# Quantum Encoding of MNIST Data

Kavya Reddy

April 2, 2025

## 1 Introduction

The MNIST dataset, comprising grayscale images of handwritten digits, is widely used for image classification tasks and serves as an ideal benchmark for quantum machine learning. In this experiment, we preprocess MNIST digits and demonstrate Basis, Angle, and Amplitude encoding using quantum circuits to analyze their representation in quantum states.

## 2 Dataset Preprocessing

We use the MNIST dataset and filter only digits **0 and 1**. The images are downsampled to $4 \times 4$ (16 pixels), normalized, and reshaped into vectors.

Listing 1: Loading and preprocessing MNIST

```
(x_train, y_train), _ = tf.keras.datasets.mnist.load_data()
x_train = x_train.astype("float32") / 255.0
x_resized = np.array([resize(img, (4, 4), mode='reflect') for img in x_train])
x_flat = x_resized.reshape(-1, 16)
mask = (y_train == 0) | (y_train == 1)
x_flat = x_flat[mask]
y_flat = y_train[mask]
x_data = x_flat[:5]
y_data = y_flat[:5]
```

## 3 Basis Encoding

Pixels are thresholded into binary values and encoded via Pauli-X gates. Only 4 qubits (first 4 bits) are used for simplicity.

Listing 2: Basis Encoding Circuit

```
x_basis = (x_data > 0.01).astype(int)

@qml.qnode(dev_basis)
def basis_circuit(x):
    for i in range(4):
        if x[i] == 1:
            qml.PauliX(wires=i)
    return qml.probs(wires=range(4))

basis_outputs = [basis_circuit(x[:4]) for x in x_basis]
```

**Output Probabilities:**

- Sample 0: [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
- Sample 1: [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

- Sample 2: [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
- Sample 3: [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
- Sample 4: [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

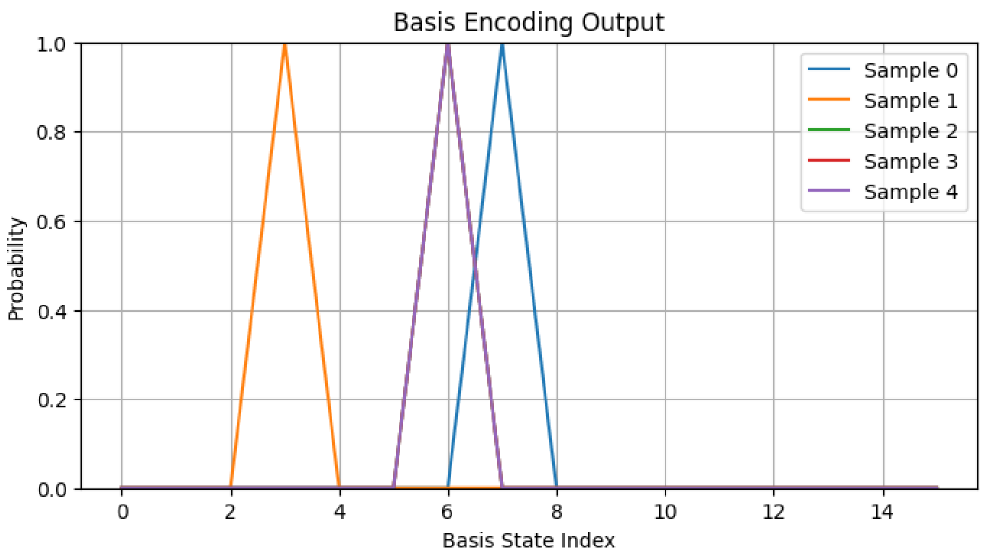


Figure 1: Basis Encoding Output

## 4 Angle Encoding

Each pixel value (scaled to $[0, \pi]$) is encoded using a RY rotation gate.

Listing 3: Angle Encoding Circuit

```
x_angle = x_data * np.pi

@qml.qnode(dev_angle)
def angle_circuit(x):
    for i in range(4):
        qml.RY(x[i], wires=i)
    return qml.probs(wires=range(4))

angle_outputs = [angle_circuit(x[:4]) for x in x_angle]
```

**Output Probabilities:**

- Sample 0: [0.846 0.003 0.142 0. 0.008 0. 0.001 0. 0. 0. 0. 0. 0. 0. 0. 0. ]
- Sample 1: [0.969 0.003 0.028 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. ]
- Sample 2: [0.96 0. 0.016 0. 0.024 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. ]
- Sample 3: [0.993 0. 0.004 0. 0.003 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. ]
- Sample 4: [0.99 0. 0.003 0. 0.007 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. ]

