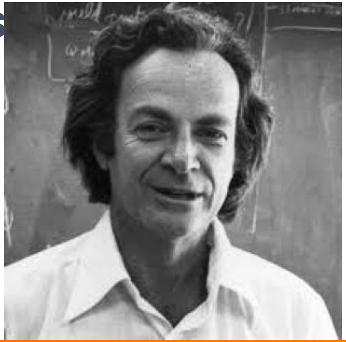
TensorFlow Quantum:

An open-source library for quantum machine learning

Google AI Quantum

Masoud Mohseni

MPML, July 30, 2020



“Nature isn’t classical, dammit, and if you want to make a simulation of nature, you’d better make it quantum mechanical!”

- Richard Feynman

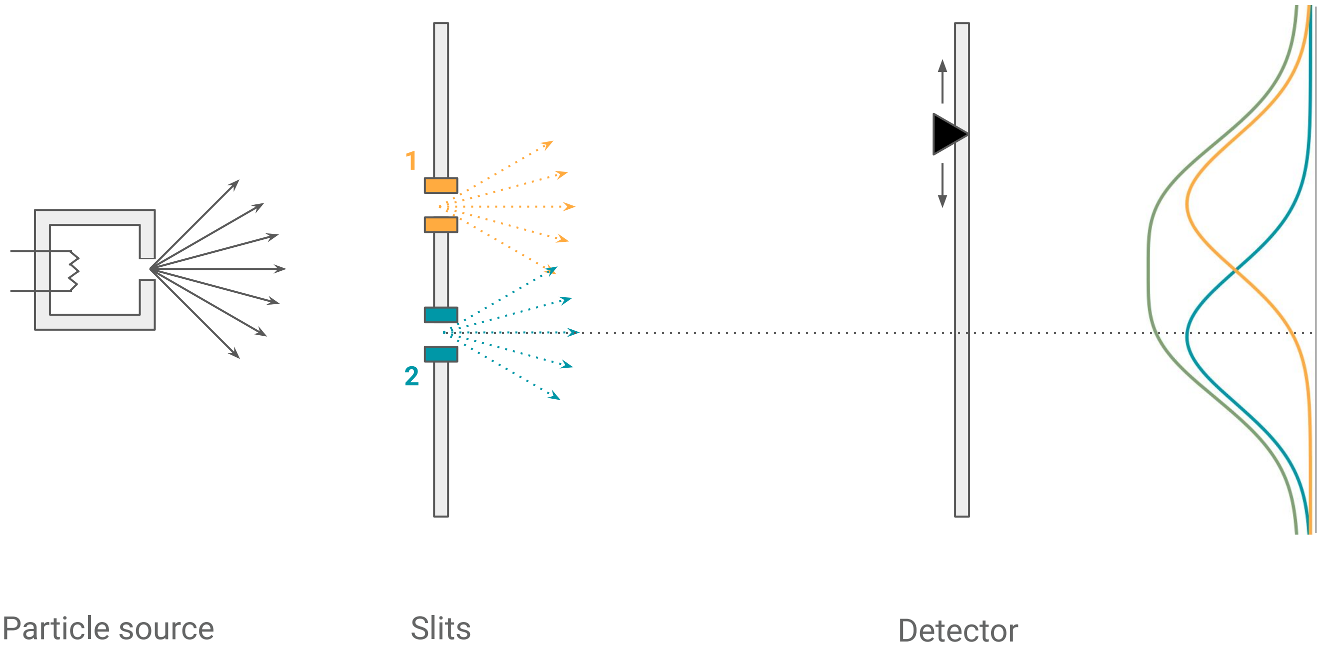
“Nature isn’t classical, dammit, and if you want to learn a model of nature, you’d better make it quantum mechanical!”

TFQ team

Quantum mechanics, quantum computing, and quantum machine learning

in the next slide!

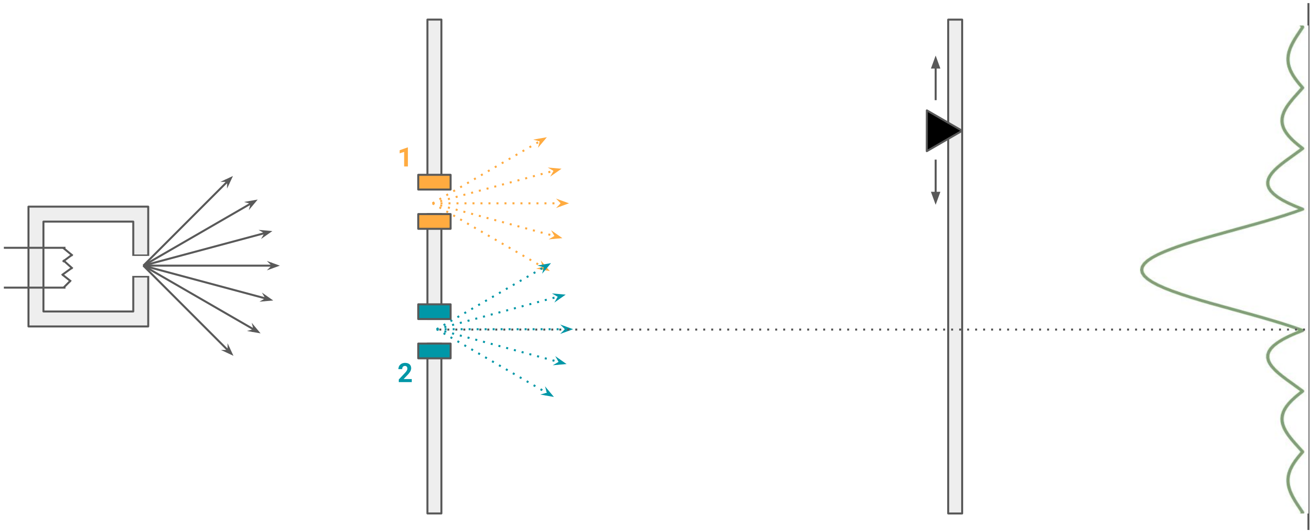
Double-slit experiment at the heart of quantum mechanics



Probabilities for classical particles are additive

$$P(x) = p_1(x) + p_2(x)$$

Double-slit experiment at the heart of quantum mechanics



Quantum particles interfere, and can cancel out

$$P(x) = |\phi_1(x) + \phi_2(x)|^2$$
$$= |\phi_1|^2 + |\phi_2|^2$$
$$\underbrace{+|\phi_1||\phi_2|\left[e^{i(\theta_1-\theta_2)} + e^{-i(\theta_1-\theta_2)}\right]}_{\text{Quantum interference}}$$

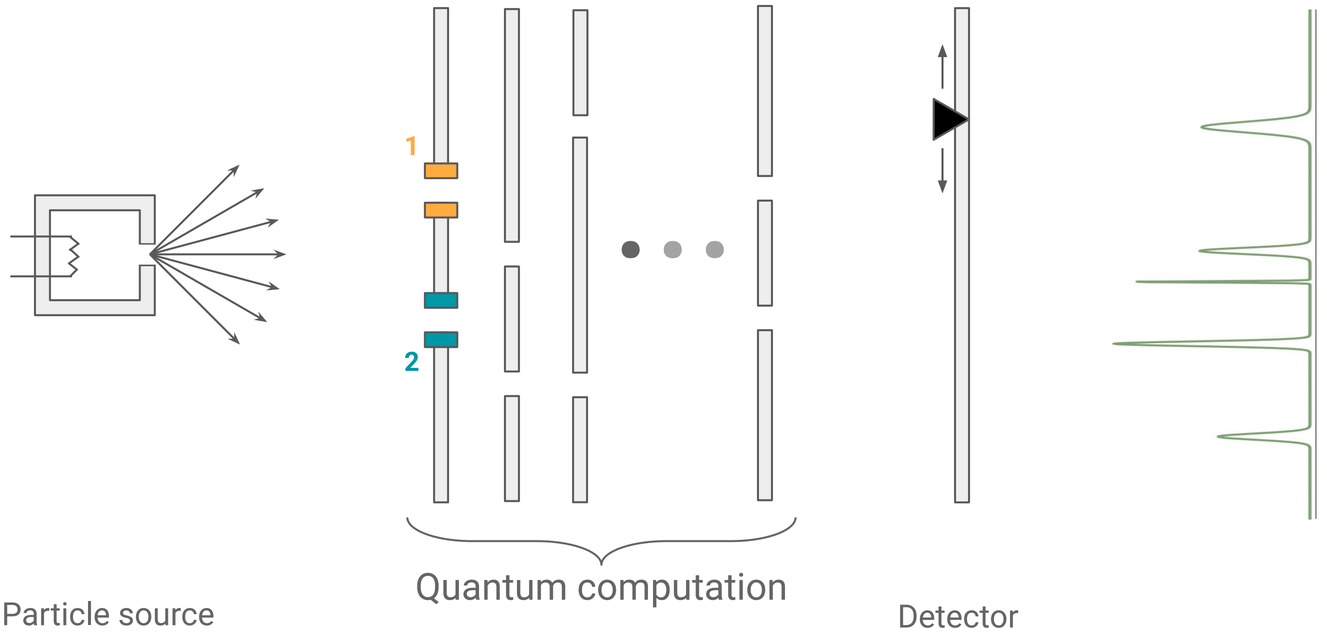
Particle source

Slits

Detector

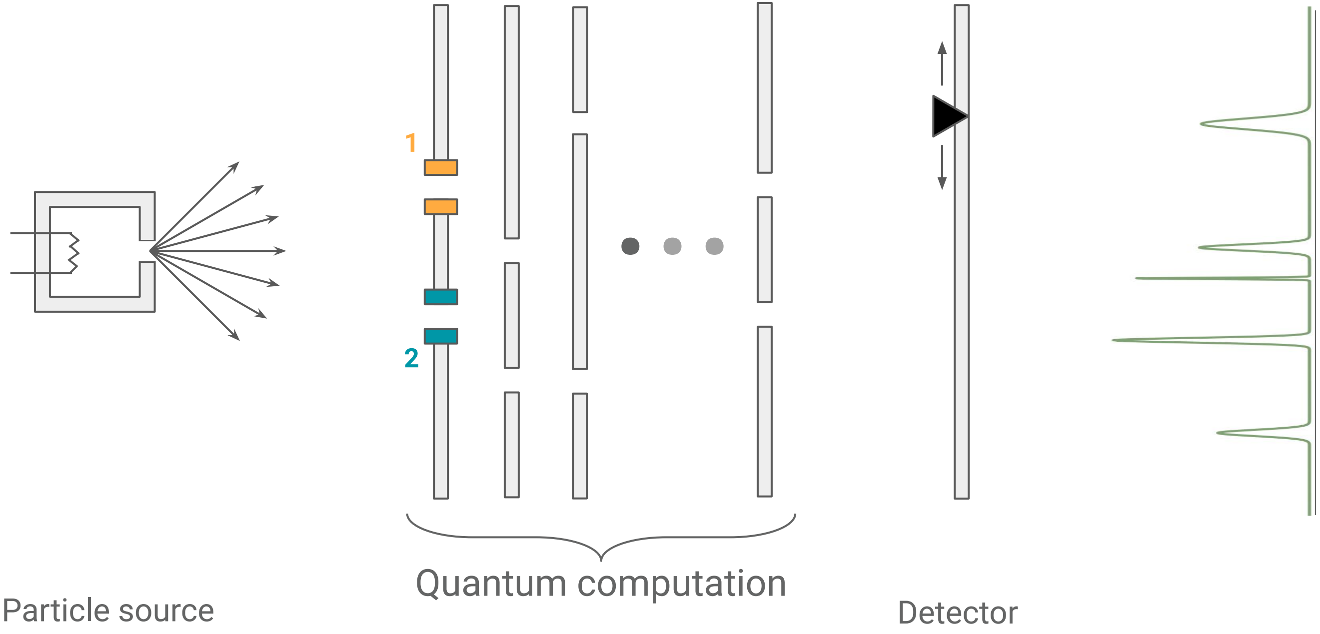
Probability

Double-slit experiments as quantum computing



Quantum computing is about creating constructive interference for correct answer(s)

Double-slit experiments as quantum computing



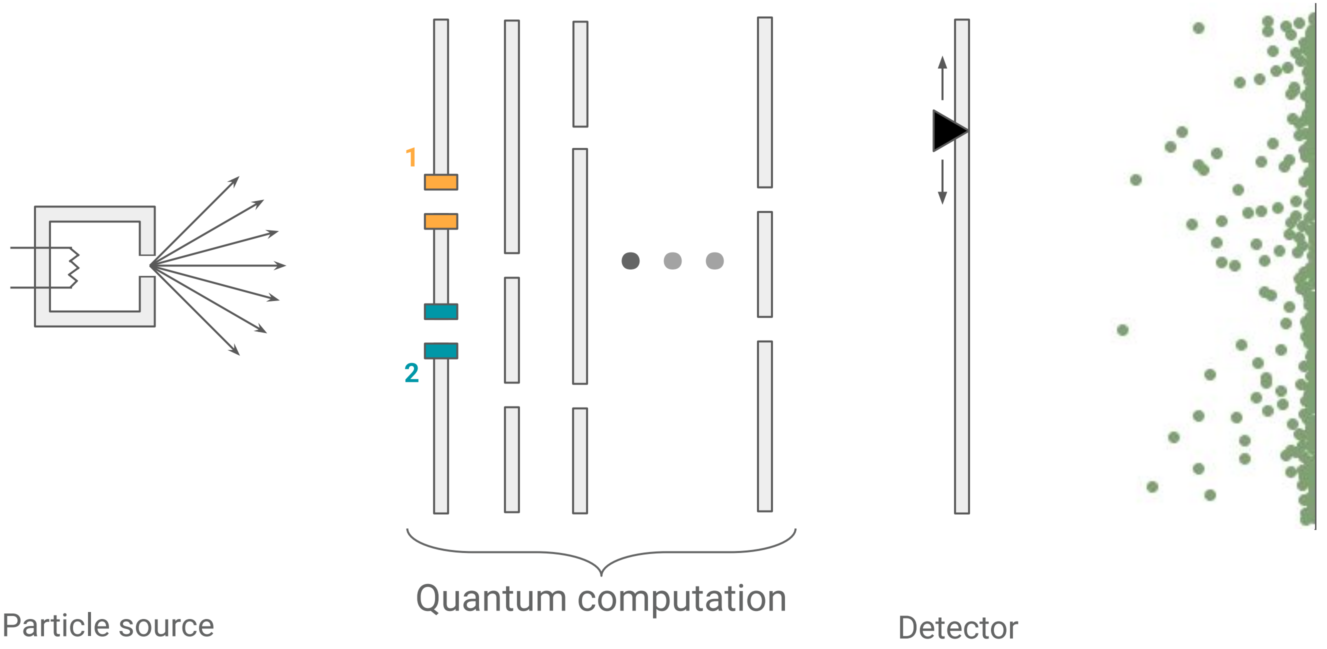
Quantum computing is about engineering constructive interference for correct answer(s)

$$\cdot|\phi_1||\phi_2|\left[e^{i(\theta_1-\theta_2)}+e^{-i(\theta_1-\theta_2)}\right]$$

Congratulations! You have just learned quantum computing!

