OAK RIDGE National Laboratory

Quantum Computing User Program at Oak Ridge National Laboratory

Presented by: Jordan Winetrout

HPC Engineer

QCUP User Assistance

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

The Quantum Computing User Program is supported by the DOE Advanced Scientific Computing Research (ASCR) program office

U.S. DEPARTMENT OF ENERGY

Quantum Computing User Program is a first step to discovery

Enable Research

Provide a broad spectrum of user access to the best available quantum computing systems

Evaluate Technology

Monitor the breadth and performance of early quantum computing applications

Engage Community

Support growth of the quantum ecosystem by engaging with users, developers, vendors, and providers

OAK RIDGE National Laboratory

olcf.ornl.gov

Our QCUP Team Today

Ashley Barker

QCUP Director

Travis Humble

QCUP Advisor

Josh Cunningham

QCUP Management

Claire Marvinney

QCUP Management

Suzanne Prentice

QCUP Software Services

Cara Kennedy

QCUP Accounts

Misty Abston

QCUP Accounts

Katie Bethea

QCUP Accounts + Communications

Chris Fuson

QCUP UA

Michael Sandoval

QCUP UA

Jordan Winetrout

QCUP UA

Ryan Landfield

QCUP Science Liaison

Alessandro Baroni

QCUP Science Liaison

Antonio Coelle Perez

QCUP Science Liaison

Peter Groszkowski

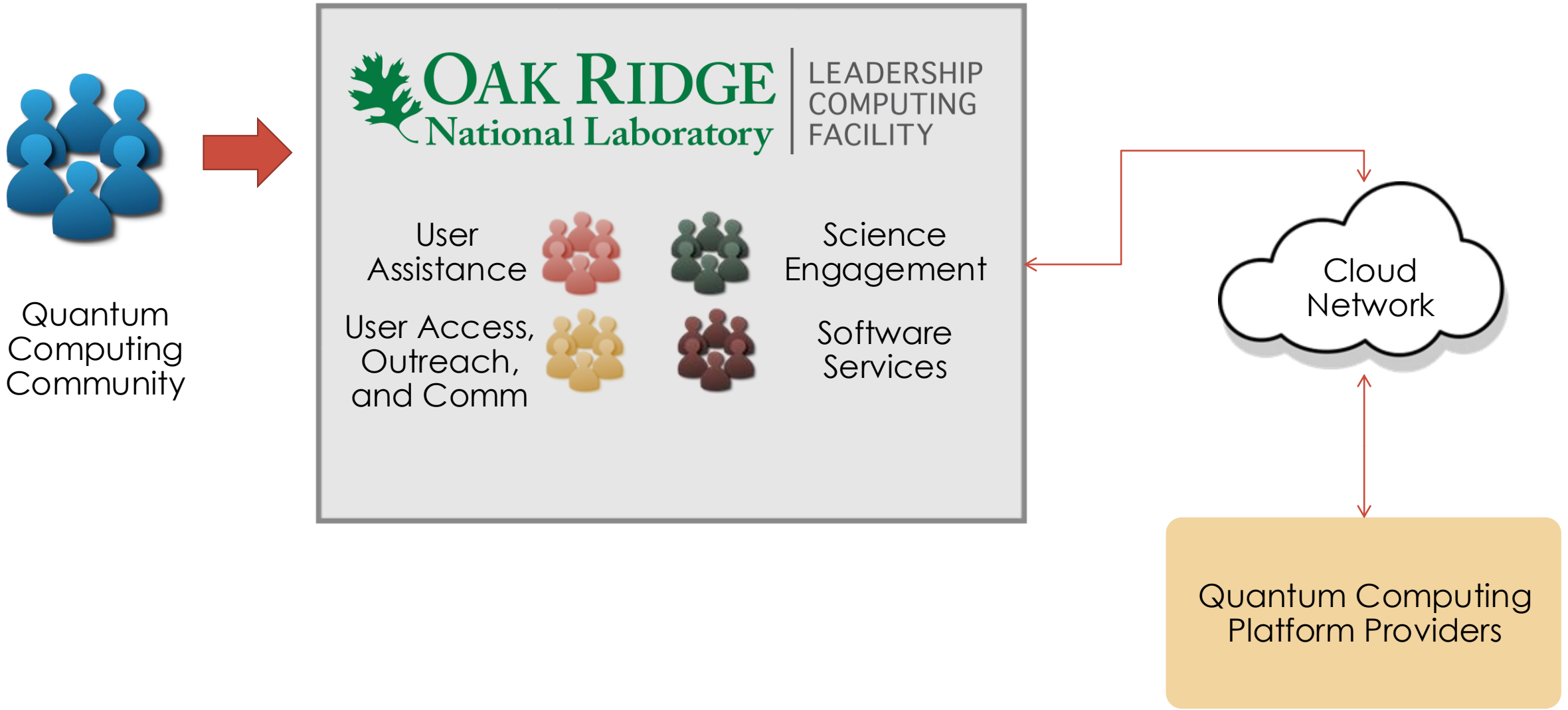
QCUP Science Liaison

In-Saeng Suh

QCUP Technology Integration

And other contributors to the program within OLCF

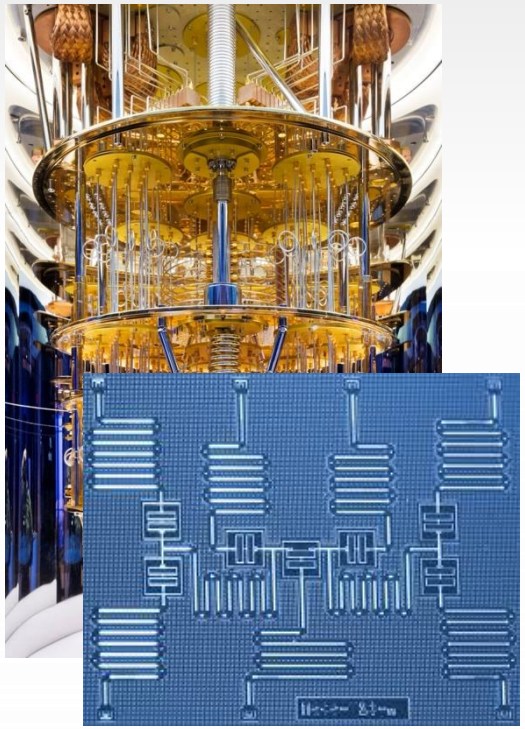
QCUP Operations Model: Cloud Access



Quantum Computing User Program Resources

IBM

- 6 superconducting transmon systems, up to 156 qubits.

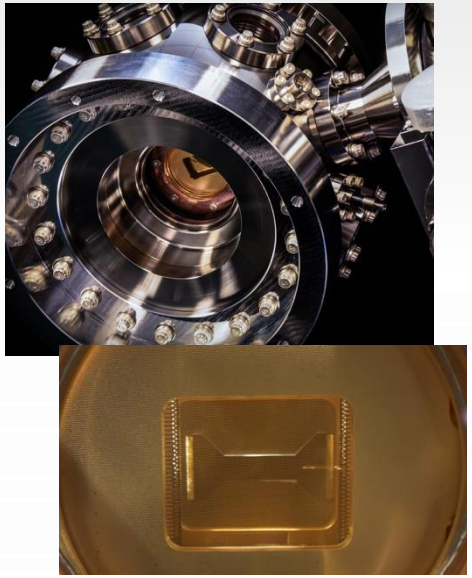


IQM

- 3 superconducting transmon systems, up to 54 qubits.

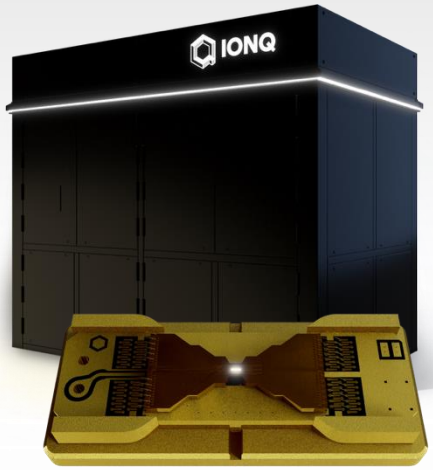
Quantinuum

- 3 trapped-ion systems, up to 98 qubits.



IonQ

- 2 trapped-ion systems, up to 36 qubits.



