Carnegie Mellon University

Quantum Image Processing

Presented By,

Meizhong Lyu
Material Science and Engineering

Prabh Simran Baweja
Electrical & Computer Engineering

Vinay Patil
Electrical & Computer Engineering

Alok Anand
Electrical & Computer Engineering

30th November, 2021

INTRODUCTION TO QUANTUM COMPUTING (18-819 F)

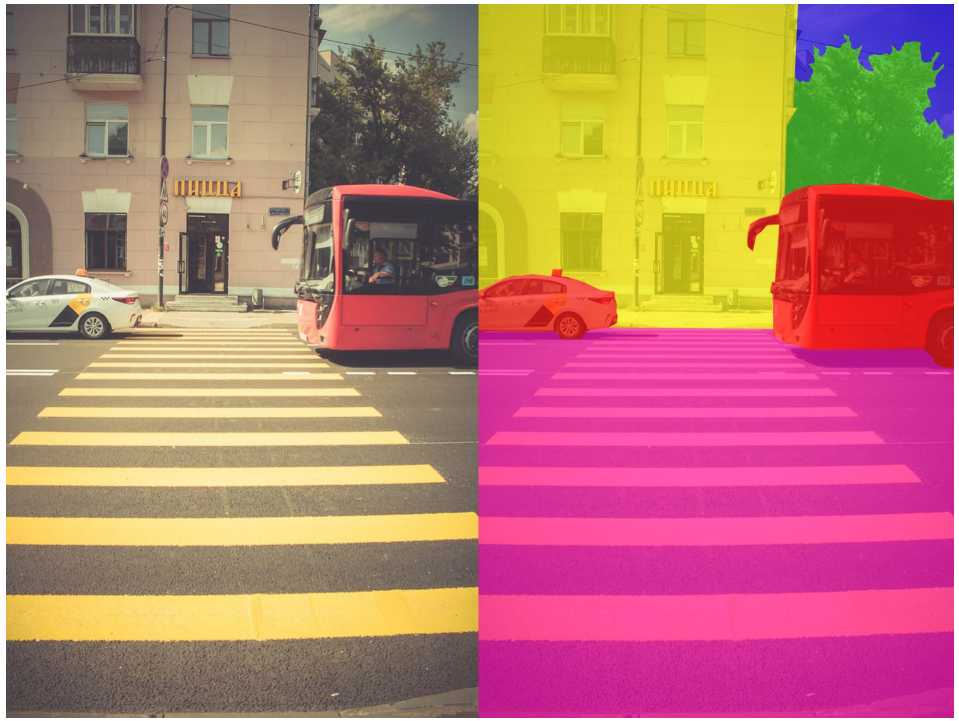
Prof. Sridhar Tayur
Prof. Davide Venturelli
Prof. Elias Towe
Prof. David E. Bernal

Contents

- **Introduction**
- **Literature review**
  - Classical and Quantum Image Representation
  - Classical and Quantum Edge Detection
  - Quantum Platforms
  - Quantum SVM and Feature Maps
- **Datasets**
- **Results**
  - classical results
  - Binary classification
  - Quantum Hadamard Edge Detection
  - Challenges
- **Impact of noise in quantum circuits**
- **Future work**

Introduction

- **Developed in response to several major problems:**
  - Picture digitization and coding to facilitate transmission, storage, printing
  - Picture enhancement and restoration: e.g to interpret surface of other planets
  - Picture segmentation and description as an early stage of Machine Vision.



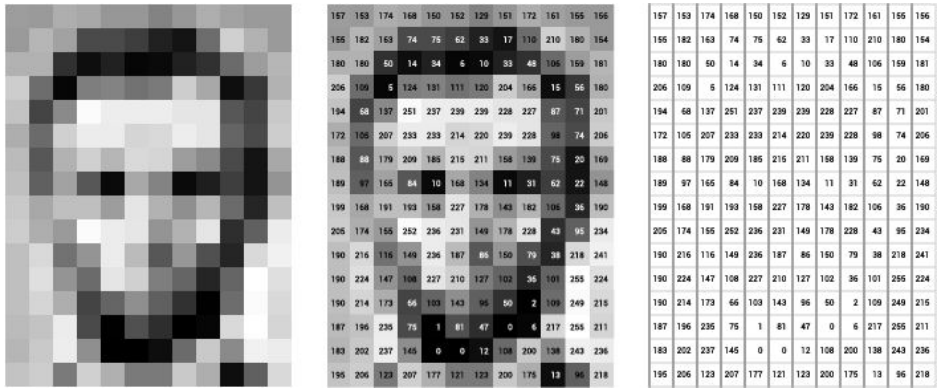
- **Applications of Image Processing**
  - Medical Image Processing
  - Computer Vision (Self driving cars, Autonomous stores)

Literature Review

A. Classical Image Representation

Images are represented as matrices of numbers, depicting discrete color/intensity value for every pixel.

(A)

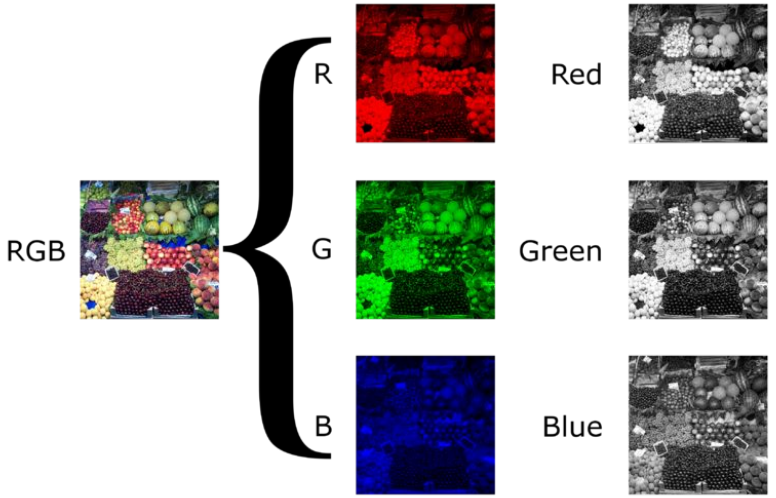


Images processing techniques :

1. Grayscale with pixel values varying from 0 (black) to 255 (white) [(A) Images composition in pixel form]

2. Color layered with 3 channels (Blue, Green, and Red [(B) Composition of RGB from three grayscale images]

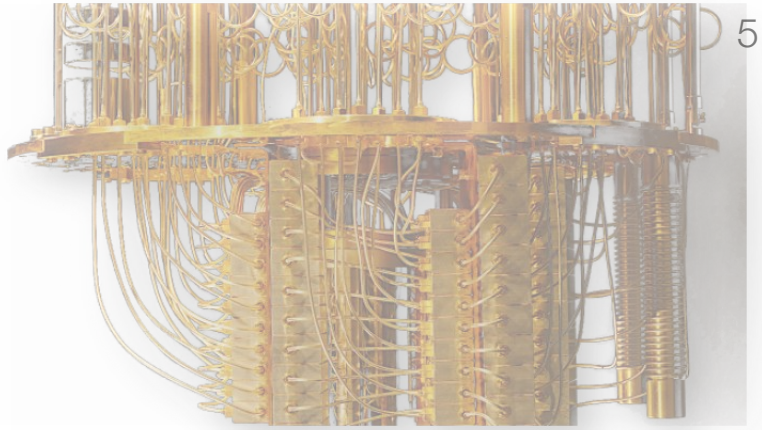
(B)



3. Binary portraying black or white values (0 or 1)

Carnegie Mellon University

B. Quantum Image Representation (QIR)



- Inherent properties in quantum computing

- Images can be represented as a superposition of quantum bits.
- QIR can improve processing efficiency for the task of real-time processing of large-scale image
- Some common models are FRQI, NEQR and QPIE .

Quantum Probability Image Encoding (QPIE)

- Uses the probability amplitudes of a quantum state to store the pixel values of a classical image.
- The number of qubits necessary to encode the image are further reduced.
- QPIE encodes image of size r x c as,

$$|I\rangle = \sum_{i=0}^{2^{2n}-1} c_i|i\rangle, n = [\log_2(rc)] \quad \text{where,} \quad I' = (I_{1,1}, I_{2,1}, \ldots, I_{r,1}, I_{1,2}, \ldots, I_{r,c})^T$$

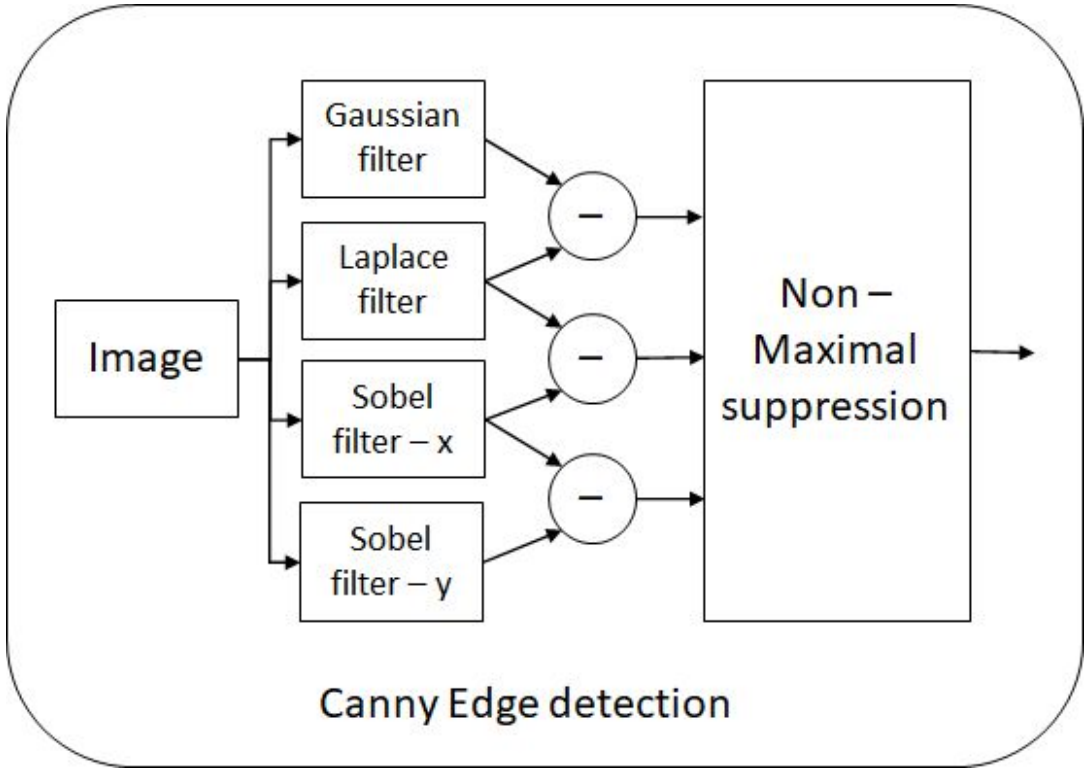
Carnegie Mellon University

C. Classical Edge and Corner Detection

(1) Edge Detection : Edge detection is an image processing technique for finding the boundaries of an object in the given image

