Quantum Image Encoding with PennyLane

Kavya Reddy

April 4, 2025

1. Objective

We explore three types of quantum encodings for images from the CIFAR-10 dataset:

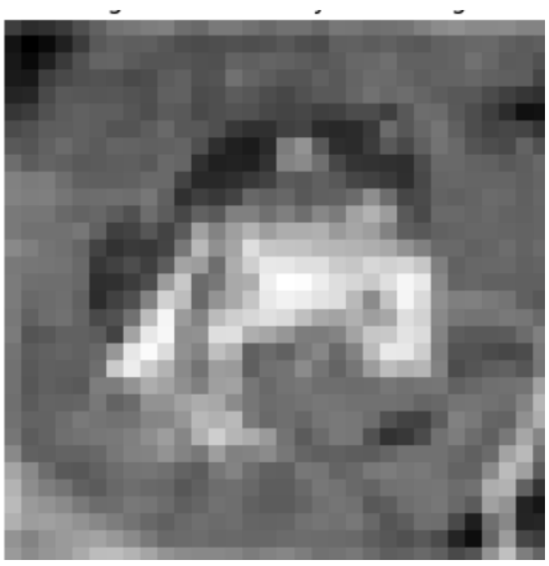
1. Basis Encoding (Binary threshold encoding)
2. Angle Encoding (Pixel intensity mapped to rotation)
3. Amplitude Encoding (Global normalized vector embedding)

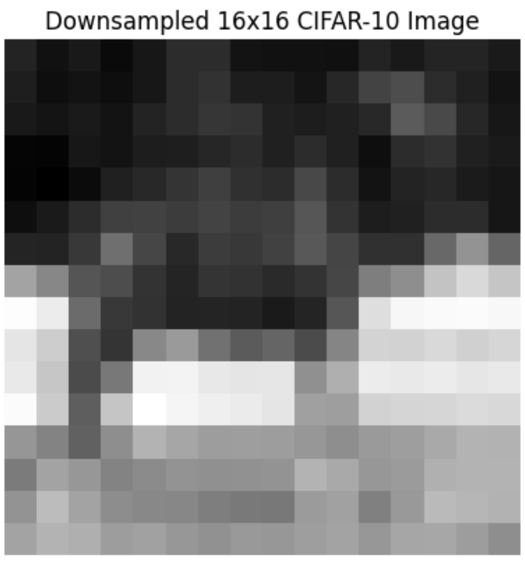
2. Preprocessing

All encodings rely on transforming CIFAR-10 images into grayscale and resizing to match the number of qubits.

```
transform = transforms.Compose([
    transforms.Grayscale(num_output_channels=1),
    transforms.Resize((image_size, image_size)),
    transforms.ToTensor()
])

dataset = torchvision.datasets.CIFAR10(
    root='./data', train=True, download=True, transform=transform
)
image_tensor = dataset[0][0].squeeze().numpy()
flat_image = image_tensor.flatten()
```

Original and Downsampled Image:



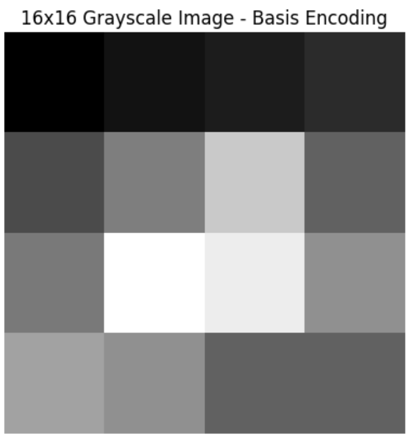


3. Basis Encoding

Concept: Flip each qubit using `PauliX` if the corresponding pixel ¿ 0.5.

```
def basis_encoding(img):
    for i, val in enumerate(img):
        if val > 0.5:
            qml.PauliX(wires=i)

@qml.qnode(dev)
def basis_encoded_circuit(img):
    basis_encoding(img)
    return [qml.expval(qml.PauliZ(i)) for i in range(n_pixels)]

result = basis_encoded_circuit(flat_image)
```

Binary Image Threshold Visualization:

