservoir approaches on forecasting tasks, showing *"substantial promise"* for time-series prediction. (Open-source efforts like CERN's CVQC repository already implement parameterized quantum circuits for time-series forecasting, underscoring the feasibility of this approach.)

High-Energy Physics Time Series. Particle accelerators and observatories generate rich time-series data that can benefit from forecasting and anomaly detection. For example, sensors in a synchrotron produce multivariate streams of magnet currents, RF settings, beam intensities, etc., on each acceleration cycle. Data-driven forecasting in accelerator control is a growing area of interest. Time series forecasting methods have already shown *"encouraging results"* in particle accelerators. The BOOSTR dataset from Fermilab, for instance, provides cycle-by-cycle time series of 55 accelerator device readings (with minimal preprocessing needed). Similarly, cosmic-ray observatories (e.g. the Pierre Auger Observatory) publish long-term counting rates (e.g. 15-minute cosmic-ray flux measurements) as open data. Predicting such physics-based quantities (magnet currents, beam energy, cosmic-ray flux, etc.) could improve machine performance and scientific understanding.

Sustainability and Interpretability. Modern ML models can be resource-intensive. In contrast, reservoir methods are noted for their minimal computational footprint. For sustainability, we will measure QRC's training time and memory versus classical baselines. QRC's sparse training (only output weights) should yield very low training cost. Interpretability is enhanced by the linear readout: one can inspect how individual reservoir observables (e.g. particular qubit measurements) influence the forecast. Our project explicitly targets this *sustainability* angle, aiming for a "green AI" solution with reduced training/energy requirements and understandable model structure.

2 Methodology

Our approach combines public physics data, a quantum reservoir model, and classical baselines. Key components include:

- **Data and Forecast Target:** We will use a public dataset such as the BOOSTR accelerator dataset (cycle-by-cycle booster magnet readings) or a cosmic-ray flux time series from an open observatory. Data preprocessing will be minimal (e.g. aligning timestamps, normalizing values). The forecasting goal will be a physics-relevant quantity, such as the next-cycle peak magnet current or the upcoming average particle energy. This ensures downstream scientific value.

- **Quantum Reservoir Design:** We will simulate a quantum reservoir in software. For example, a small register of qubits (e.g. 4–6 qubits) evolving under a fixed Hamiltonian or random quantum circuit can serve as the reservoir. Inputs from the time series will be encoded into the reservoir each timestep (e.g. by applying rotation gates whose angles depend on the input value). After each input, we simulate the quantum evolution (using Qiskit or PennyLane on a classical backend) and measure a set of observables (such as Pauli-Z on each qubit). The collection of measurement outcomes forms a high-dimensional feature vector for that timestep. These features carry temporal memory of past inputs via the quantum state.

- **Readout and Training:** Only a classical linear readout layer will be trained. We will feed the reservoir's feature vectors into a linear regression (e.g. Ridge regression using scikit-learn) to predict the next value in the series. Training this output layer is efficient since it is just solving a least-squares problem on the collected features. This echoes classical RC practice: "they work on small training sets and operate with linear optimization." We will contrast this with training a full LSTM or deep network, which requires iterative backpropagation.

- **Classical Baselines:** We will implement one or more classical forecasting models for comparison. A natural choice is an LSTM network (implemented in PyTorch) with capacity matched to the problem. We will also implement a classical Echo State Network (ESN) with a similar fixed-random reservoir (possible via existing libraries). These baselines will help benchmark accuracy. We will record their training times and model sizes to highlight QRC's efficiency advantage.

- **Evaluation Metrics:** Forecast accuracy will be measured via standard metrics (RMSE, MAE) on a held-out test set. Crucially, we will also measure resource metrics: the number of trainable parameters, CPU/GPU training time, and memory usage of each model. We expect QRC to use orders-of-magnitude fewer trainable parameters (only output weights) and to require much less training time. We will present these results in tabulated and graphical form.

- **Interpretability Analysis:** To probe interpretability, we will analyze the trained output weights of the QRC. For instance, we can see which qubit observables are most weighted for forecasting, or how reservoir states cluster. This may provide physical insight (e.g. linking certain quantum modes to input fluctuations). Such analysis is simplified because the model is linear.

