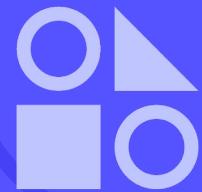


agnostiq

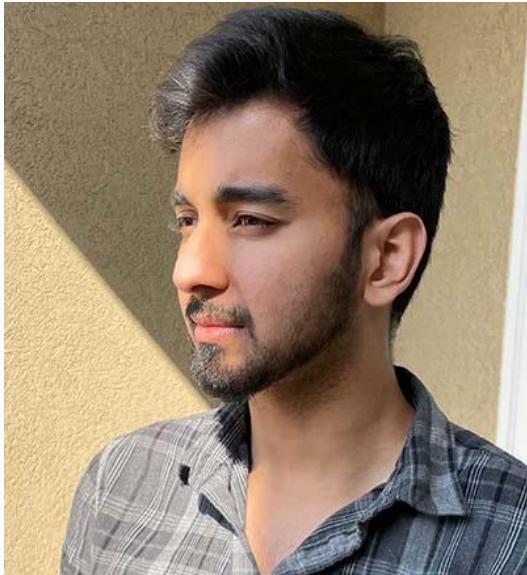


Covalent.

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
 - a. Multiple dispatches, sub-lattices, results
 - b. Covalent Executors
 - c. C/C++ integration (Classical HPC)
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

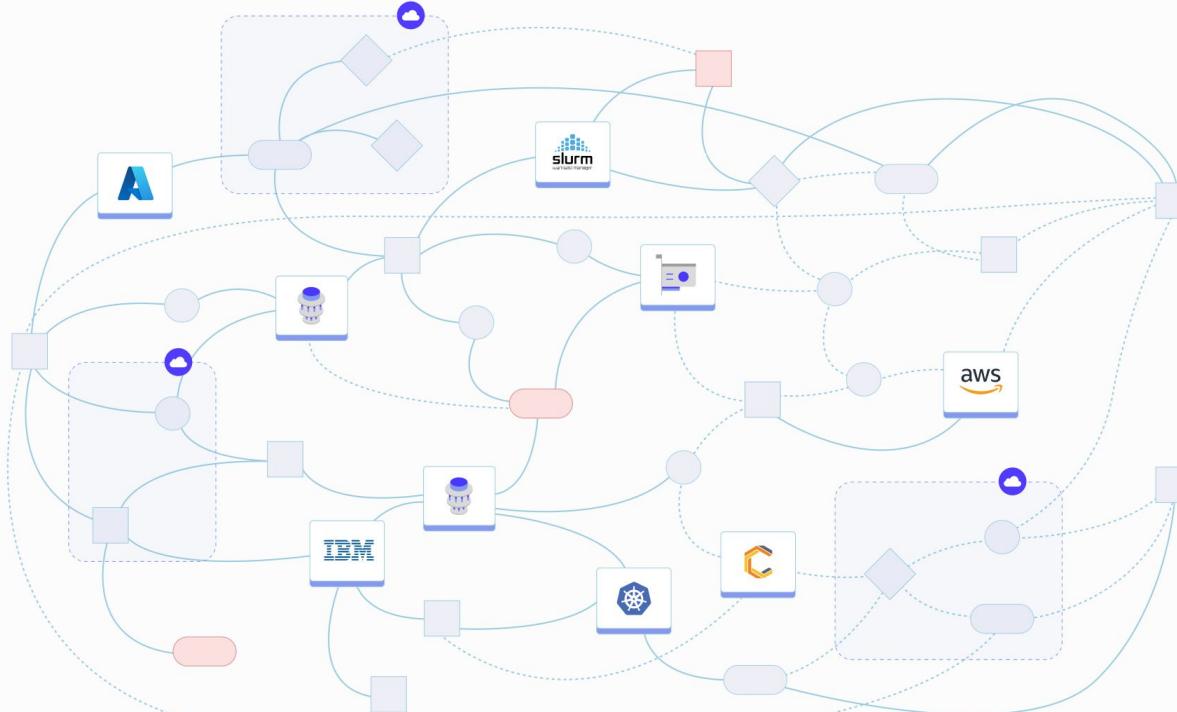
Additional Operational Challenges:

- 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

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



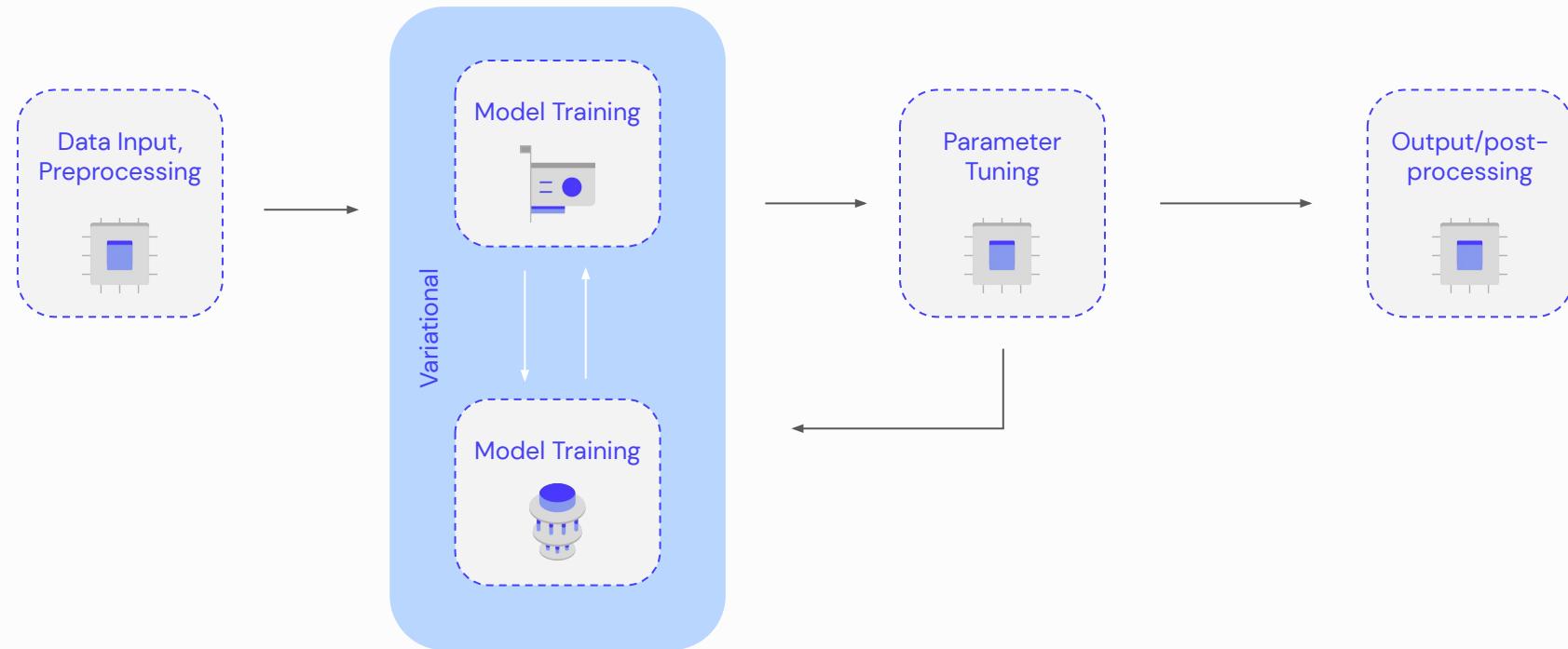
Modern computational workflows are complex.