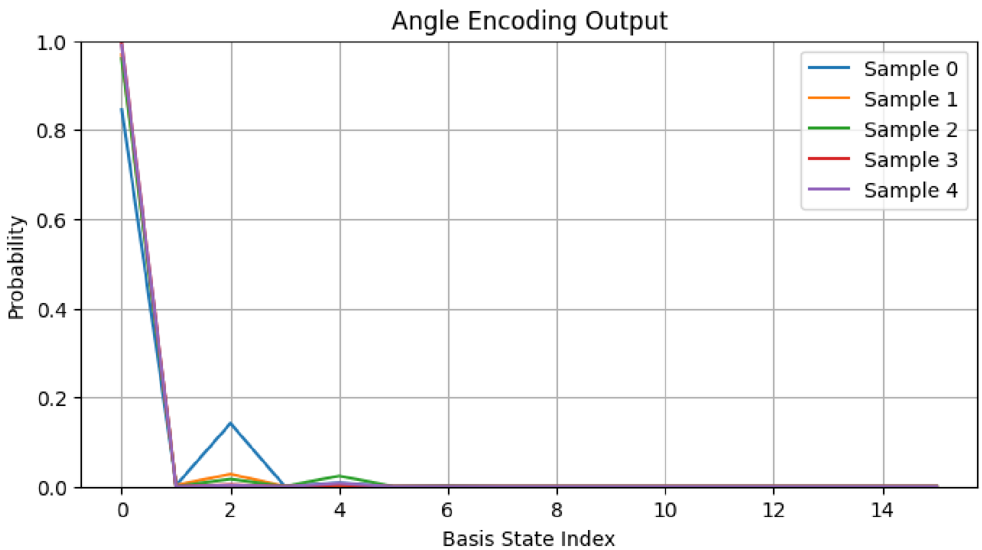


Figure 2: Angle Encoding Output

# 5 Amplitude Encoding

Each image vector is normalized to unit length and encoded as amplitudes of a quantum state.

Listing 4: Amplitude Encoding Circuit

```
def normalize_vector(v):
    norm = np.linalg.norm(v)
    return v / norm if norm > 1e-6 else np.ones_like(v) / np.sqrt(len(v))

x_amplitude = np.array([normalize_vector(x) for x in x_data])

@qml.qnode(dev_amp)
def amplitude_circuit(x):
    qml.AmplitudeEmbedding(x, wires=range(4), normalize=True)
    return qml.probs(wires=range(4))

amp_outputs = [amplitude_circuit(x) for x in x_amplitude]
```

**Output Probabilities:**

- Sample 0: [0. 0.006 0.096 0.002 0.001 0.17200001 0.26699999 0.042 0.017 0.15099999 0.102 0.018 0.003 0.108 0.016 0. ]
- Sample 1: [0. 0. 0.035 0.003 0. 0.017 0.37099999 0.005 0. 0.31600001 0.081 0. 0.001 0.168 0.002 0. ]
- Sample 2: [0. 0.032 0.022 0. 0. 0.134 0.208 0. 0. 0.077 0.414 0. 0. 0.005 0.106 0. ]
- Sample 3: [0. 0.012 0.014 0. 0. 0.164 0.241 0. 0. 0.105 0.34799999 0. 0. 0.017 0.099 0. ]
- Sample 4: [0. 0.024 0.01 0. 0. 0.207 0.197 0. 0. 0.19599999 0.237 0. 0. 0.045 0.083 0. ]

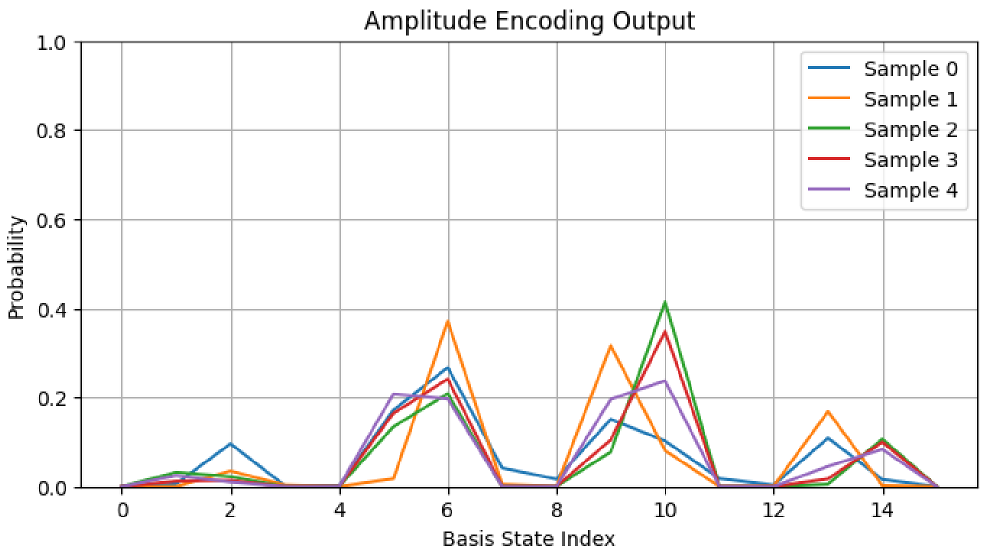


Figure 3: Amplitude Encoding Output

# 6 Conclusion

The visualization of quantum probability distributions from MNIST inputs highlights the differences between encoding methods in terms of expressiveness and state preparation. This leverages encoded MNIST features for classification and quantum advantage exploration.