Random double-slit experiments as random quantum operations

Randomly separated walls with randomly placed slits

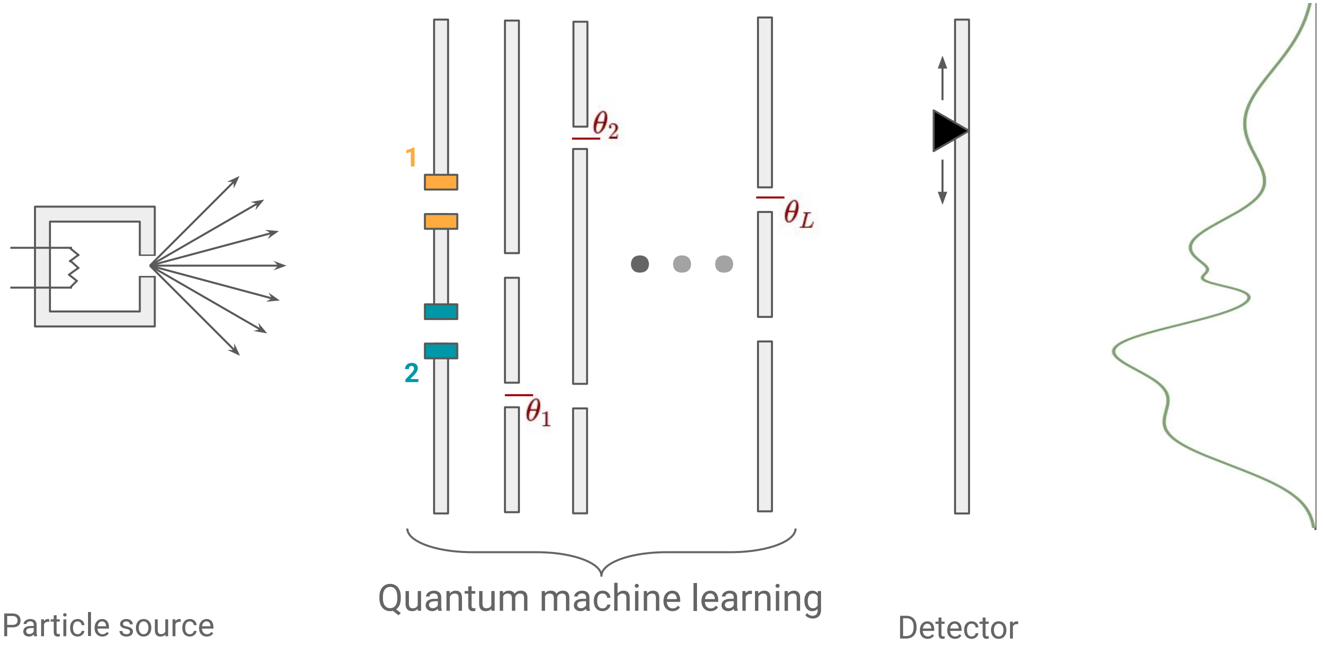


A distribution that is exponentially hard to sample classically

This particular implementation does not lead to quantum supremacy, as it is not scalable !

Iterative double-slit experiments as quantum ML

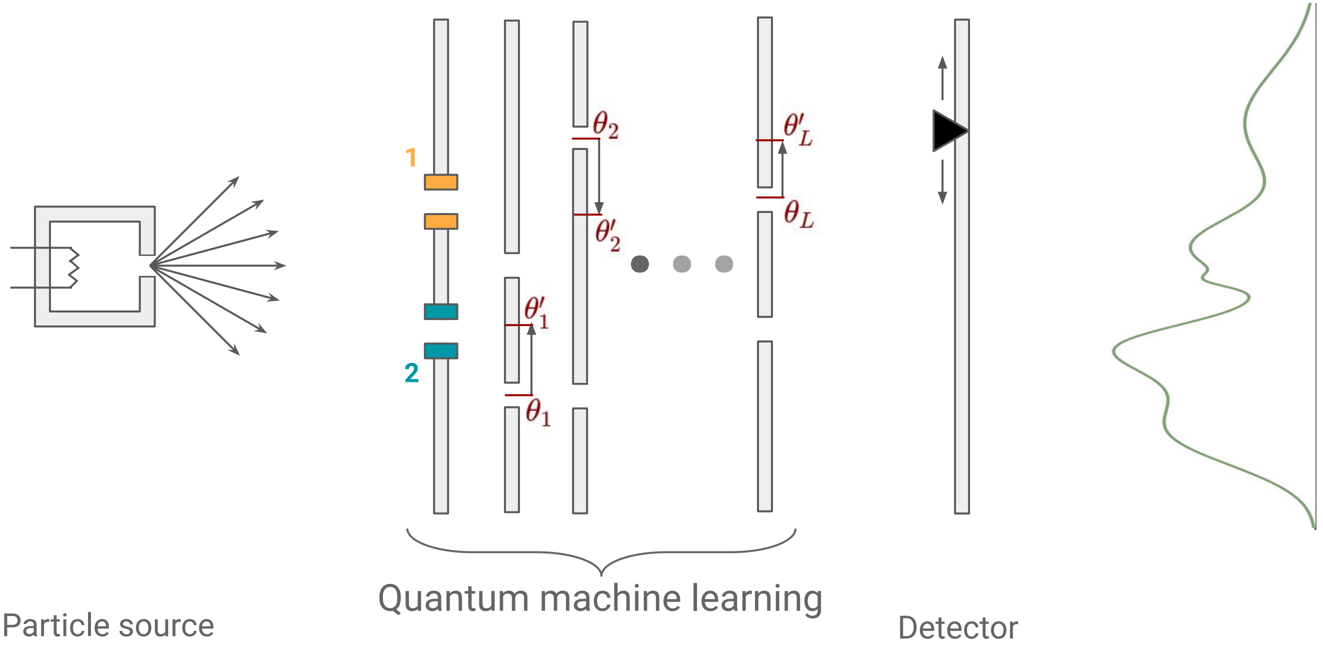
Start with randomly separated walls with randomly placed slits, iteratively change to get the correct outputs



Quantum machine learning is about iterative creation of constructive interference for correct answers

Iterative double-slit experiments as quantum ML

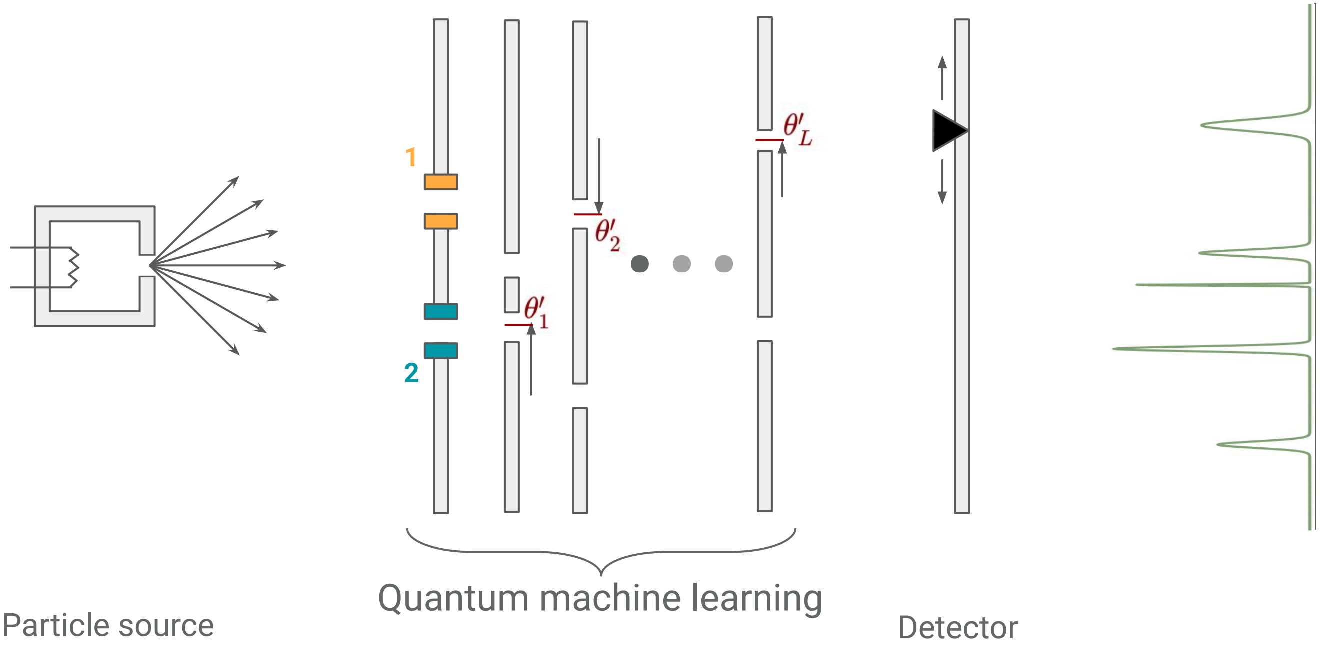
Start with randomly separated walls with randomly placed slits, iteratively change to get the correct outputs



Quantum machine learning is about iterative creation of constructive interference for correct answers

Iterative double-slit experiments as quantum ML

Start with randomly separated walls with randomly placed slits, iteratively change to get the correct outputs

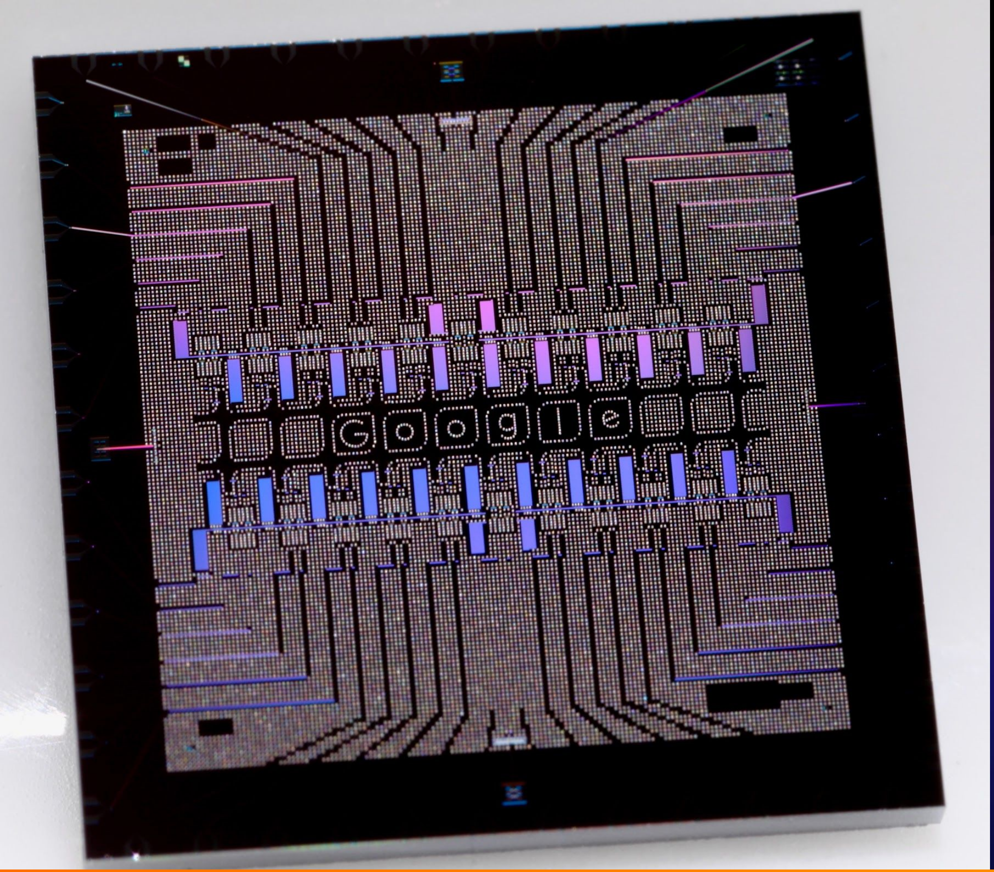


Quantum machine learning is about iterative creation of constructive interference for correct answers

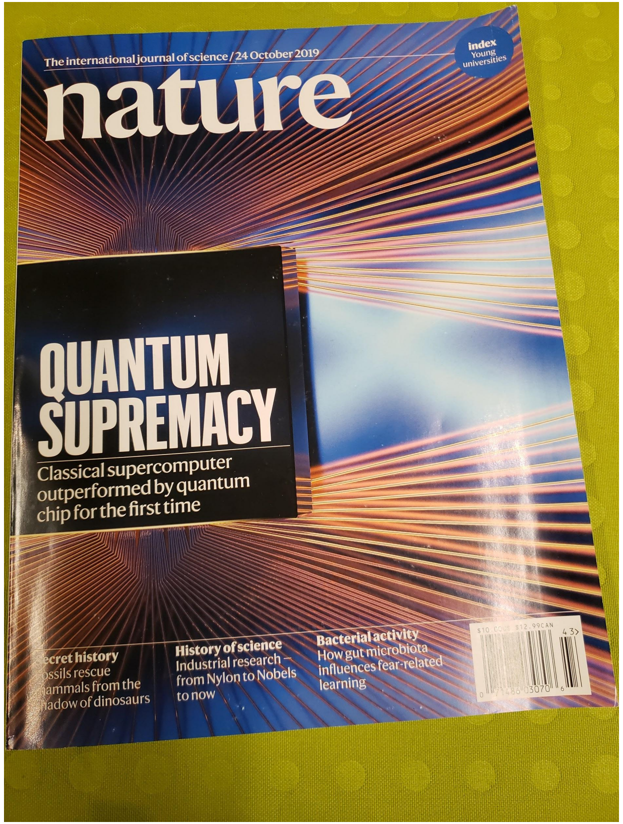
You have just learned what is quantum machine learning!

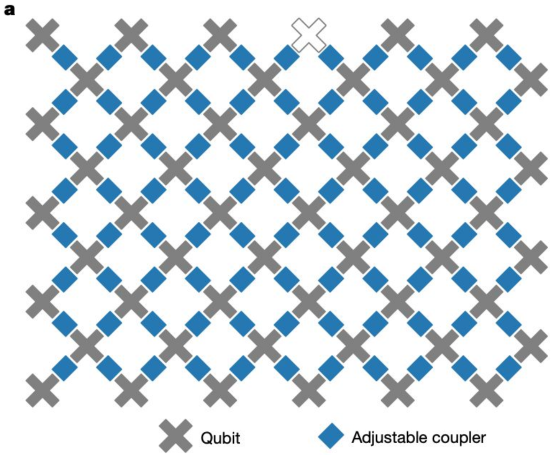
The rest of talk are just details for scalable implementations!

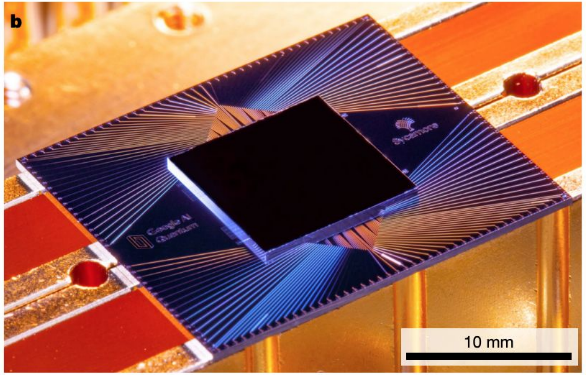
Google uses superconducting qubits



Google Sycamore processor

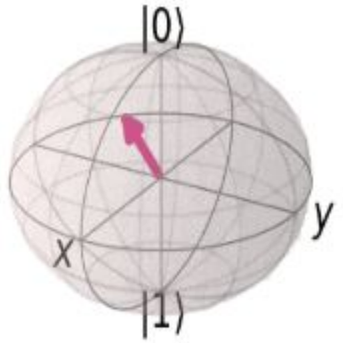






Quantum Bits (Qubits)

Classical Bit: Always has a value of 0 or 1
A bit can be copied,
Doesn't change if measured,
Measuring a bit doesn't affect other unmeasured bits.

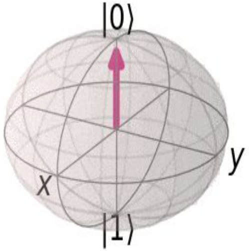


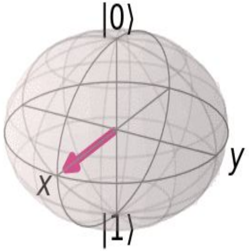
Quantum Bit (qubit): **None of the above holds in general!**

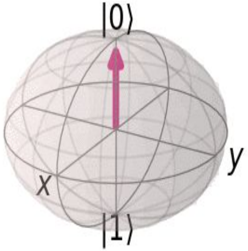
- How should we manipulate quantum information?
- How can we achieve universal quantum computation?

