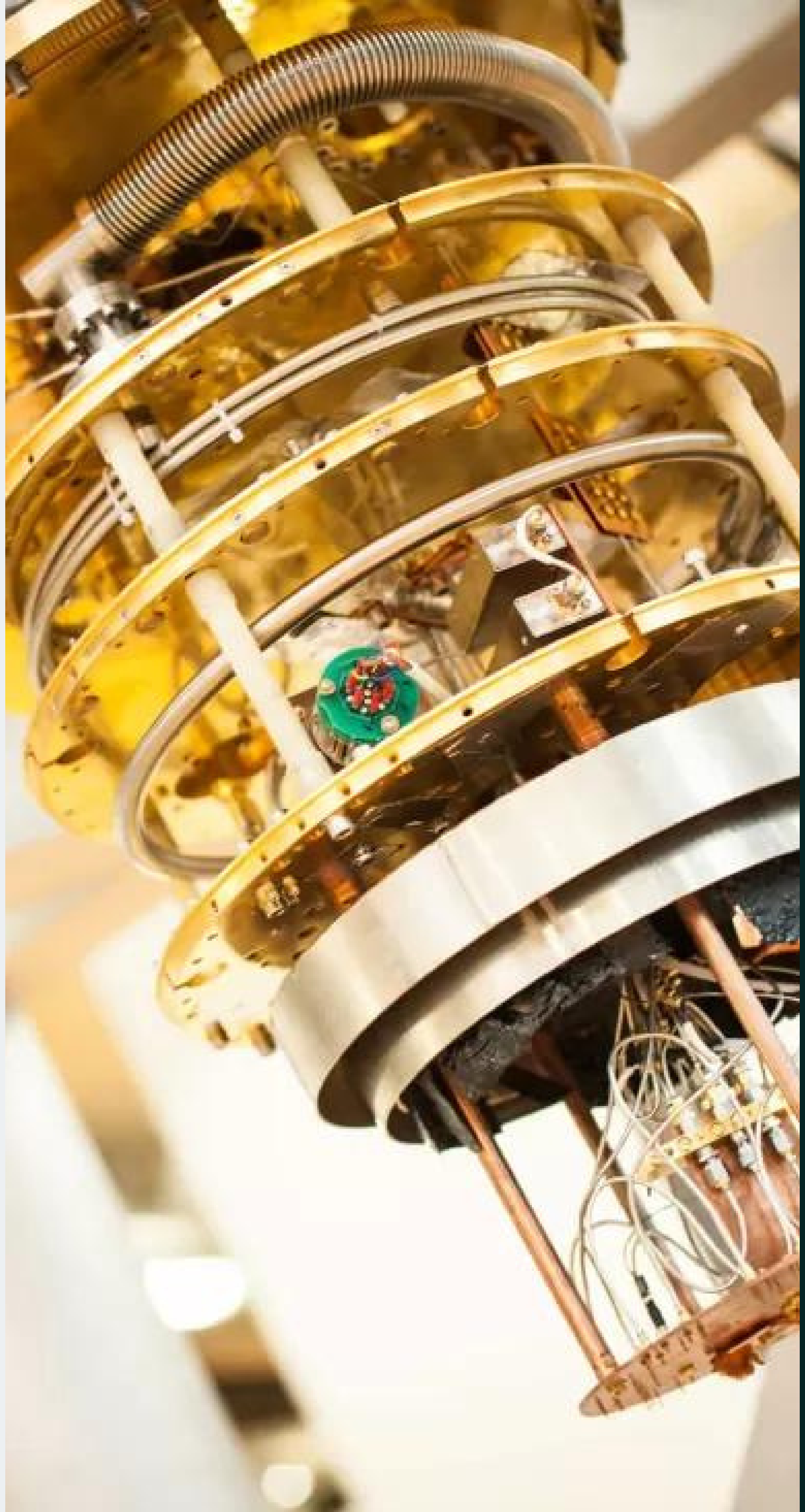
Project Presentation



QCourse 551-1

Project 27: QNN Algorithm Implementation

Mentor: Tal Michaeli

Team Members: Ashmit Gupta
Asif Saad
Roman Ledenov

Implementing a Hybrid Network to classify the MNIST Dataset

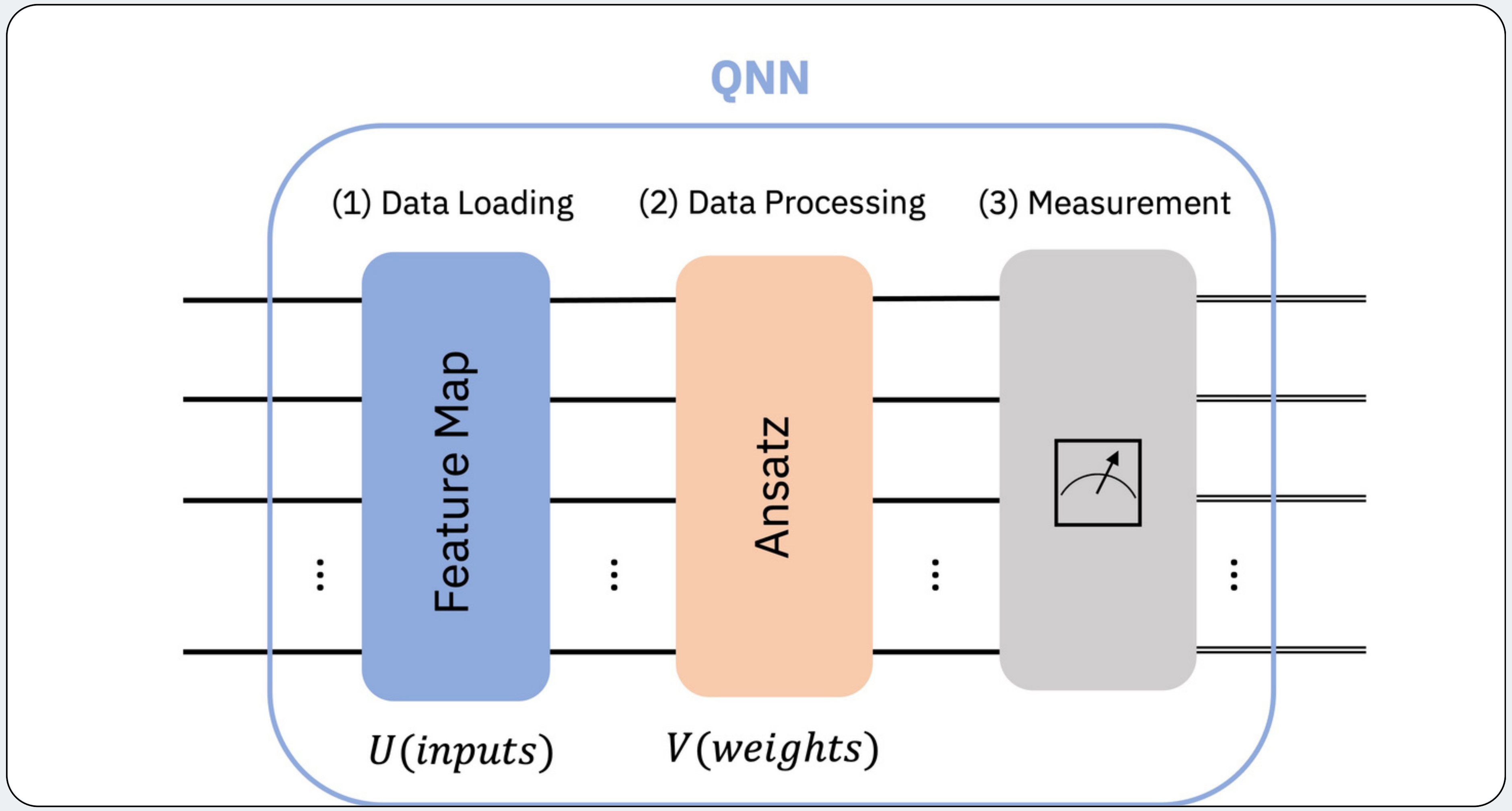
Agenda

Quantum Neural Network

Quantum Neural Networks are quantum algorithms based on parametrized quantum circuits that can be trained in a variational manner using classical optimizers.

These algorithmic models can be trained to find hidden patterns in data similar to their classical counterparts. These models can load classical data (inputs) into a quantum state, and later process it with quantum gates parametrized by trainable weights.

A Quantum layer take in classical data and return classical data.



Back to Agenda Page

Quantum Neural Network Algorithm Implementation

Problem Statement: Create a hybrid network consisting of both classical and quantum layers to classify the MNIST dataset.

Back to Agenda Page

Project Milestones

WE ARE HERE!

