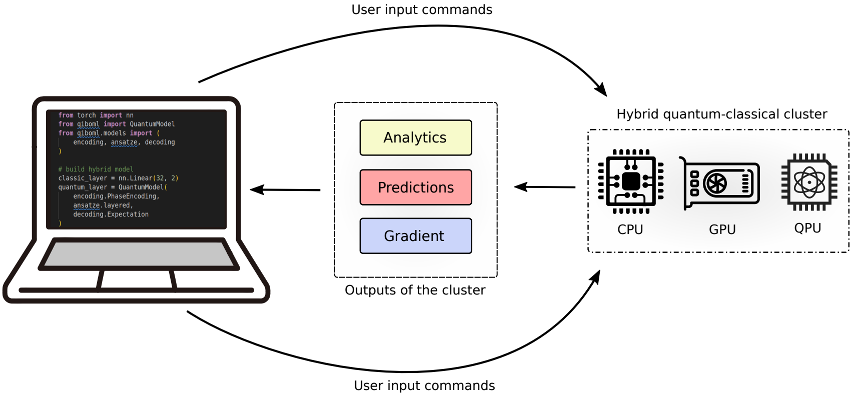
# A few slides on full-stack QML

November 22, 2024

# Quantum Machine Learning challenges

We aim to involve quantum process units (QPU) into machine learning (ML) pipelines.

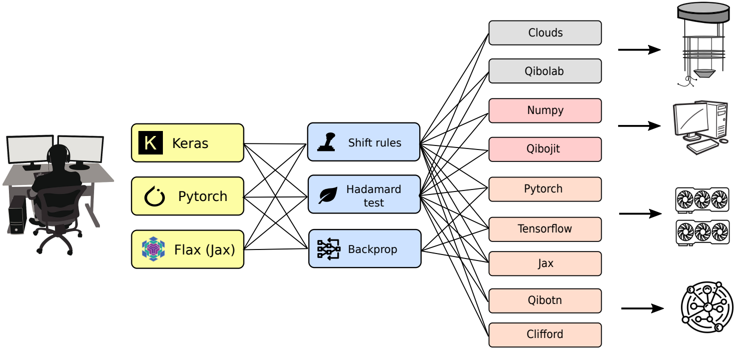



Classical and quantum components have to be executable on CPUs, GPUs and QPUs to fullfill the whole potential of an **hybrid quantum-classical cluster**.

# Qibo as a modular playground

To do so, `Qibo` stands as an intriguing playgrond thanks to its **modularity**.

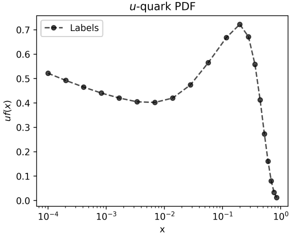
Once a favourite ML framework is chosen, a quantum circuit can be built with Qibo and included into the pipeline.



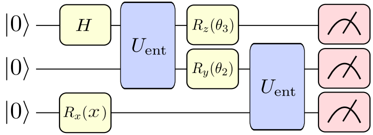

The circuit can then be executed onto the desired Qibo backend (quantum or classical).

# A QML pipeline

A target $f(x)$

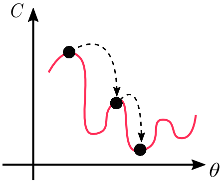



A parametric model $U(x;\theta)$




returning predictions

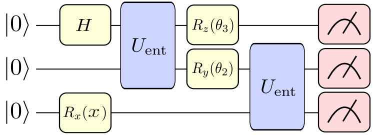
$$\tilde{f}(x;\theta) = \langle 0|U(x;\theta)^{\dagger}\hat{O}U(x;\theta)|0\rangle$$



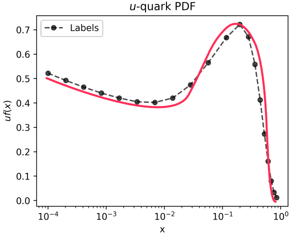

A training algorithm

# A QML pipeline

A parametric model $U(x;\theta)$

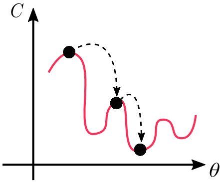



A target $f(x)$




returning predictions

$$\tilde{f}(x;\theta) = \langle 0|U(x;\theta)^\dagger \hat{O} U(x;\theta)|0\rangle$$




A training algorithm

# Training a real superconducting device

📖 arXiv:2308.06313

### High level API: Qibo

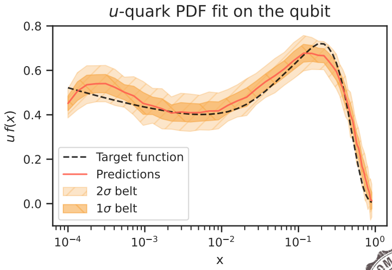
- </> define **prototypes** and models;
- </> **simulate** training and noise.