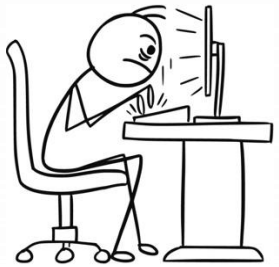
Total system count is 14

How do I get access to Quantum computers?

What are the steps to request access?

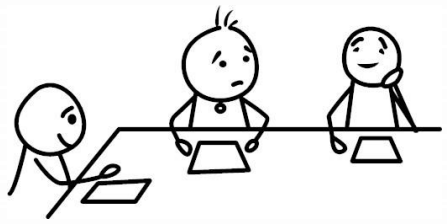
Project Request

- PI submits a proposal describing merit of idea and why it requires access to QCUP resources
- Online collects essential information
- Email notification of successful submission
- Available at olcf.ornl.gov



Project Review

- Quantum Resource Utilization Council (QRUC) receives proposals.
- QRUC reviews proposal for feasibility and merit.
- Additional review for export control, data sensitivity, user agreements



Project Award

- PI is notified that access to the system has been awarded.
- PI is notified of the allocation size, as warranted.
- PI receives unique project ID

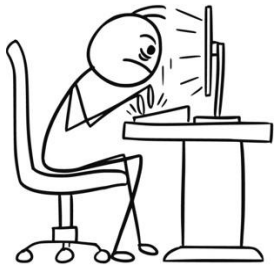
User Request

- PI is evaluated as potential system user
- PI authorizes other user account requests
- Accounts vets users for export control, sensitive information, etc.
- OLCF notifies users of account creation.

What are the steps to request time on hardware?

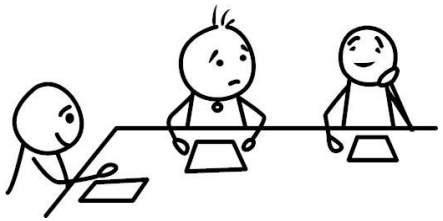
Allocation Request

- PI submits an allocation request, providing justification for resource request
- Online form collects essential information
- Required: time, description of what is to be run, justification for time request
- Available at olcf.ornl.gov



Allocation Review

- Quantum Resource Utilization Council (QRUC) receives the allocation request from User Assistance (UA)
- QRUC reviews allocation request and determines best approach to allocate time, given resources available