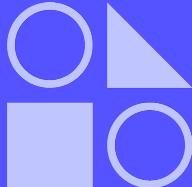


Quantum is driving the need for more heterogeneity

A simplified quantum machine learning workflow:





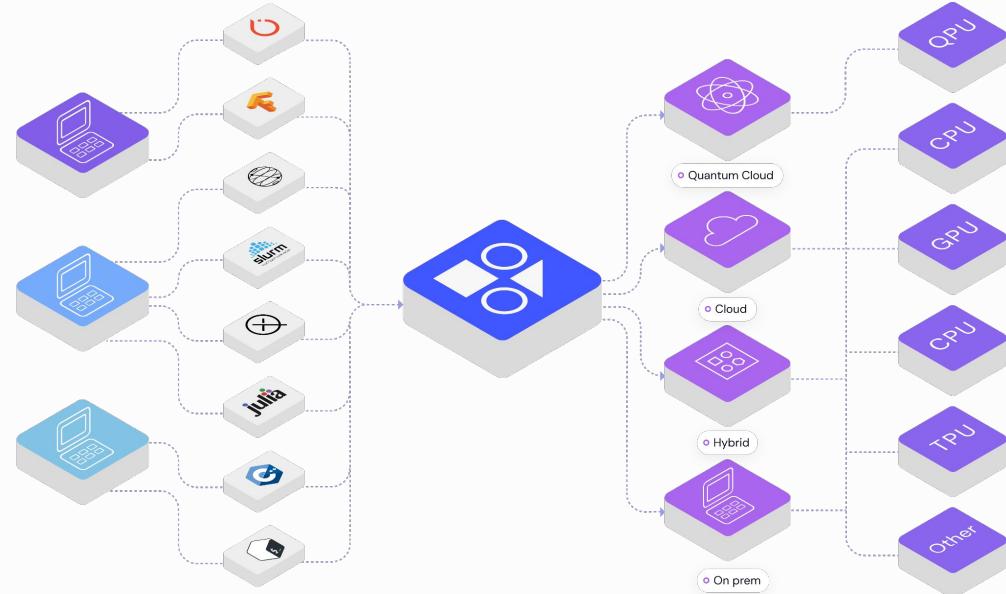
The background features several large, semi-transparent blue geometric shapes: a circle on the left, a triangle pointing down on the left, a large triangle pointing up on the right, and a circle on the right. These shapes overlap each other and the central text area.
The logo icon is composed of four white shapes arranged in a cross-like pattern. It includes a square at the bottom, a triangle pointing up to its right, and two circles, one at the top-left and one at the bottom-right.
Covalent.