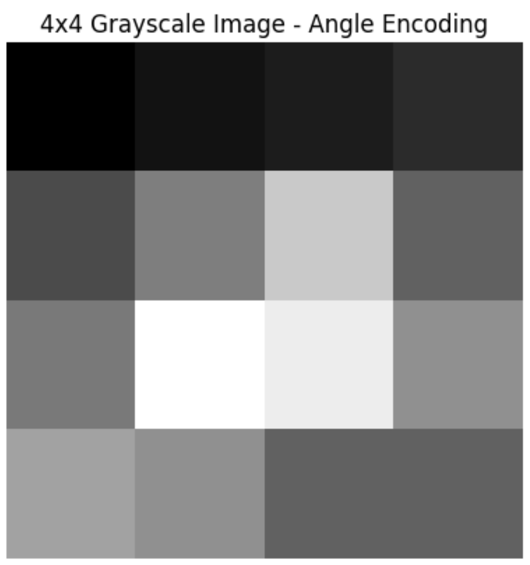


4. Angle Encoding

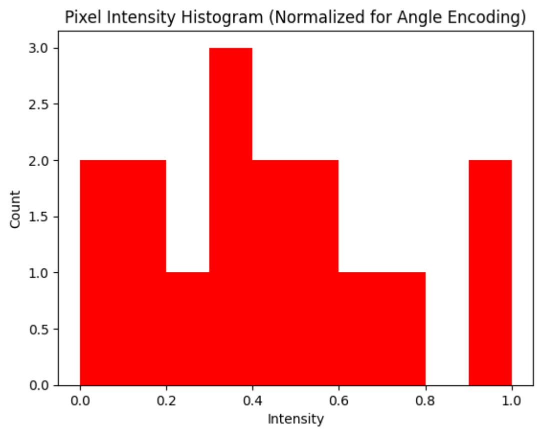
Concept: Use rotation angles proportional to pixel intensity using RY gates.

```
def angle_encoding(image):
    for i, val in enumerate(image):
        qml.RY(np.pi * val, wires=i)

@qml.qnode(dev)
def angle_encoded_circuit(image):
    angle_encoding(image)
    return [qml.expval(qml.PauliZ(i)) for i in range(n_pixels)]

normalized = (flat_image - flat_image.min()) / (flat_image.max() - flat_image.min())
result = angle_encoded_circuit(normalized)
```

Angle Encode image:



Histogram of Normalized Intensities:

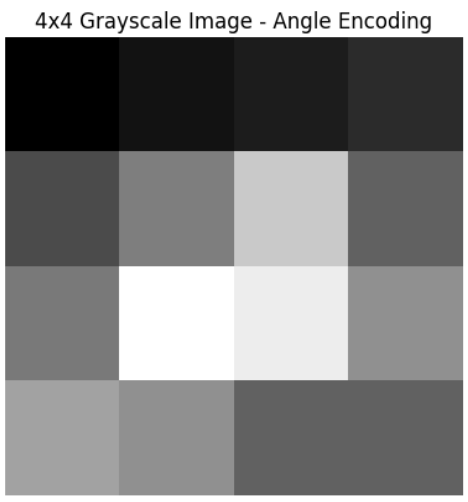


5. Amplitude Encoding

Concept: Compress full image into a quantum state vector using amplitude embedding.

```python
n_qubits = int(np.log2(n_pixels))
dev = qml.device('default.qubit', wires=n_qubits)

@qml.qnode(dev)
def amplitude_encoded_circuit(image):
    qml.AmplitudeEmbedding(
        features=image / np.linalg.norm(image),
        wires=range(n_qubits),
        normalize=False
    )
    return [qml.expval(qml.PauliZ(i)) for i in range(n_qubits)]

result = amplitude_encoded_circuit(flat_image)
```

Downsampled 16x16 Image Used for Amplitude Encoding:



6. Observations

- **Basis Encoding** is simple but only encodes binary (0 or 1) information.
- **Angle Encoding** retains grayscale variation and is well-suited for small image sizes.
- **Amplitude Encoding** is the most efficient, using only $\log_2(n)$ qubits, but must normalize data globally.

7. Conclusion

We successfully encoded grayscale CIFAR-10 images into quantum circuits using three encoding strategies. Each encoding type has tradeoffs in terms of qubit count, fidelity, and feasibility.