Single Qubit Gates

$-\boxed{Y}-$ + $-\boxed{Z}-$ = Arbitrary single-qubit rotations

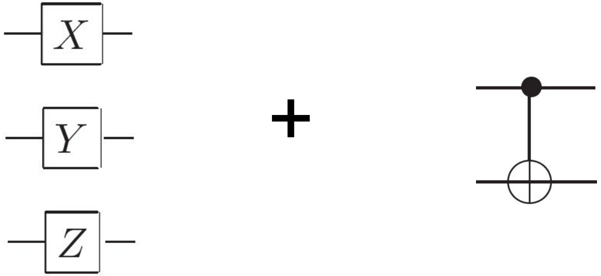






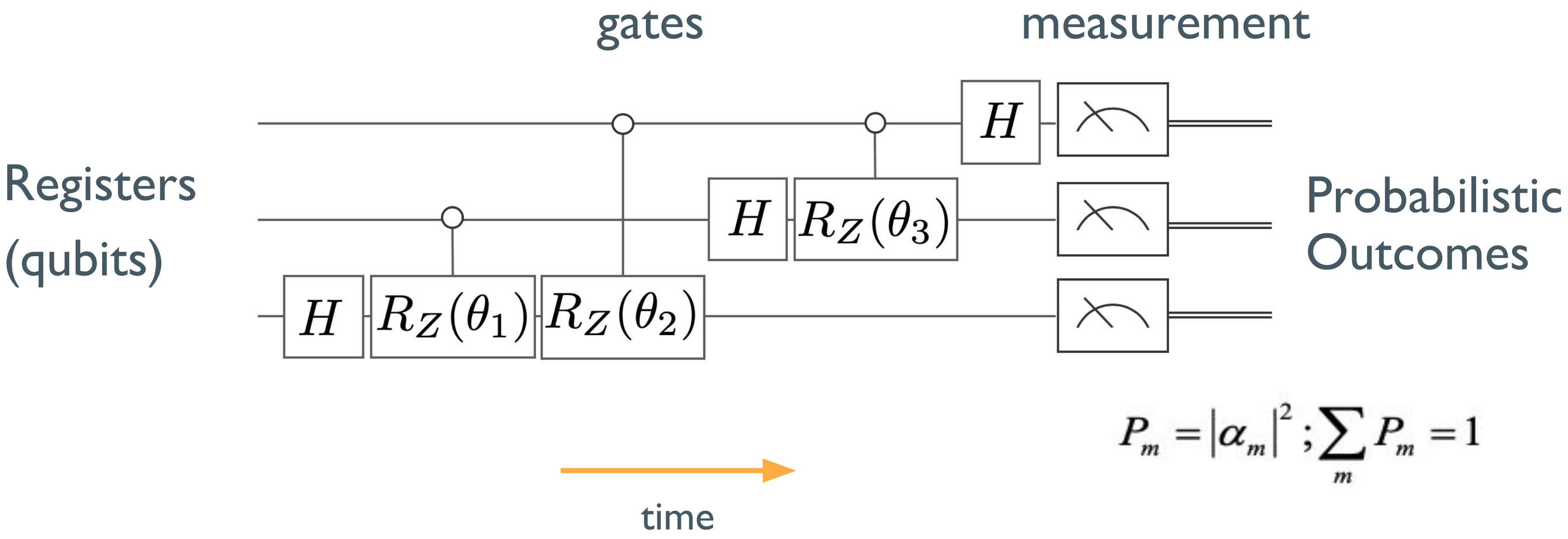
“Arbitrary” or “Universal” Quantum Computation

1- Ability to perform arbitrary single qubit rotations
2- Any nontrivial two-qubit rotations



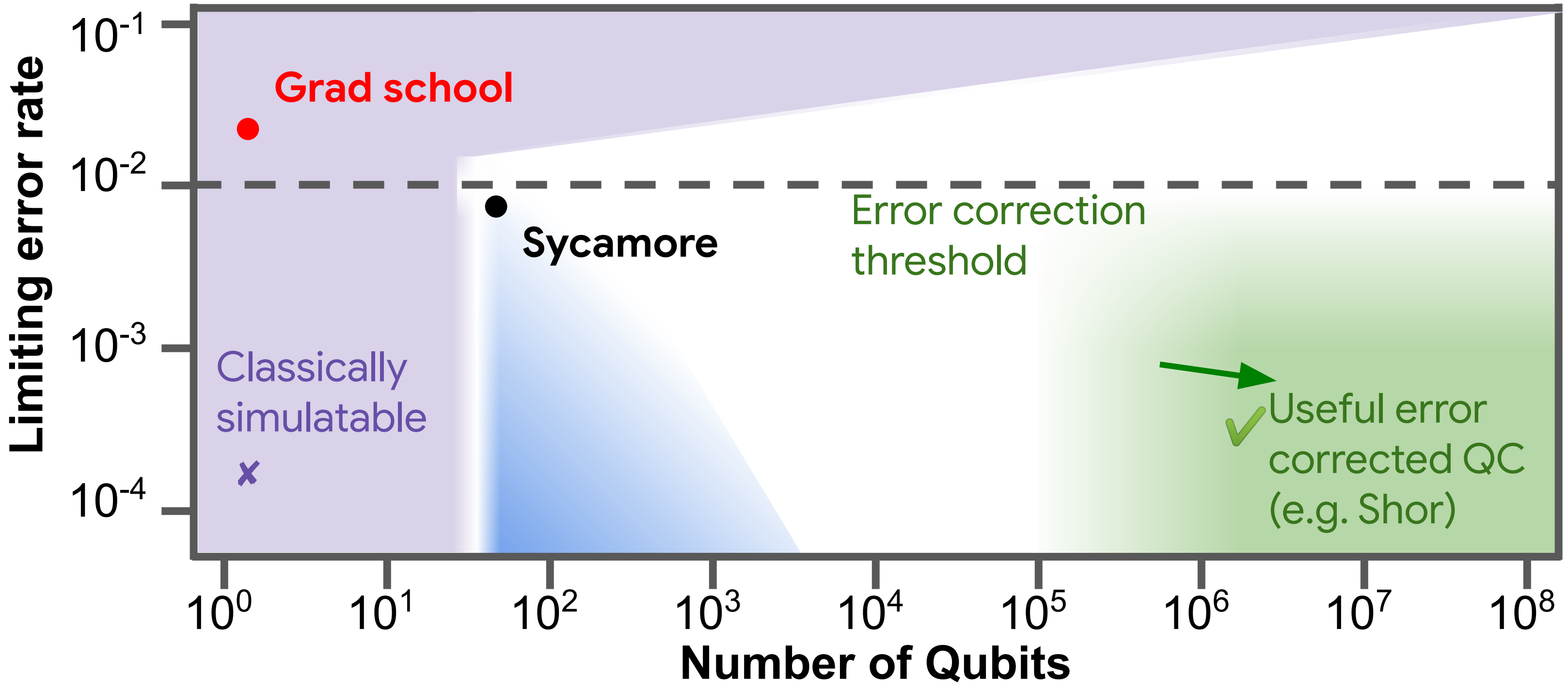
Understanding quantum circuits

Quantum circuit: sequence of evolutions of a quantum state



$$P_m = |\alpha_m|^2 ; \sum_m P_m = 1$$

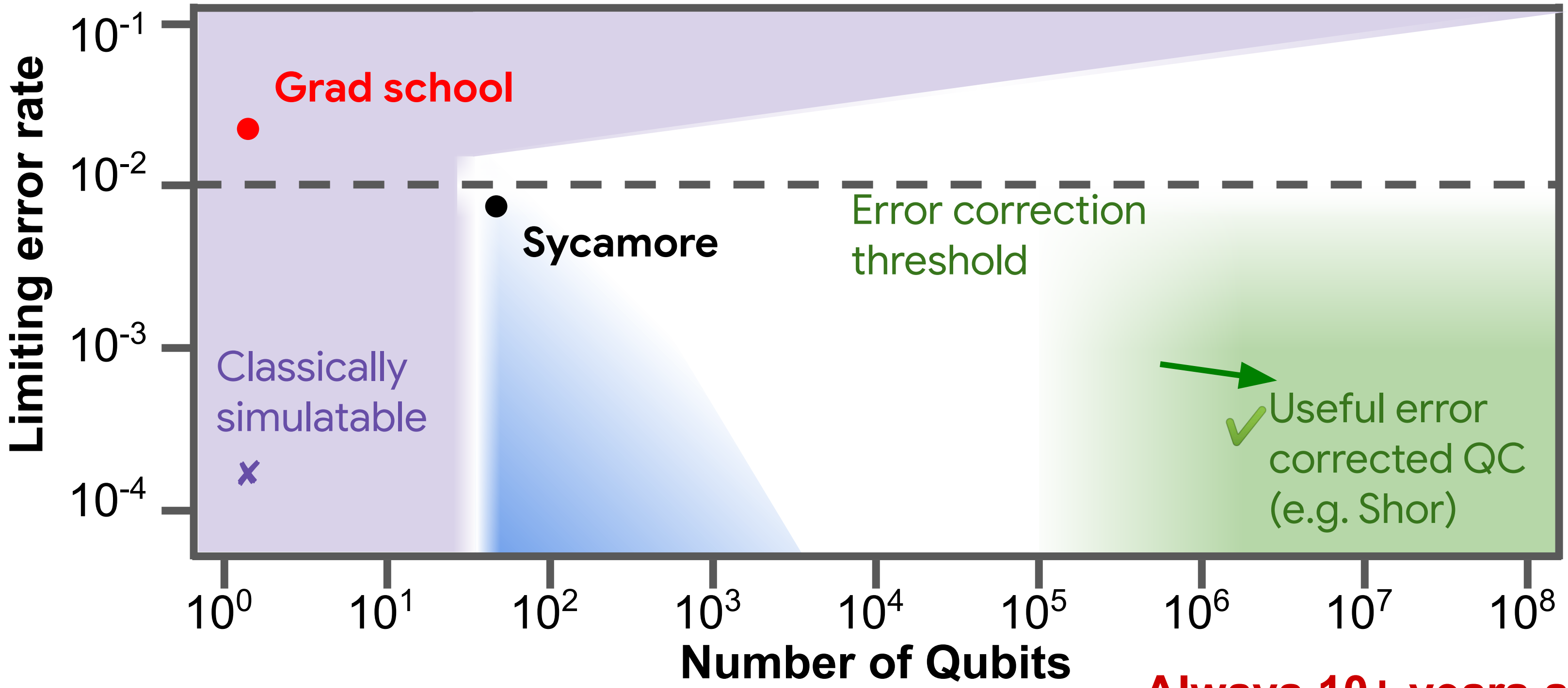
Near-term Quantum Computing Landscape



Google AI Quantum

TensorFlow Quantum

Near-term Quantum Computing Landscape

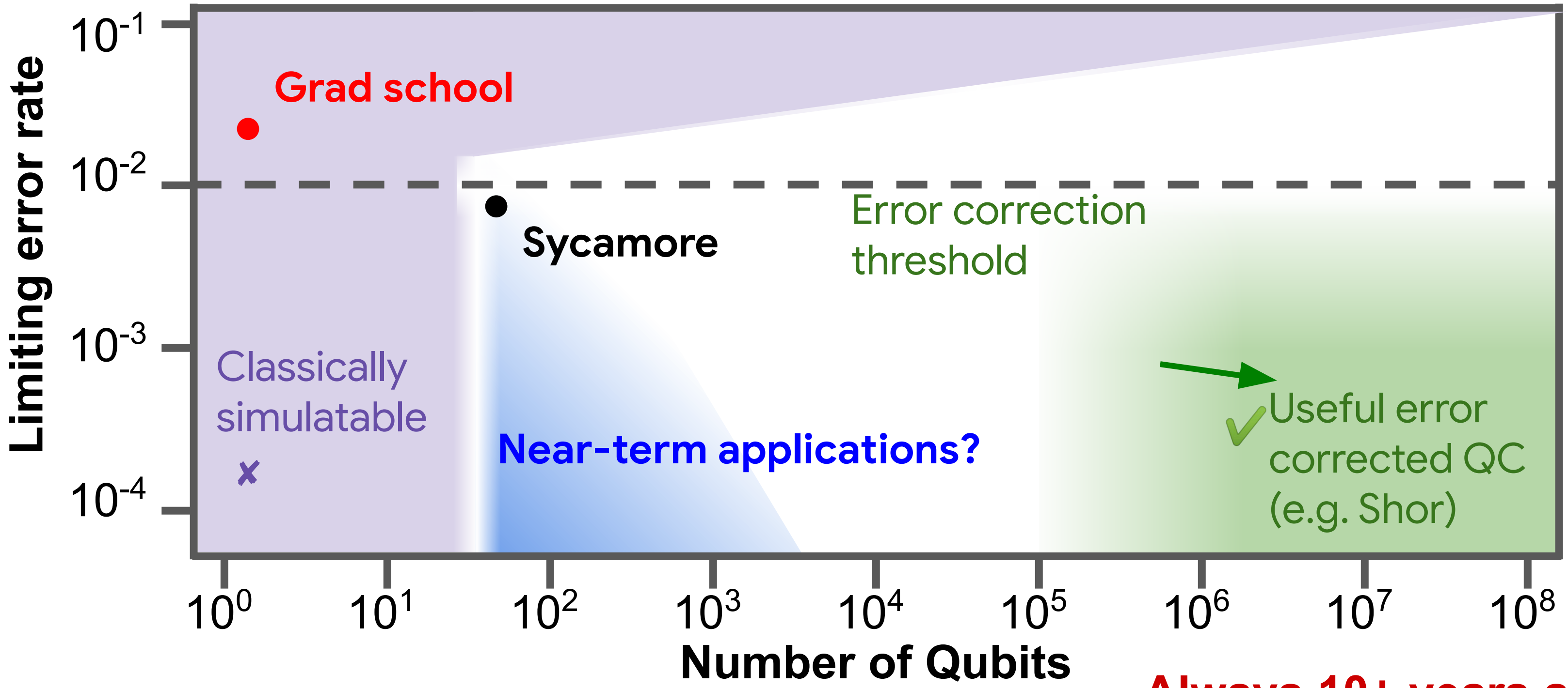


Always 10+ years away!?

Google AI Quantum

TensorFlow Quantum

Near-term Quantum Computing Landscape

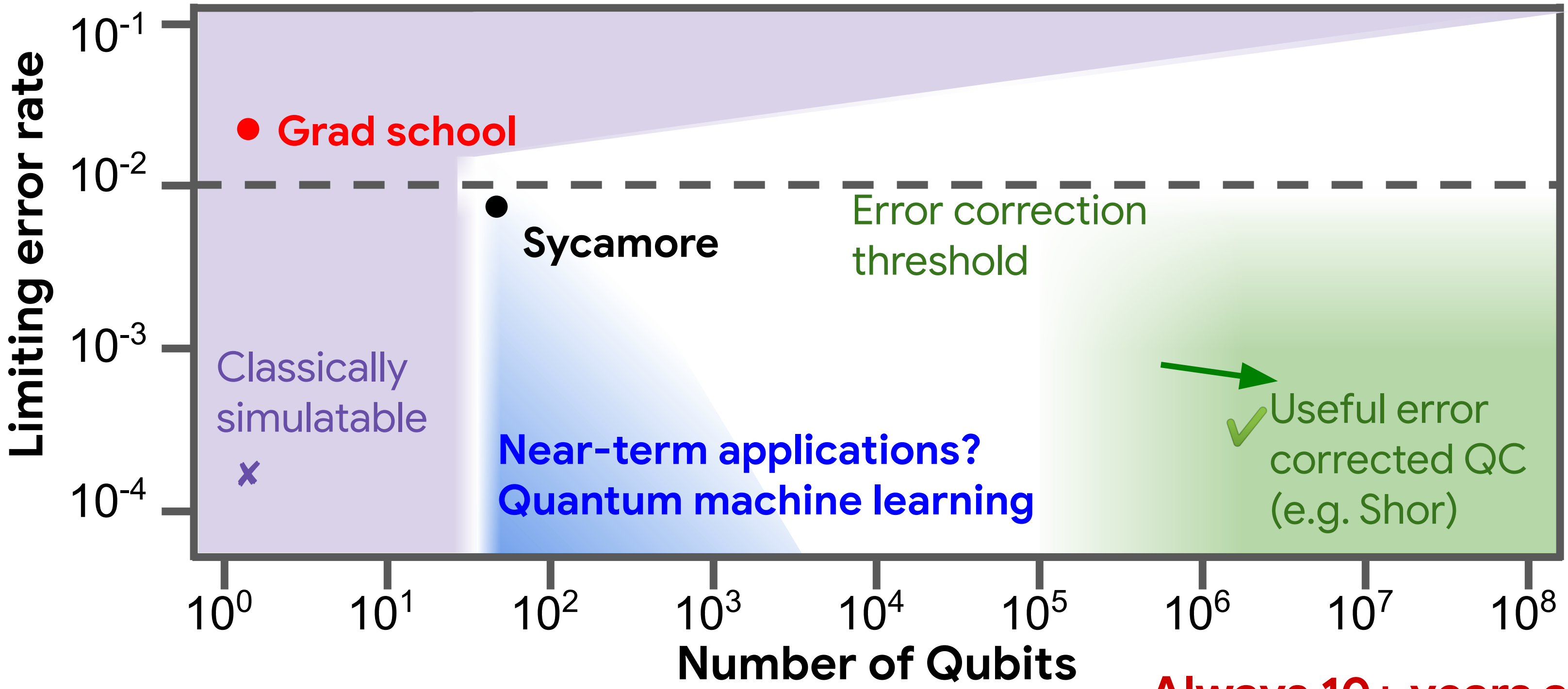


Always 10+ years away!?