DIfferent methods: Prewitt edge detection, Sobel edge detection, Laplacian edge detection,

Canny edge detection



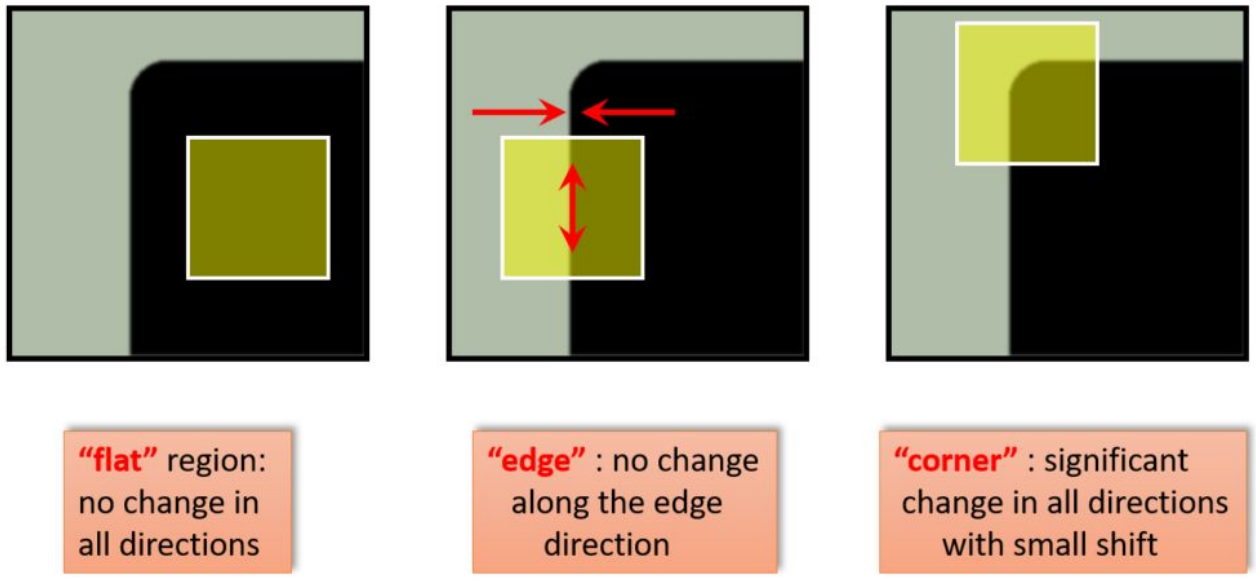
Carnegie Mellon University

C. Classical Edge and Corner Detection

(2) Corner Detection : Window based algorithm to detects corner of objects.

DIfferent methods: The Moravec corner detection, The Forstner corner detector, SUSAN (Smallest Univalue Segment Assimilating Nucleus) corner detector [20],

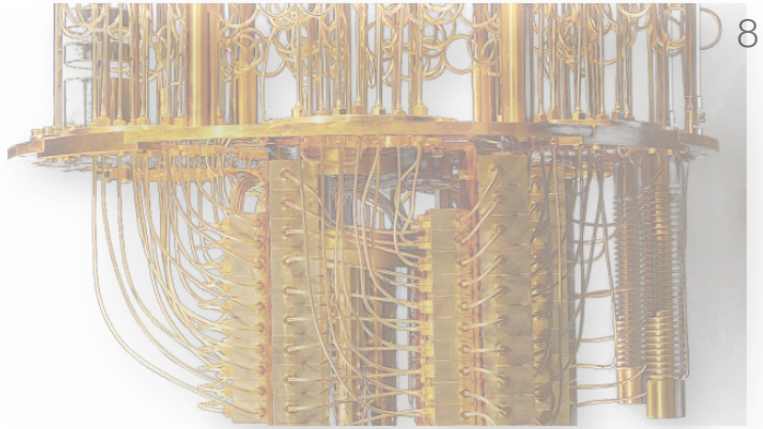
Harris corner detector



Carnegie Mellon University

Reference: https://datahacker.rs/opencv-harris-corner-detector-part1/

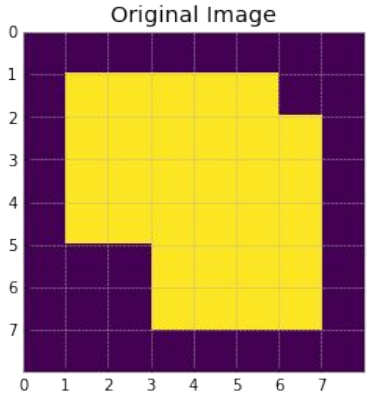
D. Quantum Edge Detection

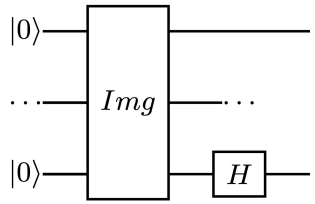


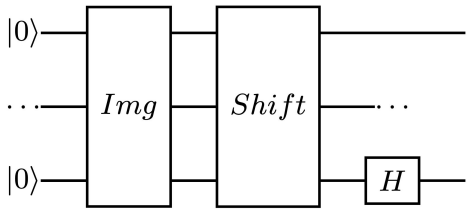
➢ **QIR + classical edge detection**
- QSobel: FRQI + Sobel

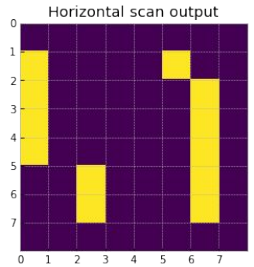
➢ **Quantum Hadamard Edge Detection(QHED) :**

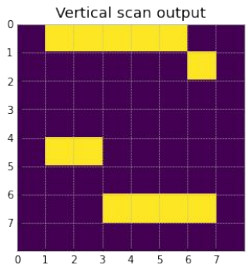
Apply a single Hadamard gate to the first qubit of the quantum circuit to encode the image, the resulting state encodes the edge information in the amplitudes of the basis states.

