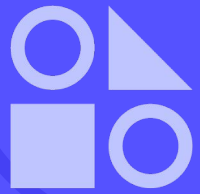
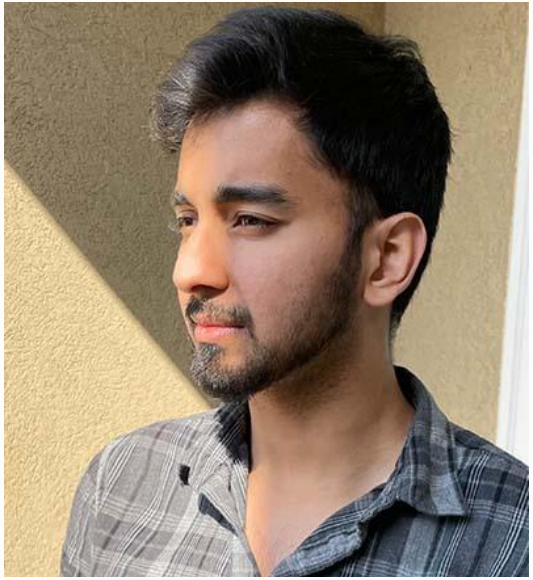
agnostiq



Covalent.

Heterogeneous Workflow Orchestration
for
Quantum and HPC

Santosh K. Radha, PhD
Head of Quantum Algorithms



Ara Ghukasyan, PhD
Research Software Engineer

Overview

1. Introduction to Covalent
2. Covalent 101
3. Code Examples
4. Introduction to Quantum Machine Learning
5. Focus on Similarity Learning
6. Heterogeneous Similarity learning with Covalent **(hands-on workshop)**

Workflow Orchestration

Why is workflow orchestration is needed?

Primary Challenges:

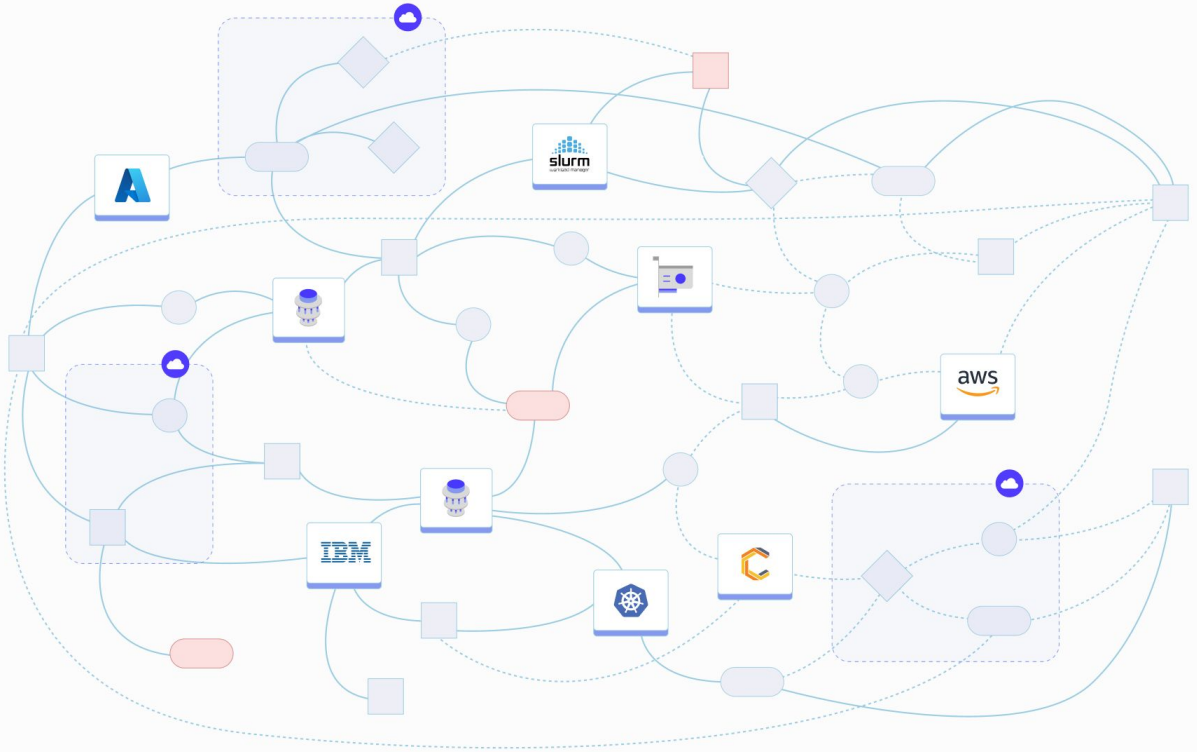
- Explore new research problems
- Apply interdisciplinary techniques
- Push the boundaries of hardware
- Find hidden patterns in data

Lots of work, little reward… everyone comes up with their own bespoke solutions to these

Additional Operational Challenges:

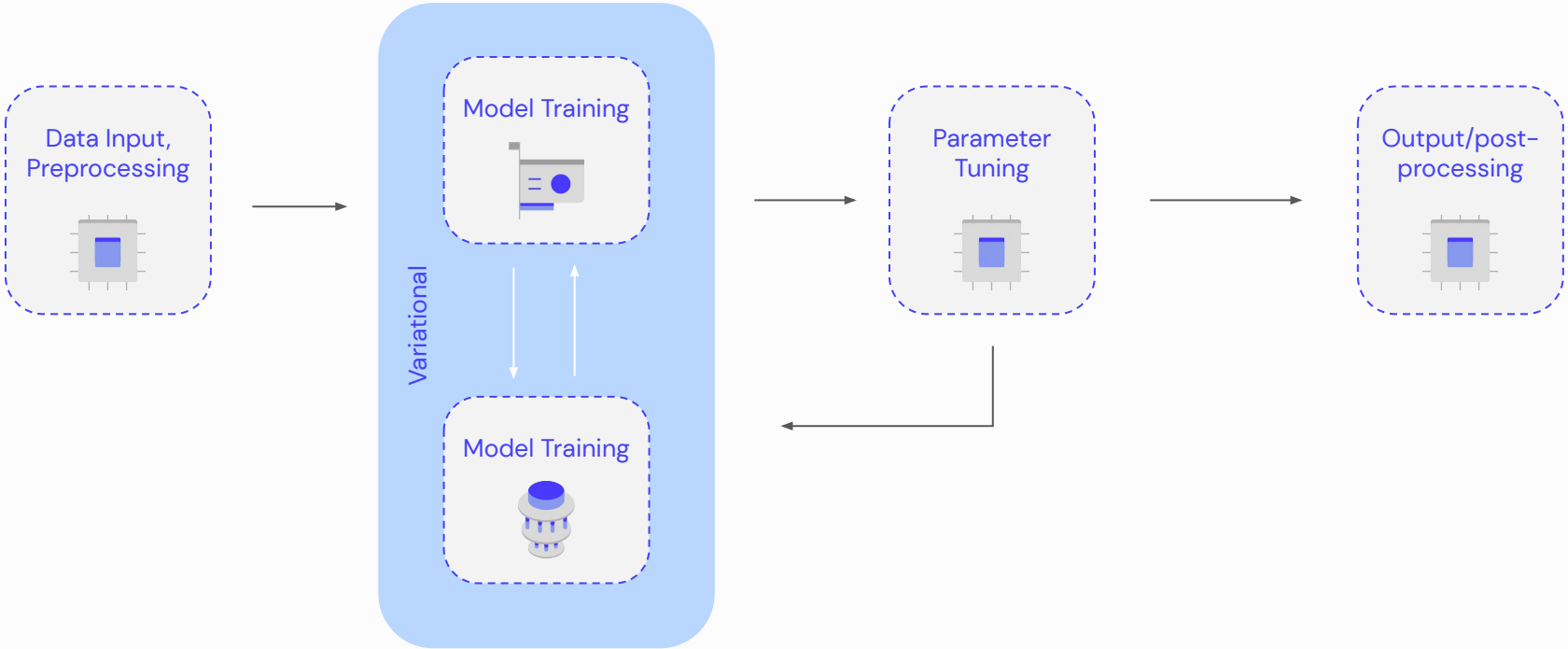
- Organizing and versioning experiments
- Managing and sharing data
- Managing accounts on multiple clusters
- Checkpointing long simulations
- Reproducing (your own) work
- Package version conflicts
- Compiling colleagues' code
- Incorporating legacy code
- Identifying optimal hardware resources
- Infrastructure management
- Working within a budget constraint

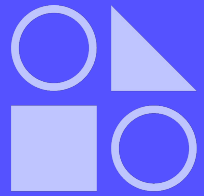
Modern computational workflows are complex.