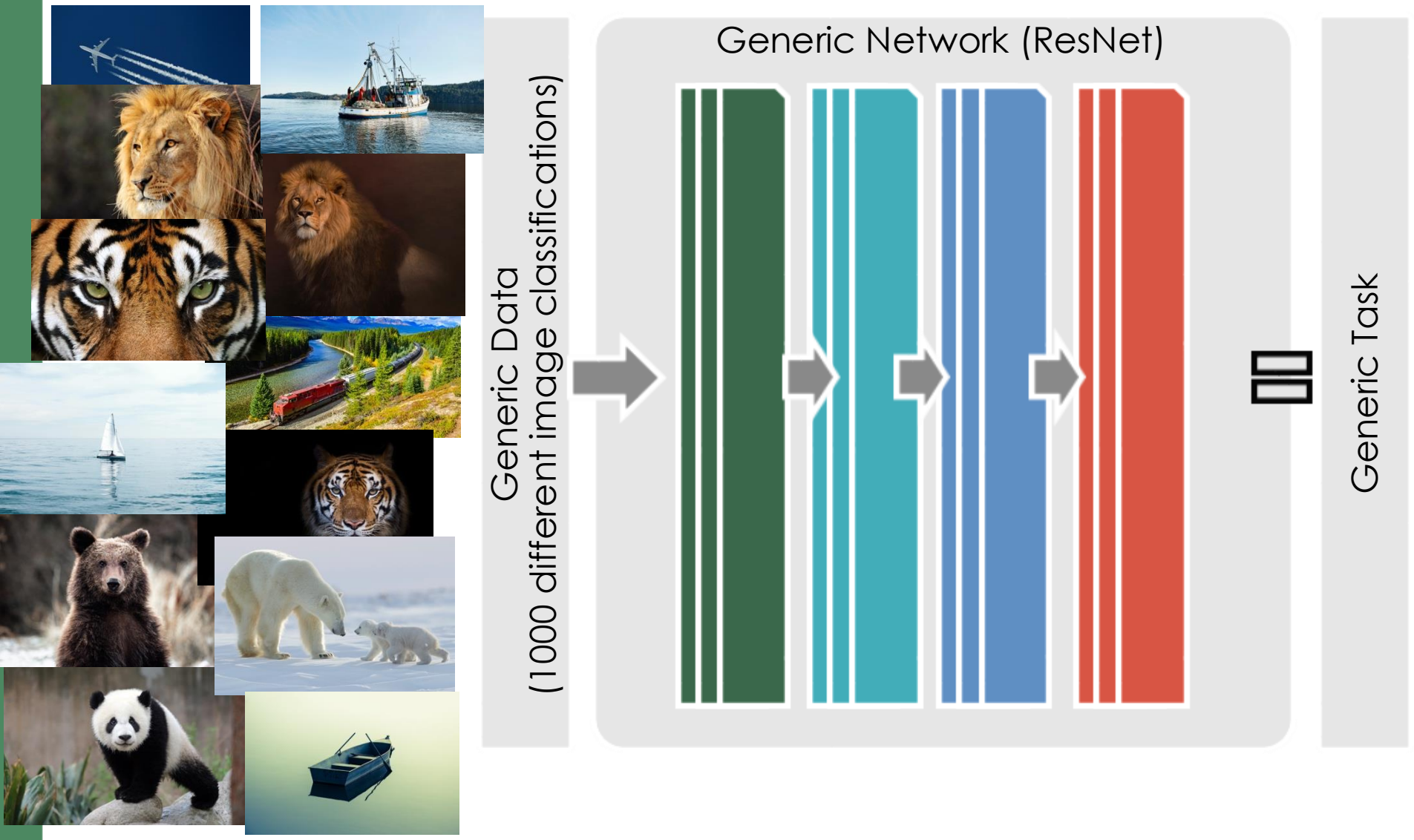
Allocation Received

- PI is emailed by UA about allocation size, and timeline for running their jobs
- UA provides allocation to project
- Project can run their request on hardware