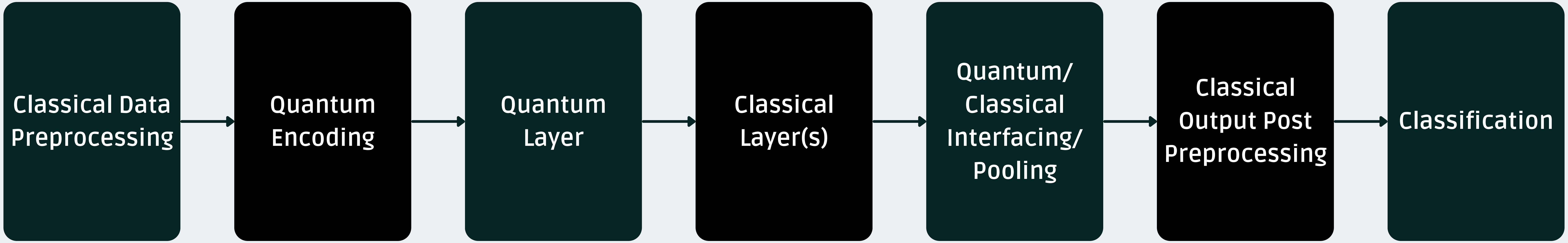
01	02	03	04	05	06
WEEK 1-3	**WEEK 4-5**	**WEEK 6-7**	**WEEK 8-9**	**WEEK 10-11**	**WEEK 12**
Discuss project, understanding the prerequisites and creating a work plan.	Pre-processing MNIST dataset and encoding it into a Quantum Circuit.	Design and Implement circuit for Quantum Layer.	Training and Testing of our QNN,	Extend our research and start writing our project research paper.	Finishing up and final presentation.

Step 1: Deciding the QNN Architecture

This Hybrid QNN architecture combines classical and quantum processing for efficient MNIST digit classification. Classical data preprocessing is followed by quantum encoding, where classical information is translated into quantum states.

The Quantum Layer utilizes quantum gates to perform computations, followed by Classical Layer(s) for further processing. The Quantum/Classical interfacing or pooling layer integrates quantum and classical information, leading to classical output post-processing. Finally, the classification step provides the ultimate prediction.

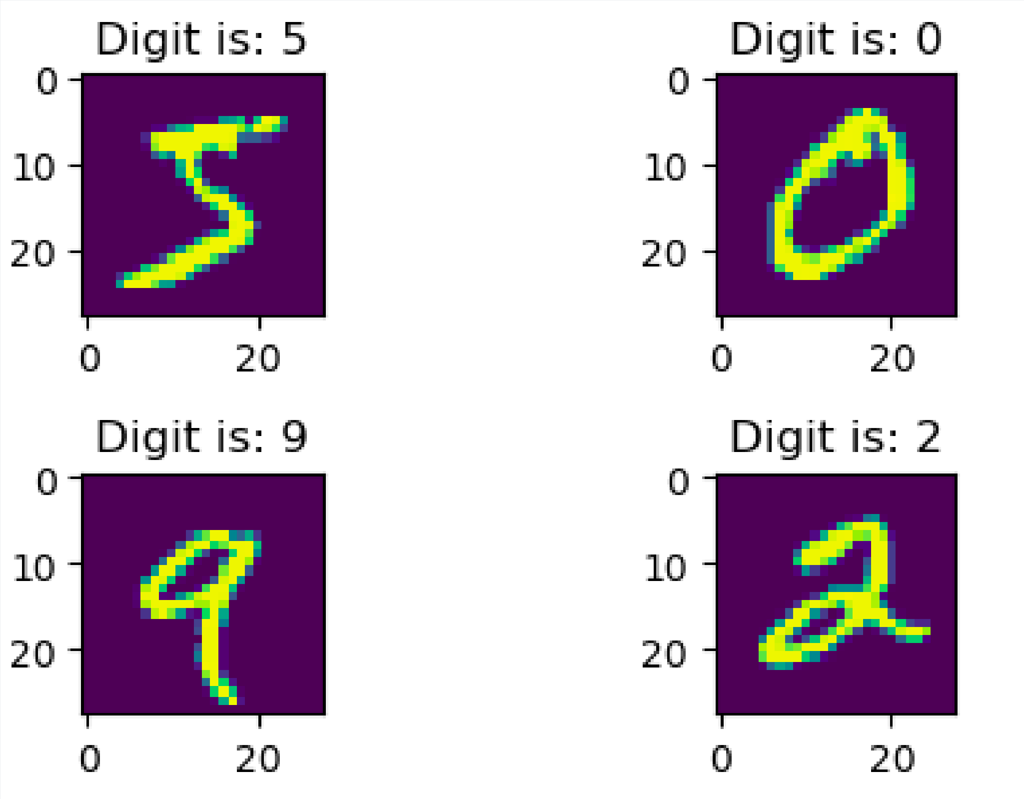
This hybrid approach leverages the strengths of quantum computing while maintaining compatibility with classical methods, offering a promising paradigm for enhanced machine learning tasks on quantum devices.