Quantum is driving the need for more heterogeneity

A simplified quantum machine learning workflow:





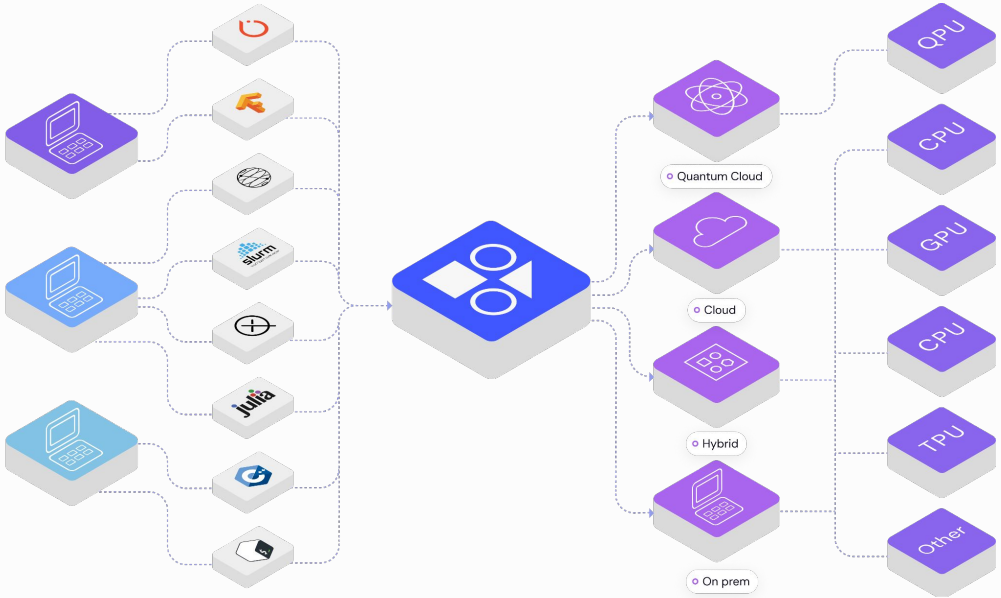
Covalent.

by Agnostiq

Covalent is a free & open source workflow orchestration platform for heterogeneous computing

Covalent.

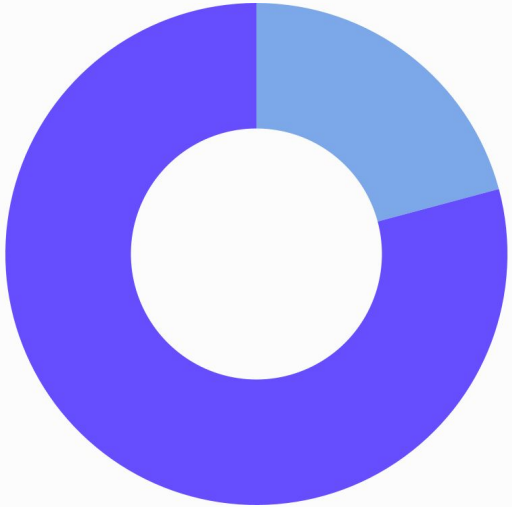
- Support for quantum + HPC
- Distributed computation
- Heterogeneous hardware and software
- Workflow management interface



AgnostiqHQ/covalent ⭐

Why does Covalent exist?

Computational research



Time spent on research

Time spent on
Writing,
Debugging,
Executing,
Maintaining,
Research code

How Covalent can help

Solution 1.

Hybrid research / experiments. A single QML experiment now contains CPU + GPU/TPU + QPU.

Solution 2.

HPC research requires rapid prototyping and experimentation across software parameters.

Solution 3.

Execute high-throughput calculations. Run massively parallel jobs at scale across clusters.

Solution 4.

Avoid long HPC queue times and redeploy to different quantum queues using serverless HPC.

Where Covalent fits in the (Python) stack

Covalent supports a growing ecosystem of hardware and software

Classical Resources

- Slurm executor
- AWS Fargate executor
- AWS Batch executor
- Azure executor
- GCP executor
- Kubernetes executor

Quantum Resources

- AWS-Braket Jobs
- Qiskit runtime (coming soon)
- More to come
- **Write your own plugin!**

Software Packages

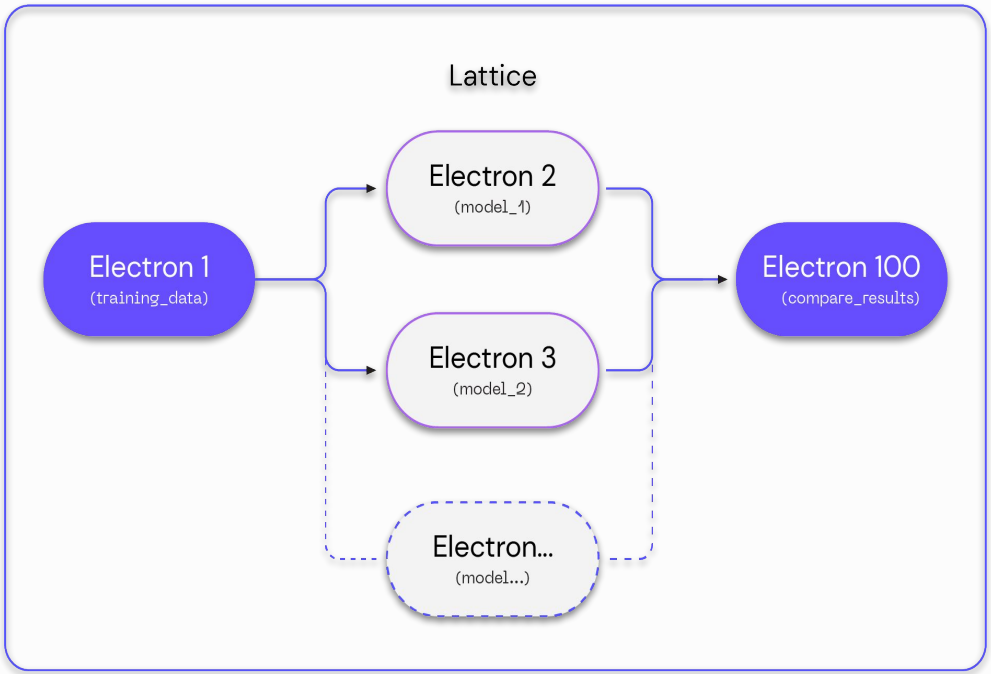
Any and all packages!