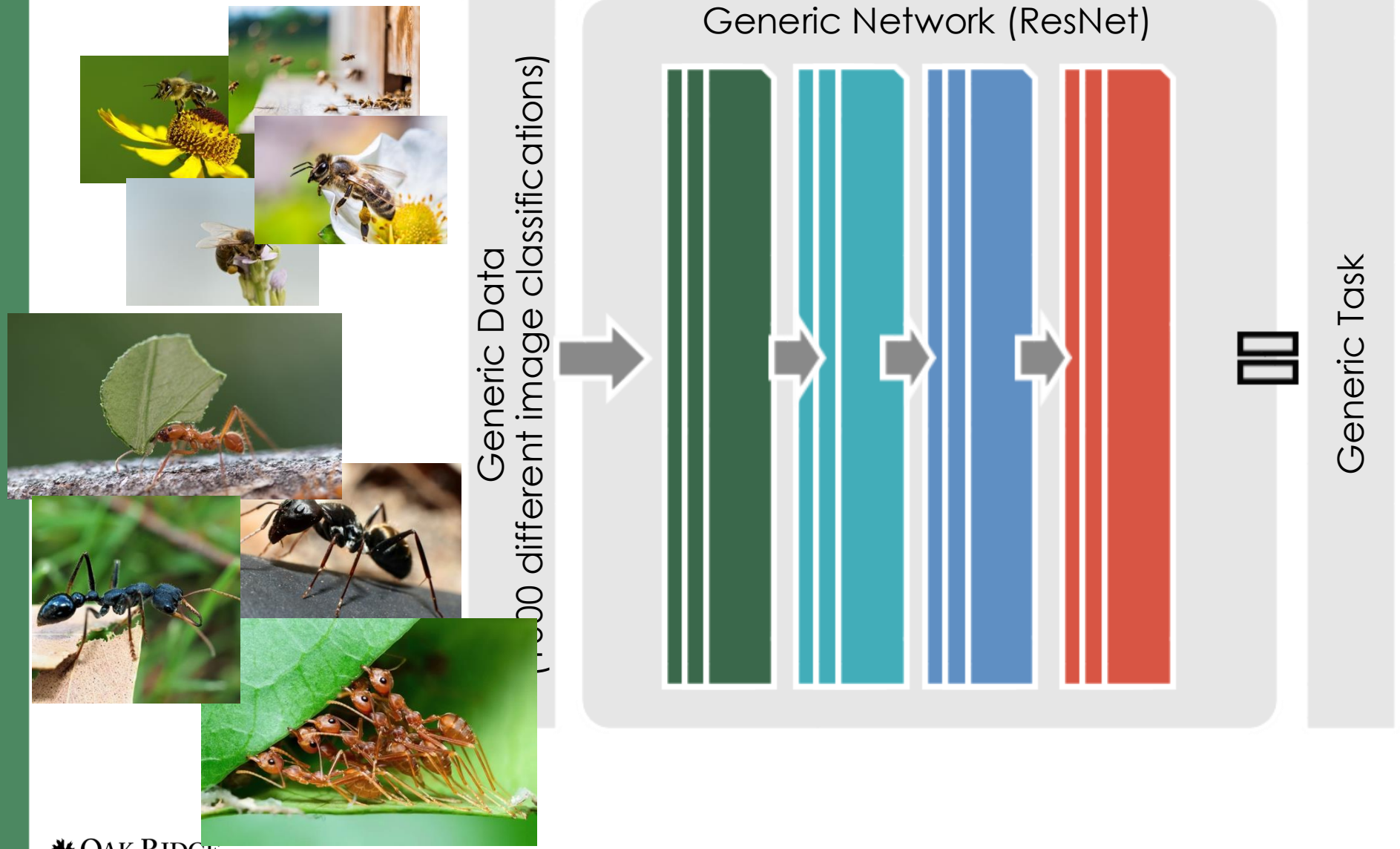
Challenge: Quantum Machine Learning

HPC + AI + QC

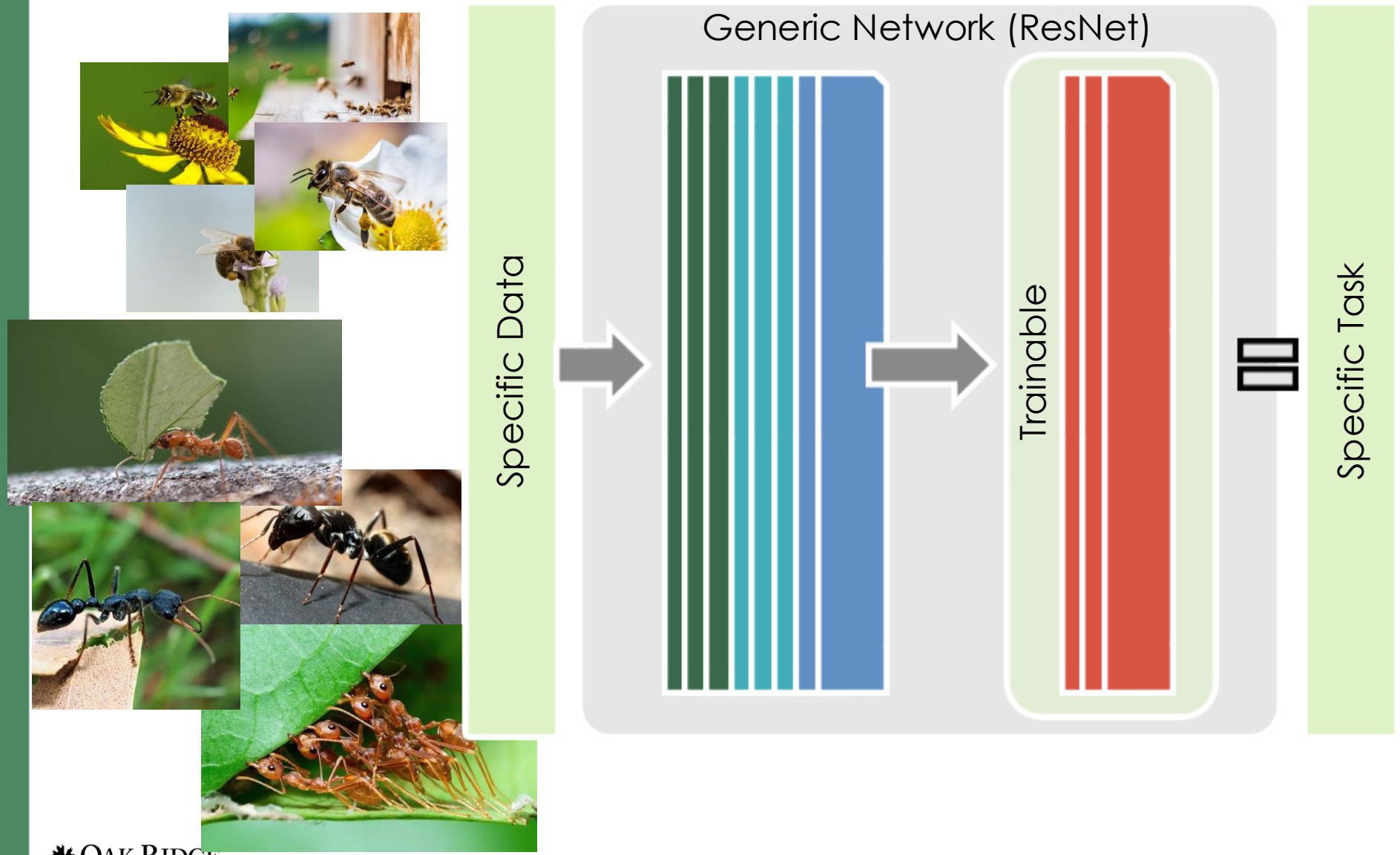
Workflow Overview with Pictures - Transfer Learning