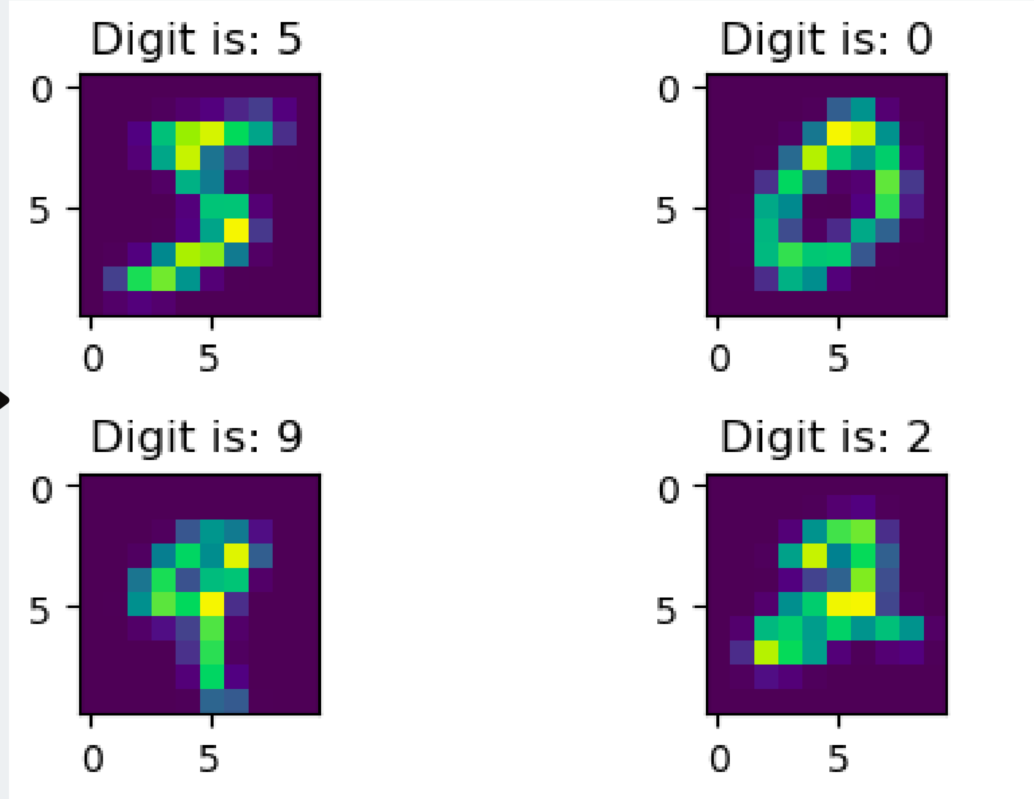
Back to Agenda Page

Step 2: Pre-processing MNIST Dataset

We are using PyTorch transforms. Resize to resize our dataset images of 28 x 28 px to shrink them into 8 x 8 px images. Then we plan to encode these images using to angle encoding such that it takes 1 qubits per two pixels.




```
def resize_image(image, resize_value=4):
    resize_transform = transforms.Compose([
        transforms.ToPILImage(),
        transforms.Resize((resize_value, resize_value)),
        transforms.ToTensor()
    ])
    re_image = resize_transform(image)
    re_image = re_image.squeeze()
    return re_image
```