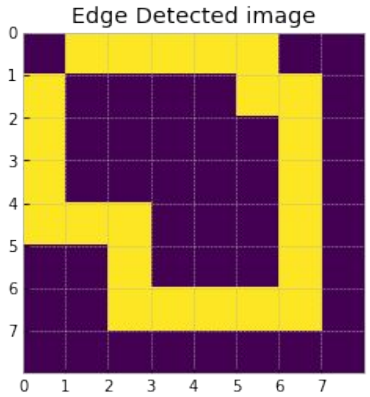






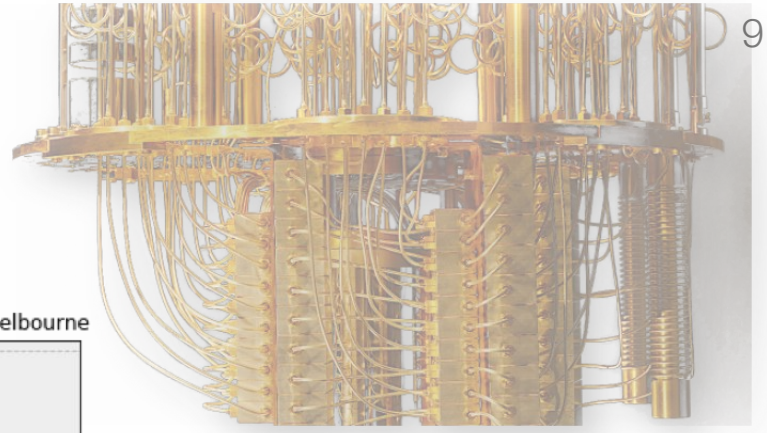


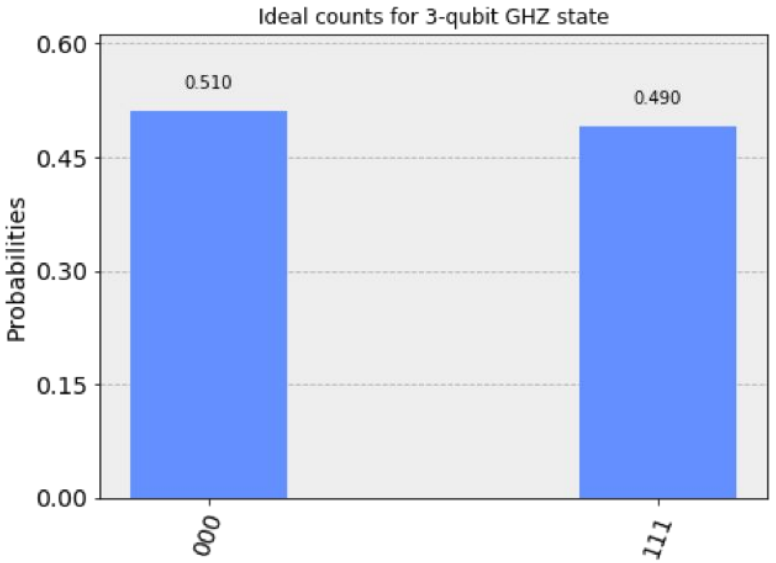


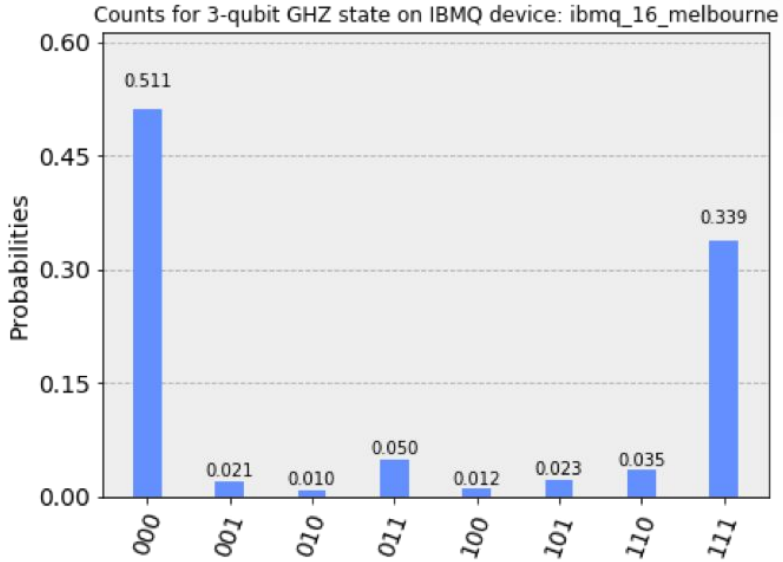


Carnegie Mellon University

E. Quantum Platforms







QASM Simulator

Statevector Simulator

IonQ

Rigetti

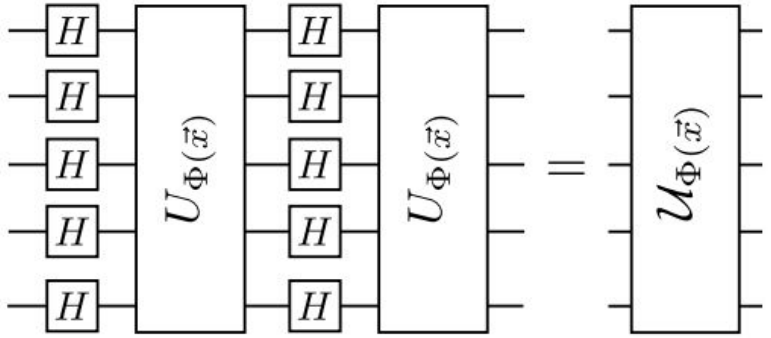
Carnegie Mellon University

F. Quantum Feature Maps

- Classical data is encoded to the quantum state space using a quantum feature map.
- Quantum feature map : utilises classical feature vector to quantum state by applying unitary operation on initial state $(|0>)^n$ where, n is number of qubits for encoding.
- **Pauli Feature Map** : constitutes P_0= X, P_1 = Y, P_2 = ZZ.
- **Z feature Map** : k =1 , and constitutes P_0 = Z Pauli matrix in entangling block.
- **ZZ feature Map** : when k = 2, constitutes P_0 = Z and P_1 = ZZ.
- Feature Map : Contains layers of Hadamard gates with entangling blocks , $U_{\phi(x)}$ encoded as,

$$\mathcal{U}_{\Phi(\mathbf{x})} = \prod_d U_{\Phi(\mathbf{x})} H^{\otimes n}, \quad \text{where} \quad U_{\Phi(\vec{x})} = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\vec{x}) \prod_{i \in S} Z_i \right)$$

Entangling blocks, $U_{\phi(x)}$: $P_i \epsilon \{I, X, Y, Z\}$ denotes Pauli matrices.
S, denotes connection between different qubits/datapoints
S $\epsilon \{\binom{n}{k} combinations, where\ k = 1 \ldots . n.\}$



Circuit Depth = 2

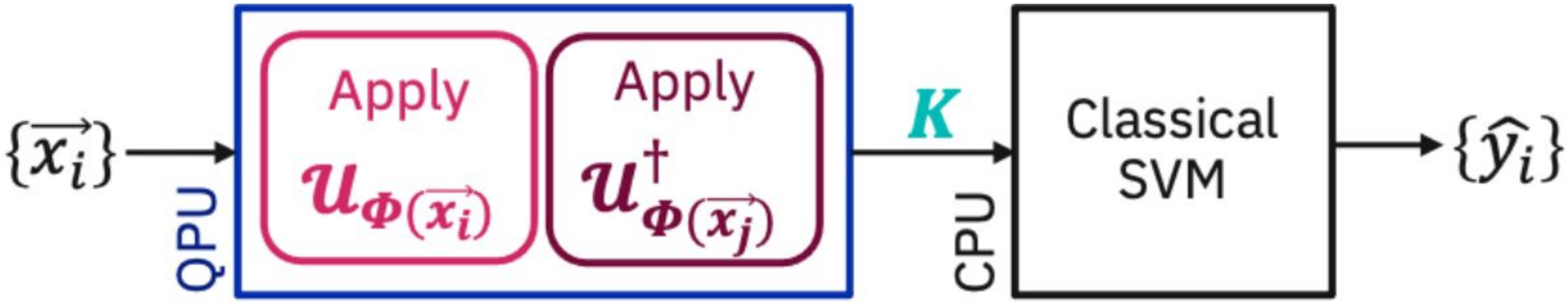
G. Quantum SVM

Steps involved in Quantum Kernel support vector classification algorithm

1. **Preparation of quantum kernel matrices for training and testing**

- In each pair of datapoints in training datasets $\boldsymbol{x_i}, \boldsymbol{x_j}$,feature map is applied.
- In each pair of training data points $\boldsymbol{x_i}$ and testing data points $\boldsymbol{y_j}$, feature map is applied.

2. **Using the train and test Quantum kernel matrices in a classical support vector machine classification**



Datasets

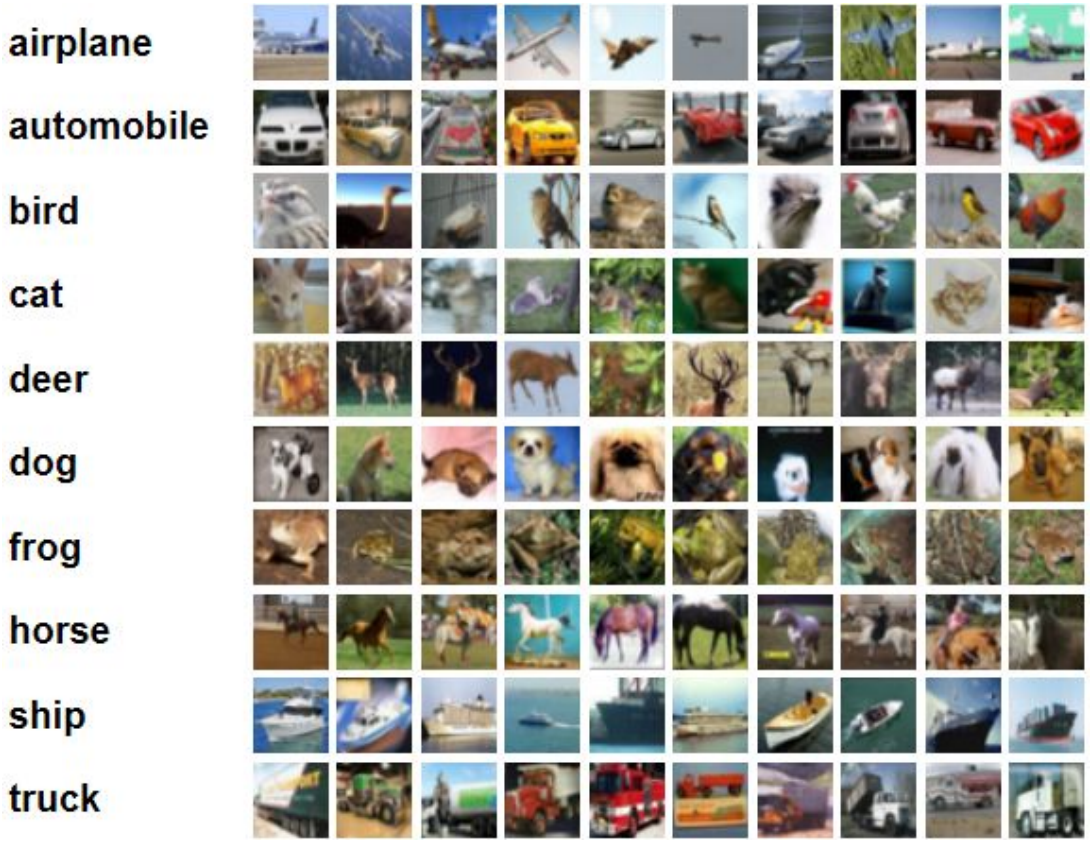
A. MNIST Dataset

- This dataset [32] is an extensive set of 60,000 handwritten digits.
- For Image encoding, every image has a dimension of 28x28 pixels.
- 50,000 training data and 10,000 test dataset consisting of equally distributed digits from 0-9.

Sources : MnistExamples - MNIST database - Wikipedia

B. CIFAR-10 Dataset

- This dataset consists of 60,000 images distributed equally across 10 different classes.
- For Image encoding, every image has a dimension of 32x32 pixels.
- 50,000 images are used as train dataset, and 10,000 images are used as test dataset.



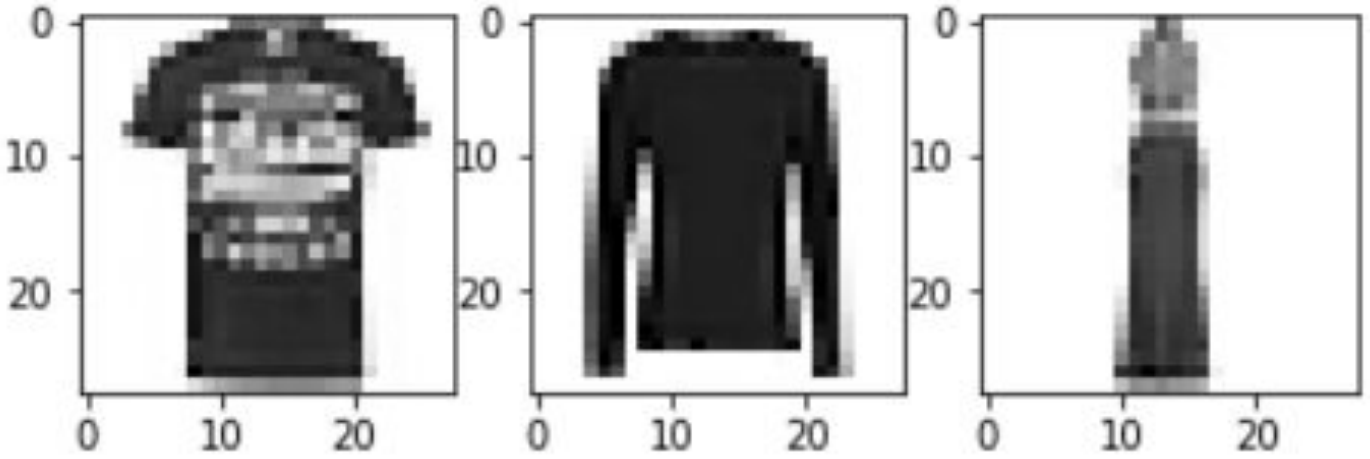
Sources : How-To: Train CIFAR-10 classification model from scratch using Kibernetika.AI - Kibernetika

C. Fashion MNIST Dataset

- This dataset is an extensive set of 60,000 clothing images, 50,000 training data and 10,000 test dataset, with label from 10 different classes
- For Image encoding, every image has a dimension of 28x28 pixels.
- Here, we are using only 3 of equally distributed clothing data class labelled :
- T-shirt – Label 0