Languages

Covalent 101: The Basics

Covalent Concepts

- Workflows are comprised of inter-dependent tasks
- Workflows are called “**lattices**”
- Sub-tasks are called “**electrons**”
- Electrons can be written in other languages
- Electrons can each run on different hardware backends

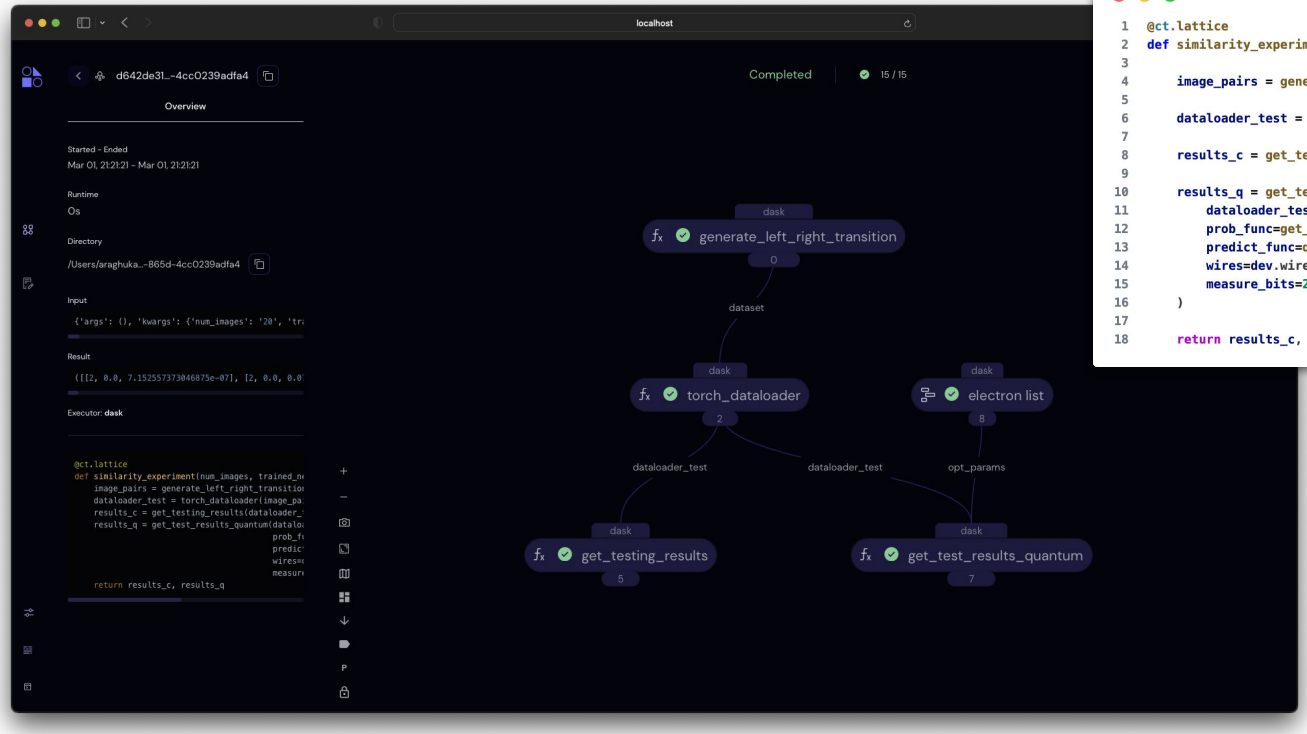


Minimal code changes required

```python
def optimize(x, y):
    return min(x, y)

def task(x, y):
    z = optimize(x, y)
    return z

def train(model):
    import time
    time.sleep(1)
    return model

def workflow(inputs):
    model = task(inputs[0], inputs[1])
    return train(model)

print(workflow([1, 2]))

```

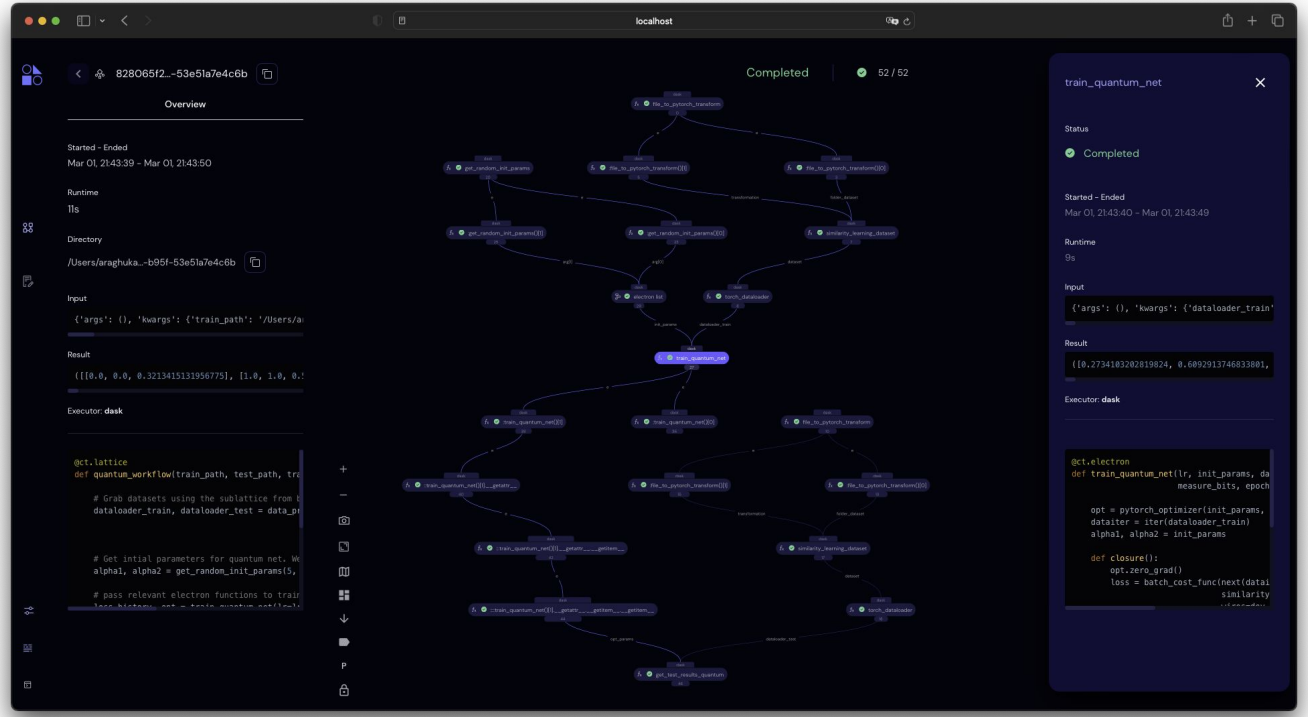
```python
import covalent as ct

def optimize(x, y):
    return min(x, y)

@ct.electron
def task(x, y):
    z = optimize(x, y)
    return z

@ct.electron
def train(model):
    import time
    time.sleep(1)
    return model

@ct.lattice
def workflow(inputs):
    model = task(inputs[0], inputs[1])
    return train(model)

dispatch_id = ct.dispatch(workflow)([1, 2])
print(ct.get_result(dispatch_id, wait=True))

```

Covalent UI




```python
@ct.lattice
def similarity_experiment(num_images, trained_net_c, trained_opt_q):

    image_pairs = generate_left_right_transition(num_images)

    dataloader_test = torch_dataloader(image_pairs, shuffle=False, batch_size=1)

    results_c = get_testing_results(dataloader_test, trained_net_c)

    results_q = get_test_results_quantum(
        dataloader_test, trained_opt_q.param_groups[0]['params'],
        prob_func=get_quantum_prob,
        predict_func=quantum_predict,
        wires=dev.wires,
        measure_bits=2
    )

    return results_c, results_q
```