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



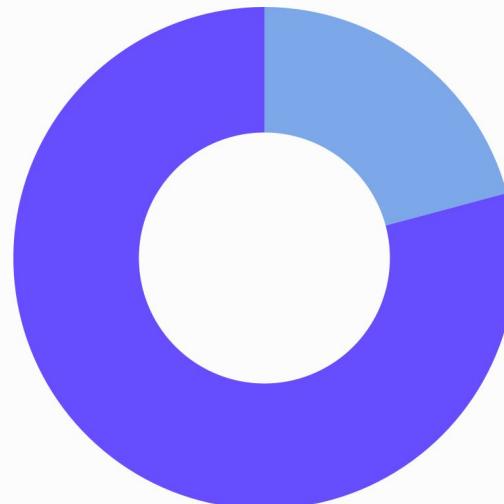
AgnostiqHQ/covalent

covalent.xyz



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



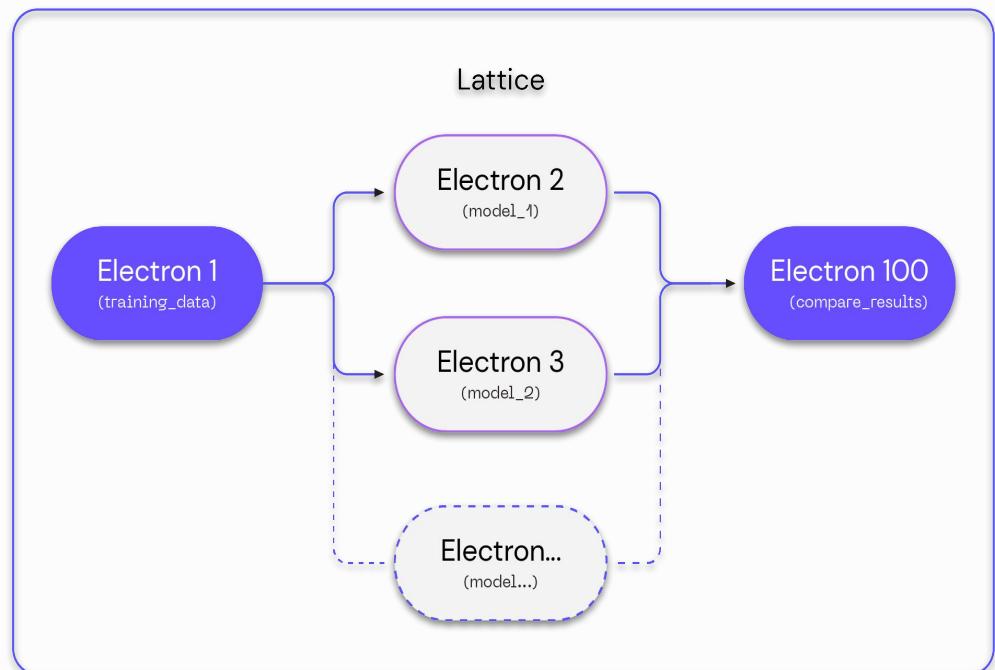


Covalent 101: The Basics



Covalent Concepts

- Workflows are comprised of inter-dependent tasks
- Workflows are called “lattices”
- Tasks are called “electrons” – these are fundamental blocks
- Tasks can be written in other languages
- Tasks can each run on different hardware backends





Minimal code changes required



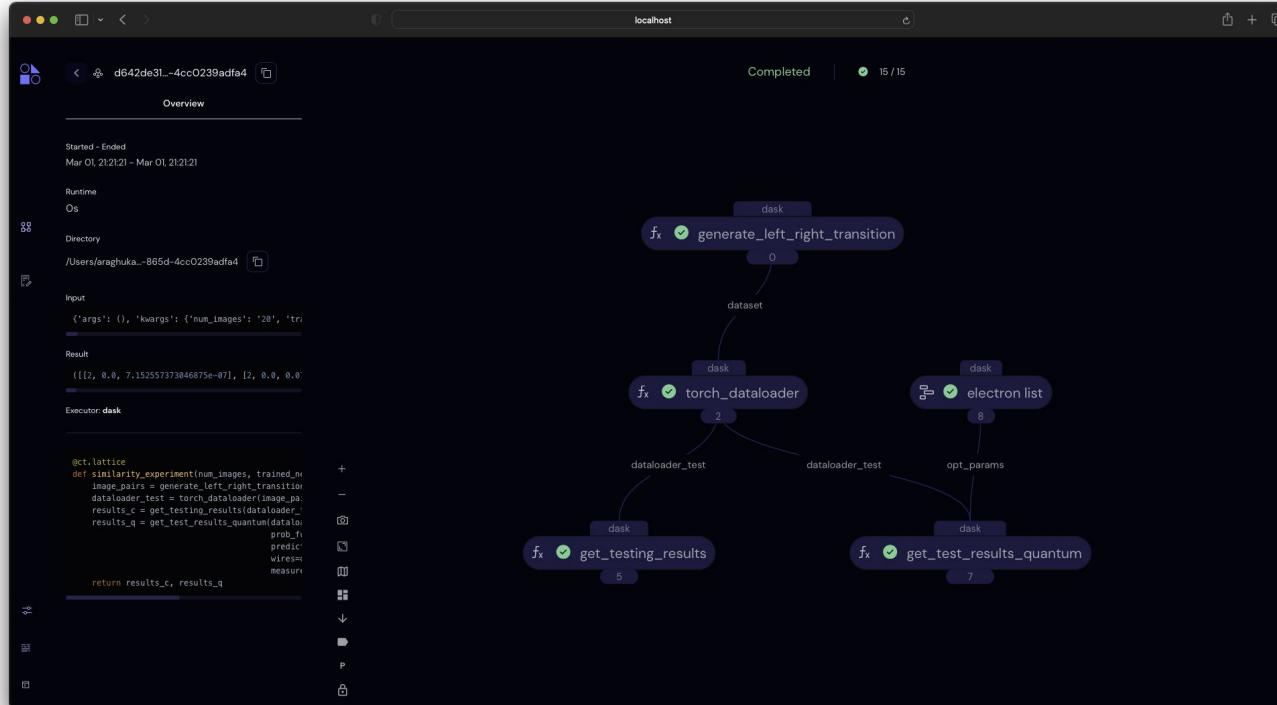
```
1 def optimize(x, y):
2     return min(x, y)
3
4 def task(x, y):
5     z = optimize(x, y)
6     return z
7
8 def train(model):
9     import time
10    time.sleep(1)
11    return model
12
13 def workflow(inputs):
14     model = task(inputs[0], inputs[1])
15     return train(model)
16
17 print(workflow([1, 2]))
18
```



```
1 import covalent as ct
2
3 def optimize(x, y):
4     return min(x, y)
5
6 @ct.electron
7 def task(x, y):
8     z = optimize(x, y)
9     return z
10
11 @ct.electron
12 def train(model):
13     import time
14     time.sleep(1)
15     return model
16
17 @ct.lattice
18 def workflow(inputs):
19     model = task(inputs[0], inputs[1])
20     return train(model)
21
22 dispatch_id = ct.dispatch(workflow)([1, 2])
23 print(ct.get_result(dispatch_id, wait=True))
24
```



Covalent UI





Visualize and monitor complex workflows

The screenshot shows the Covalent web interface for monitoring and visualizing complex workflows.

Overview Panel:

- Status:** Completed (52 / 52)
- Started - Ended:** Mar 01, 21:43:39 – Mar 01, 21:43:50
- Runtime:** 9s
- Input:** {'args': (), 'kwargs': {'train_path': '/Users/.../b95f-53e5la7/e4c6b'}}
- Result:** ([[0.0, 0.0, 0.3213415131956775], [1.0, 1.0, 0.]])
- Executor:** task

Code View:

```
@ct.lattice
def quantum_workflow(train_path, test_path, trai...
```

Detailed Task View (train_quantum_net):