  Pullover – Label 2

  Dress - Label 3



T-shirt – Label 0 Pullover – Label 2 Dress - Label 3

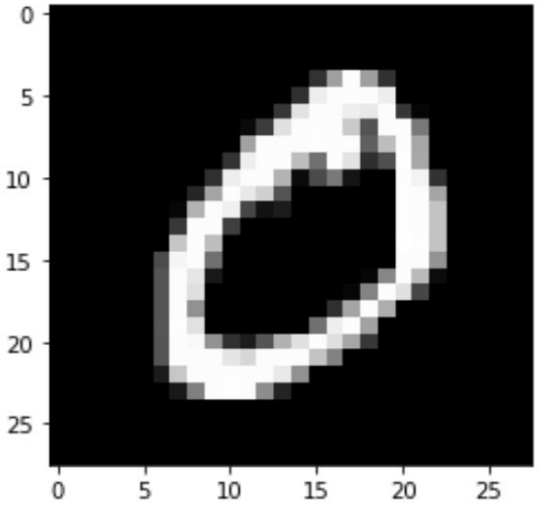
Carnegie Mellon University

Classical Results

A. MNIST character recognition using Harris-corner and K-means or SVM

STAGE -1

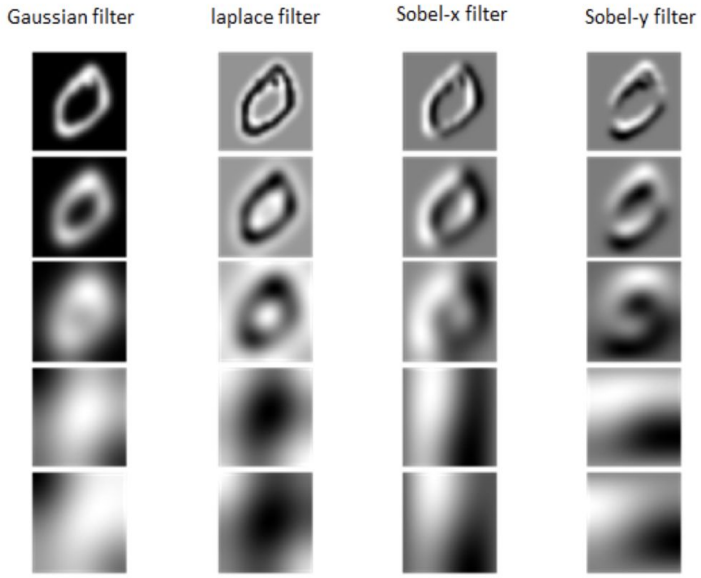
Normalization and smoothing image



Stage 1

STAGE - 2

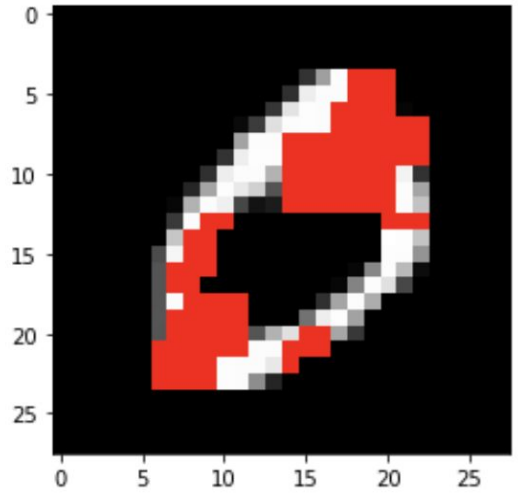
Getting edge related information using Gaussian, Laplace, Sobel filter



Stage 2

STAGE - 3

Harris corner detector to detect edges in the digits



Stage 3

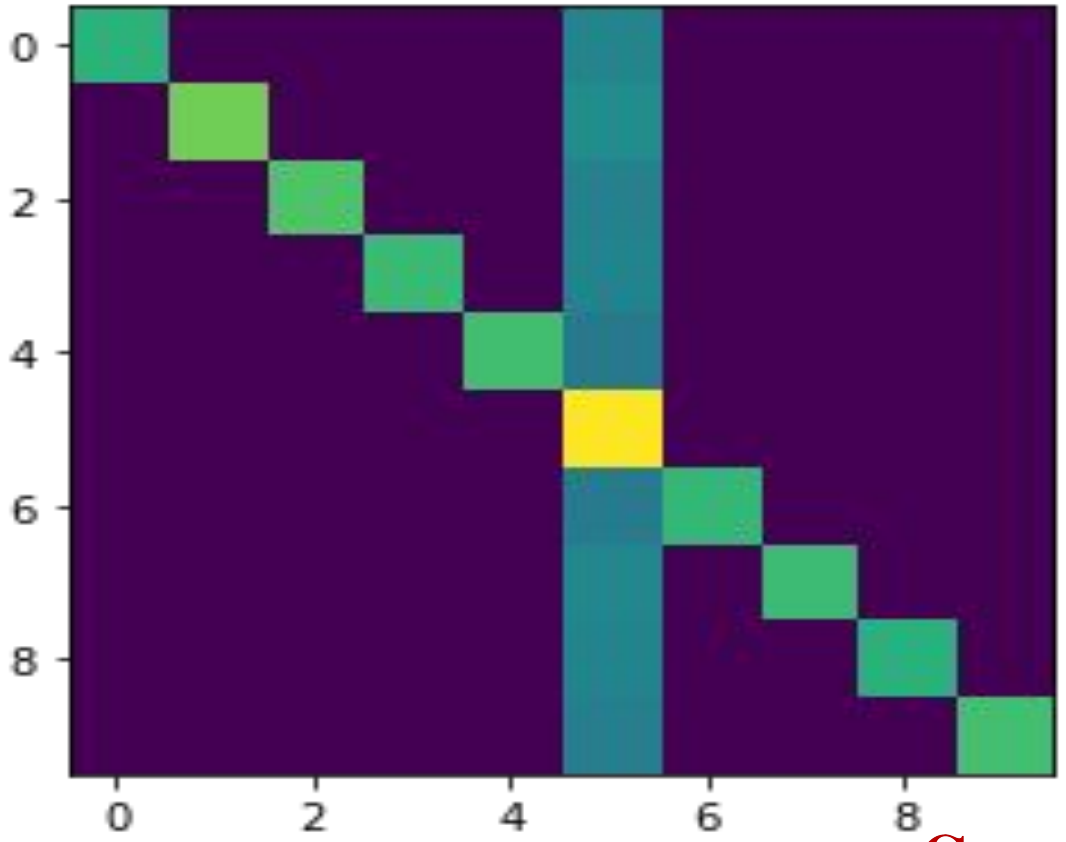
Carnegie Mellon University

A. MNIST character recognition using Harris-corner and K-means or SVM

Stage 4
Conversion to bag-of-words representation

Stage 5
Histogram representation

Stage 6
Classification using K-means/SVM



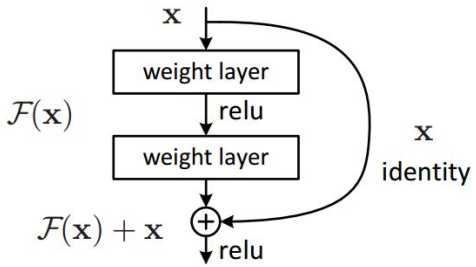
Confusion Matrix: 64% Accuracy

Carnegie Mellon University

B. Deep Learning models for MNIST and CIFAR-10

1) Deep Learning Models

- Architecture Design
  - CNN Layers with Residual Blocks
  - Batch Normalization
  - ReLU activation
- Optimizer: Stochastic Gradient Descent (SGD)
- Loss Function: Cross Entropy Loss
- Test Accuracy:
  - MNIST Dataset: 99.6% in 12 epochs
  - CIFAR10 dataset: 77.5% in 15 epochs



Residual Block

```
conv_and_res(1, 8),
conv_and_res(8, 16),
conv_and_res(16, 32),
conv_and_res(32, 16),
conv2(16, 10),
Flatten()
```

MNIST Model architecture

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 128)               524416
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 591,274
Trainable params: 591,274
Non-trainable params: 0
_________________________________________________________________
```

CIFAR10 Model architecture

Carnegie Mellon University

Quantum Results

C. MNIST-Fashion 3-class classification using QSVM-

STAGE - 1

Data Preparation

Here, small set of image dataset is considered and split into sample, validation and test data and labels.
Data consists of following labels:

1. Label 0 : T-shirt
2. Label 2 : Pullover
3. Label 3 : Dress

Visualizing the data samples, then preprocessing the dataset using Standardization, PCA and Normalization.

STAGE - 2

Data Modeling

For Multi-class classification using binary classifiers, we use One-vs-Rest approach.

We create three QSVM binary classifiers, classifying a particular label e.g., label 0 as positive (1) and rest as negative (0).

1. Label 0 vs the Rest
2. Label 2 vs the Rest
3. Label 3 vs the Rest

STAGE - 3

Data Encoding

A. Quantum Feature Map

Encoding classical data to quantum state, we use Quantum feature Map, depending on dataset.

Implemented feature map:

1. PauliFeature Map
2. ZZFeature Map
3. Zfeature Map

Carnegie Mellon University

C. MNIST-Fashion 3-class classification using QSVM

STAGE - 4

Data Encoding

B. Quantum Kernel Matrices

Quantum kernel class is prepared using feature map of training and testing datasets, to estimate training and validation kernel matrices.

STAGE - 5

Prediction

Finally, we predict the result for test data from the trained kernel feature matrices.

Then, calculate the probability of predicting the labels 0, 2 and 3 from the test data.

STAGE - 6

Analyzing Results

Result obtained from different feature map and corresponding kernel matrices, is analyzed on the data for the prediction and comparing the accuracy of training obtained on validation data. Help to realize the feature map suitable for 3-class image classification with more accuracy, with same input parameters.

C. Binary QSVM Image Classification for MNIST

LABEL - 0

```
Original validation labels:      [3 3 2 0 3 0 3 2 3 2 2 3 2 2 2 3 0 2 3 3]
Validation one label vs Rest:    [0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0]
Accuracy of ZZFeature Map discriminating between label and others: 90.0%
Accuracy of ZFeature Map discriminating between label and others: 95.0%
Accuracy of Pauli Feature Map discriminating between label and others: 85.0%
Probability of label : [0.38 0.55 0.29 0.24 0.26 0.23 0.42 0.45 0.45 0.39 0.23 0.23 0.29 0.3
 0.25 0.21 0.29 0.46 0.29 0.24]
Probability of label : [0.98 0.98 0.11 0.37 0.08 0.03 0.93 0.31 0.98 0.88 0.32 0.62 0.05 0.2
 0.3  0.04 0.24 0.93 0.04 0.48]
Probability of label : [0.45 0.37 0.31 0.27 0.3  0.32 0.34 0.29 0.4  0.28 0.27 0.3  0.24 0.39
 0.32 0.3  0.27 0.42 0.31 0.22]
```

LABEL - 2