Google AI Quantum

TensorFlow Quantum

Near-term Quantum Computing Landscape



Always 10+ years away!?

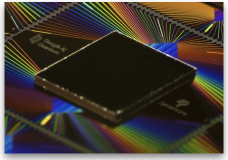
Google AI Quantum

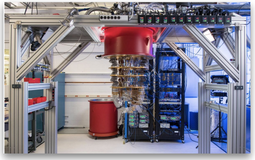
TensorFlow Quantum

TensorFlow Quantum marks another important milestone for Quantum at Google

Google AI Quantum Launch Roadmap

2018	Oct 2019	Mar 2020	2020
Cirq and OpenFermion Launch	Quantum Supremacy Announcement	TF Quantum Launch	Quantum Engine Early Acces Partners (EAP)





TensorFlow

Main Objectives for TF Quantum

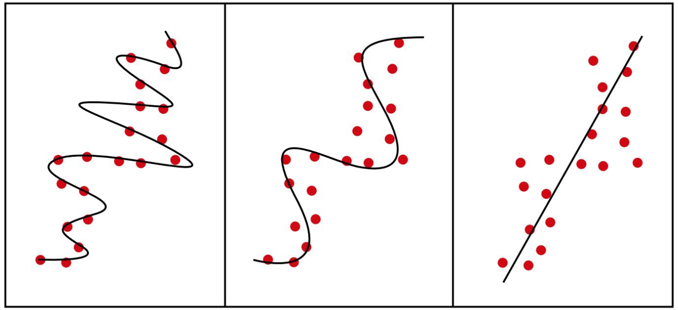
- A software framework for hybrid quantum-classical machine learning under TensorFlow and Cirq

- Fast prototyping, training, inference, and testing of quantum models over quantum data

- Discovering new quantum algorithms for NISQ devices or error-corrected quantum computers

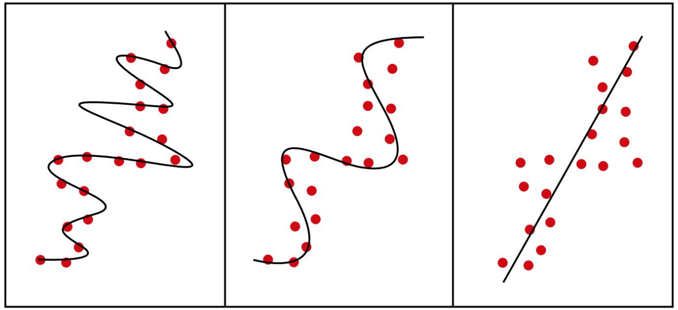
Three main features of machine learning

- **Representation power (efficiently representing data)**
- **Optimization power (efficient iterations)**
- **Generalization power (minimizing the error in prediction)**



Three main features of machine learning

- Representation power (efficiently representing data)
- Optimization power (efficient iterations)
- Generalization power (minimizing the error in prediction)



Can we improve any (combinations) of these features quantum mechanically?

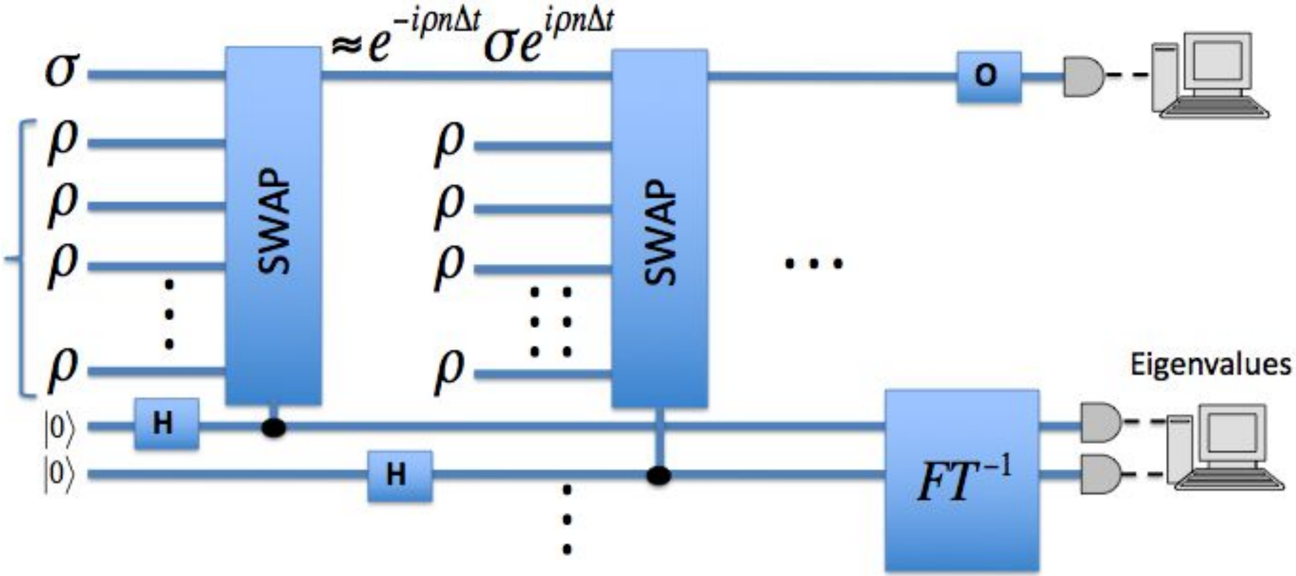
Types of Quantum machine learning

- Quantizing classical neural networks (pre-history 90s)

- Accelerated linear algebra on quantum computers (2013)
  - First generation of QML
  - Only applies to fault-tolerant quantum computers

- Low-depth Quantum circuit learning or “Quantum Neural Networks”
  - Second generation of QML
  - Applicable to Noisy-Intermediate Quantum (NISQ) processors

qPCA: Efficient diagonalization of low-rank density matrices



Lloyd, Mohseni, Rebnetrost,
qK-mean, arXiv:1307.0411
qPCA, Nature Physics (2014), qSVM,
PRL (2014)

$$\sigma \otimes \rho^n \otimes |0\rangle^m \rightarrow \sum_i \psi_i |V_i\rangle |\lambda_i\rangle$$

Eigenvectors Eigenvalues

$$n = O(\varepsilon^{-3} \log d)$$

Google AI
Quantum

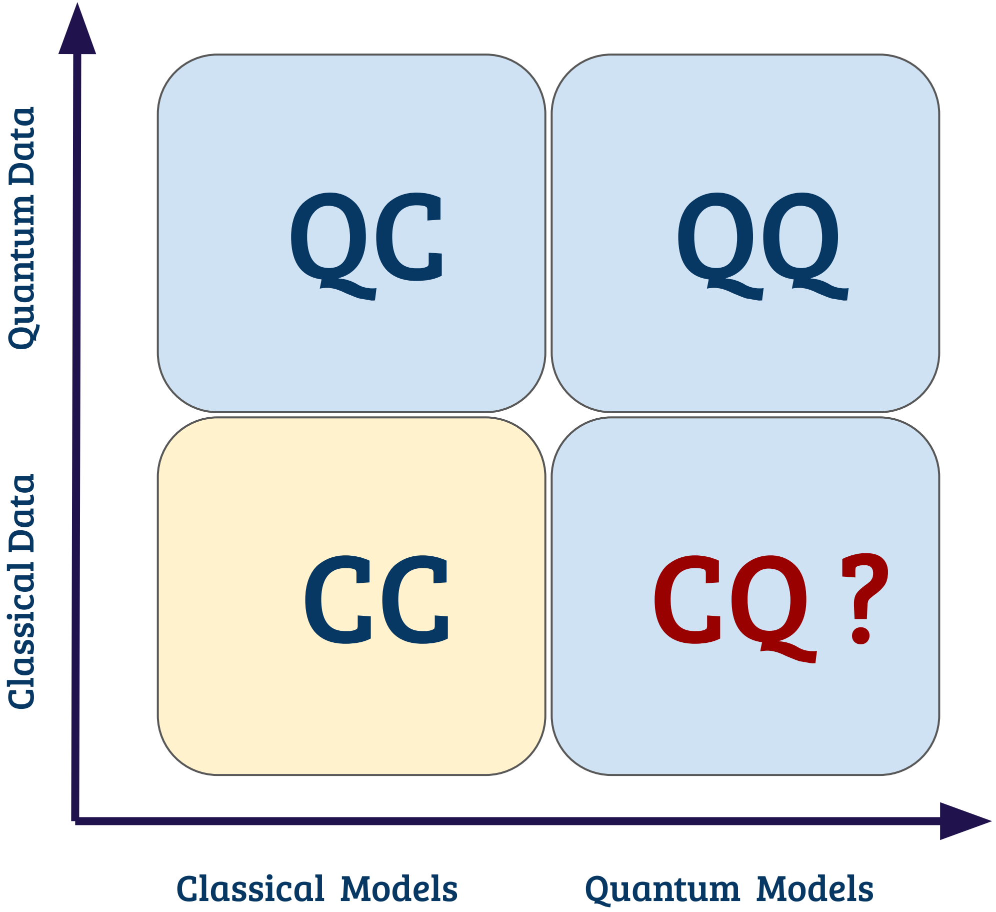
Types of Quantum machine learning

- Quantizing classical neural networks (pre-history 90s)

- Accelerated linear algebra on quantum computers (2013)
  - First generation of QML
  - Only applies to fault-tolerant quantum computers

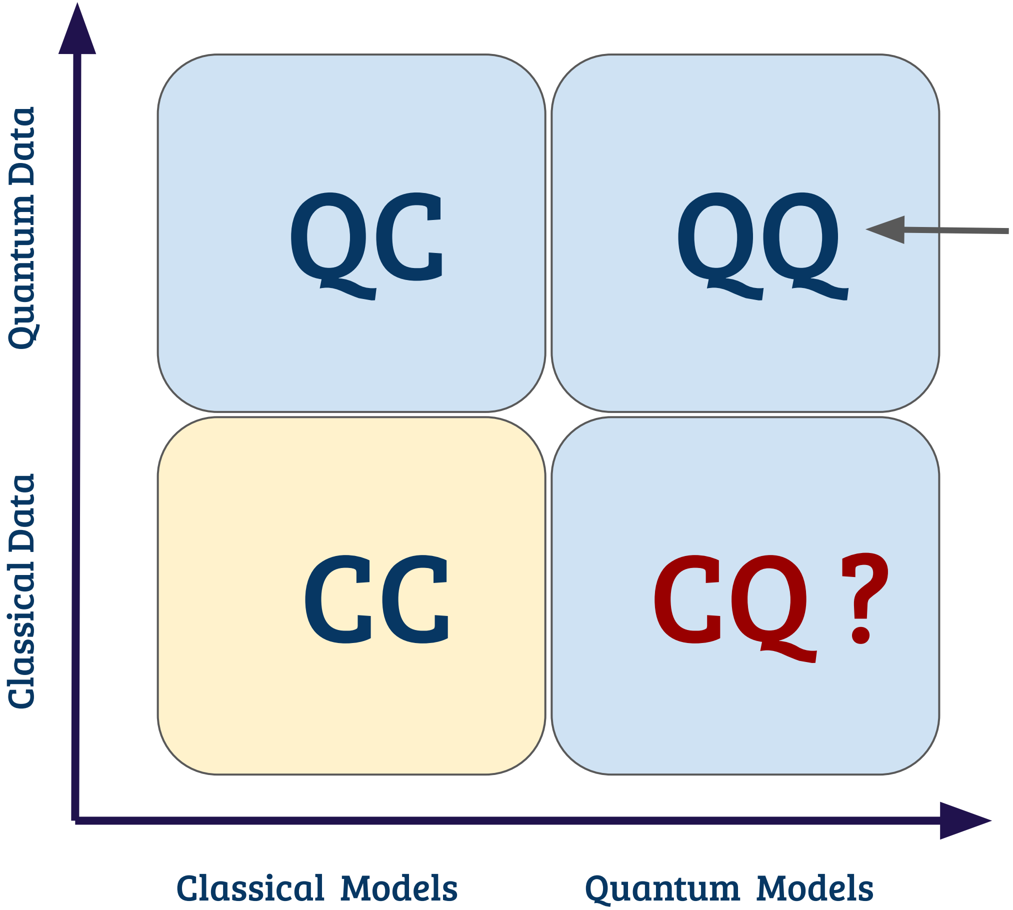
- **Low-depth Quantum circuit learning or “Quantum Neural Networks”**
  - **Second generation of QML**
  - **Applicable to Noisy-Intermediate Quantum (NISQ) processors**

Types of Machine Learning



TensorFlow Quantum

Types of Machine Learning

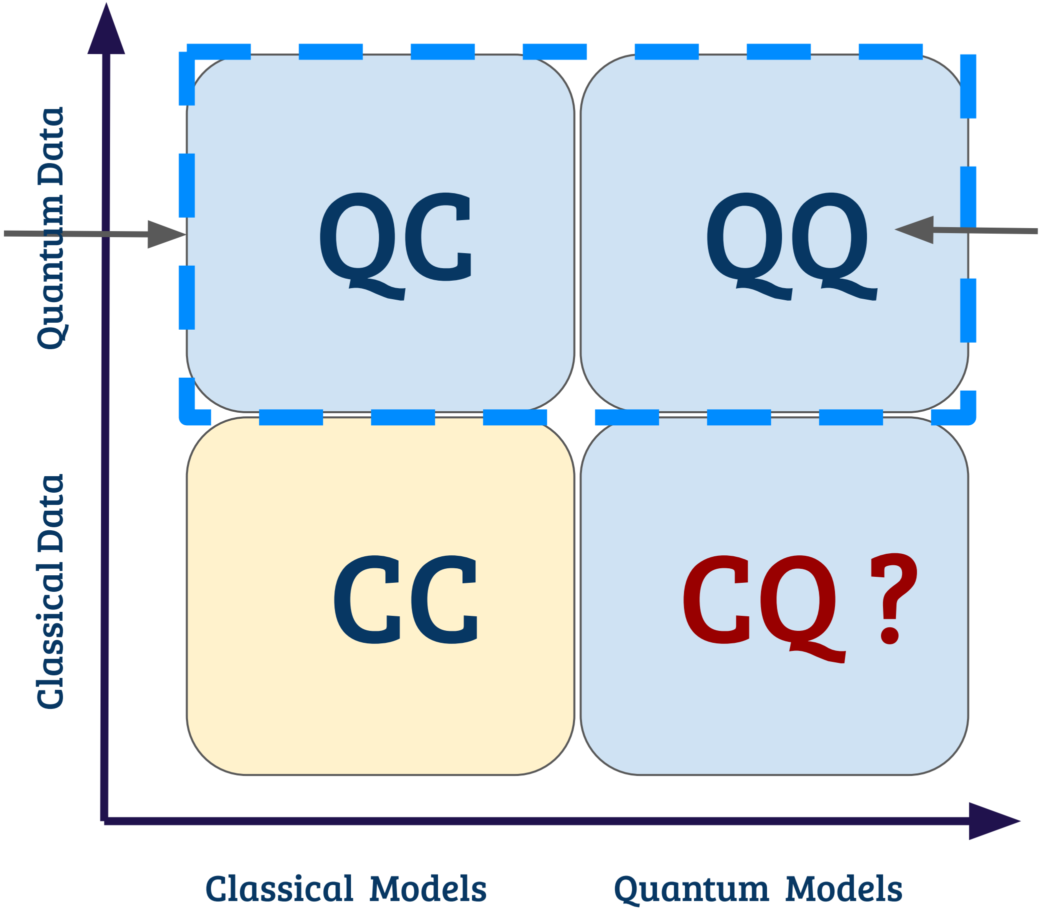


Disentangling quantum data could be exponentially hard for classical models in worse cases (e.g., near a quantum critical point)

TensorFlow Quantum

Types of Machine Learning

Hybrid classical quantum models for quantum data



Disentangling quantum data could be exponentially hard for classical models in worse cases (e.g., near a quantum critical point)

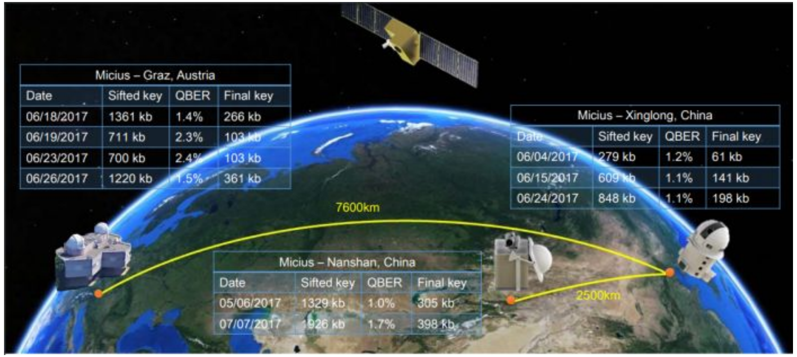
TensorFlow Quantum

Types of hybrid quantum-classical computing

- Classical control of quantum circuits (all quantum computers are hybrid!)
- Classical preprocessing or post-processing
- Classical algorithms with quantum subroutines
- Classical variational outer loops for optimizing quantum circuits
- **Classical-Quantum co-processors / hybrid quantum-classical models**

Different Kinds of Quantum Data

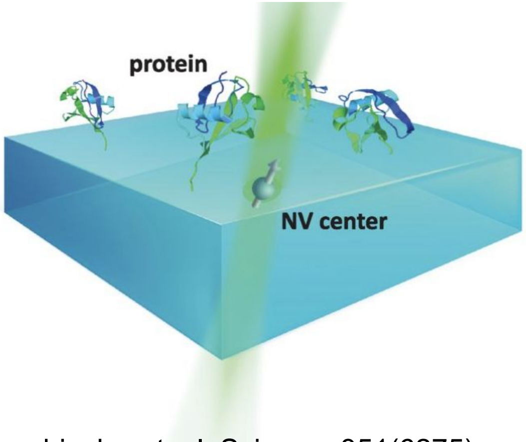
- **Data in quantum communication networks**:
  - quantum key distribution
  - quantum repeaters
  - quantum receivers



The Micius satellite, by MIT Technology review.