- **Tools and Code Structure:** The implementation will be done in Python. We will use Qiskit or PennyLane to define and simulate the quantum circuits, PyTorch for any neural-network baselines, and scikit-learn for regression. The GitHub repository will be organized into modules (data loading, QRC model, baselines, training scripts, evaluation). The code will be containerized or well-documented so that QIntern users can reproduce experiments.

3 Implementation Plan

The project will be executed over an 8-week span with the following milestones:

- **Weeks 1–2:** *Data and Environment Setup* – Acquire the chosen dataset (e.g. BOOSTR from Zenodo) and perform any required minimal preprocessing (sorting by time, scaling values). Install and configure Python libraries (Qiskit, PennyLane, PyTorch, scikit-learn). Validate data loading through example scripts.

- **Weeks 2–3:** *Classical Baseline Development* – Implement and train classical forecasting models. For example, develop a PyTorch LSTM to predict the target time series. Also implement a classical reservoir (ESN) model as an additional baseline. Record baseline performance and resource usage.

- **Weeks 3–4:** *Quantum Reservoir Construction* – Design the QRC architecture. For instance, configure a 4-qubit circuit with fixed random couplings. Encode inputs as qubit rotations and simulate the circuit at each timestep. Extract measurement results as features. Validate that the reservoir retains input information (by checking echo state property).

- **Weeks 4–5:** *QRC Training and Tuning* – Collect reservoir features over training data and train the linear output layer (e.g. Ridge regression). Tune hyperparameters such as number of qubits, circuit depth, and regularization. Ensure the QRC can make multi-step forecasts if needed. Iterate on design to optimize accuracy.

- **Week 5:** *Resource Measurement and Comparison* – Measure and compare the computational cost of each model. For QRC and each baseline, record total training time on the same hardware, memory usage, and number of parameters. Plot training curves and metrics. Confirm that QRC requires significantly fewer resources (as theory predicts).

- **Week 6:** *Interpretability and Analysis* – Analyze the trained QRC weights and reservoir states. Possibly use dimensionality reduction (PCA) to visualize reservoir dynamics. Document any physical insights (e.g. which qubit modes correspond to leading time-series components). Prepare figures comparing feature importances.

- **Weeks 7–8:** *Documentation and Write-up* – Clean and modularize code for release (README, instructions). Write the short paper/preprint (4–6 pages) with sections: Introduction, Methodology, Experiments, Results, Discussion. Emphasize novelty, sustainability, and forecasting gains. Cite relevant work (e.g. QRC papers, RC reviews). Prepare the GitHub repository for QIntern with clear instructions and data links.

4 Expected Outcomes

- **GitHub Project:** A complete, modular repository containing data loading scripts, QRC implementation, classical baseline code, and evaluation notebooks. The code will be well-documented, with instructions in a README for reproducing the experiments.
- **Forecasting Results:** Empirical results showing that QRC achieves comparable or better accuracy than classical models on the chosen physics time series. We anticipate that QRC will capture the dynamics effectively due to its large feature space, yielding low forecast error.
- **Sustainability Metrics:** Quantitative demonstration that QRC uses far fewer computational resources. For example, we expect the QRC to have only $O(n)$ trainable parameters (versus $O(n^2)$ in an LSTM) and significantly lower training time. We will report relative improvements (e.g. "80
- **Interpretability Insights:** Analysis showing how reservoir features map to the forecast. For instance, we might identify that certain qubit observables align with specific physical signals. This provides evidence that the QRC model is interpretable and physically meaningful.
- **Paper/Preprint:** A short manuscript suitable for a Tier-1 venue. The paper will clearly state the novelty (first QRC application to [accelerator/cosmic] time series, focus on sustainability), methodology, and key results (accuracy vs. resource trade-offs). Figures will include benchmarking plots of error vs. resource use.

5 Target Venues for Publication

Potential high-impact venues include **Nature Scientific Reports** (broad-audience ML and science), **npj Quantum Information** or **Quantum** (specialized in quantum computing), **IEEE Transactions on Quantum Engineering**, and machine learning conferences with quantum tracks. The *NeurIPS Datasets and Benchmarks* track or an ICML/NeurIPS workshop on quantum ML could be appropriate if emphasis is on the dataset and benchmark. Submission to *Physical Review X* or *PRX Quantum* is also possible given the physics-ML crossover. We will tailor the final manuscript to the chosen venue's style and focus.