```
Original validation labels:      [3 3 2 0 3 0 3 2 3 2 2 3 2 2 2 3 0 2 3 3]
Validation one label vs Rest:    [0 0 1 0 0 0 0 1 0 1 1 0 1 1 1 0 0 1 0 0]
Accuracy of ZZFeature Map discriminating between label and others: 85.0%
Accuracy of ZFeature Map discriminating between label and others: 90.0%
Accuracy of Pauli Feature Map discriminating between label and others: 90.0%
Probability of label : [0.27 0.18 0.31 0.35 0.38 0.52 0.37 0.24 0.19 0.38 0.06 0.57 0.73 0.48
 0.08 0.7  0.43 0.29 0.5  0.64]
Probability of label : [0.   0.   0.03 0.   0.06 0.02 0.   0.06 0.   0.   0.   0.04 1.   0.73
 0.   1.   0.   0.   0.   0.27]
Probability of label : [0.14 0.29 0.25 0.18 0.23 0.36 0.24 0.36 0.26 0.46 0.45 0.37 0.49 0.3
 0.34 0.5  0.26 0.21 0.18 0.71]
```

Carnegie Mellon University

C. Binary QSVM Image Classification for MNIST

LABEL - 3

```
Original validation labels:      [3 3 2 0 3 0 3 2 3 2 2 3 2 2 2 3 0 2 3 3]
Validation one label vs Rest:    [1 1 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 1 1]
Accuracy of ZZFeature Map discriminating between label and others: 65.0%
Accuracy of ZFeature Map discriminating between label and others: 90.0%
Accuracy of Pauli Feature Map discriminating between label and others: 70.0%
Probability of label : [0.26 0.13 0.3  0.68 0.67 0.69 0.05 0.16 0.23 0.07 0.96 0.23 0.06 0.31
 0.89 0.11 0.41 0.11 0.4  0.18]
Probability of label : [0.02 0.04 1.   1.   1.   1.   0.13 0.44 0.03 0.03 1.   0.03 0.   0.39
 1.   0.   1.   0.69 1.   0.  ]
Probability of label : [0.13 0.24 0.49 0.44 0.72 0.13 0.24 0.35 0.18 0.24 0.46 0.26 0.24 0.18
 0.14 0.16 0.39 0.09 0.5  0.16]
```

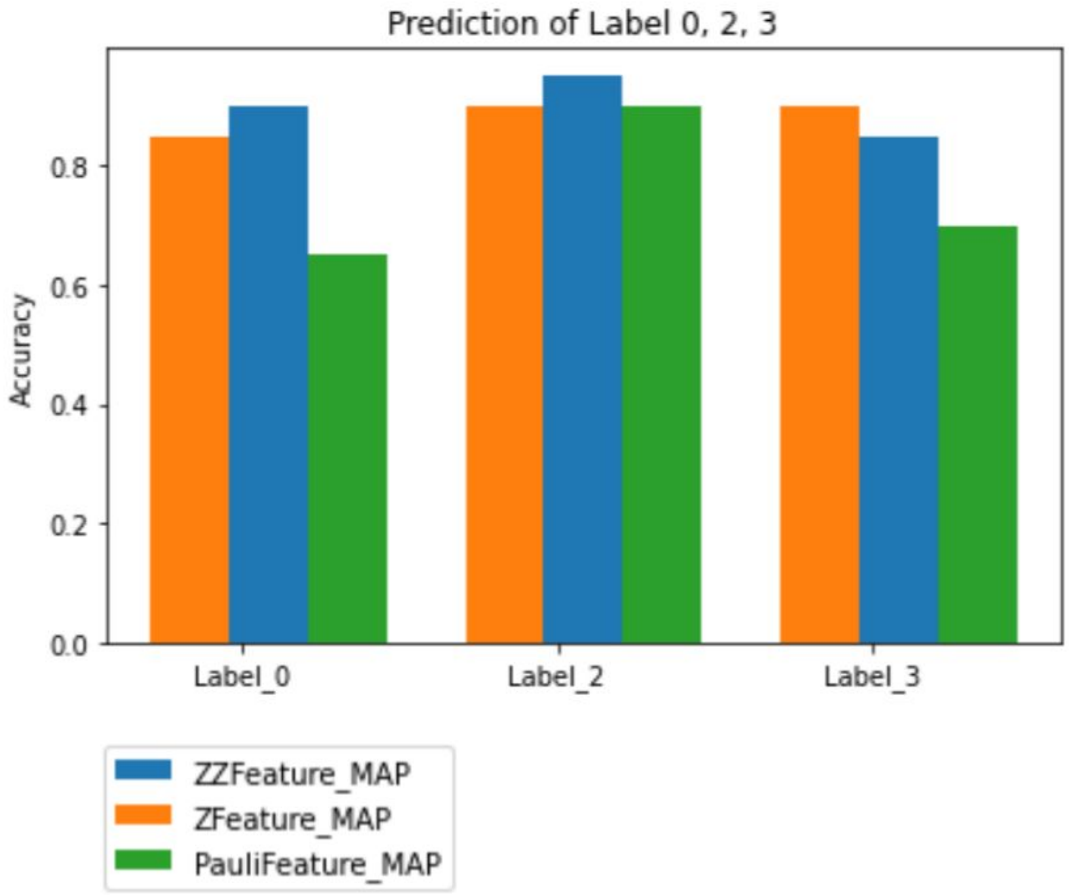
Final : TEST- Data Classification (size = 20)

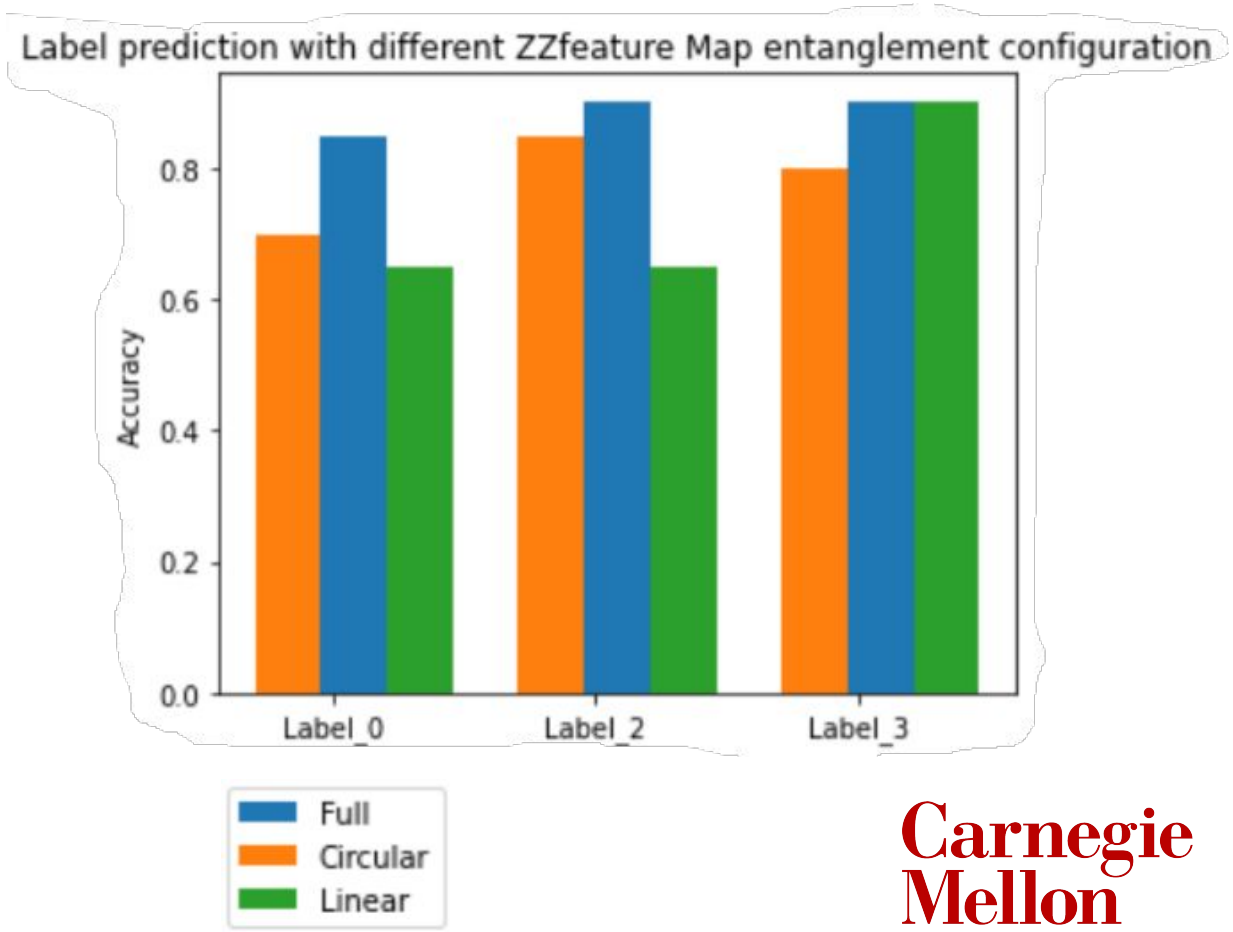
```
ZZ Feature MAP     : Prediction: [array([0, 0, 2, 3, 3, 3, 0, 0, 0, 0, 3, 2, 2, 2, 3, 2, 2, 0, 2, 2])]
Z Feature  Map     : Prediction: [array([0, 0, 3, 3, 3, 3, 0, 3, 0, 0, 3, 0, 2, 2, 3, 2, 3, 0, 3, 0])]
Pauli Feature MAP  : Prediction: [array([0, 0, 3, 3, 3, 2, 0, 2, 0, 2, 3, 2, 2, 0, 2, 2, 3, 0, 3, 2])]
```

Carnegie Mellon University