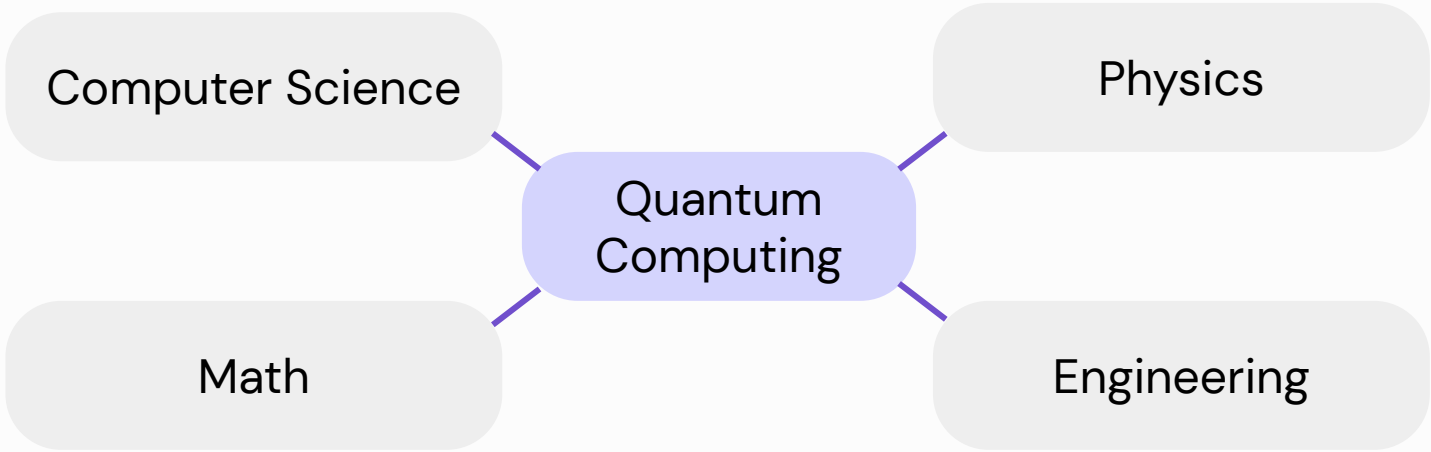
Visualize and monitor complex workflows



You can contribute too!

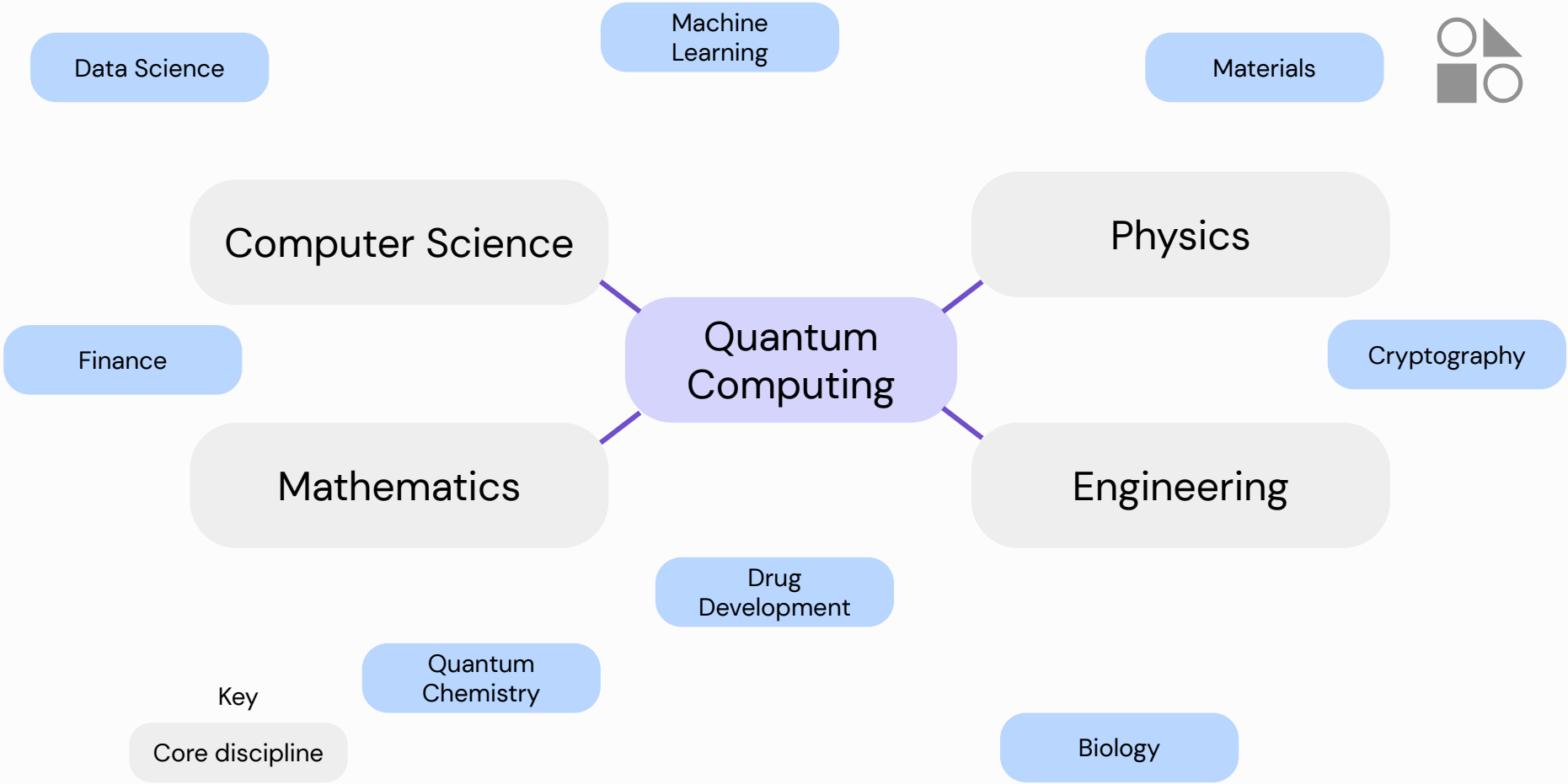
Pythonic workflows	Automatic checkpointing	Multiple language support	Little-to-no overhead	Customizable
Reproducibility	Code locally, run anywhere	Covalent. It's Open-Source!	Intuitive User-interface	Natively hybrid workflows
Native parallelization				Variety of executors
Code isolation	Parameter caching	Cloud agnostic	Interactive jobs	Start locally and scale

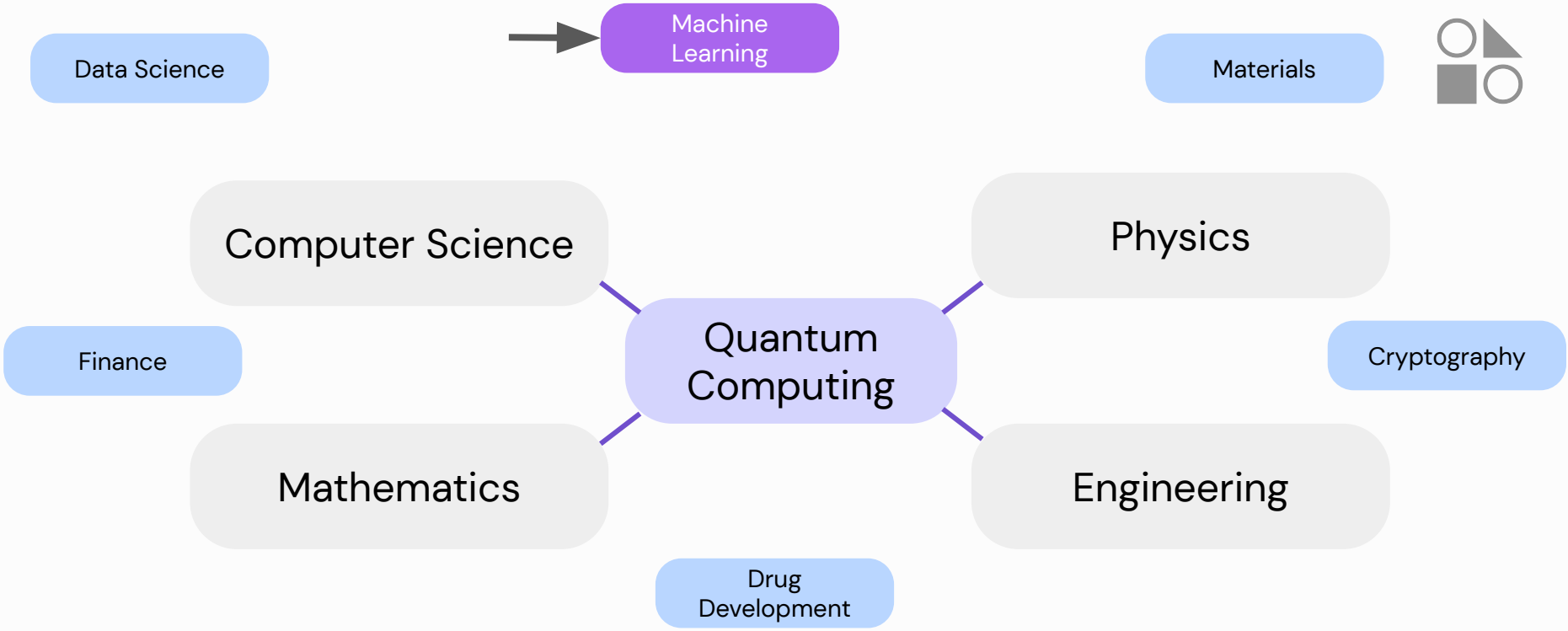
A Brief Introduction to Quantum Machine Learning



