Project Documentation: QML Torch

Project Summary

QML torch is a toolkit that integrates quantum computing backends like Qiskit and PennyLane with familiar machine learning frameworks like PyTorch and scikit-learn. This will help lower the learning barriers to quantum computing by letting users add layers of quantum computing into ML models, this will make hybrid ML practical for students and researchers alike. This project matters because it transforms QML from theory into hands-on exploration, providing benchmarks and applied use cases in fields such as cybersecurity, optimization, and bioinformatics. PennyLane is a low-level framework; it gives you the tools to build quantum circuits and connect them to PyTorch, but you still have to design everything yourself. QML Torch takes that foundation and makes it plug and play pre-built quantum layers, benchmark scripts, and applied demos so people can drop a quantum block into a model and immediately test if it helps.

Project Executives

Manager: Ajay Sudhir - asudhir@charlotte.edu
Project Lead: Summer Malik - smalik12@charlotte.edu

Project Information

Background/Motivation:
Quantum machine learning (QML) is often taught at the level of theory or toy examples, but very few students ever get to use it hands on. Libraries like PennyLane and Qiskit are powerful, but they require a steep learning curve and aren't plug and play for students who just want to test hybrid models. QML Torch is motivated by the need to close that gap: to make quantum ML accessible inside the familiar PyTorch/scikit-learn ecosystem while also generating practical case studies that show when hybrid approaches actually help. The bigger vision is to build not just a toolkit, but a CAIR resource that students can learn from, extend, and use in real research or outreach.

Objectives/Goals:
1. Build a pip-installable Python toolkit with pre-built quantum layers and quantum kernels for scikit-learn.
2.Deliver a benchmark harness that compares classical vs hybrid models on toy and applied datasets.
3.Run multiple applied case studies (cybersecurity, optimization, misinformation detection, recommender systems, and bioinformatics).
4.Develop educational deliverables: tutorials, Jupyter notebooks, Streamlit demos, and workshop ready materials

5.Create a long-term, reproducible CAIR project resource with documentation, demos, and benchmarks that future students can build on.

Methodology/Tasks:
(subteam, is described to break apart tasks and group members, subteam can obviously be combined if needed)

Phase 0: On boarding and Teaching (weeks 0-1 or 2)
goal: get everyone on the same page technically and methodologically
1. intro workshop
  -quantum basics qubits, gates, superposition, entanglement.
  -PennyLane & Qiskit walkthrough: building and simulating small circuits.
  - tool set up: Python env (PyTorch, PennyLane, Qiskit, sklearn), GitHub repo, branching workflow, Docker install.,CI pipeline intro.
Mini Exercise: each member runs a quantum kernel SVM on the moons dataset, pushes results.
Team Roles Assignment: assign leads for toolkit, tracks, benchmarks, docs.

Phase 1: Core toolkit (weeks 1-6)
goal: deliver the foundation library that makes quantum layers feel native in pytorch and sklearn.
torch quantum layer:
- encoder subteam: implement angle encoding, implement amplitude encoding, document trade-offs between encoding.
-Anzsats subteam: Build variational templates (RY, RX-CNOT, hardware-efficient) and test shallow vs deeper circuits.
-fine tuning subteam: Compare performance across encoders/ansät and adjust depth/parameters to balance accuracy vs runtime.
quantum kernel:
implementation subteam: Write sklearn-compatible class with fit/transform/predict and add ability to drop into existing sklearn pipelines.
efficiency subteam:Implement caching of kernel matrix and add batching for large inputs
finetuning: Test linear vs polynomial feature maps and benchmark kernel scalability

Phase: 2 Apply QML torch to real world domains, this is the applied track, these are just examples, not all need to be done.
Cybersecurity track: Dataset prep (normalize features, balance classes), modeling (classical), modeling (hybrid, ex. quantum kernel SVM +MLP +Q layer), finetuning (test different encodings vary qubit count, adjust circuit depth)
Optimization Track
Misinformation/NLP track
Recommender Systems Track

Bioinformatics Track

Phase: 3 QML torch is a resource for teaching and learning (hopefully), not everything is practical so these are hypothetical benchmarks
Documentation Site (MkDocs)
API sub-team: write reference docs for functions/classes.
Finetuning: peer review docs, ensure clarity for beginners.
Jupyter notebook series:
Content sub-team: notebooks for Intro to QML, hybrid moons dataset, case studies.
fine tuning: run notebooks in clean environments to confirm reproducibility.
Streamlit demo app:
frontend team
backend team
Outreach material
Distribution

Scope:
The scope of QML Torch is focused but ambitious. The project will deliver a Python library that includes a PyTorch-native TorchQuantumLayer, an sklearn-compatible QuantumKernel, and a benchmark CLI for reproducible experiments. Beyond the toolkit, the team will carry out applied case studies in cybersecurity, optimization, misinformation detection, and recommender systems, with an optional stretch track in bioinformatics. Educational deliverables are also in scope, including Jupyter notebook tutorials, a full documentation site, a Streamlit demo app, and workshop-ready teaching materials for CAIR outreach. To ensure reproducibility, the project will provide Docker images, continuous integration testing, and archived benchmark reports. Out of scope are large-scale production deployments on real quantum hardware, as the project will rely on simulators and the IBM Quantum free tier for small-scale hardware validation.

Team Size:
8–10 members. The project naturally splits into multiple sub teams: core toolkit (encoders, ansätze, kernels), benchmarking, applied case studies (cybersecurity, optimization, NLP, recommender systems, bioinformatics optional), and documentation/UI. Each area requires focused ownership, making a mid-sized team ideal for both productivity and educational value.

GitHub:
https://github.com/Charlotte-AI-Research/qml_torch