C. Binary QSVM Image Classification for MNIST

Comparison of classification results (Quantum)





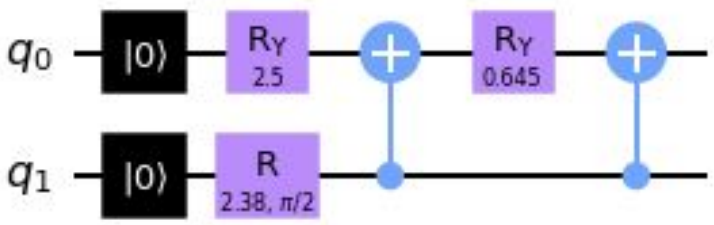
Carnegie Mellon University

D. Challenges in Binary QSVM Image Classification MNIST

- **Handling large datasets, training and testing.**
  For handling larger datasets, during training and testing takes lot of time. With limited memory space available, it often causes kernel instance to interrupt.

- **Realizing Quantum circuits for larger dimension data such as image is difficult to realize.**
  To evaluate performance of quantum circuit for large dimension image and feature, makes it complex and difficult - requires splitting image data to smaller dimensions to be efficient.

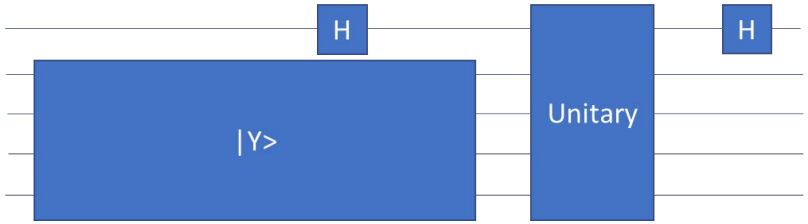
E. Quantum Hadamard Edge Detection



Equation: $|\text{Img}\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle$

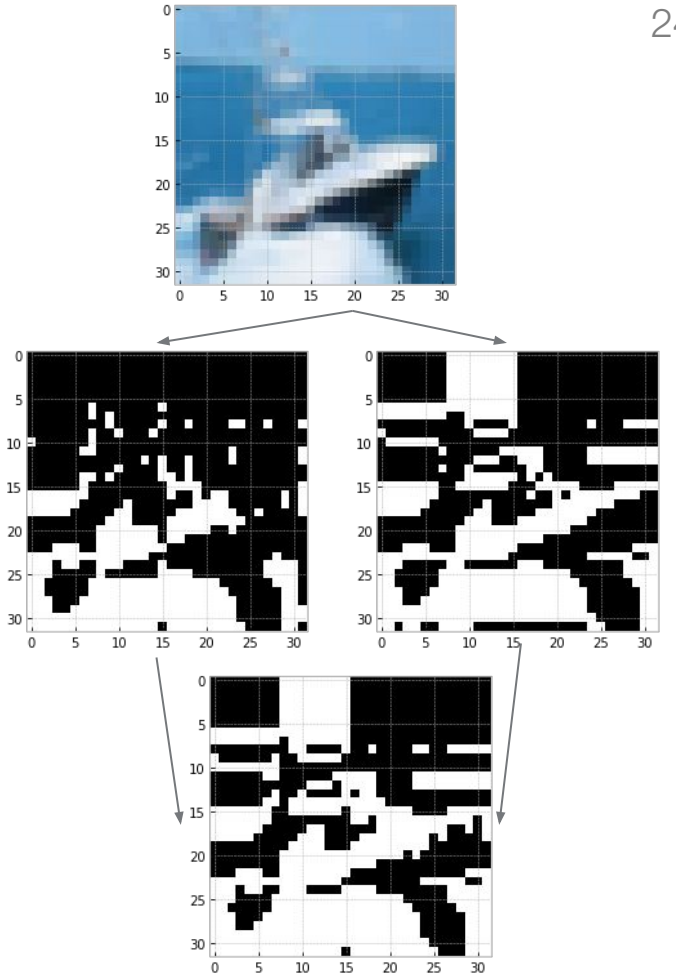
Quantum Probability Image Encoding(QPIE)

Step 1



Quantum circuit creation using Hadamard operation

Step 2

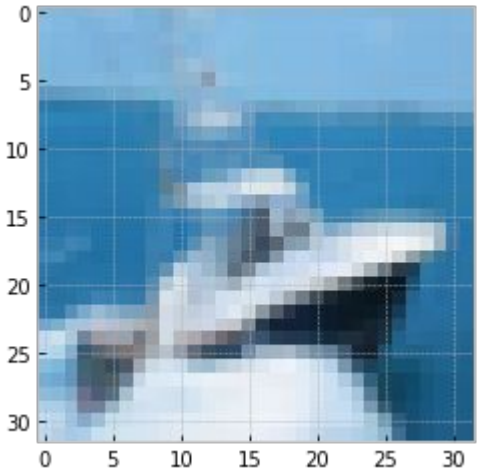


Read classical bits depending on auxiliary bit, and merge both scan results

Step 3

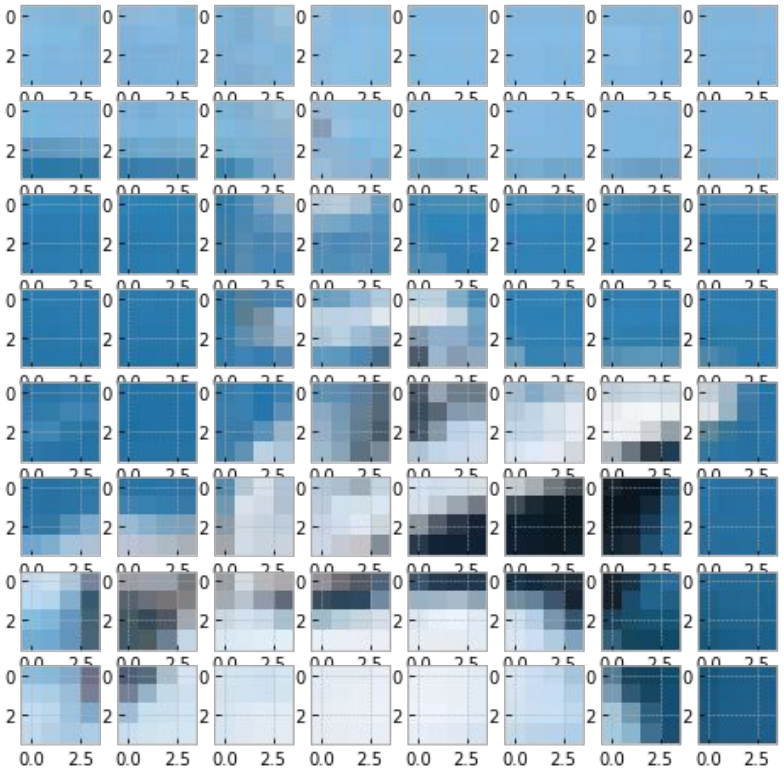
Carnegie Mellon University

E. Handling Large images with QHED



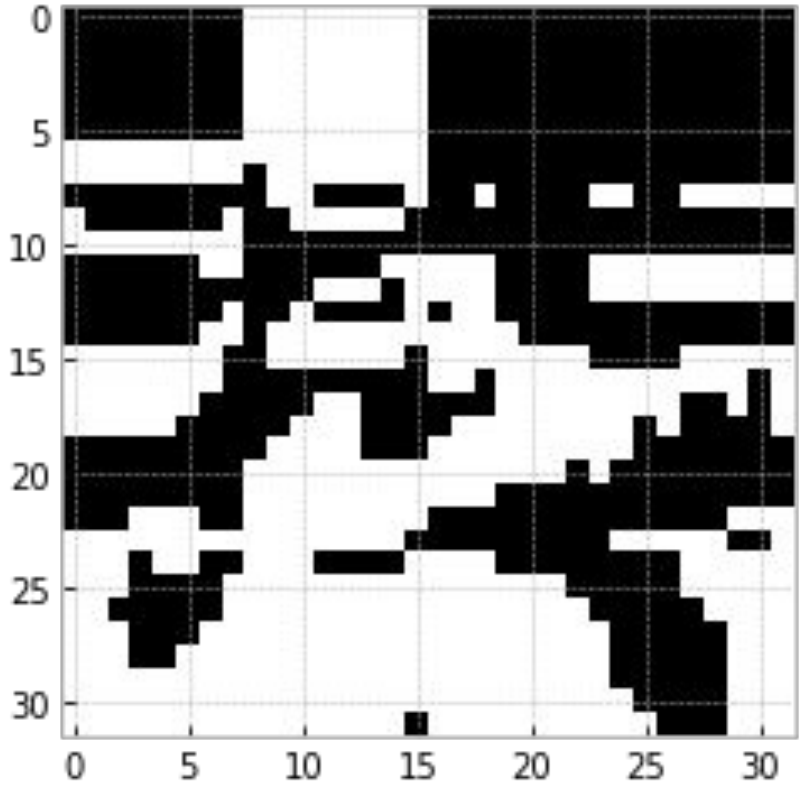
read Input image

Step 1



split image into small patches
4x4 recommended because of 2 qbit requirement on real systems

Step 2



perform QHED on every patch to detect edges and merge patches to form edge detected image

Step 3

Carnegie Mellon University

E. Classification with QHED

Model description:

- **LogisticRegression** with all-vs-one classification
- Parameter tuning:
  - Solver: saga
  - penalty : l1
  - intercept terms
  - maximum iterations : 1000

Runtime description:

- IBM qiskit quantum Machines
  - **state_vector** simulator : better result in accuracy but slower
  - **qsam_simulator** : better performance however accuracy is low also adds circuit complexity

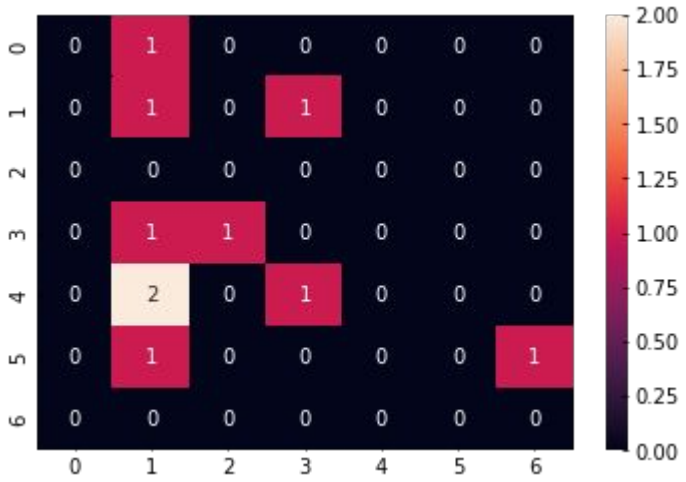
MNIST dataset classification: **78% over test dataset**