Back to Agenda Page

Step 3: Encoding the MNIST Datasets

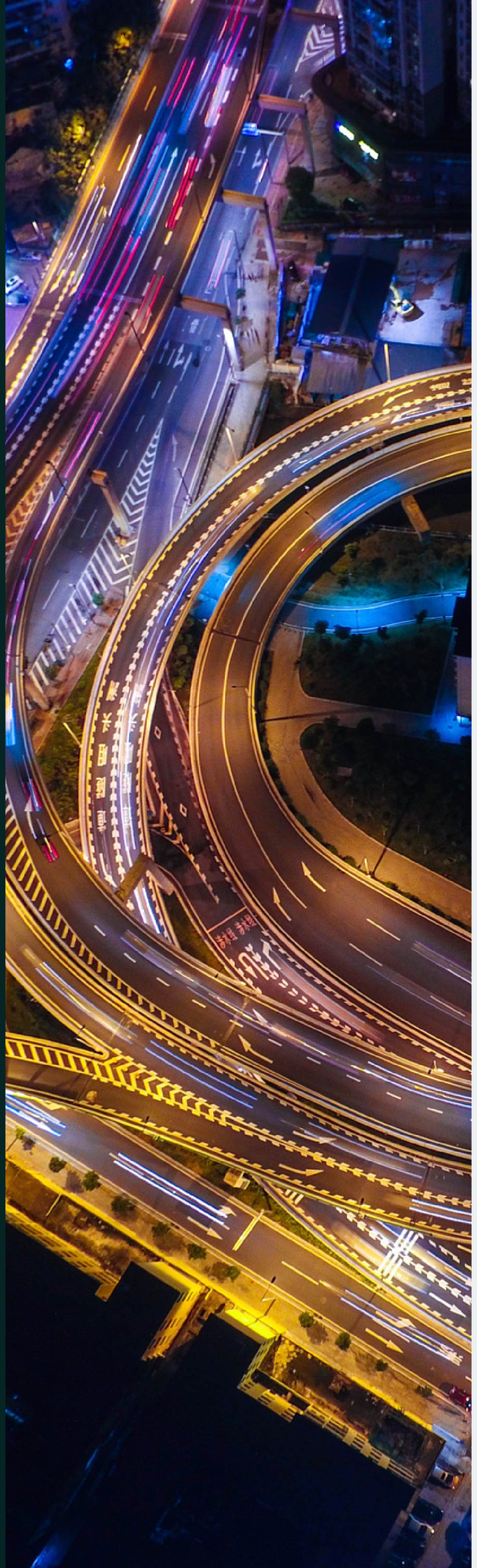
We use Angle encoding to encode our image pixels into angles of controlled X and Z gates. In general, it requires around 8 to 10 qubits per pixel. For a 28x28 grayscale image, a rough estimate suggests the need for approximately (28 x 28 x 8) to (28 x 28 x 10) qubits. Precision in representing pixel intensities plays a key role in determining the qubit requirements.

```python
def angle_encode_image(image, size):
    # Create a quantum circuit
    num_qubits = size * size
    qc = QuantumCircuit(num_qubits, num_qubits)

    # Angle encode each pixel value into the quantum state
    for i in range(size):
        for j in range(size):
            intensity = image[i, j]
            angle = intensity * (2 * np.pi / 256)  # Map intensity to angle
            qc.ry(angle, i * size + j)


    return qc
```

For our purpose, we use one qubit to encode two pixel information of our MNIST Dataset Images. Since, after classical pre-processing, our images are or size 8 x 8 px, we need a total 32 qubits to encode our image. We are currently, exploring ways to reduce this number to somewhere around 6-8 qubits.

Future Plans

Back to Agenda Page



Layers

- Parameterized Quantum Circuit
- Classical Circuit interfaced with PQC.

Optimizers

- Stochastic Gradient Descent (most likely)
- Adam (as testing)

Pooling

- Combines outputs of classical and quantum computation.
- Two approaches: Direct interfacing and Indirect interfacing.
- Various types of pooling layers (e.g., Averaging, Max)

Challenges and Obstacles

Back to Agenda Page

- We currently require 32 qubits for encoding single dataset image. We need to find ways to reduce this number to somewhere around 6-8 qubits.
- The shortage of computation power of simulation of classiq hardware.
- Inability to run the system on NISQ device, thus can’t face real world issues, errors and short comings.

Resources

- Paper on Quantum On-Chip Training with Parameter Shift and Gradient Pruning
- Classiq QNN User Guide: https://docs.classiq.io/latest/user-guide/built-in-algorithms/qml/qnn/
- Project Report by Ashmit JaiSarita Gupta: Implementing Quantum Neural Network Algorithm
- Project Repository on GitHub

Conclusion

Goal:
To create an hybrid Quantum Neural Network which can classify MNIST Dataset Images.

Current Stage:
We are currently in the process of decreasing the qubit requirements and designing the Quantum Layer.

Short term plans:

- Design the quantum layer
- Explore classical post processing

Back to Agenda Page

Thank you!

Do you have any questions?

Back to Agenda Page