Different Kinds of Quantum Data

- **Data in quantum metrology:**
  - nuclear magnetic resonance detection, NV centers, Rydberg atoms,...
  - quantum sensing
  - quantum imaging



I. Lovchinsky, et. al. Science, 351(6275), pp.836-841.

Different Kinds of Quantum Data

- **Control and calibration of quantum processors:**
  - Quantum measurement and parameter estimation
  - Quantum control and calibration
  - Quantum tomography

Google AI Quantum

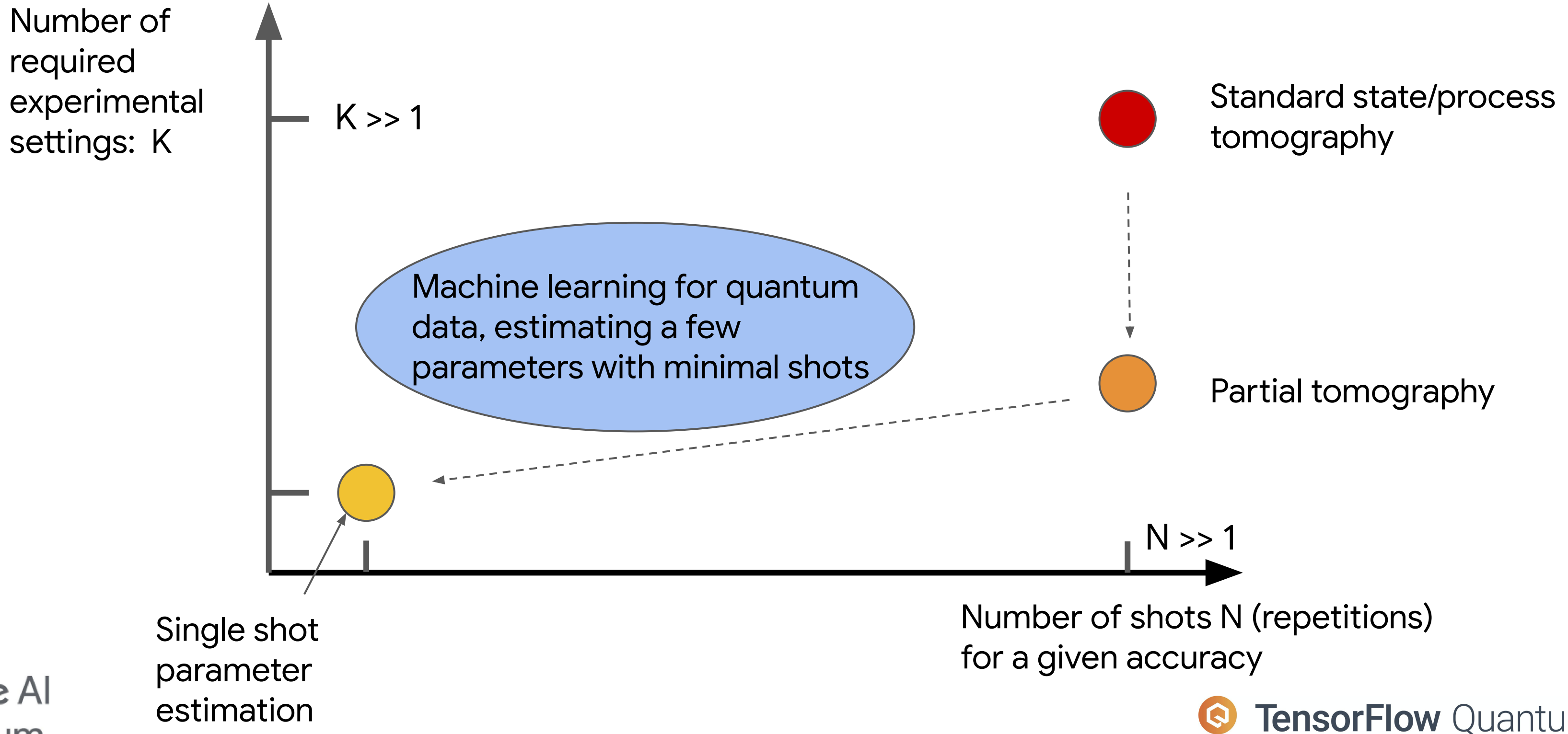
TensorFlow Quantum

Different Kinds of Quantum Data

- **Output states of quantum computers:**
  - quantum verification, quantum (nonlocal) inference
  - Simulation of chemical systems, material science, pharmaceutical
  - Simulation of quantum matter (classification and generative models for quantum many-body systems, quantum critical systems, e.g., high T superconductivity).
  - quantum algorithm discovery

Examples of QML for quantum data: parameter estimation

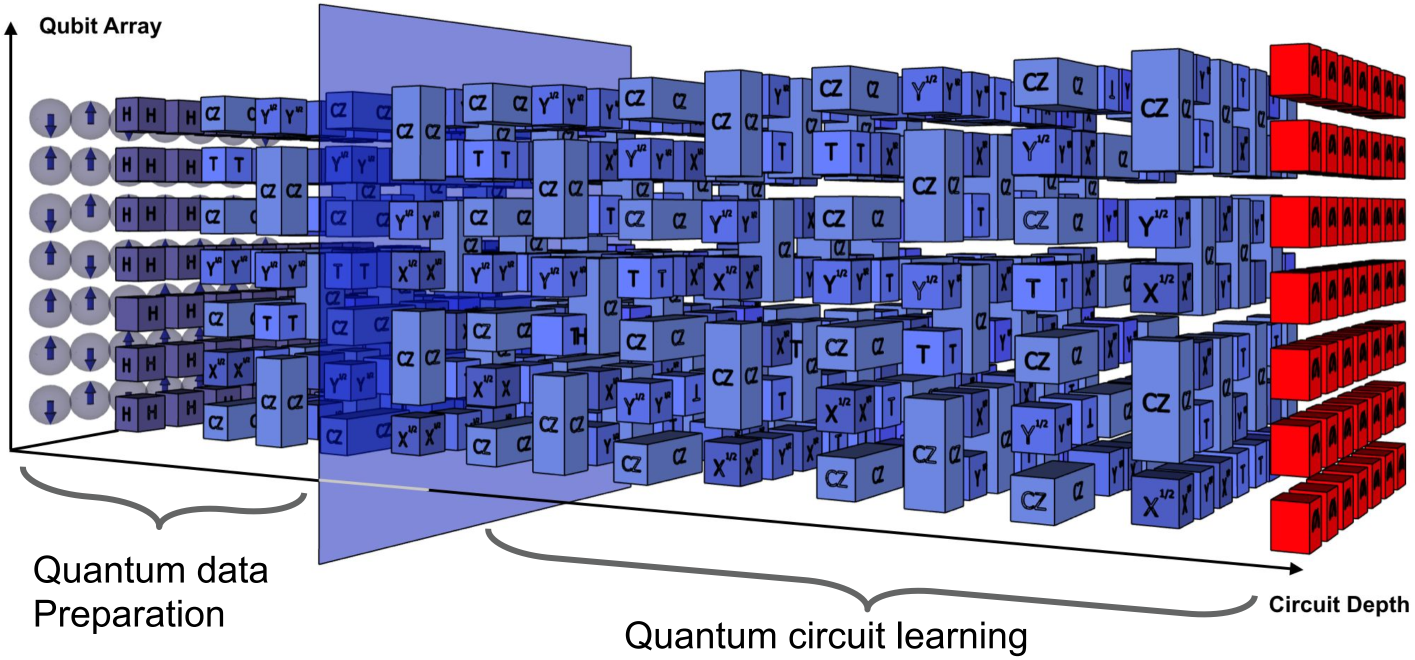
state discrimination, error-detection, state/procss tomography



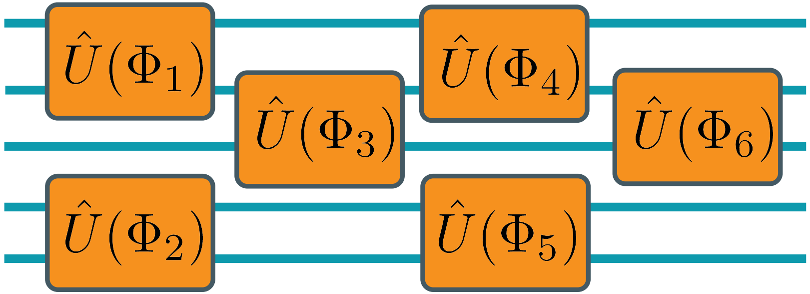
Google AI Quantum

TensorFlow Quantum

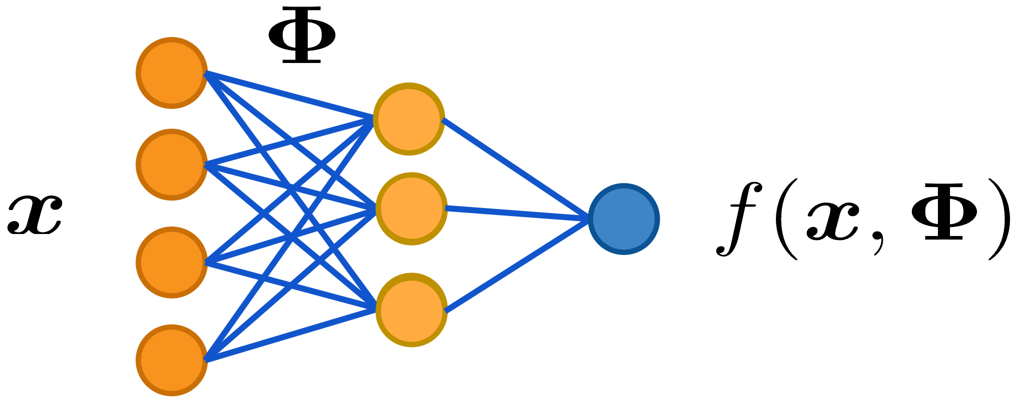
QML on finite space-time volume of parameterized quantum circuits



Parameterized Quantum Circuits



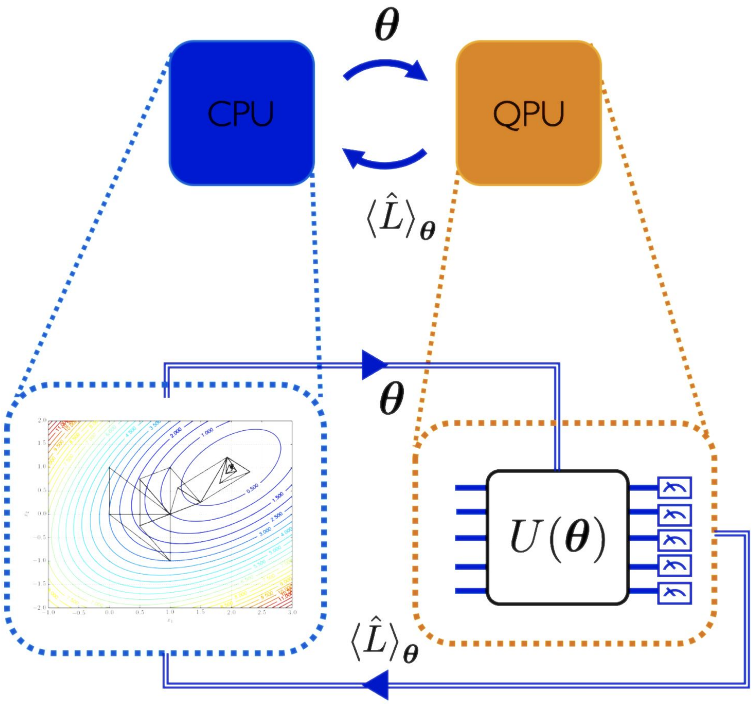
- Sequence of continuously-parameterized “rotations”
- Forms a parameterized quantum circuit, also known as a ***quantum neural network***



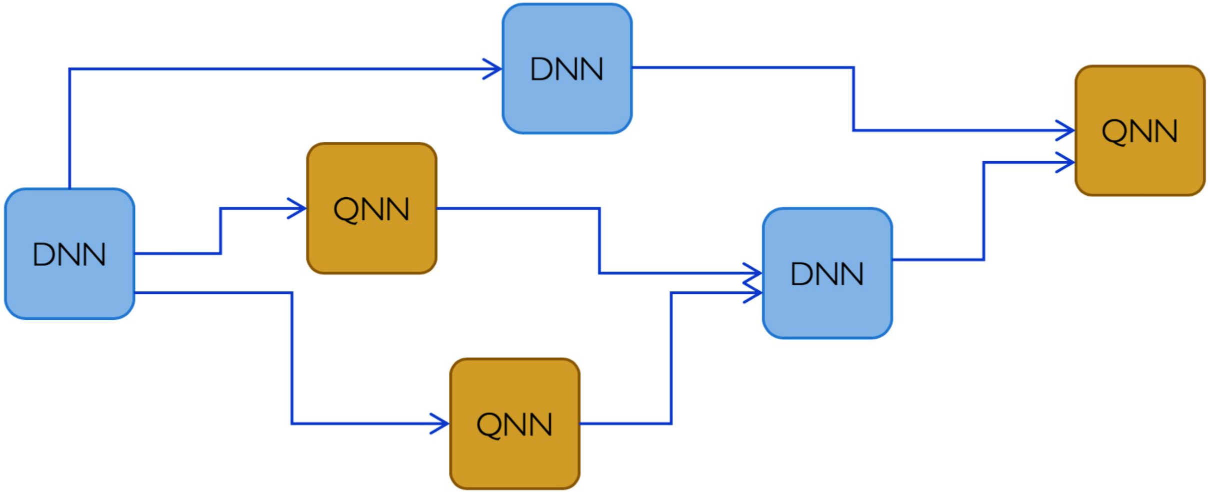
Variational Quantum Algorithms

Iterative quantum-classical optimization

- Execute parametric quantum circuit on QPU
- Measure observable expectation value <L> over multiple runs
- Relay information to classical processing unit (CPU)
- CPU optimization algorithm suggests new parameters



Hybrid quantum-classical learning



What are the existing toolboxes?

- **Cirq**
  - Quantum circuit construction and simulation language
  - Focused on NISQ devices

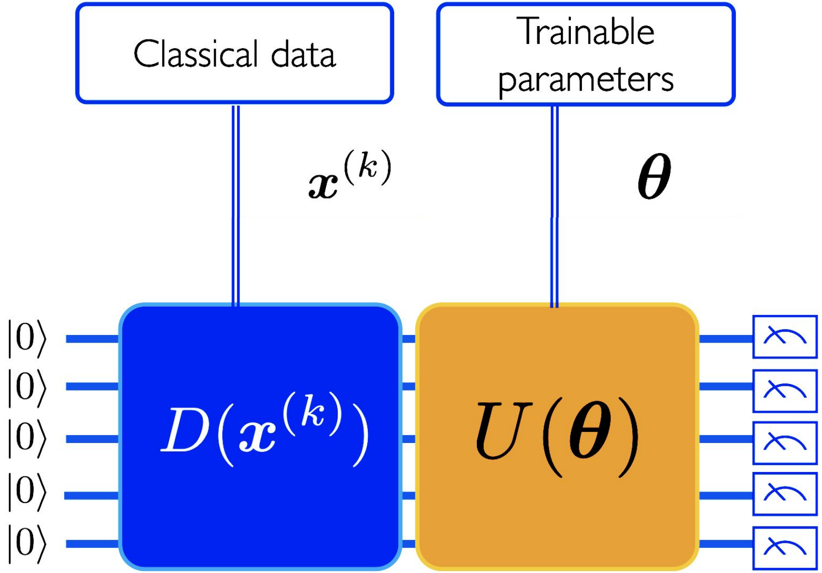
- **TensorFlow**
  - One of the most widely used machine learning platform
  - Designed for heterogeneous computation

Can we combine them?

How can build hybrid models by combining TF and Cirq?

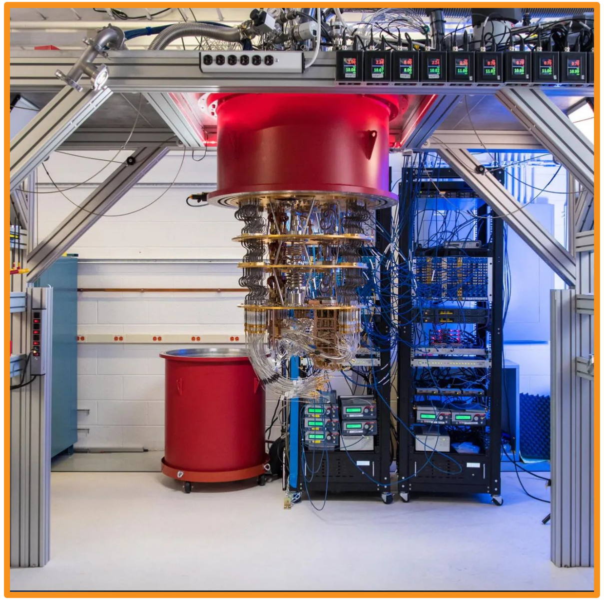
Technical Hurdle 1

- **Quantum data cannot be imported**
  - Quantum data must be prepared on the fly
  - Both data and the model are layers in the quantum circuit
  - Graph is highly dynamic



Technical Hurdle 2

- **QPU needs full quantum program for each run**
  - QPU run is a few microseconds
  - Relatively high latency CPU-QPU (ms)
  - Batches of jobs are relayed to quantum computer



Our design principles

1. **Differentiability:** Must support differentiation of quantum circuits and hybrid backpropagation.

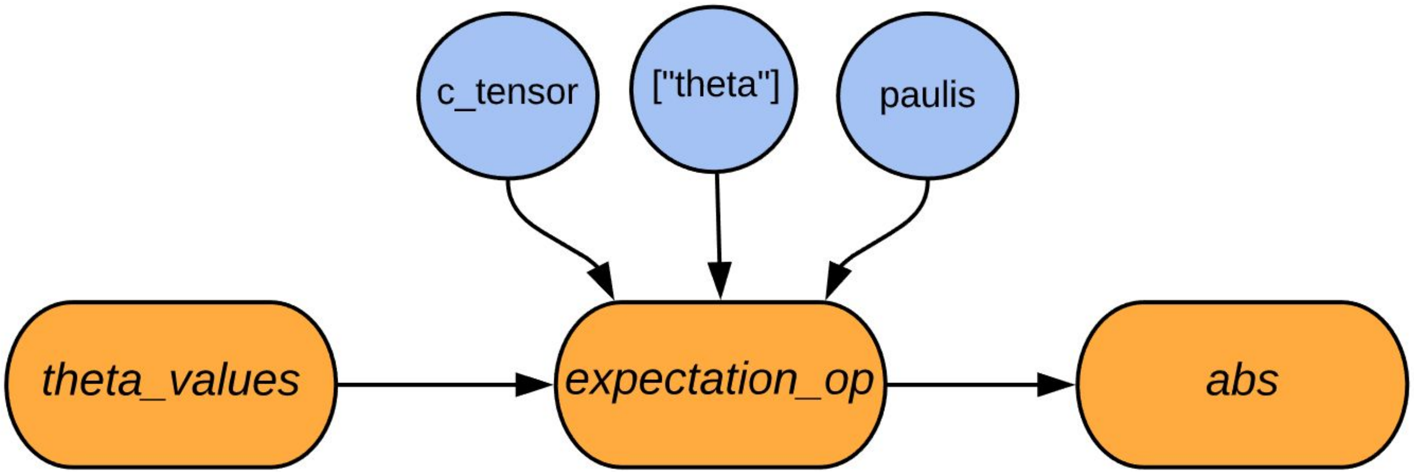
2. **Circuit Batching:** Quantum data loaded as quantum circuits, training over many different quantum circuits in parallel.

3. **Execution Backend Agnostic**: Switch from a simulator to a real device easily with few changes.

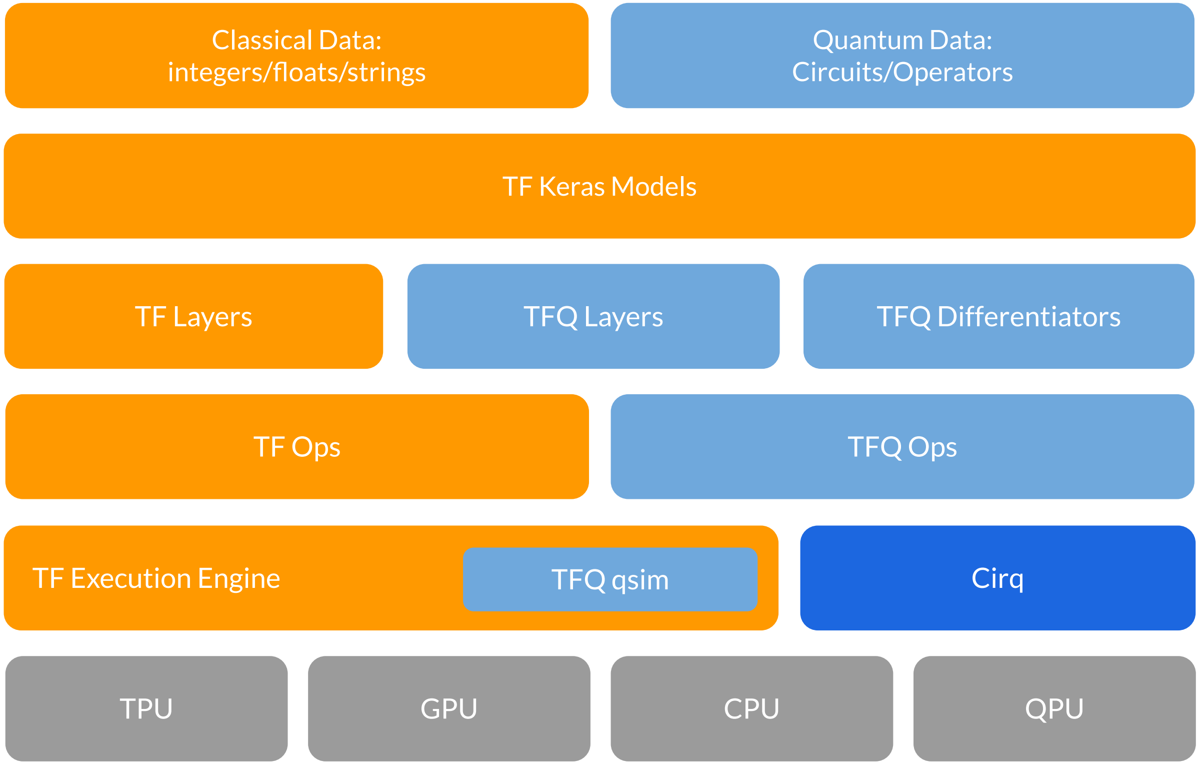
4. **Minimalism:** A bridge between Cirq and TF; does not require users to re-learn how interface with quantum computers or solve problems using machine learning.

Software architecture

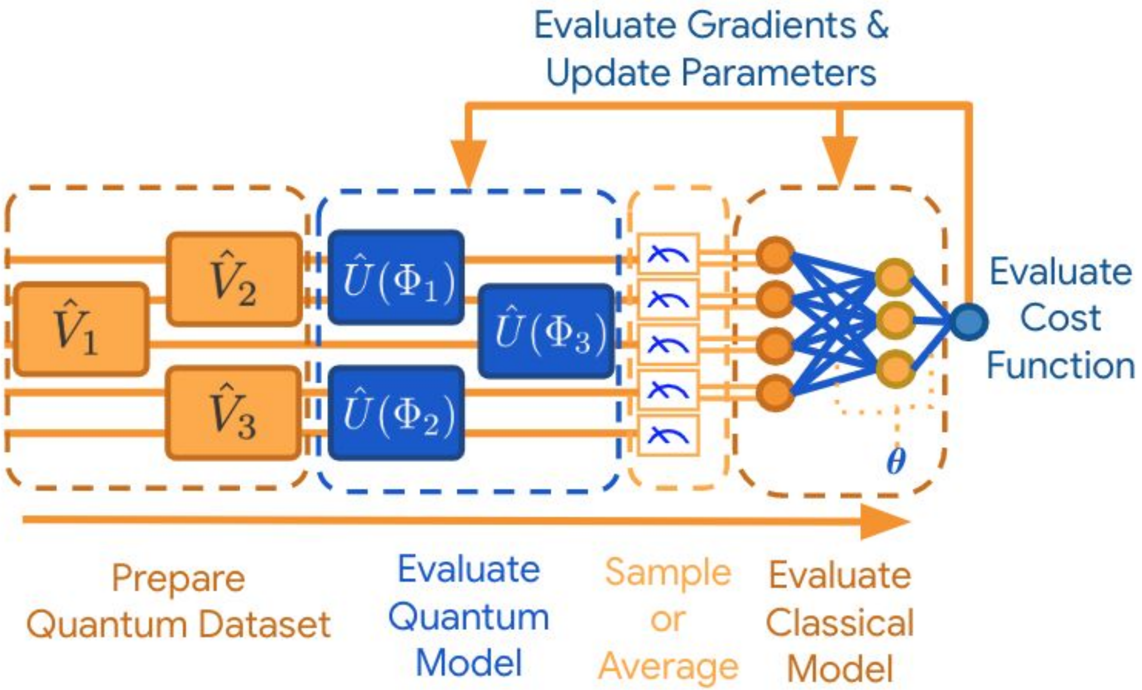
- Circuits are **TENSORS**, use Cirq constructs to generate these tensors
- Converting circuits to classical data (aka running or simulating them) can be done by **OPs**



Software stack



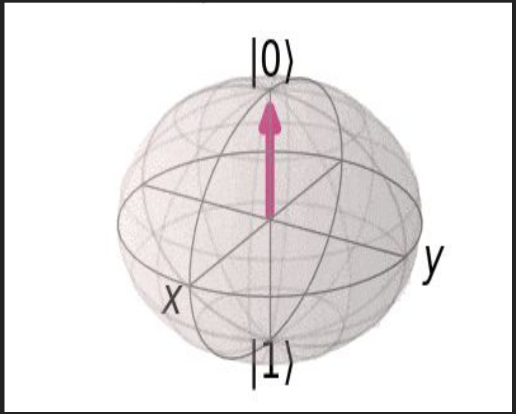
TFQ pipeline for a hybrid discriminative model



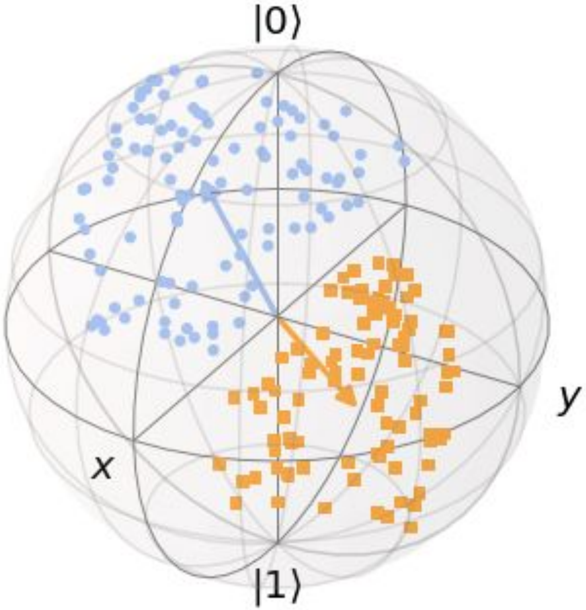
Hello Many-Worlds

You can use TFQ to perform a ‘hello world’-type task; Binary classification of quantum states for a single qubit

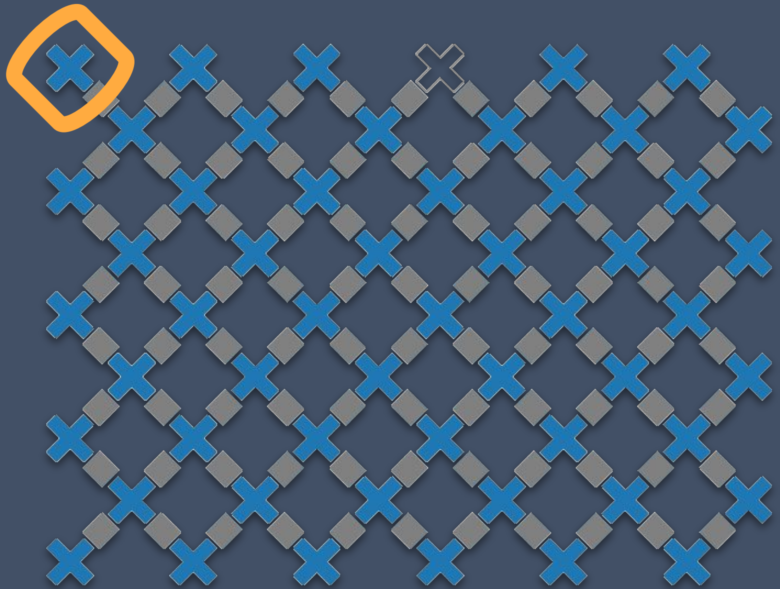
arXiv:2003.02989



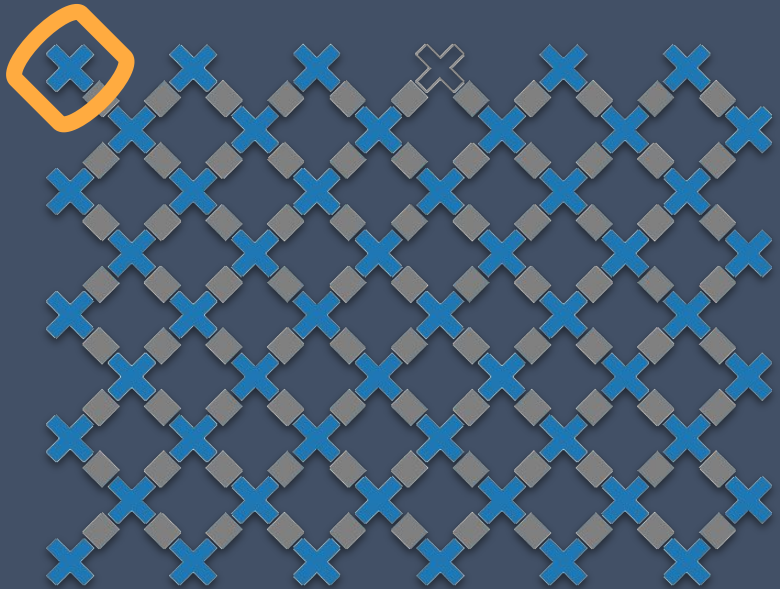
Quantum dataset for a single qubit



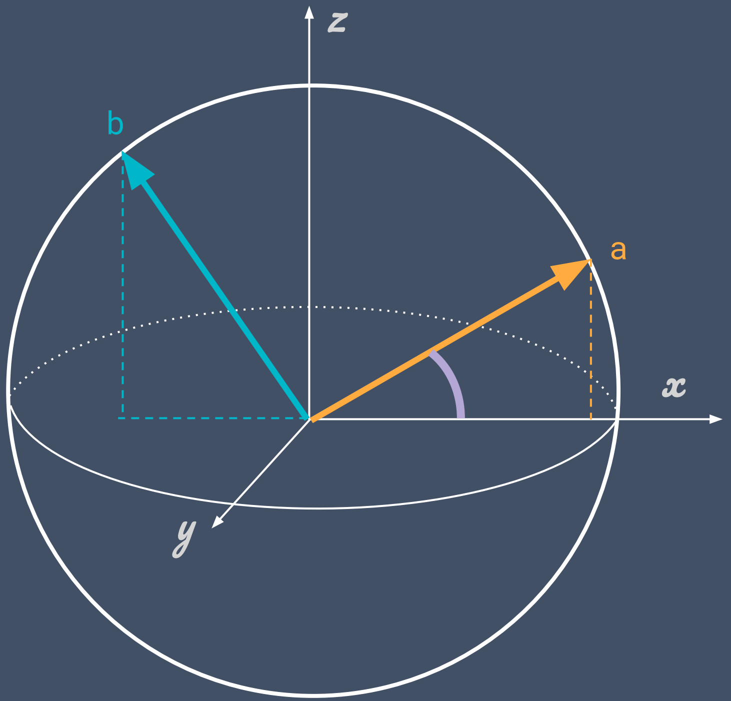
```
import cirq, random, sympy
import numpy as np
import tensorflow as tf
import tensorflow_quantum as tfq

qubit = cirq.GridQubit(0, 0)
```



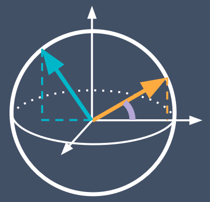
```
import cirq, random, sympy
import numpy as np
import tensorflow as tf
import tensorflow_quantum as tfq

qubit = cirq.GridQubit(0, 0)

# Quantum data labels
expected_labels = np.array([[1, 0], [0, 1]])

# Random rotation of X and Z axes
angle = np.random.uniform(0, 2 * np.pi)
```



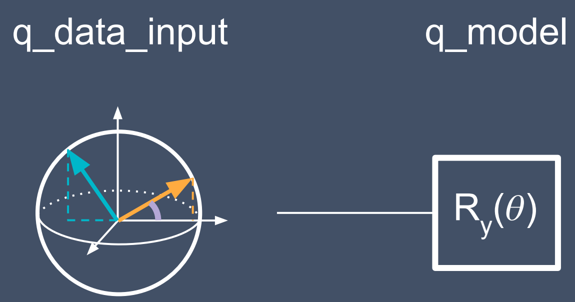
```python
import cirq, random, sympy
import numpy as np
import tensorflow as tf
import tensorflow_quantum as tfq

qubit = cirq.GridQubit(0, 0)

# Quantum data labels
expected_labels = np.array([[1, 0], [0, 1]])

# Random rotation of X and Z axes
angle = np.random.uniform(0, 2 * np.pi)

# Build the quantum data

a = cirq.Circuit(cirq.Ry(angle)(qubit))
b = cirq.Circuit(cirq.Ry(angle + np.pi/2)(qubit))
quantum_data = tfq.convert_to_tensor([a, b])
```



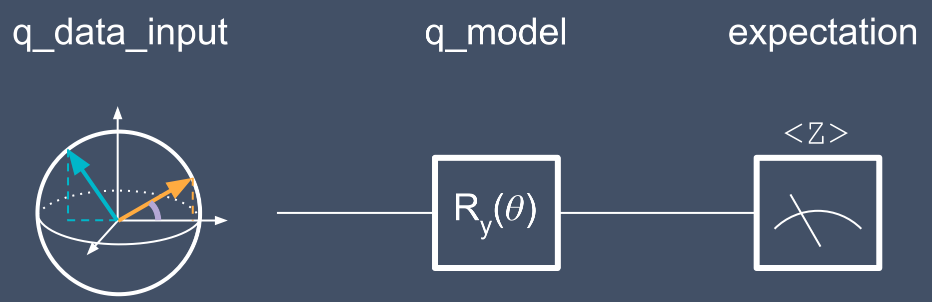
q_data_input



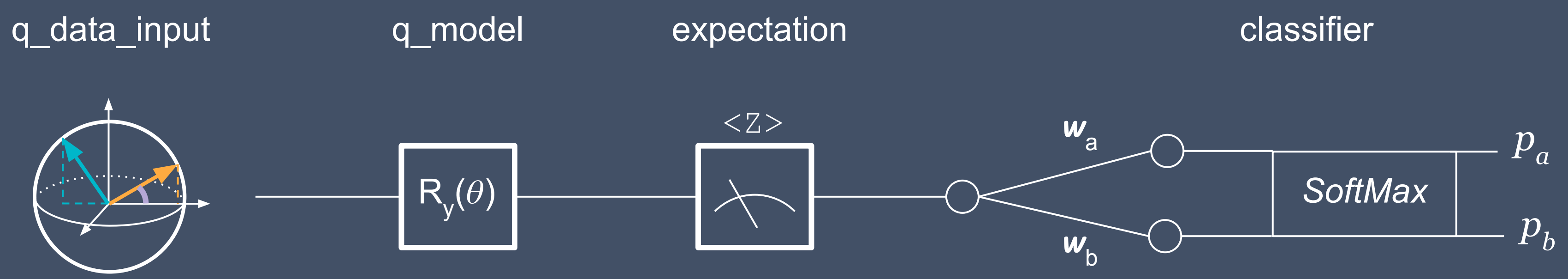
```
# Build the quantum model
q_data_input = tf.keras.Input(shape=(), dtype=tf.dtypes.string)
```



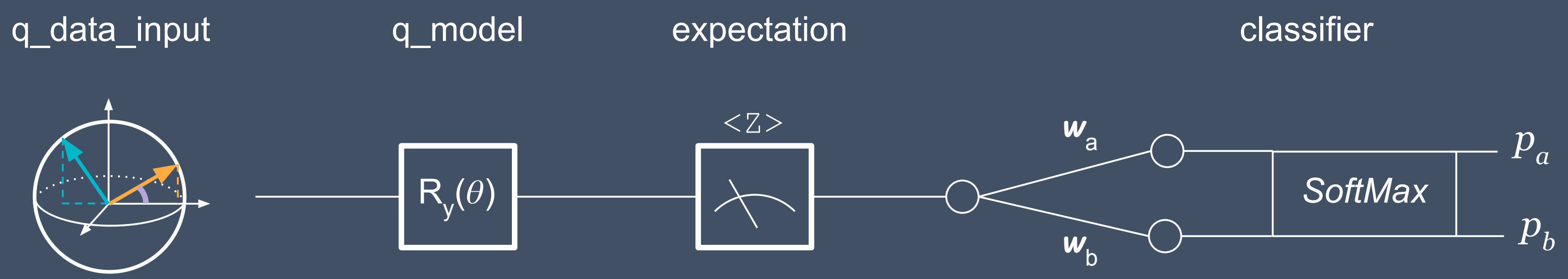

```
# Build the quantum model
q_data_input = tf.keras.Input(shape=(), dtype=tf.dtypes.string)

theta = sympy.Symbol('theta')
q_model = cirq.Circuit(cirq.Ry(theta)(qubit))
```



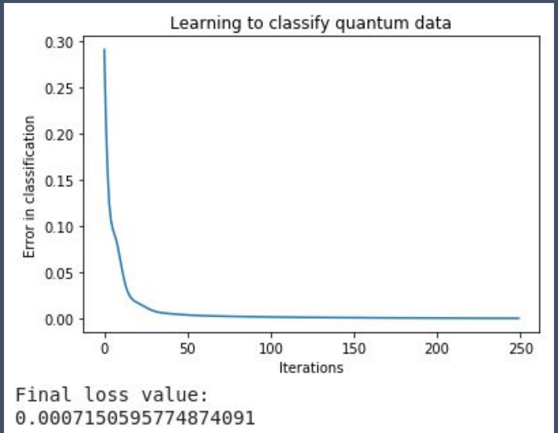

```python
# Build the quantum model
q_data_input = tf.keras.Input(shape=(), dtype=tf.dtypes.string)

theta = sympy.Symbol('theta')
q_model = cirq.Circuit(cirq.Ry(theta)(qubit))

expectation = tfq.layers.PQC(q_model, cirq.Z(qubit))
expectation_output = expectation(q_data_input)
```




```python
# Build the quantum model
q_data_input = tf.keras.Input(shape=(), dtype=tf.dtypes.string)

theta = sympy.Symbol('theta')
q_model = cirq.Circuit(cirq.Ry(theta)(qubit))

expectation = tfq.layers.PQC(q_model, cirq.Z(qubit))
expectation_output = expectation(q_data_input)

# Attach the classical SoftMax classifier
classifier = tf.keras.layers.Dense(2, activation=tf.keras.activations.softmax)
classifier_output = classifier(expectation_output)
```




```python
# Build and train the hybrid model
model = tf.keras.Model(inputs=q_data_input,
     outputs=classifier_output)
model.compile(
     optimizer=tf.keras.optimizers.Adam(learning_rate=0.1),
     loss=tf.keras.losses.CategoricalCrossentropy())
history = model.fit(x=quantum_data, y=expected_labels,
     epochs=250, verbose=0)
```



```python
# Check inference on noisy quantum datapoints
noise = np.random.uniform(-0.25, 0.25, 2)
test_data = tfq.convert_to_tensor([
    cirq.Circuit(
        cirq.Ry(random_angle + noise[0])(qubit)),
    cirq.Circuit(
        cirq.Ry(random_angle + noise[1] + np.pi/2)(qubit))
])
predictions = model.predict(test_data)
```

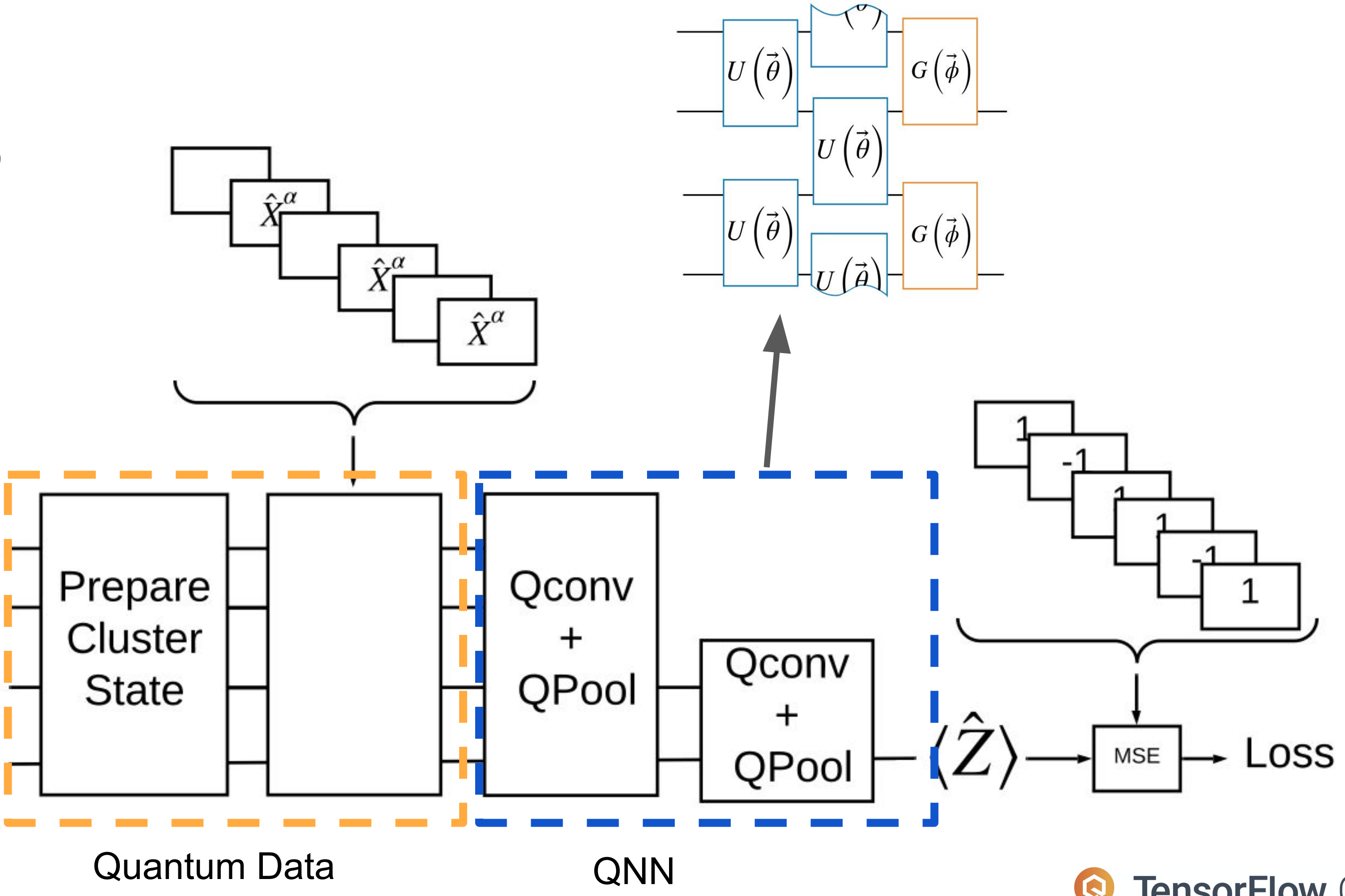
```
Noisy element from a:
prob(0)=0.9995, prob(1)=0.0005
Noisy element from b:
prob(0)=0.0025, prob(1)=0.9975
```

Hybrid Quantum-Classical

Convolutional Neural Networks (CNN)

Quantum CNN

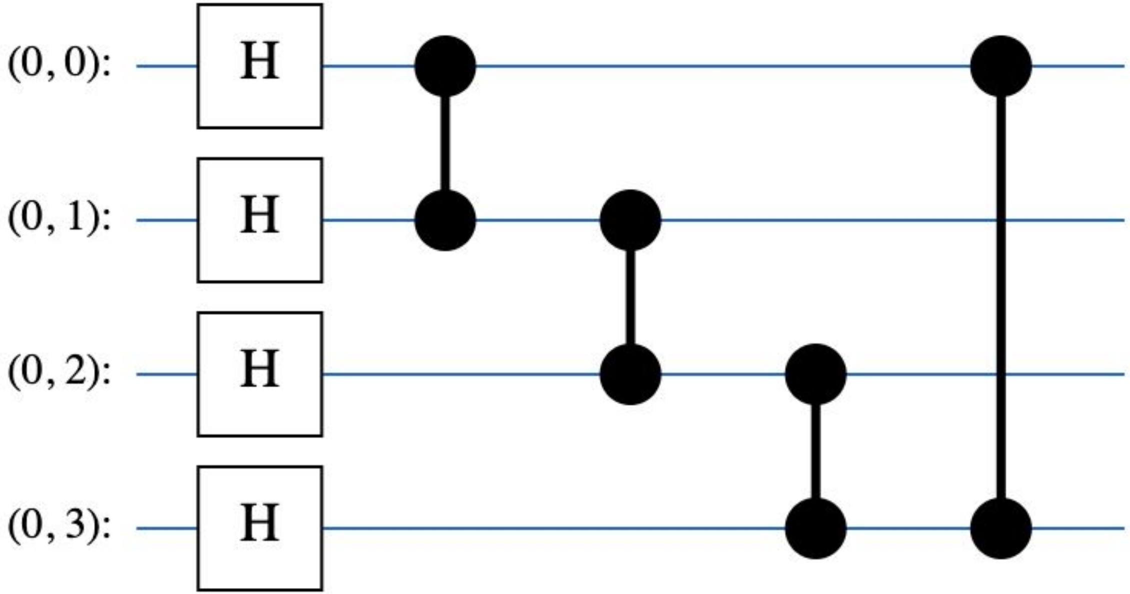
I. Cong, M. Lukin,
Nature Physics (2019)