```
[Parallel(n_jobs=5)]: Using backend ThreadingBackend with 5 concurrent workers.
convergence after 30 epochs took 1 seconds
Accuracy on test dataset = 0.78
[Parallel(n_jobs=5)]: Done   1 out of   1 | elapsed:    0.5s finished
```

CIFAR10 dataset classification: **10% over test dataset**

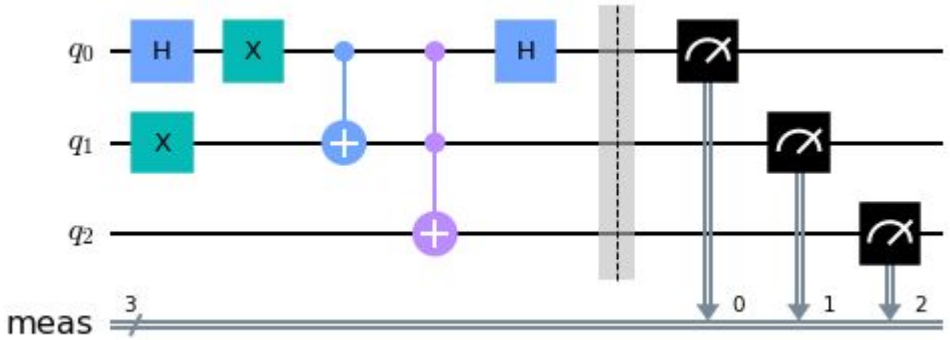
- Limited dataset training and testing
- model trained on 100 images and tested over 10 images from test dataset

```
[Parallel(n_jobs=5)]: Using backend ThreadingBackend with 5 concurrent workers.
convergence after 32 epochs took 0 seconds
Accuracy on test dataset = 0.10
[Parallel(n_jobs=5)]: Done   1 out of   1 | elapsed:    0.7s finished
```

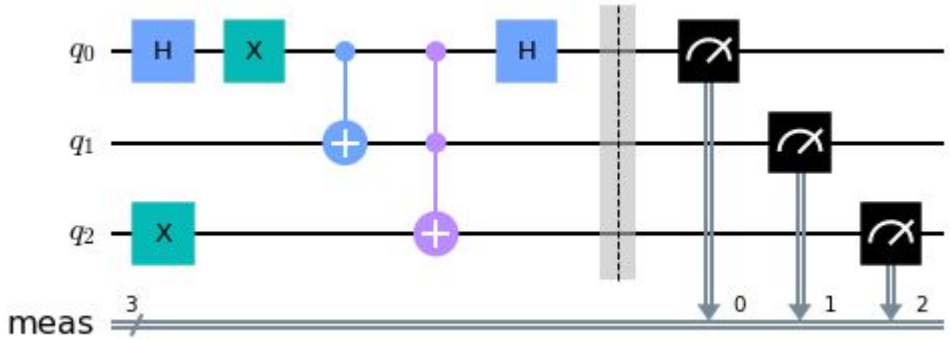


Carnegie Mellon University

F. Quantum Qiskit qsam circuit



Horizontal scan



Vertical scan

