# NeuroTKET: An ML-Enhanced Quantum Circuit Compiler Using TKET

## September 20, 2025


### Nathan Delcid

**BS Computer Science | CU Boulder '26**


## Abstract


This project proposes a novel approach to quantum circuit compilation that integrates machine learning with Quantinuum's TKET compiler. The goal is to automatically optimize quantum circuits beyond the capabilities of standard rule-based compilers by developing a lightweight learning-based optimizer that can:

- Adapt to hardware constraints
- Learn circuit simplification patterns
- Leverage existing TKET optimizations as a baseline

Leveraging Quantinuum's open-source TKET toolchain and APIs, the goal is to implement an ML-guided optimization module that works alongside TKET's native passes. This work would include building and testing the system on Quantinuum's device emulator (and real hardware if available), measuring improvements in gate count and expected fidelity. The outcome will be a prototype **"neuro-compiler" (NeuroTKET)** capable of reducing circuit depth or gate count with minimal overhead, plus a demonstration of its effectiveness on example circuits using open-source tools.


## Novelty/Innovation

Existing quantum compilers rely heavily on deep RL agents to find gate sequences. This project explores a new machine learning paradigm for quantum compilation by adhering to the following guidelines:

1. **Avoid heavy pre-training and complex action-space handling of RL-based compilers.**
    - Prior RL compilers (e.g. Quarl [3] and others) had to address enormous action spaces and state representations with specialized neural architectures, and required long training phases to learn a policy. In contrast, this approach would use self-supervised learning to guide circuit optimizations, with the hopes of significantly simplifying the learning problem.
2. **Instead of models learning from scratch, leverage TKET's powerful existing optimizations as a baseline and then learn residual improvements.**
    - This means the ML module focuses on identifying non-obvious circuit transformations that TKET's built-in passes might miss – for example, merging rotations or reordering gates in ways that standard heuristics don't attempt. Notably, Quarl's [3] RL agent surprisingly learned complex optimizations like rotation merging that usually require a dedicated pass; this project aims to achieve similar advanced optimizations via a different learning technique.
3. **Prioritize incorporating hardware-awareness directly into the compiler.**
    - Using Quantinuum's API and device noise models, the ML module could learn to prefer transformations that improve fidelity on Quantinuum hardware. Traditional compilers optimize mostly for gate count or depth, but this compiler, enhanced by ML, can be tuned to also account for hardware specifics (e.g. which gate types or qubit mappings yield higher fidelity on H1-1). This kind of adaptive machine-learned optimization goes beyond prior RL compilers that mostly focused on gate count minimization.

In summary, the project's novelty lies in **combining TKET's deterministic compiling strength with a data-driven ML layer to achieve faster, approximate optimizations without the heavy infrastructure of deep RL**. It could chart a new path for quantum compiling by using modern ML algorithms (graph neural networks, pattern recognition, etc.) to automate circuit improvements.

This strategy has seen only limited exploration (e.g. Smith et al. [6] targeted a few compiler passes, but no comprehensive ML-driven optimizer exists). This project will conclude with a demonstration prototype that can automatically discover circuit reductions/reorders (that a human or default compiler wouldn't easily find) by leveraging the open-source resources at hand.

## References


[1] Moro, L., et al. "Quantum compiling by deep reinforcement learning." *Communications Physics* 4, 178 (2021). https://www.nature.com/articles/s42005-021-00684-3

[2] Zhang, Y., et al. "RL-based quantum compiler for a superconducting processor." *arXiv preprint arXiv:2406.12195* (2024). https://arxiv.org/abs/2406.12195

[3] Li, G., et al. "Quarl: A Learning-Based Quantum Circuit Optimizer." *arXiv preprint arXiv:2307.10120* (2023). https://arxiv.org/abs/2307.10120

[4] Quantinuum TKET Documentation – pytket and pytket-quantinuum API guides. https://docs.quantinuum.com/

[5] Quantinuum Backend Info – Device and emulator availability (H1-1, H1-1E, etc.) and usage of default compilation passes. https://docs.quantinuum.com/

[6] Smith, J., et al. "MQT Predictor: predicting optimal quantum compiler passes." In *Proceedings of the International Conference on Quantum Computing* (2022). https://pages.cs.wisc.edu/ (illustrating prior ML approach limited to few passes).