### Calibration: Qibocal

- **calibrate** qubits;
- generate **platform configuration**;

### Execution: Qibolab

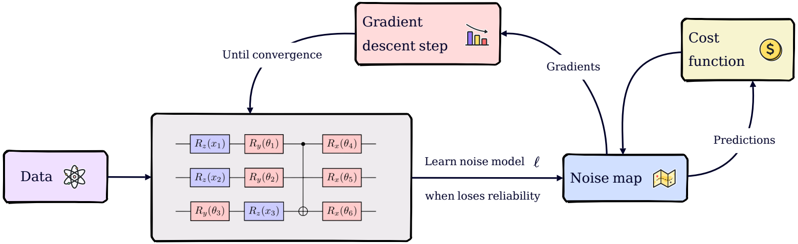
- allocate **calibrated** platform;
- **compile** and **transpile** circuits;
- execute and return **results**.




| Parameter | Value |
|---|---|
| $N_{\mathrm{data}}$ | 50 |
| $N_{\mathrm{shots}}$ | 500 |
| MSE | $\sim 10^{-3}$ |
| Electronics | Xilinx ZCU216 |
| Training time | $\sim$ 2h |

# Exploit the hybrid environment: real-time quantum error mitigation

We want to mitigate the noise of the QPU using **error mitigation** techniques.




1. consider a parametric quantum circuit trained with gradient descent;
2. learn the noise map $\ell$ every time is needed over the procedure;
3. use $\ell$ to clean up both predictions and gradients.

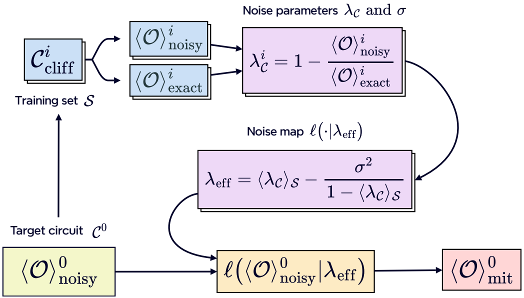
This procedure requires **real-time interaction** between quantum and classical devices: the former returns the output of the quantum system, the latter mitigate the noise.

# Mitigation requires quantum and classical resources

We use a "learning-based" technique, which exploits **efficient classical simulators** to reconstruct surrogates of the target circuit and learn its noise.

In particular:

1. define a set of circuits $\mathcal{S}$ similar to the target[1];
2. collect noisy predictions from the QPU $\forall s \in \mathcal{S}$;
3. simulate exact predictions on C(G)PU $\forall s \in \mathcal{S}$;
4. learn the noise map applying classical regression to noisy-exact data.

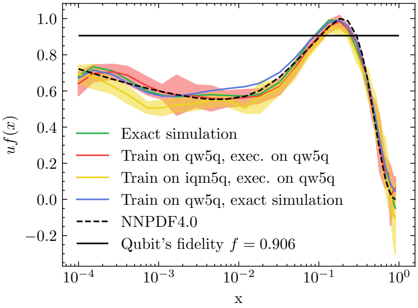



[1]We construct them as Clifford so that we can simulate high number of qubits with good performances.

# RTQEM in action

We perform a gradient descent on two different quantum devices (and noises!)

| Parameter | $N_{\text{train}}$ | $N_{\text{params}}$ | $N_{\text{shots}}$ | Epochs | Optimizer | Learning rate |
|---|---|---|---|---|---|---|
| **Value** | 15 | 16 | 500 | 100 | Adam | 0.1 |




- `qw5q` from QuantWare and controlled using Qblox instruments;
- `iqm5q` from IQM and controlled using Zurich Instruments.

| Train. | Epochs | Pred. | Config. | MSE |
|---|---|---|---|---|
| `qw5q` | 100 | `qw5q` | RTQEM | 0.0013 |
| `iqm5q` | 100 | `qw5q` | RTQEM | 0.0037 |
| `qw5q` | 100 | `sim` | RTQEM | 0.0016 |

All the hardware results are obtained deploying the $\boldsymbol{\theta}_{\text{best}}$ on `qw5q`.