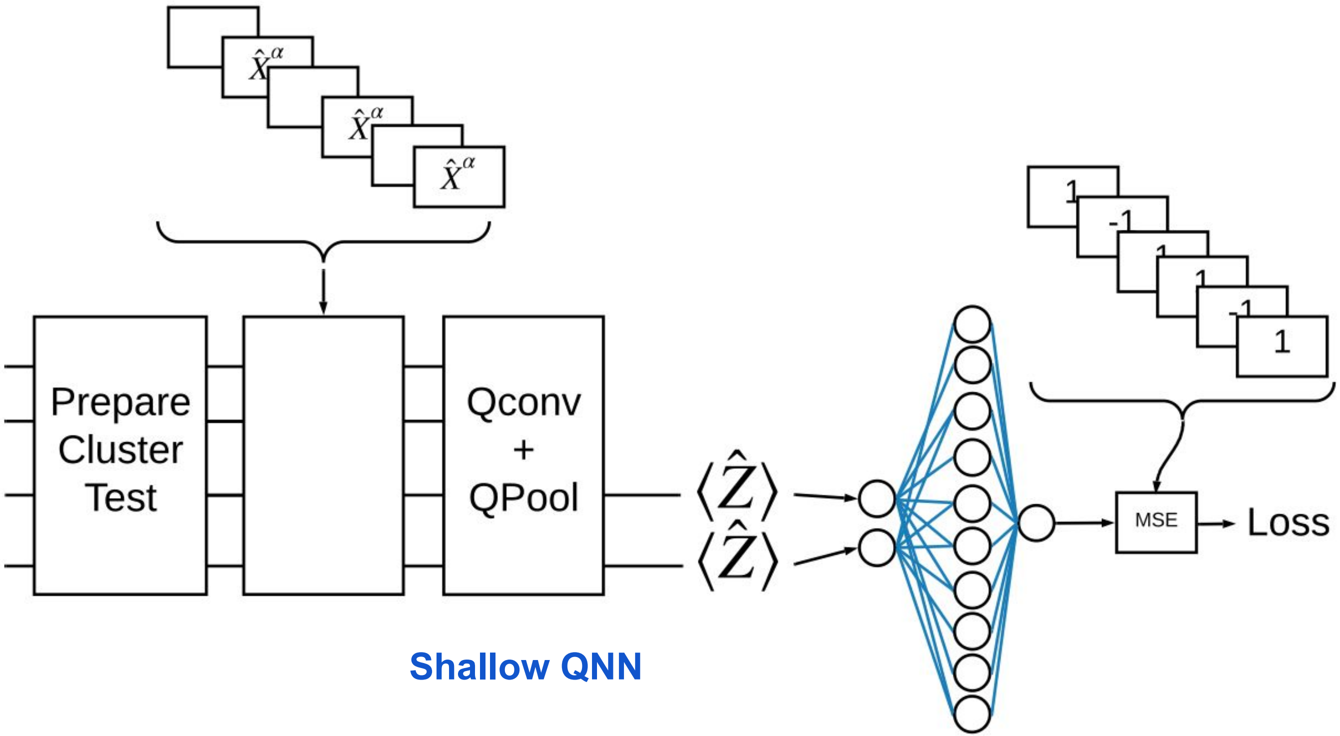
Google AI Quantum

TensorFlow Quantum

Cluster State Prepartion



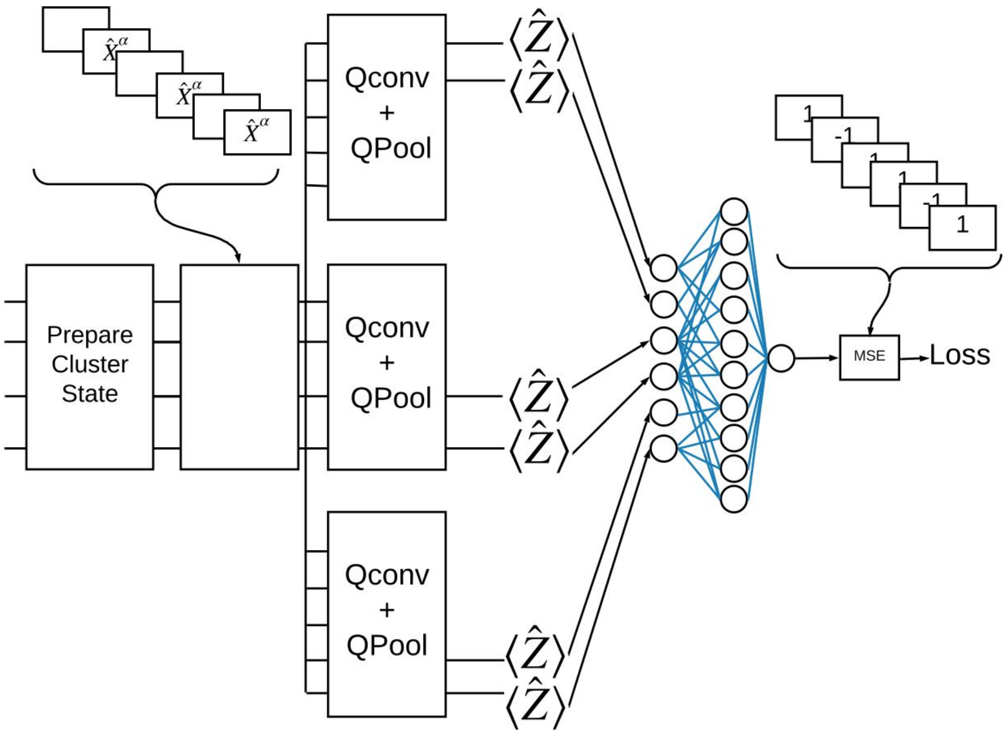
Hybrid Quantum-Classical CNN



Google AI Quantum

TensorFlow Quantum

Hybrid quantum-classical CNNs: Distributed NISQ Computing

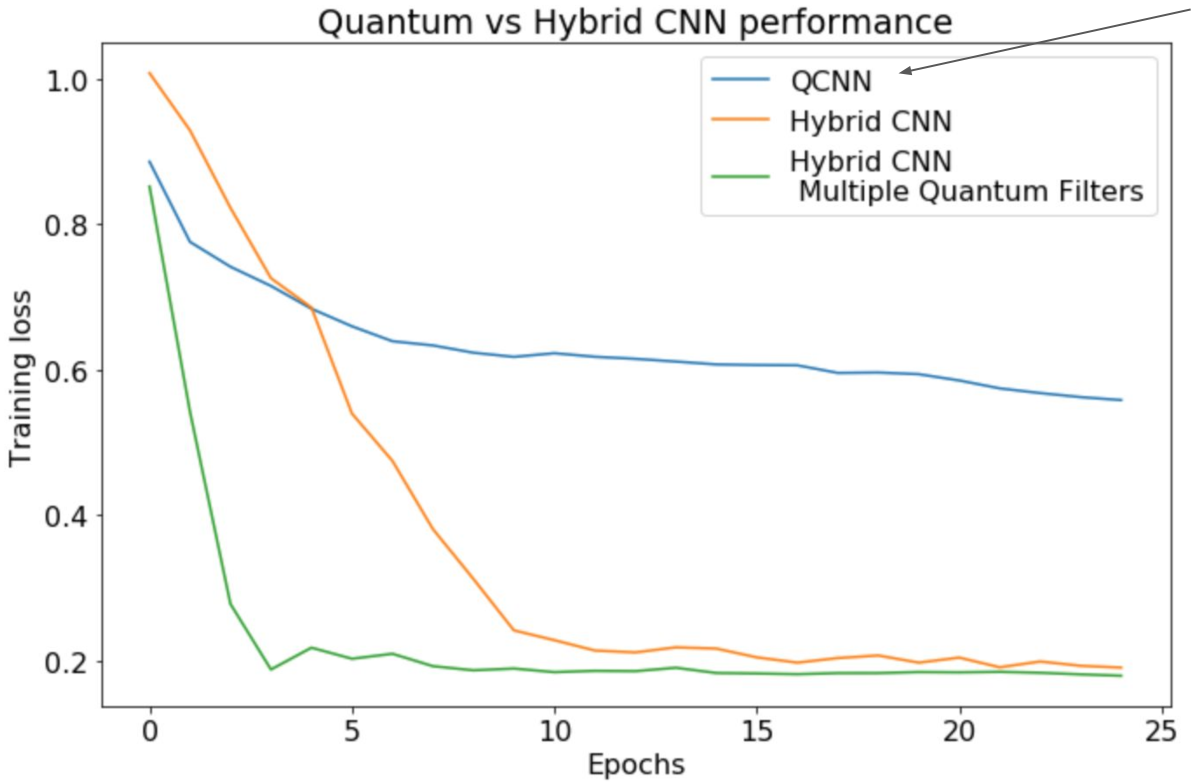


Parallelized Shallow QNN

Google AI Quantum

TensorFlow Quantum

Hybrid CNN Results



Cong, Lukin, et al
Nature Physics (2018)

TensorFlow Quantum

TFQ Benefits to Researchers

1. Reduce prototyping time from weeks to hours
    a. High level API integration with Keras
    b. High performance circuit simulator via qsim

2. Support for Hybrid Models & Quantum Data
    a. Access to algorithmic features of TensorFlow
    b. Integration with Cirq
    c. Automatic differentiation of quantum circuits

3. Exposure to TensorFlow Community (Millions of Users)

Next Steps

Research collaborations with academia:

- Quantum Dataset Initiative
- Practical quantum supremacy for QML on quantum data
- Novel quantum control & error mitigation schemes

Engineering:

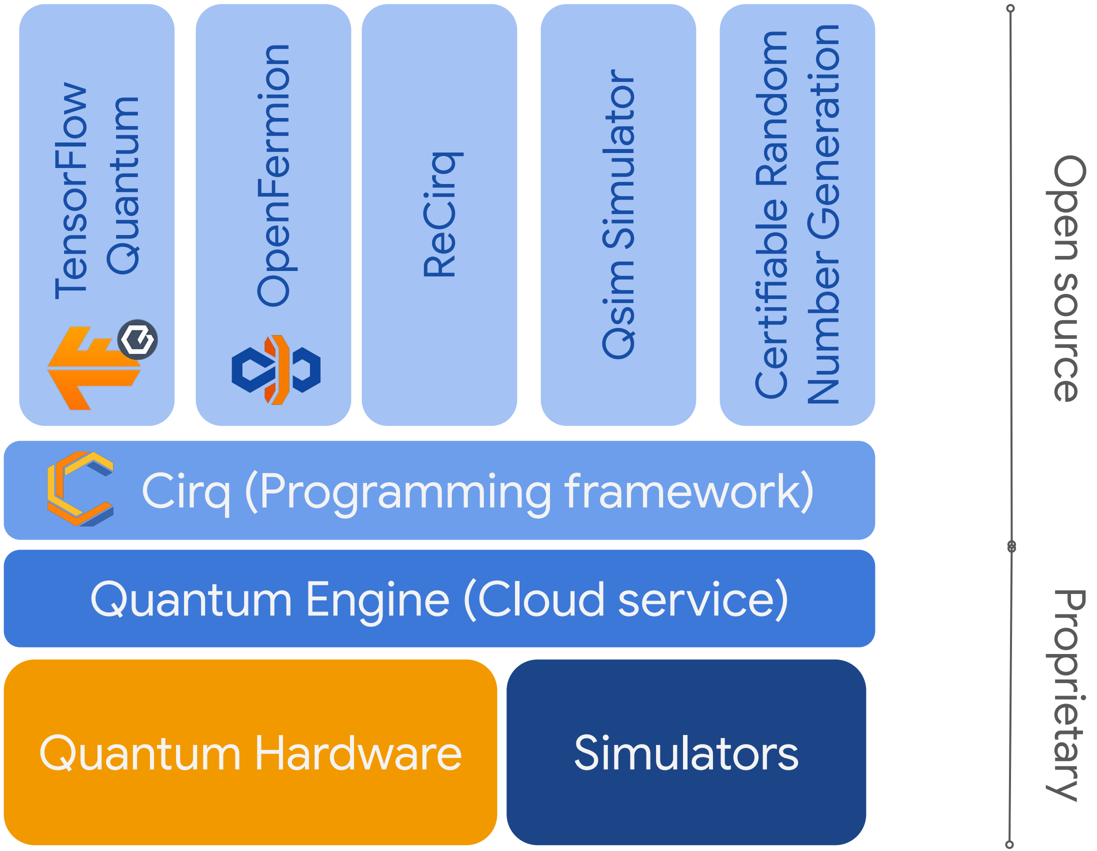
- Integration with Quantum Engine

Adoption:

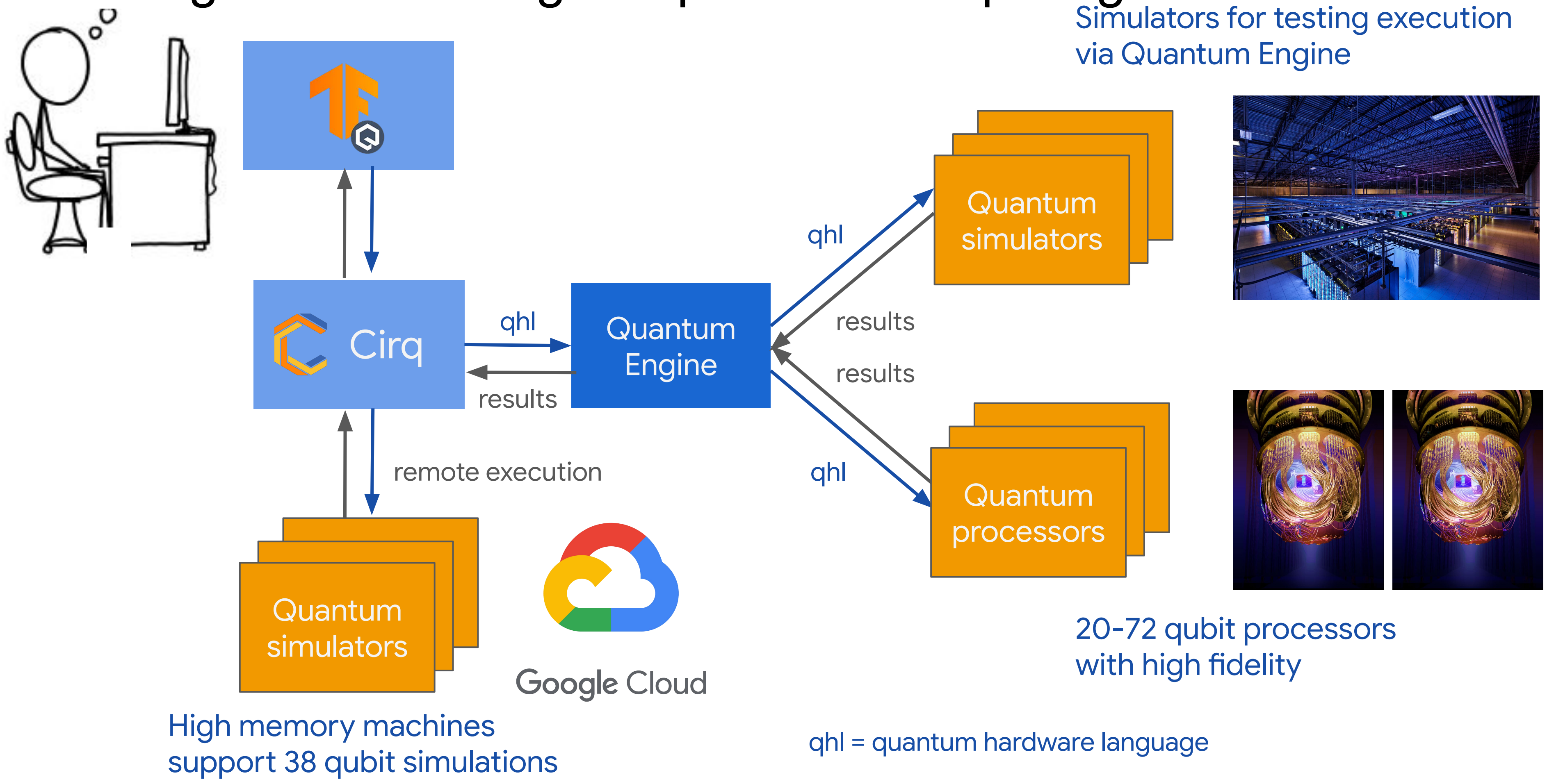
- Integrate more academic partners (U of Toronto / Caltech / Harvard) & Google Brain & Deepmind

Software stack

Libraries and tools



Running TFQ via Google's quantum computing service

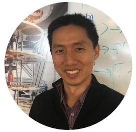


The Team

Tech Lead

Masoud Mohseni
Google AI Quantum

Product Manager



Alan Ho
Google AI Quantum

Theory

Guillaume Verdon
Quantum @ X

Engineering

Michael Broughton
Google AI Quantum

Trevor McCourt
Google AI Quantum

Jae Yoo
Tensorflow

Murphy Niu
Google AI Quantum

Antonio Martinez
Google AI Quantum

Philip Massey
Google

Evan Peters
UWaterloo

Thank You!

https://www.tensorflow.org/quantum

arXiv:2003.02989

TensorFlow

DEV SUMMIT 2020