Key

Core discipline


Data Science
Machine Learning
Materials
Computer Science
Physics
Finance
Quantum Computing
Cryptography
Mathematics
Engineering
Drug Development
Quantum Chemistry
Key
Core discipline
Application
Biology


Data Science
Machine Learning
Materials
Computer Science
Physics
Finance
Quantum Computing
Cryptography
Mathematics
Engineering
Drug Development
Quantum Chemistry
Key
Core discipline
Application
Biology

What is ~~Quantum~~ Machine Learning?

- Before addressing Quantum Machine Learning (QML), we must first address ML
- Model parameters are tuned while being fed a training dataset
- The predictive capability of a model is assessed using a testing dataset

What is ~~Quantum~~ Machine Learning?

- Before addressing Quantum Machine Learning (QML), we must first address ML
- Model parameters are tuned while being fed a training dataset
- The predictive capability of a model is assessed using a testing dataset

ML: The construction of predictive models without *explicit instruction*.

What is ~~Quantum~~ Machine Learning?

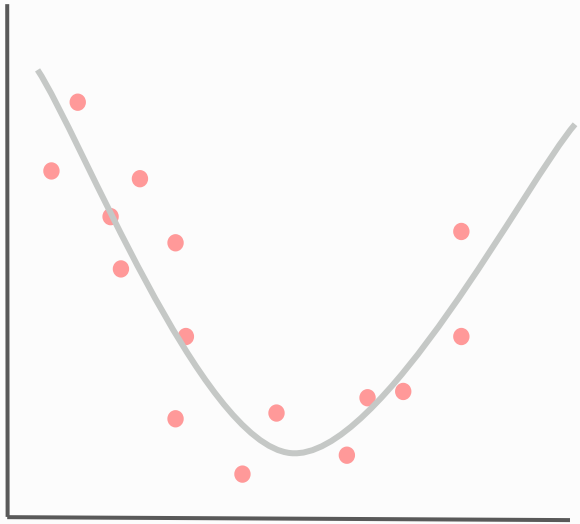
- Before addressing Quantum Machine Learning (QML), we must first address ML
- Model parameters are tuned while being fed a training dataset
- The predictive capability of a model is assessed using a testing dataset

ML: The construction of predictive models without *explicit instruction*.

Often, validation is performed between training and testing

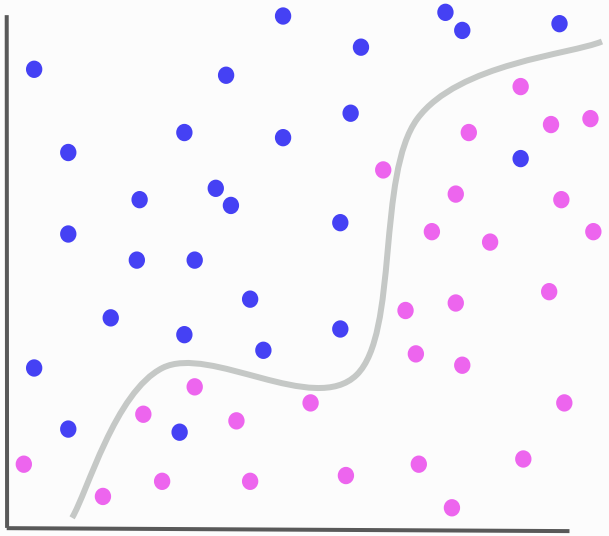
What is ~~Quantum~~ Machine Learning?

Regression



Logistic regression, random forest...

Classification



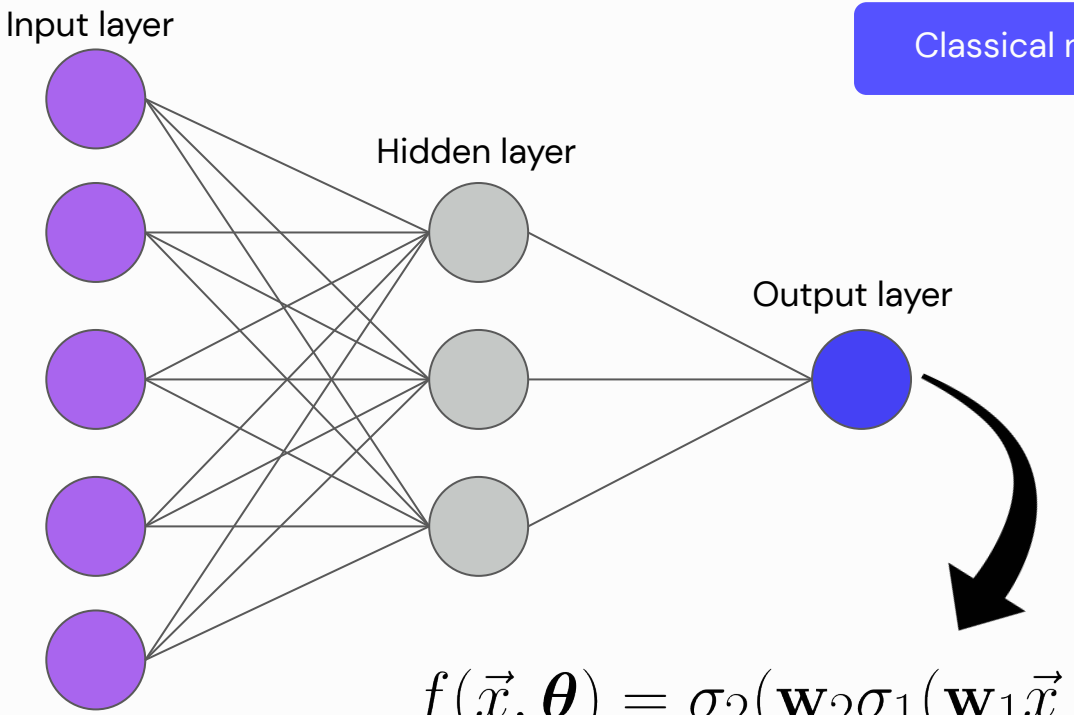
Support vector machines, autoencoders....

*Just two of the most common applications

What is ~~Quantum~~ Machine Learning?