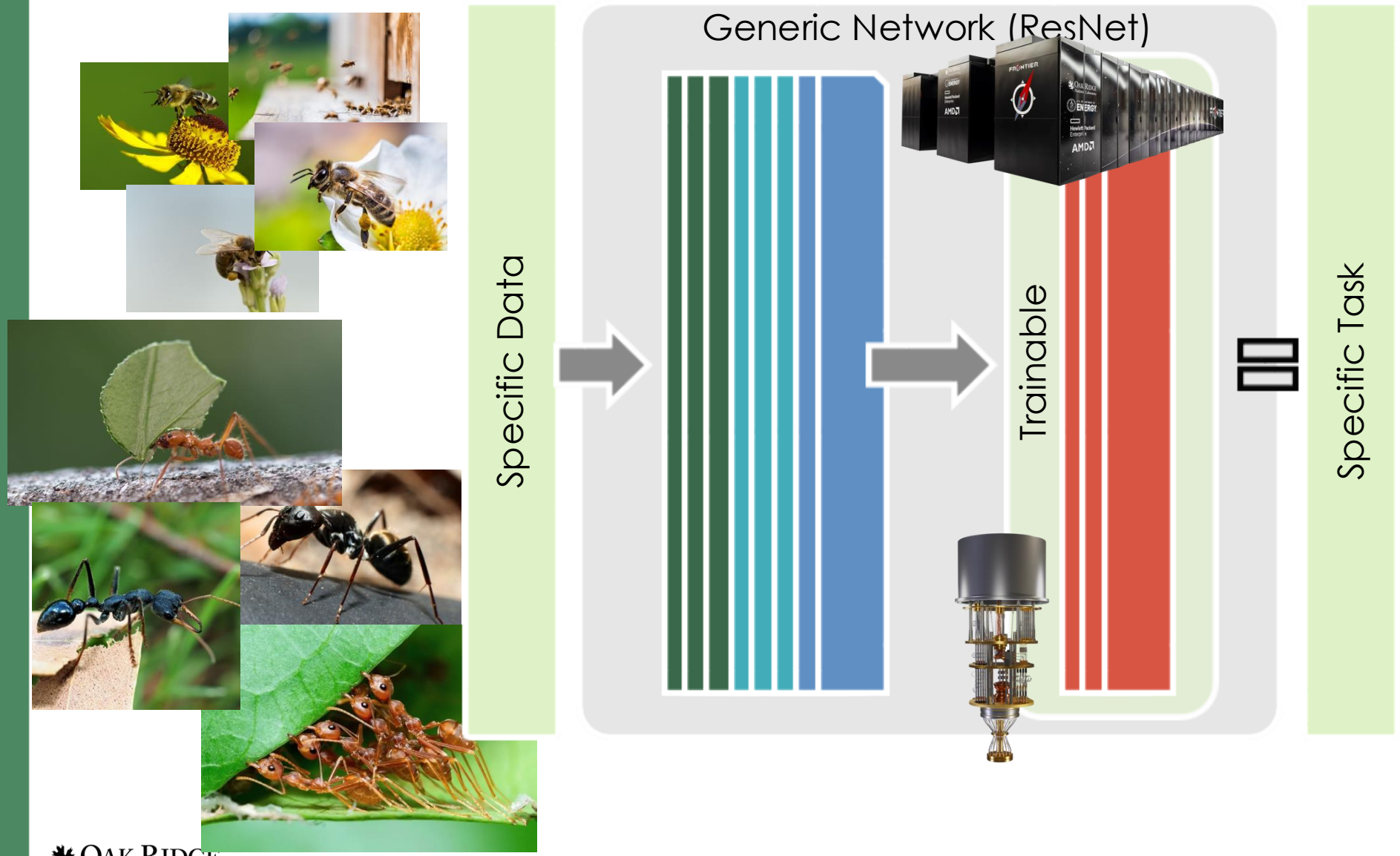
Workflow Overview with Pictures - Transfer Learning