```
qc_small_h = QuantumCircuit(total_qb)
qc_small_h.x(1)
qc_small_h.h(0)
qc_small_h.x(0)
qc_small_h.cx(0, 1)
qc_small_h.ccx(0, 1, 2)
qc_small_h.h(0)
qc_small_h.measure_all()
```

- Vertical and horizontal scan differs by not-gate location.
- in horizontal scan, q1 is connected to not-gate. and in vertical scan q2 is connected to not-gate

Carnegie Mellon University

F. Challenges in QHED

- **Handing large images:-**
  - the qbit requirement for large images increases which makes it subject to more noise and inaccurate edges
  - **solution:** splitting image into smaller chunks

- **Training time on statevector simulator:-**
  - statevector simulator is slower in performing edge detection which in-turn increases the training time.
  - **solution:** running the gate-based circuit on actual quantum computer or qsam simulator which gives better performance but it is more exposed to the noise from the devices

- **Optimal image split ratio:-**
  - The algorithm to handle larger images is dependent on image slice dimensions.
  - if the image is sliced into very small images then the output of quantum circuit is quicker however, the outputs are not acceptable as edges
  - The larger image slice chunks provides better accuracy however, the the time needed-for computation is higher.

G. Edge detection comparison classical vs quantum

Classical Edge detection	Quantum Edge quantum
1. Runtime complexity : o(mn.log(mn))	1. Runtime complexity using sobel filter: o(n^2) using hadamard operator : o(log^2 n)
2. Space complexity : o(mn) classical bits	2. Space complexity : ceil(log2(N)) + 1 (auxiliary bit)
3. Slower when comes to larger images	3. Performance remains same irrespective of image size, however, for accuracy in output the image is divided into small chunks for accurate processing.

Impact of Noise

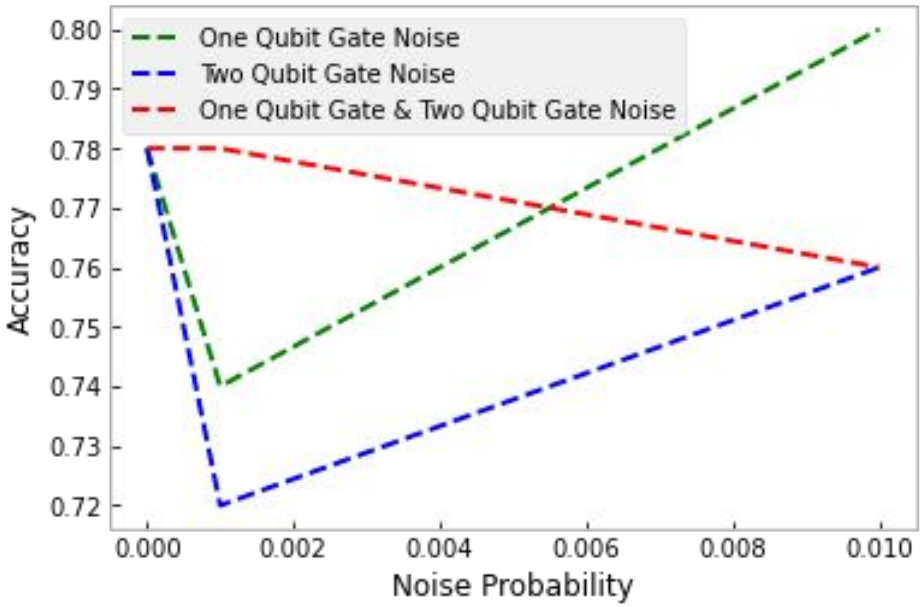
Different Noise Models for Quantum Edge Detection:

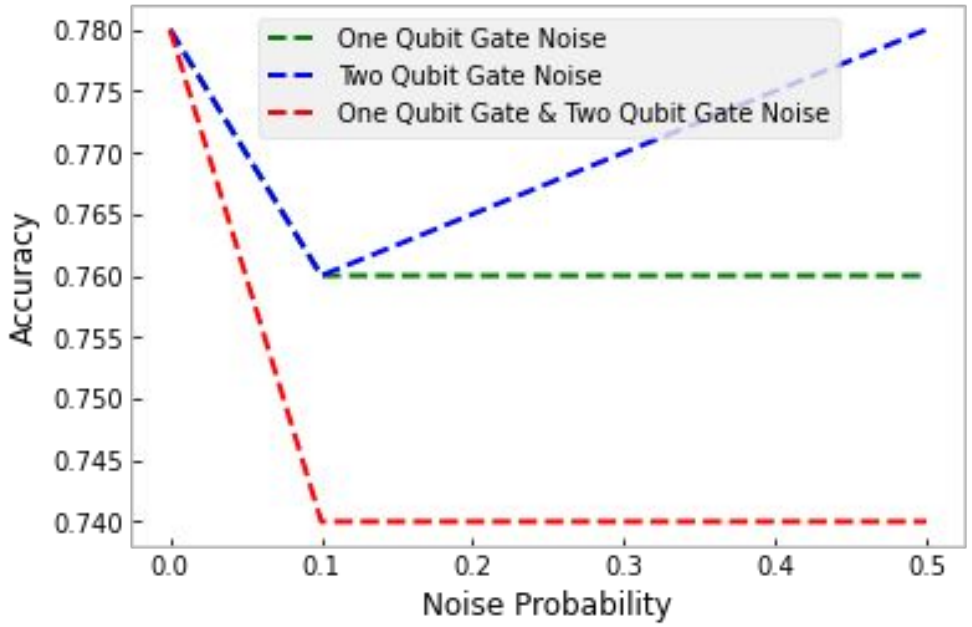
1. **Depolarization Noise**
   Spontaneous transitions among eigenstates
   Induce the sudden death of maximally entangled qubits
   The amount of depolarization is controlled by the parameter provided to the Noise Model

2. **Pauli Noise**
   The amplitude and the phase of the qubit flip
   The percentage of flip depends on the parameter provided to the Noise Model

A. Depolarization Noise

- Analyze the impact of depolarization noise on single qubit gates and two qubit gates





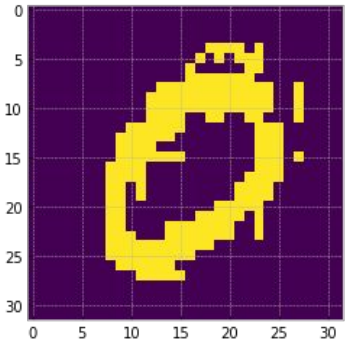
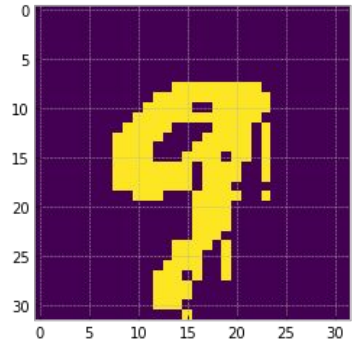
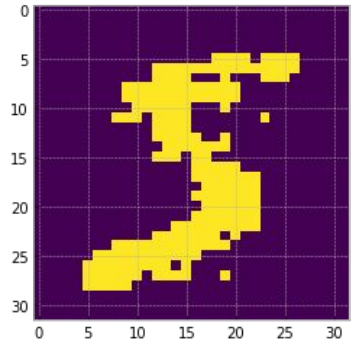
- Combination of depolarization noise on both types of gates impacts the accuracy

Carnegie Mellon University

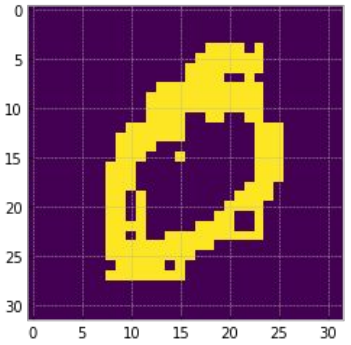
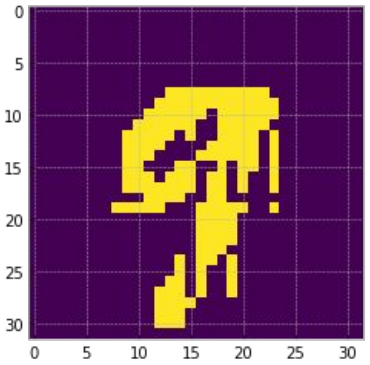
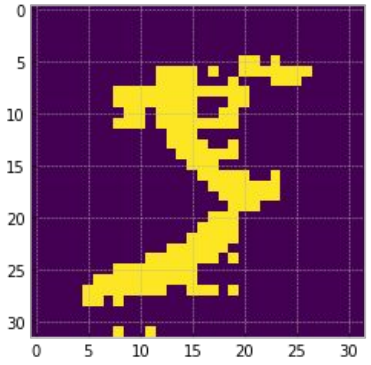
A. Depolarization Noise

- Edge detection results from the quantum circuit:

Noiseless Model

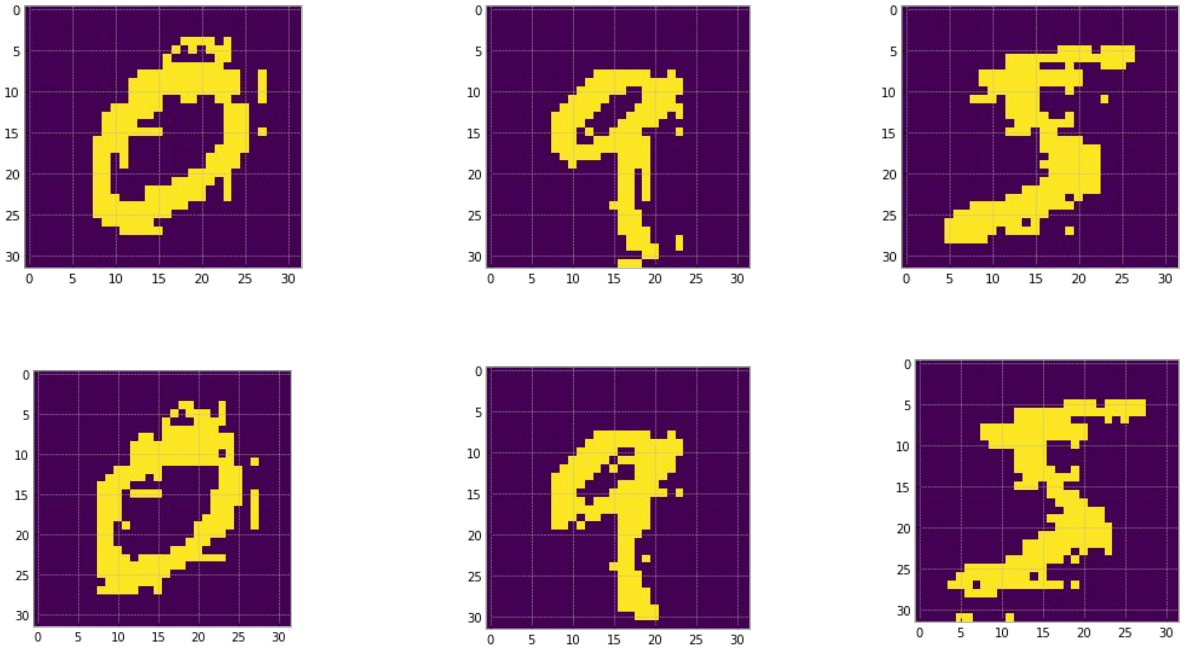
Noise Model with 0.5 Single Qubit Gate and 0.5 Two Qubit Gate Depolarizing Noise

Carnegie Mellon University

B. Pauli Noise

- Analyze the impact of pauli noise on single qubit gates with both bit flips and phase flips



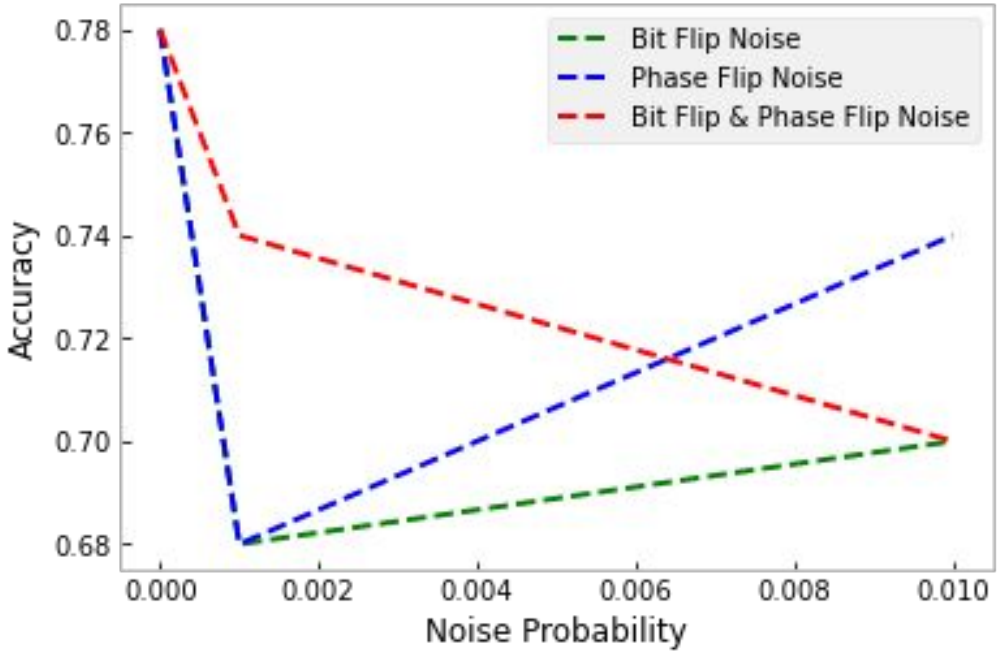
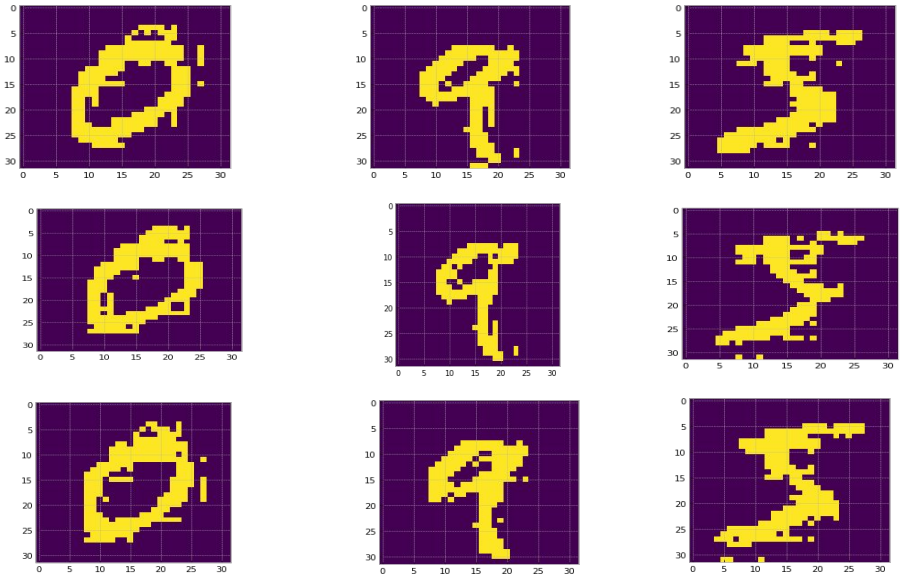


Figure: Top 3 images: Noiseless Model, Bottom 3 images: Noise Model with 0.01 Pauli Noise with both bit flip and phase flip

Carnegie Mellon University

C. Comparison: Depolarizing Noise and Pauli Noise

- Pauli Noise impacts the model performance more than depolarization noise



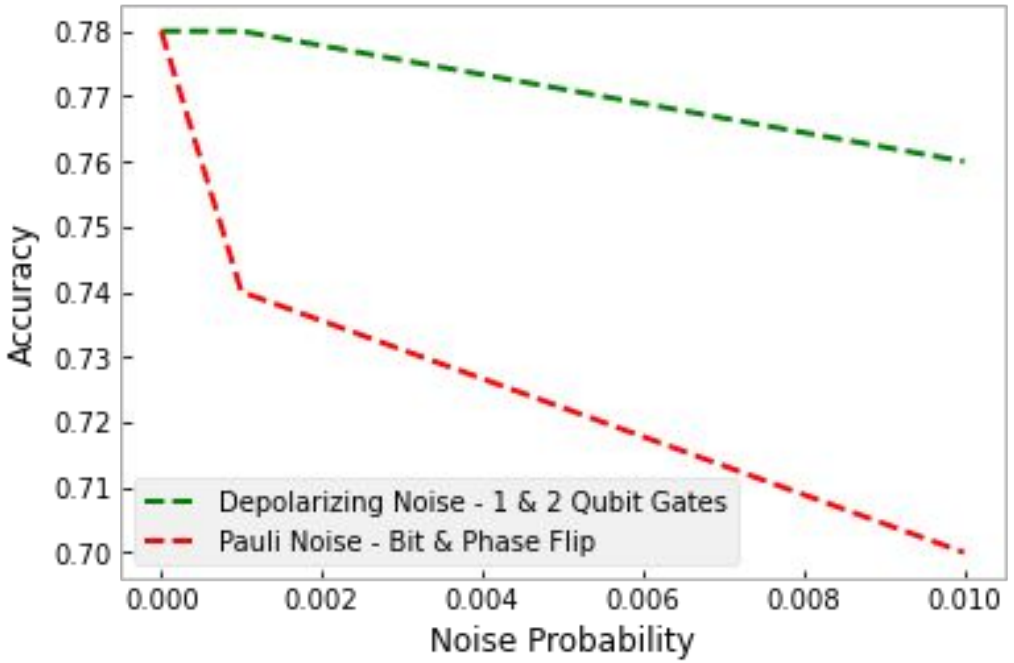


Figure: *Top 3 images:* Noiseless Model
Middle 3 images: Depolarizing Noise Model with 0.5 Single Qubit Gate and 0.5 Two Qubit Gate
Bottom 3 images: Noise Model with 0.01 Pauli Noise with both bit flip and phase flip

Carnegie Mellon University

Future Plan

- **Impact of different Noise models**
  Impact of other Noise Models on the quantum circuit results.

- **Image Verification**
  Image obtained from quantum image processing, is verified with original/real-time images for accuracy.

- **Handling large dimension data**
  Application of quantum image processing for larger dimension real-time images, to explore practical applications with greater accuracy.

Thank you!