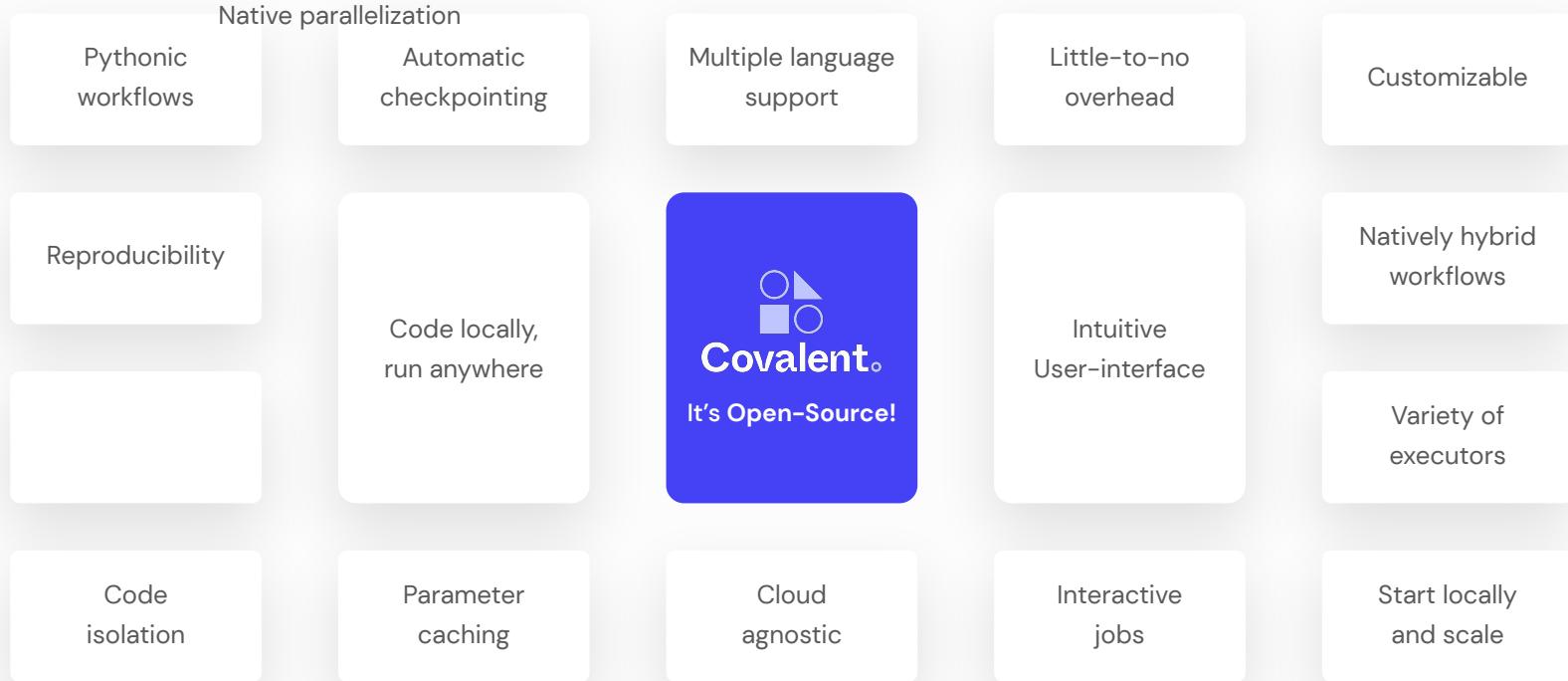
- Status:** Completed
- Started - Ended:** Mar 01, 21:43:40 – Mar 01, 21:43:49
- Runtime:** 9s
- Input:** {'args': (), 'kwargs': {'dataloader_train': ...}}
- Result:** ([0.2734103202819824, 0.6892913746833801, ...])
- Executor:** task

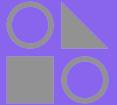
Code View (train_quantum_net):

```
@ct.electron
def train_quantum_net(llr, init_params, do...
```



You can contribute too!





Code Samples

Example 1: Curve Fitting



SETUP

- Clone the Github repository

```
git clone https://github.com/AgnostiqHQ/tutorials_covalent_aps_march_2023.git
```

- Download [Miniconda](#)
- Install the conda environment (`environment.yml` in root directory)

```
conda env create -f environment.yml
```

- Activate environment

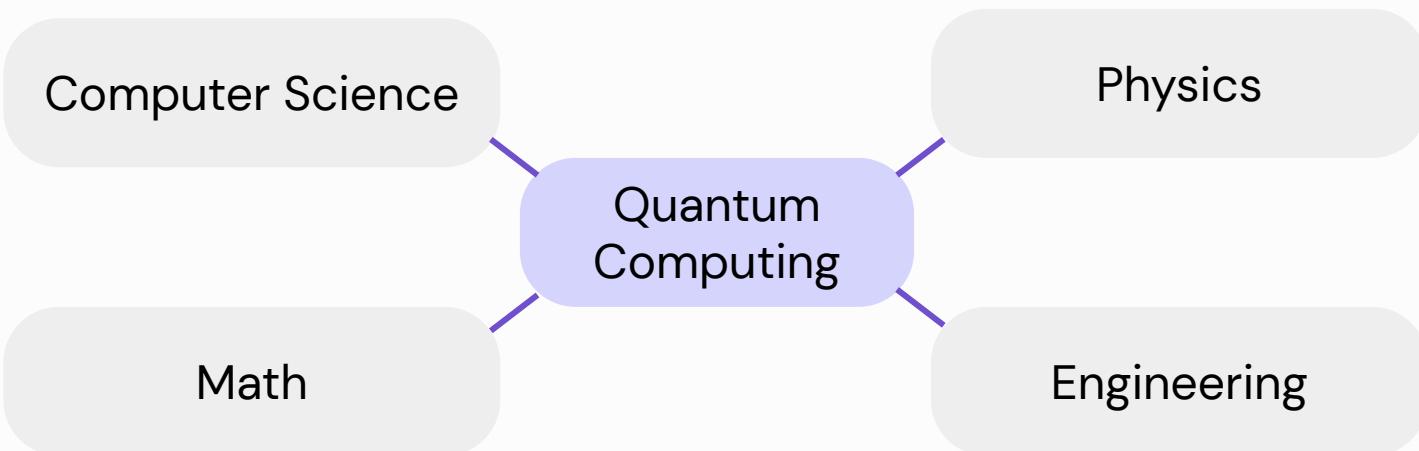
```
conda activate aps_march_covalent
```

- Add to kernel list

```
python -m ipykernel install --user --name=aps_march_covalent
```



A Brief Introduction to Quantum Machine Learning



Key

Core discipline



Data Science

Machine
Learning

Materials

Computer Science

Finance

Physics

Cryptography

Quantum Computing

Mathematics

Engineering

Drug
Development

Key

Quantum
Chemistry

Core discipline

Biology

Application



Data Science

Machine Learning

Materials

Computer Science

Finance

Physics

Cryptography

Quantum Computing

Mathematics

Engineering

Drug Development

Key

Quantum Chemistry

Core discipline

Biology

Application

aQ

covalent.xyz

86



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



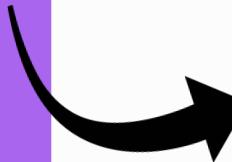
ML: The construction of predictive models without *explicit instruction*.