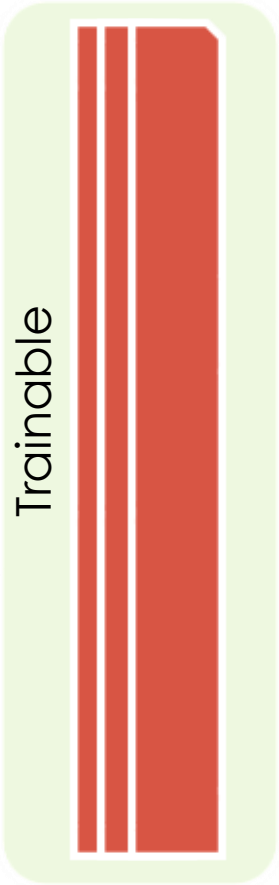
Workflow Overview with Pictures - Transfer Learning



Workflow Overview with Pictures - Transfer Learning



Workflow Overview with Pictures - Transfer Learning

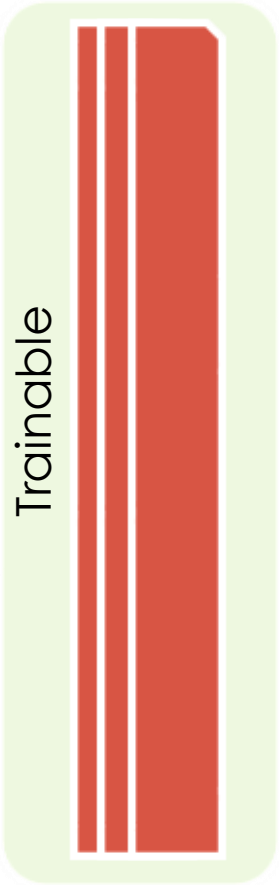


```
# Expectation values in the Z basis
exp_vals = [qml.expval(qml.PauliZ(position)) for position in range(n_qubits)]
return tuple(exp_vals)
```

Must measure the qubits to extract data from the quantum state

- Measurements result in probabilities
- For our code, we take these probabilities and compute **"expectation values"**
- Expectation values are weighted averages of measuring a specific observable
- In our code, computing the expectation value for a qubit results in a value between [-1, 1]
- If we measure 4 qubits, you can get 4 output numbers like: [-.2 , .8, .1, -.3]
- These outputs become input features for the final classical layer (next slide) that converts things to scores/logits
- What this does
- Retrieves data to be transformed later for classification
- You can repeat this over multiple shots to refine the expectation values

Workflow Overview with Pictures - Transfer Learning



The VQC is combined with a classical pre-processing layer and classical post-processing layer

- First classical layer reshapes input features to match number of qubits
- Final classical layer maps the quantum outputs to the number of classes

What this does:

- Compresses initial data to the size of our quantum system
- Converts quantum data back to the size of our classes

```
class DressedQuantumNet(nn.Module):
    """
    Torch module implementing the *dressed* quantum net.
    """

    def __init__(self,device):
        """
        Definition of the *dressed* layout.
        """

        super().__init__()
        self.pre_net = nn.Linear(512, n_qubits)
        self.q_params = nn.Parameter(q_delta * torch.randn(q_depth * n_qubits))
        self.post_net = nn.Linear(n_qubits, 2)

        self.device=device
```