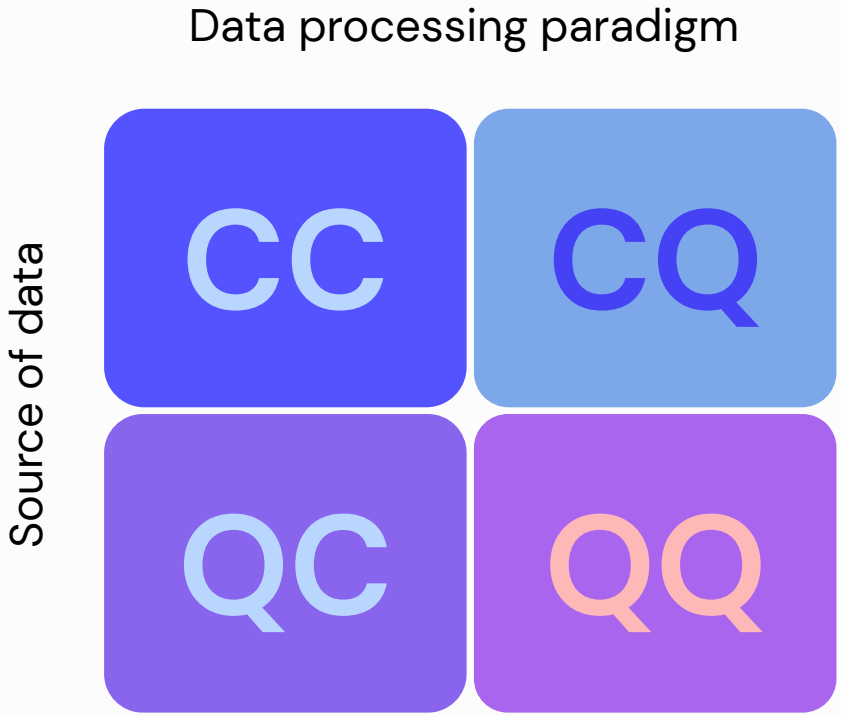
Classical neural networks

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix}$$

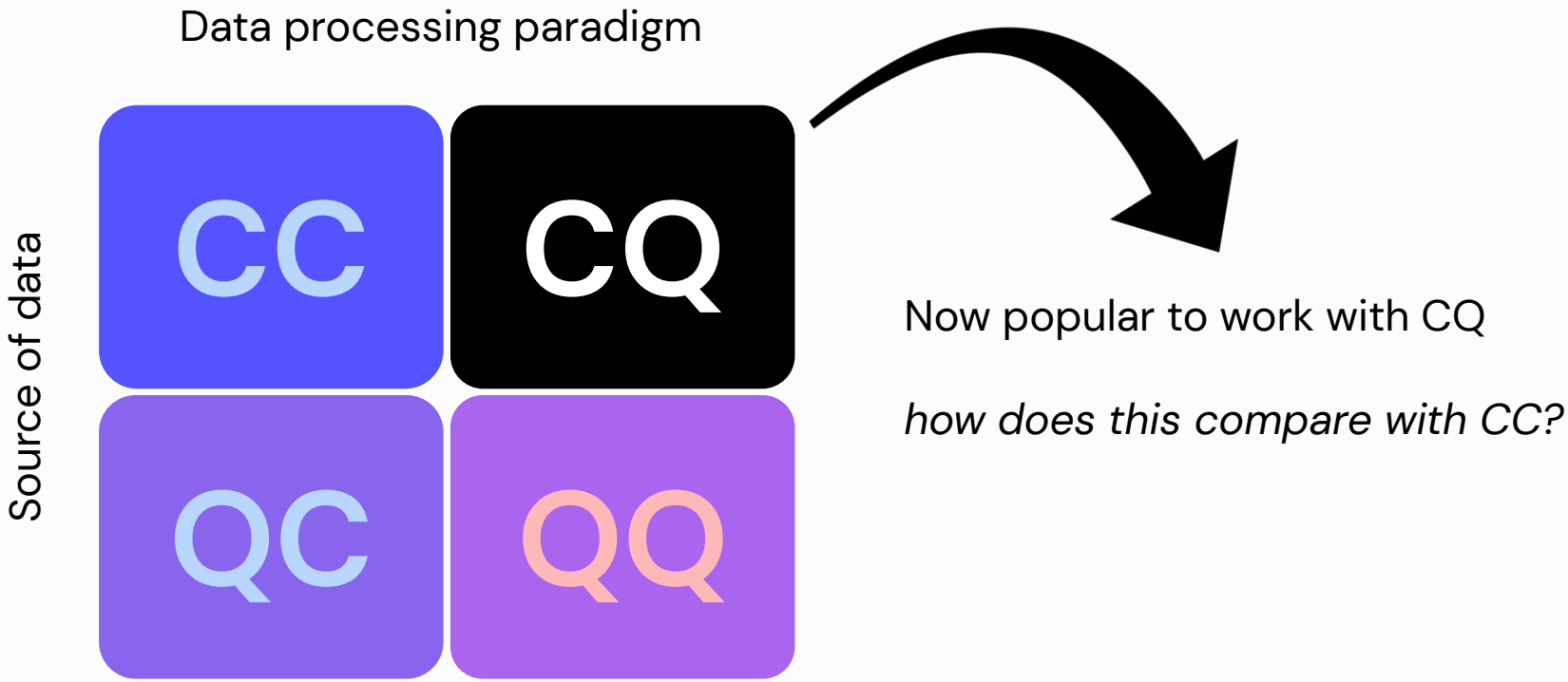


$$f(\vec{x}, \boldsymbol{\theta}) = \sigma_2(\mathbf{w}_2\sigma_1(\mathbf{w}_1\vec{x} + b_1) + b_2)$$

What is Quantum Machine Learning?



What is Quantum Machine Learning?

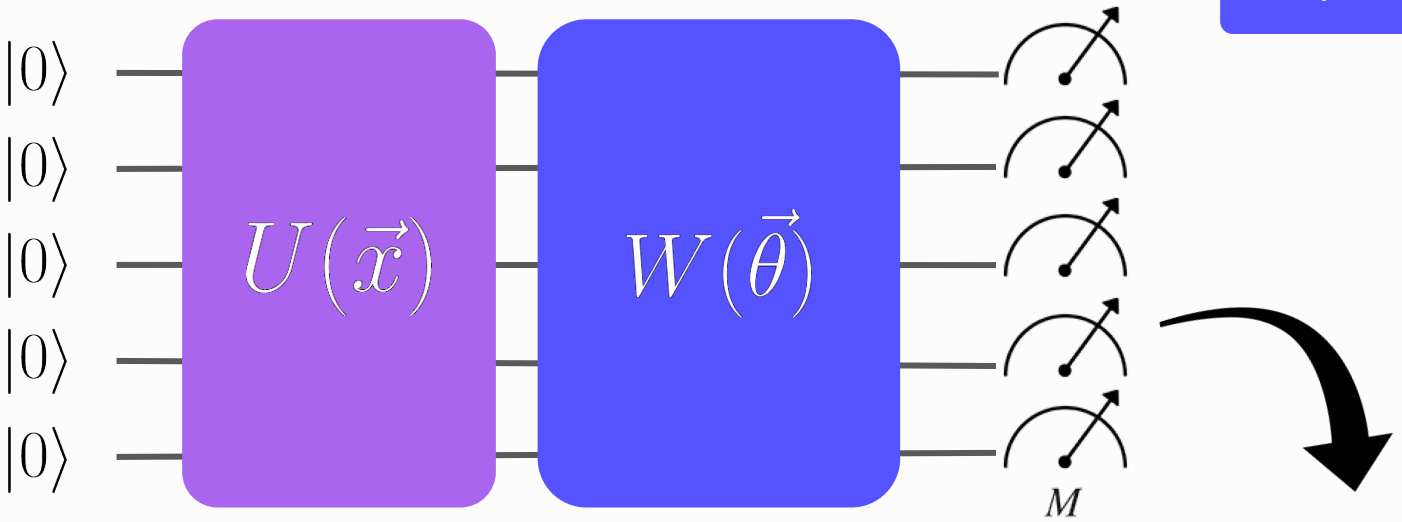


Now popular to work with CQ

how does this compare with CC?

What is Quantum Machine Learning?

Quantum neural networks



$$f(\vec{x},\vec{\theta}) = \langle 0\ldots|U^{\dagger}(\vec{x})W^{\dagger}(\vec{\theta})HW(\vec{\theta})U(\vec{x})|0\ldots\rangle$$

Classical and Quantum Similarity Learning

What is similarity learning?

Similarity learning involves learning the notion of what makes two (or more) objects similar

One way of handling this classically is to train a Siamese network. Basically two neural networks.

To handle this “quantumly”, we can use quantum similarity networks, or, GQSim (arXiv:2201.02310)

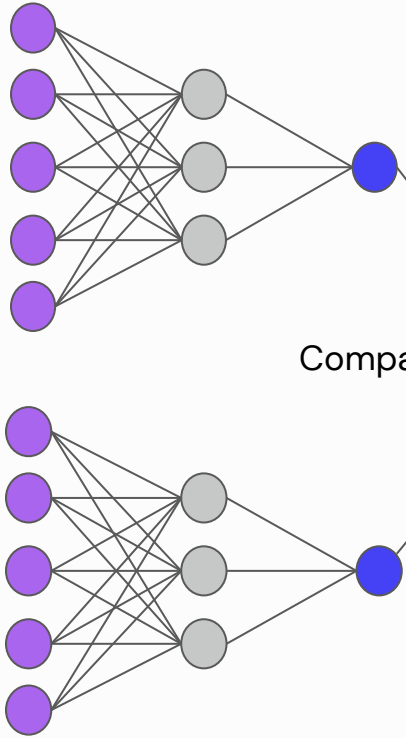
$$f(\vec{x}_i, \vec{x}_j, \vec{\theta}) \approx y_{ij}$$

$\vec{x}_i$	$\vec{x}_j$	y_{ij}
		0 (similar)
		1 (dissimilar)

Classical Siamese Network

$\vec{x}_i$

$\vec{x}_j$

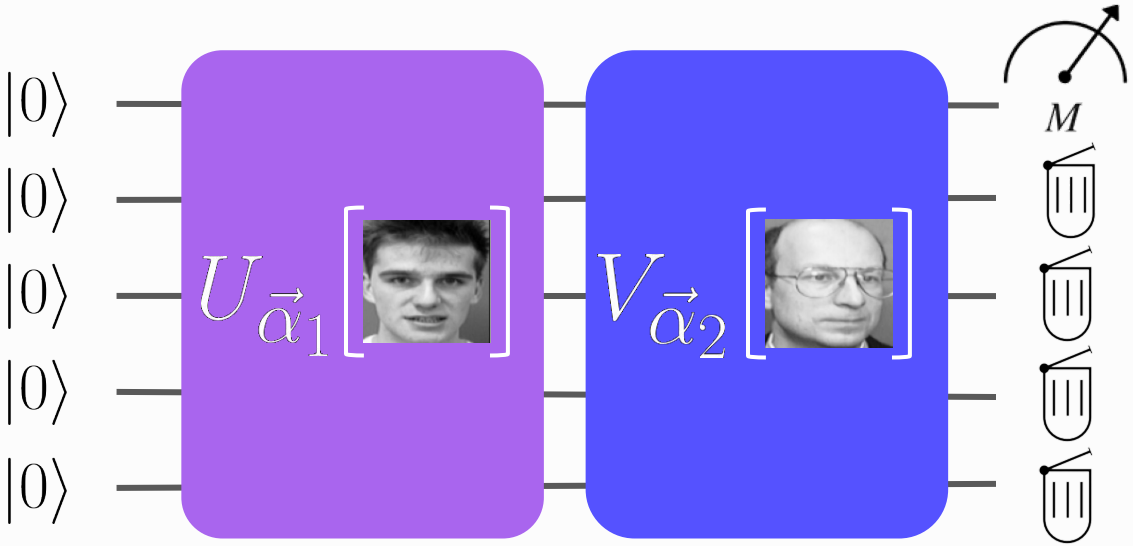


$$f(\vec{x}_i, \vec{x}_j, \vec{\theta}) \approx y_{ij}$$

y_{ij}

PyTorch

Quantum Similarity Network



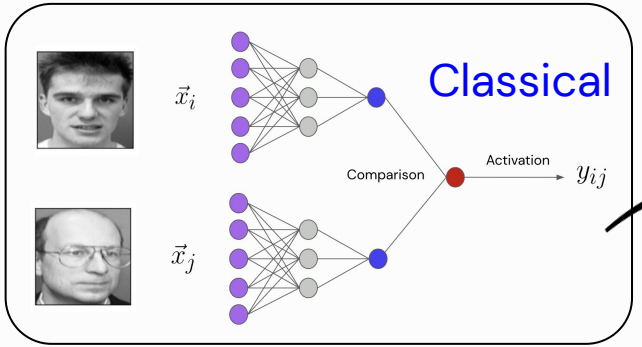
$$f(\vec{x}_i, \vec{x}_j, \vec{\theta}) \approx y_{ij}$$

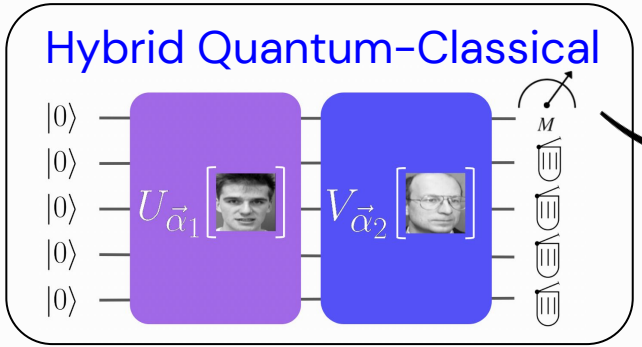
Can be as simple as the probability of measuring a single qubit in the 1 or 0 state.

PENNYLANE

Radha, S.K. and Jao, C., 2022. *Generalized quantum similarity learning.* *arXiv preprint* *arXiv:2201.02310*.

Enter Covalent





Covalent

Classical HPC

aws

slurm
workload manager

Quantum-classical compute

Amazon Braket

Qiskit

IBM

dffc97eb...-8e6a4c7eab6d

Completed | 86 / 86

Overview

Started - Ended
Mar 01, 21:20:33 - Mar 01, 21:21:18

Runtime
45s

Directory
/Users/araghuka...-bf4b-8e6a4c7eab6d

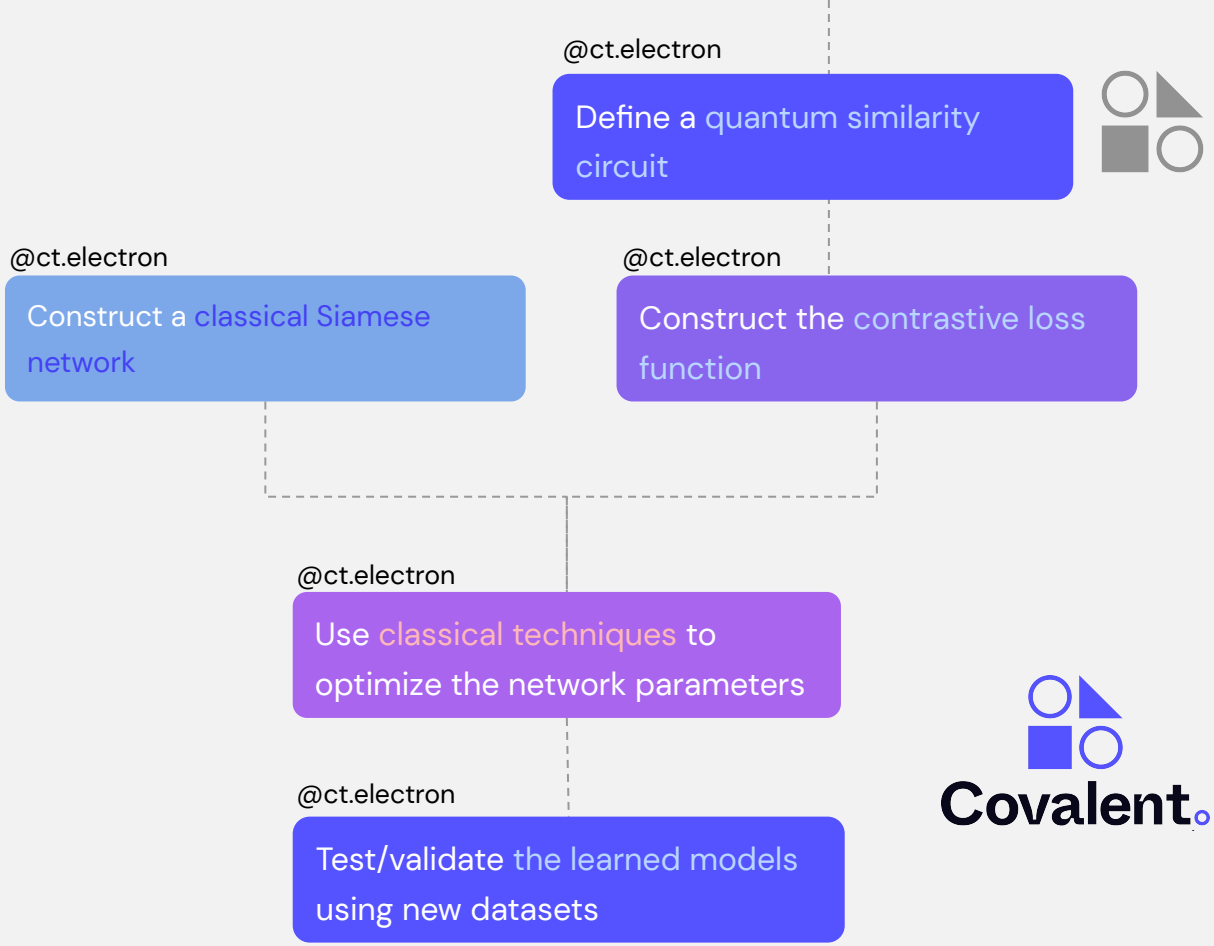
Input
{'args': (), 'kwargs': {'train_path': '/Users/a

Result
(100.0, 91.0, SiameseNetwork(\n (cnn1): Sequen

Executor: **dask**

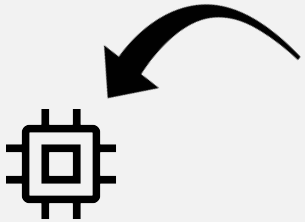
```
@ct.lattice
def quantum_classical_workflow(train_path, test

    # Run the classical sublattice
    results_classical, loss_history_classical,
        classical_workflow(train_path=pixel_t
                           test_path=pixel_te
                           train_batch=train_
                           pytorch_optimizer=
                           epochs=epochs,
                           lr=lr_c,
                           print_intermediate=
```

Overview



Covalent.

Overview



Can dispatch specific @ct.electrons to *different* remote devices

@ct.electron

Define a quantum similarity circuit

@ct.electron

Construct a classical Siamese network

@ct.electron

Construct the contrastive loss function

@ct.electron

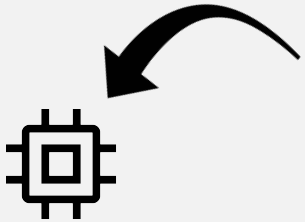
Use classical techniques to optimize the network parameters

@ct.electron

Test/validate the learned models using new datasets

Covalent.

Overview



Can dispatch specific @ct.electrons to *different* remote devices

@ct.lattice

Call all electrons in the desired order to create a **heterogeneous** workflow

@ct.electron

Define a quantum similarity circuit

@ct.electron

Construct a classical Siamese network

@ct.electron

Construct the contrastive loss function

@ct.electron

Use classical techniques to optimize the network parameters

@ct.electron

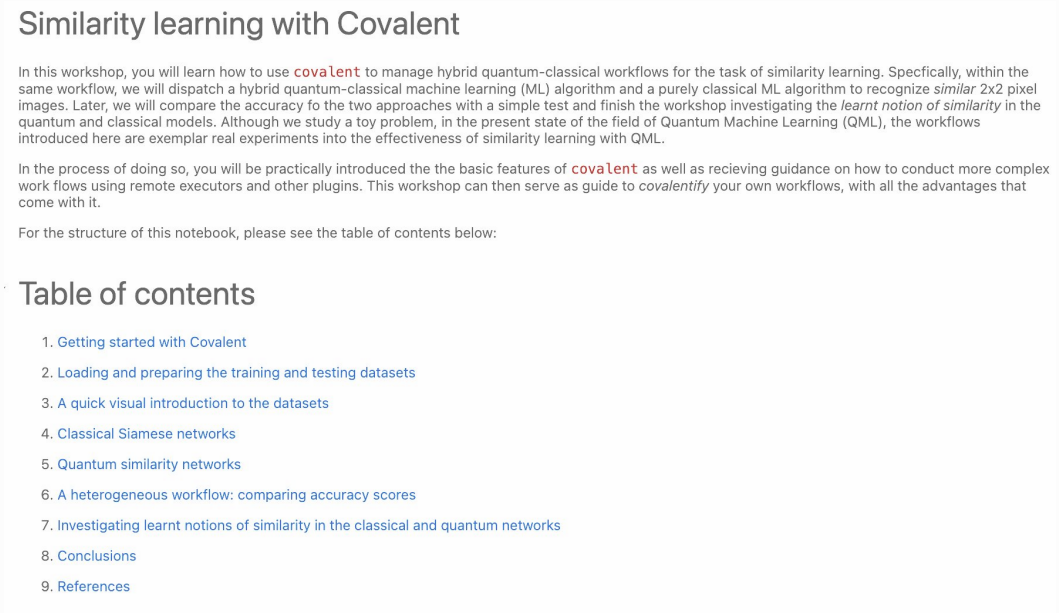
Test/validate the learned models using new datasets

Covalent.

The workshop

- Use classical and quantum methods for similarity learning and compare them.
- Let us know about your own workflows!

similarity_learning.ipynb



Similarity learning with Covalent

In this workshop, you will learn how to use `covalent` to manage hybrid quantum-classical workflows for the task of similarity learning. Specfically, within the same workflow, we will dispatch a hybrid quantum-classical machine learning (ML) algorithm and a purely classical ML algorithm to recognize *similar* 2x2 pixel images. Later, we will compare the accuracy fo the two approaches with a simple test and finish the workshop investigating the *learnt notion of similarity* in the quantum and classical models. Although we study a toy problem, in the present state of the field of Quantum Machine Learning (QML), the workflows introduced here are exemplar real experiments into the effectiveness of similarity learning with QML.

In the process of doing so, you will be practically introduced the the basic features of `covalent` as well as recieving guidance on how to conduct more complex work flows using remote executors and other plugins. This workshop can then serve as guide to *covalentify* your own workflows, with all the advantages that come with it.

For the structure of this notebook, please see the table of contents below:

Table of contents

1. Getting started with Covalent
2. Loading and preparing the training and testing datasets
3. A quick visual introduction to the datasets
4. Classical Siamese networks
5. Quantum similarity networks
6. A heterogeneous workflow: comparing accuracy scores
7. Investigating learnt notions of similarity in the classical and quantum networks
8. Conclusions
9. References

Covalent.

AgnostiqHQ/covalent ⭐

agnostiq.ai

@agnostiqHQ

covalent.xyz

Thanks for listening!