Model parameters are tuned while being fed a training dataset



The predictive capability of a model is assessed using a testing dataset

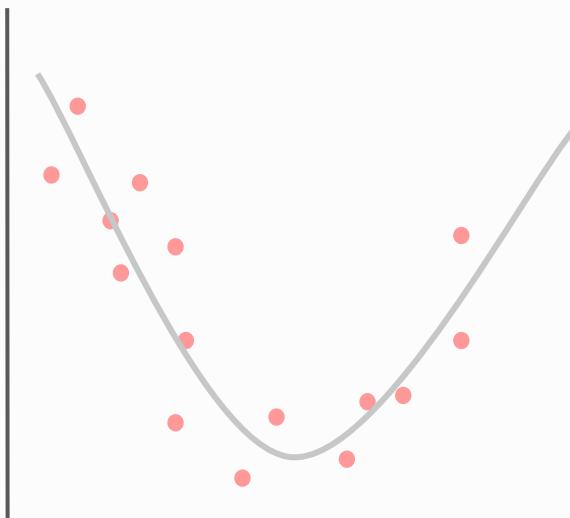


Often, validation is performed between training and testing



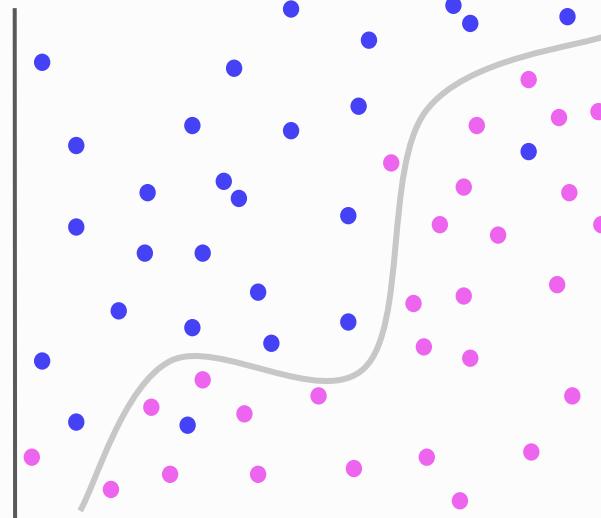
What is Quantum Machine Learning?

Regression



Logistic regression, random forest...

Classification



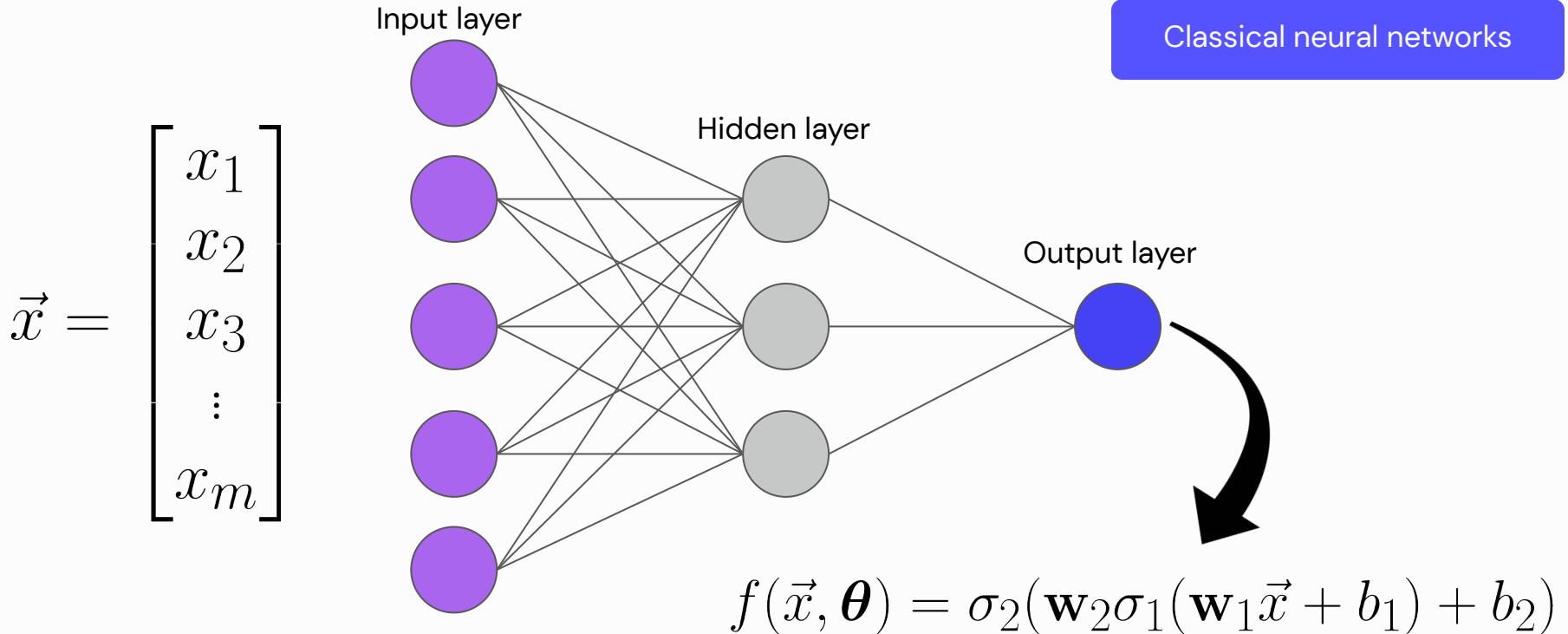
Support vector machines, autoencoders....



*Just two of the most common applications



What is Quantum Machine Learning?

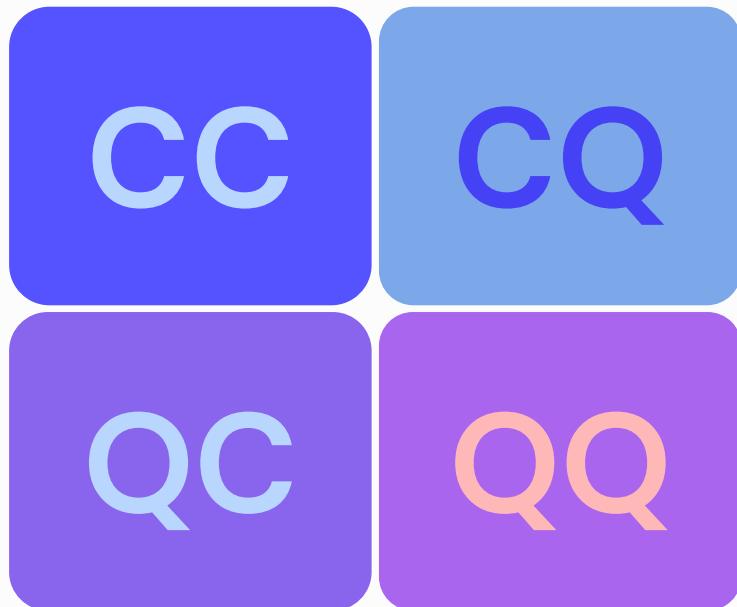




What is Quantum Machine Learning?

Data processing paradigm

Source of data





What is Quantum Machine Learning?

Source of data

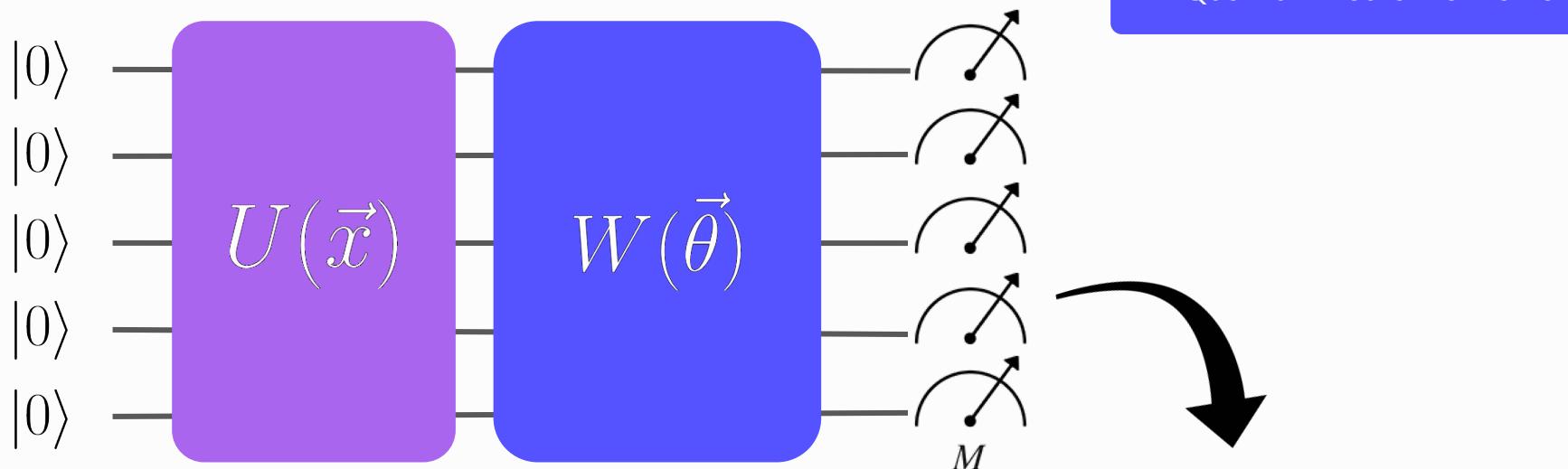


Data processing paradigm

Now popular to work with CQ, but *how does this compare with CC?*



What is Quantum Machine Learning?



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



Classical and Quantum Similarity Learning



What is similarity learning?



Similarity learning is the act of 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

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

\vec{x}_i

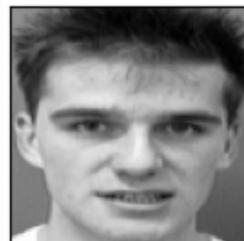


\vec{x}_j



y_{ij}

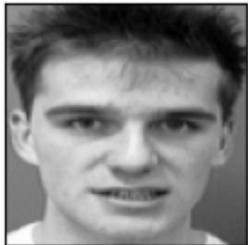
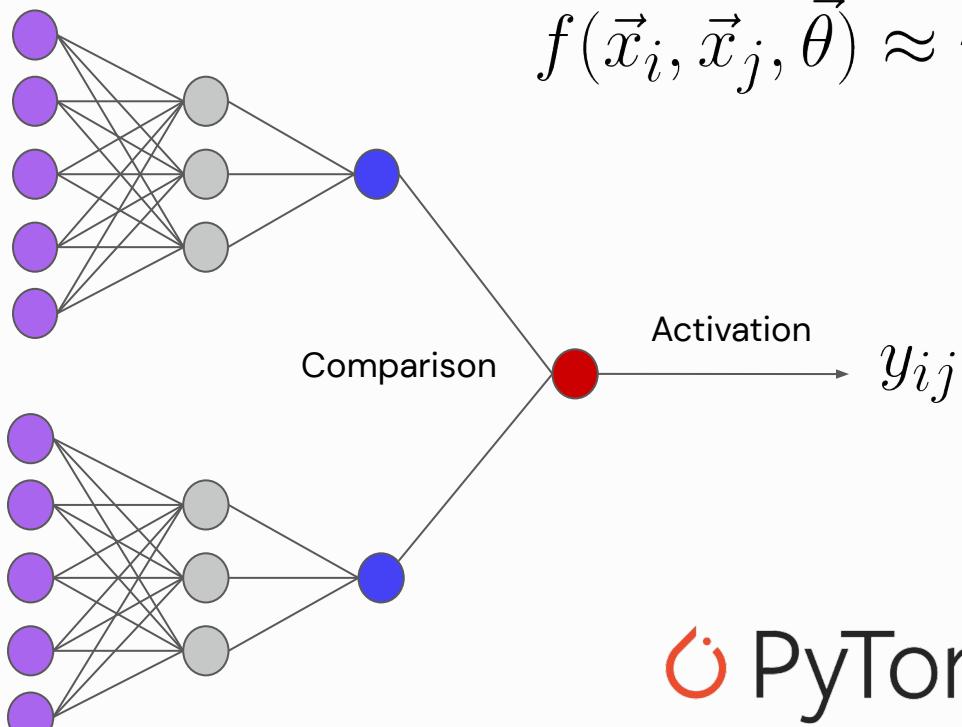
0 (similar)



1 (dissimilar)



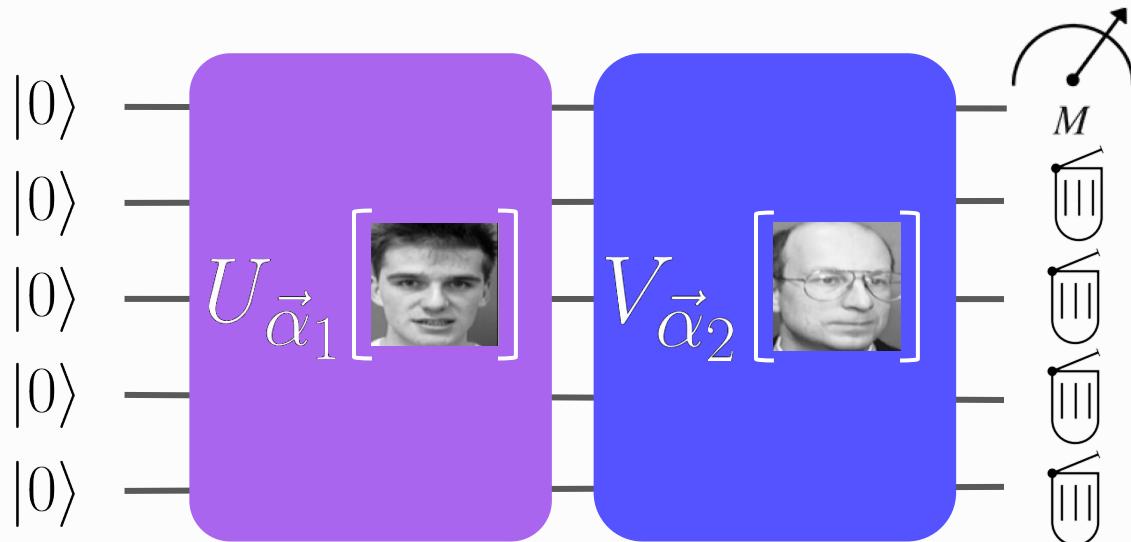
Classical Siamese Network

 \vec{x}_i  \vec{x}_j 

PyTorch



Quantum Similarity Network



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

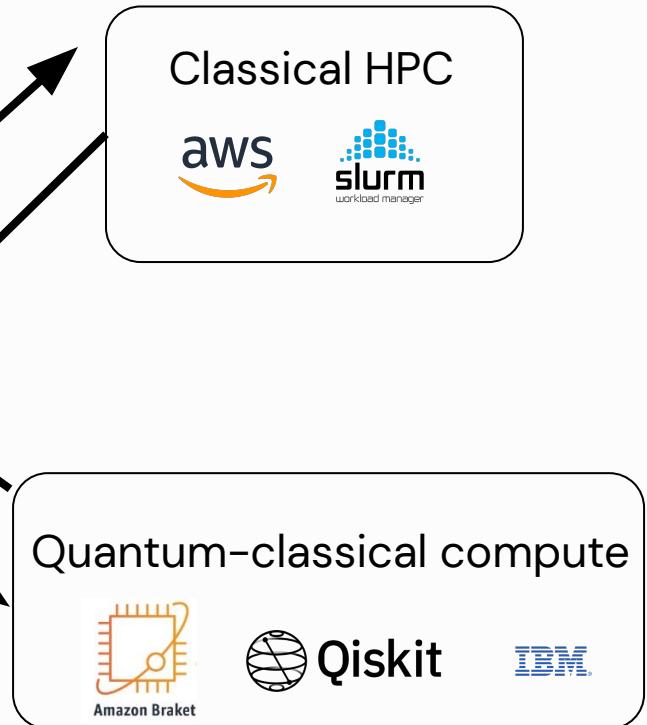
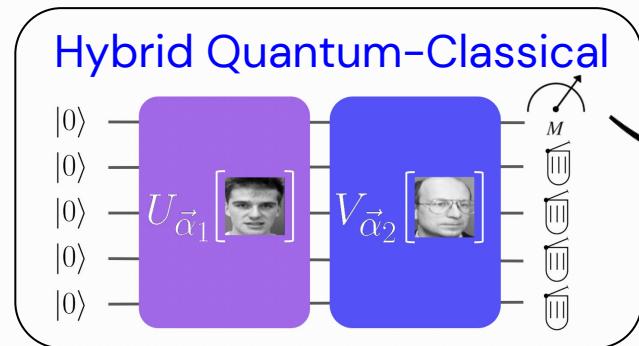
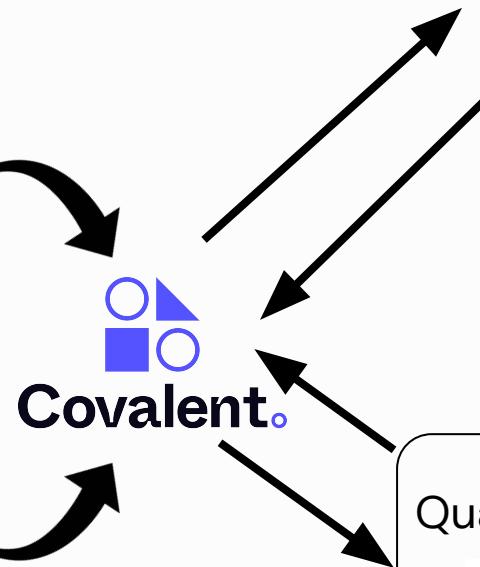
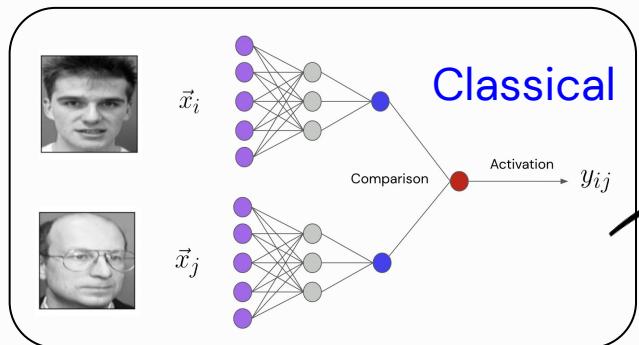
Can be as simple as the probability of measuring a single qubit in the 1 or 0 state.

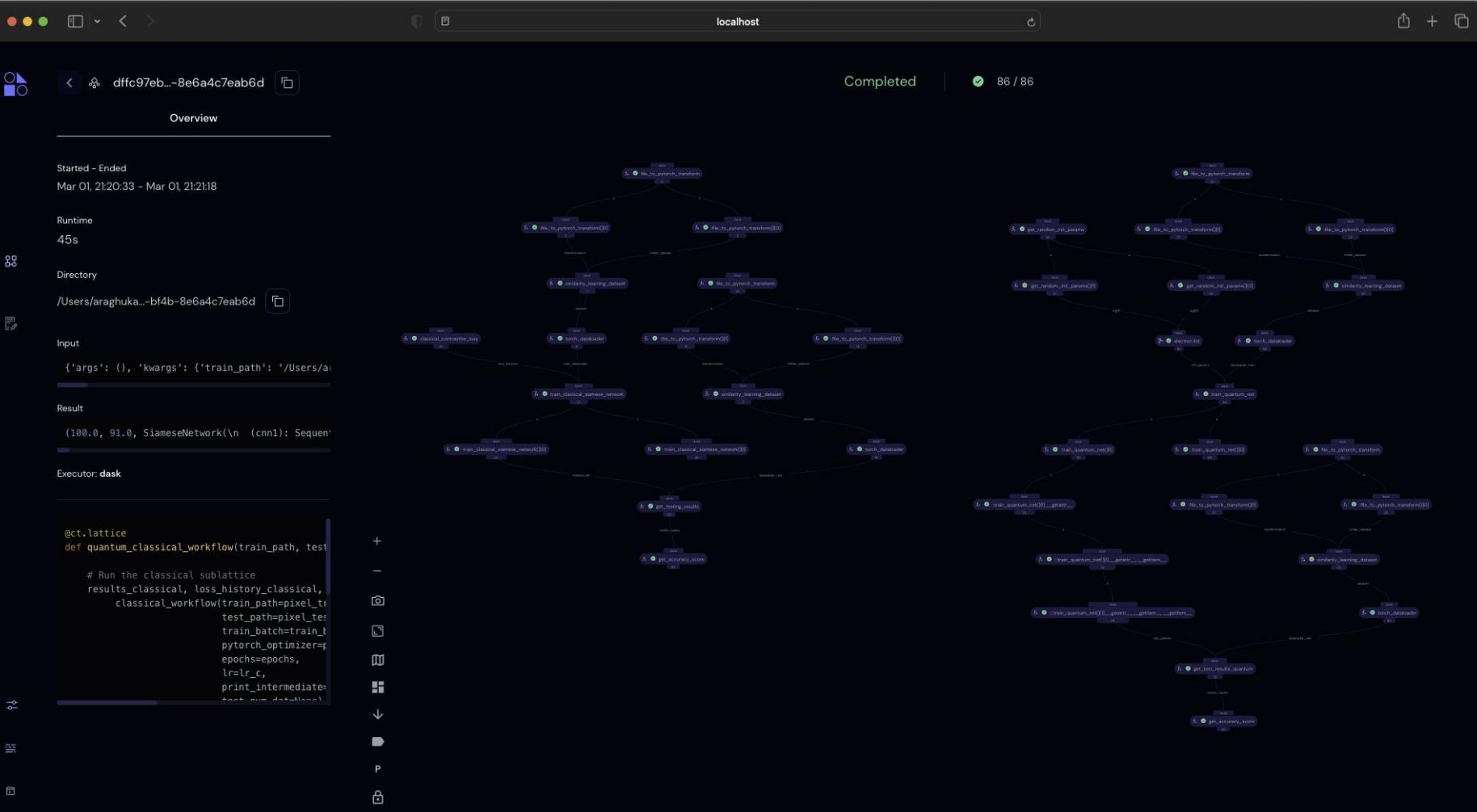


Radha, S.K. and Jao, C., 2022. Generalized quantum similarity learning.
arXiv preprint [arXiv:2201.02310](https://arxiv.org/abs/2201.02310).



Enter Covalent





Modularize your experiments

@ct.electron

Define a quantum similarity circuit

@ct.electron

Construct a classical Siamese network

@ct.electron

Construct the contrastive loss function

@ct.electron

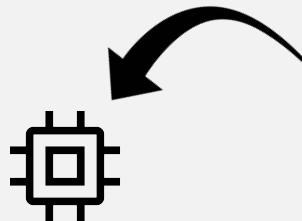
Use classical techniques to optimize the network parameters

@ct.electron

Test/validate the learned models using new datasets



Modularize your experiments



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

`@ct.electron`

Construct a classical Siamese network

`@ct.electron`

Define a quantum similarity circuit

`@ct.electron`

Construct the contrastive loss function

`@ct.electron`

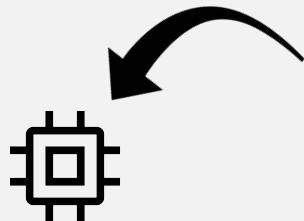
Use classical techniques to optimize the network parameters

`@ct.electron`

Test/validate the learned models using new datasets



Modularize your experiments



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`

Construct a **classical Siamese network**

`@ct.electron`

Define a quantum similarity circuit

`@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



The workshop



- 1 hour 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. Specifically, 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 to 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 receiving 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



- [AgnostiqHQ/covalent ★](https://github.com/AgnostiqHQ/covalent)
- agnostiq.ai
- [@agnostiqHQ](https://twitter.com/agnostiqHQ)



covalent.xyz

Thanks for listening!