Workflow Overview

- Compare Classical and Hybrid "QC+HPC" methods for a basic transfer learning problem

1. Freeze ResNet18 model (previously trained on the ImageNet dataset)
2. Replace final layer with Dressed Quantum Circuit
3. Classify subset of ImageNet (ants/bees)
4. 245 training images, 153 validation images
5. Each image has 512 features associated when we take over from the pre-trained model
6. Using 4 qubits, so size transformations looks like: $512 \rightarrow 4 \rightarrow 2$
7. MPI can further split up the work but will affect accuracies
8. Uses PyTorch, mpi4py, and PennyLane
9. Run things using either simulator or real quantum device

Transfer Learning

- In a nutshell, transfer learning is:
  - A machine learning technique where you repurpose a pre-trained model
  - Freeze early layers that identifies general features
  - Let the final layers train on a smaller, related dataset to fine-tune on specific features
  - You're "transferring" the learning from one task and using it for another task

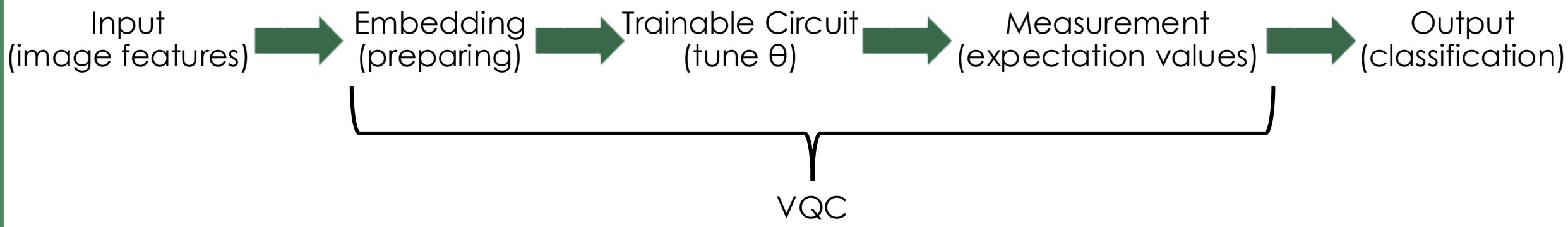
Quantum Motivation

- Classical layers use standard math, while a quantum circuit mixes information using superposition, interference, and entanglement
- This can extract more complex features not capturable by a classical network
- For some natural phenomena, it can be harder for a classical computer to simulate or approximate at scale --nature is inherently "quantum"
- Using a quantum circuit may use less parameters to train than a classical network
- For transfer learning specifically, can live in the realm of smaller qubits because smaller targeted training
- Realistic Note: in this era of quantum computing (QC), a purely classical model can match or beat these "hybrid" QC+Classical demos
- The key takeaway is to explore where quantum could add value, and to pathfind QC/HPC use-cases as hardware improves
- You will be investigating using a variational quantum circuit (VQC) in transfer learning and how viable it is

Variational Quantum Circuits (VQC)

- A VQC is an adjustable quantum program with some knobs you can tune or train
- Knobs: real numbers (parameters)
- Tuning: changing the parameters so the circuit's output helps solve a task (e.g., image classification)
- Quantum programs are composed of quantum "gates"
- So, what are the knobs/parameters?
- Rotation gates require an angle (θ) as input
- Adjusting θ can help minimize loss (classification error)
- Like weights in a neural network, θ values can be trained via machine learning
- A VQC will be a layer we implement within an existing ML model

VQC Process


Input
(image features)
Embedding
(preparing)
Trainable Circuit
(tune θ)
Measurement
(expectation values)
Output
(classification)
VQC

Pennylane Devices

- PennyLane is a cross-platform Python library for quantum computing, quantum machine learning, and quantumbchemistry
- https://pennylane.ai/
- Allows use of different quantum simulators
- Solves expectation values analytically by default (ideal scenario), but can still generate sample
- distributions by passing a number of shots (introduces noise)
- Devices we will be exploring
- lightning.kokkos (let's us run on AMD GPUs through the Kokkos library)
- qiskit.remote (let's us run on real quantum backends)
- For a more complete demonstration of PennyLane running on Frontier, please see recording here: