On Classifying Images using Quantum Image Representation

Ankit Khandelwal
TCS Research
Tata Consultancy Services
Bengaluru, India
ankanit27.kh@gmail.com

M Girish Chandra
TCS Research
Tata Consultancy Services
Bengaluru, India
m.gchandra@tcs.com

Sayantan Pramanik
TCS Incubation
Tata Consultancy Services
Bengaluru, India
sayantan.pramanik@tcs.com

Abstract—Quantum Image Representation is researched from last few years, and more active in the recent past. Set to examine how these representations would be useful for Image Processing in a quantum way, we considered the Quantum Machine Learning problem of image classification in this paper. Encouraging results have been provided on classifying benchmark datasets of grayscale and colour images using two different classifiers and their combination. Multiclass classification performance has also been tested.

Index Terms—variational classifier, FRQI, MCQI, autoencoder, quantum image representation

I. INTRODUCTION

Quantum Image Representation (QIR) is a catch-all term for methods to encode an image as a quantum state. General data encoding methods [1] exist, such as Angle Encoding and Amplitude Encoding. But, these do not take advantage of the specific structure of image data. Thus, unique representation methods have been invented to tackle image data and its flexible use in the paper. In Section V, the implementation details used in the paper. In Section VI, the implementation details are mentioned. Section VII presents the classification results obtained from the simulations. Conclusion and a few markers for future work are provided in section VII.

II. QUANTUM IMAGE REPRESENTATION METHODS

This section gives a summary of the QIR methods used in the present work. Classically, an image in stored as an array of pixel values. If the image is $w \times h$ in size, it will have $w \times h$ pixels. Each pixel will have a single pixel value if the image is grayscale or it will be an array of three values representing RGB values. In summary, for this work:

1) Image dimension $= 2^n \times 2^n$ ($= 2^{2n}$ pixels) ($n = 1$ means a $2 \times 2$ image with 4 pixels),
2) Grayscale images $\rightarrow$ Pixel values $\in [0, 255]$ $\rightarrow$ In binary $\in \{0, 1\}^8$,
   Number of Matrix Elements $= 2^n \times 2^n$,
3) Colour images $\rightarrow$ RGB, pixel values of each channel $\in [0, 255]$,
   Number of Matrix Elements $= 2^n \times 2^n \times 3$

A. Flexible Representation of Quantum Images

FRQI encodes the image data into a quantum state given by:

$$|I(\theta)\rangle = \frac{1}{2^n} \sum_{i=0}^{2^{2n} - 1} [\cos \theta_i |0\rangle + \sin \theta_i |1\rangle] \otimes |i\rangle \quad (1)$$

$$\theta_i \in \left[0, \frac{\pi}{2}\right], \theta = (\theta_0, \theta_1, \ldots, \theta_{2^{2n} - 1})$$

Here, $|0\rangle$ and $|1\rangle$ are single qubit computational basis states and $|i\rangle, i = 0, 1, \ldots, 2^{2n} - 1$ are $2n$ qubit computational basis states. The $\cos \theta_i |0\rangle + \sin \theta_i |1\rangle$ part is used to encode the pixel values while $|i\rangle$ encodes the pixel location.

The circuit to encode the image can be constructed using Hadamard ($H$) and Control Rotation, $C^{2n}R_y(2\theta)$, gates. It needs to be prepared and measured multiple times to get back the image from the state. The image retrieval process is probabilistic, and the result will depend on the number of shots used.

The number of qubits used in this representation is $2n + 1$ with $2n$ qubits to encode the pixel location and 1 qubit to encode the pixel values. Pixel values are encoded as angles and are thus scaled to fit in the range $[0, \frac{\pi}{2}]$. 

B. Multi-Channel Representation for Quantum Image

MCQI representation uses $2n + 3$ qubits to encode colour images while using $2n$ qubits to encode the pixel location like FRQI and the 3 remaining qubits to encode the pixel values of the RGB channels. This encoding is inspired by FRQI. MCQI encodes the image into a quantum state given by:

$$|I(\theta)\rangle = \frac{1}{2^n + 1} \sum_{i=0}^{2^n-1} |C^i_{RGB}\rangle \otimes \left| i \right\rangle $$

The colour information is encoded in:

$$|C^i_{RGB}\rangle = \cos \theta_R^i |000\rangle + \cos \theta_G^i |001\rangle + \cos \theta_B^i |010\rangle + \sin \theta_R^i |100\rangle + \sin \theta_G^i |101\rangle + \sin \theta_B^i |110\rangle + \cos 0 |011\rangle + \sin 0 |111\rangle$$

Colour encoding angle is applied to the R channel qubit using Control Rotation ($C^2 R_y(2\theta)$) gates where it is controlled by the G and B channel qubits, and for each pixel, three $C^{2n}(C^2 R_y(2\theta))$ gates are applied to encode the position and value information. The $\theta$ values are calculated from pixel values:

$$\theta = \cos^{-1} p$$

where, $p$ is the pixel values $\in [0, 1]$. This range is achieved by dividing the integer pixel values $\in [0, 255]$ by 255.

As before, the image retrieval process is probabilistic and depends on the number of shots.

III. QUANTUM CLASSIFIERS

Two different methods to classify the images have been used in this paper. These methods are described below.

A. Variational Quantum Classifier

To classify the images, some value to distinguish the two classes is required. The $Z$ expectation value ($ez$) of the first qubit gives a natural split. A variational ansatz is applied to the quantum image, and the expectation value of the colour qubit for FRQI and the R channel qubit for MCQI is measured. The expectation value lies in the range $[-1, 1]$, and thus a split can be formed such that:

$$\text{class} = \begin{cases} 
-1 & \text{if } ez \leq s \\
1 & \text{if } ez > s 
\end{cases}$$

where $s$ is the split set to 0 by default but can be trained to get optimal results.

1) Ansatz: A straightforward ansatz, as shown in Fig. 1a, that consists of a general single-qubit rotation on each qubit, and then a layer of CNOT gates is used in this paper. The number of layers of the ansatz is a hyperparameter. The number of parameters in the classifier = $3 \times N \times l$ where $N = \text{number of qubits and } l = \text{number of layers}$.

B. Autoencoder Classifier

An autoencoder is a tool to reduce the dimension of data. The autoencoder’s quantum analogue [5] is used here to compress the image state into a single qubit state. To use it as a classifier, the autoencoder is trained to only compress the positive class. Generally, the compressed output from an autoencoder is fed to a different classifier (see for instance [6]). Here, we use a different approach to use the autoencoder for classification. The autoencoder is trained by maximizing the fidelity of the trash qubits with the zero state ($\langle 0 | 0 \rangle^T$) where $T$ is the number of trash qubits. Once the autoencoder is trained on the positive class of the training data, it is then used to compress the validation data. Then the fidelity of the trash qubits is measured again with the zero state and the images are classified depending on the resulting fidelity. The positive class should have higher fidelity values than the negative class. A single layer of the autoencoder is shown in Fig. 1b.

IV. DATASET DETAILS

Two grayscale and one colour image dataset have been used in this paper.
Fig. 3: MNIST data. Top row shows digit 0 and bottom row shows digit 1 for \( n \in [1, 5] \) from left to right.

**A. Bars and Stripes (BAS)**

This dataset contains black and white images of dimension \( 2^n \times 2^n \). Example images are shown in Fig. 2 for \( n = 5 \). These images are randomly generated. Horizontal stripes are one class, and vertical bars are another class. This dataset is used for binary classification.

**B. MNIST**

This is the famous dataset of handwritten digits [7]. This has been used for both binary (0 and 1) and multiclass (0, 1 and 2) classification. The original images are of \( 28 \times 28 \) size with 784 pixels which is between \( n = 4 \) and \( n = 5 \) thus, they first need to be squared into \( 2^n \times 2^n \). Bilinear interpolation is used to transform the data for different \( n \). There also exists another version of this data which contains 15 different corruption variations [8]. Rather than considering only the regular MNIST data set, these corrupted data sets have also been classified. Fig. 3 shows the MNIST images, and Fig. 4 shows the corrupted images.

Fig. 4: Corrupted MNIST data. Example of all 15 corruptions. \( n = 4 \).

**C. 2 x 2 Colour Images**

This is randomly generated \( 2 \times 2 \) colour image data for classification. The image has 4 pixels of random colours. For positive class the pixel values of the \( 4^{th} \) pixel are changed to \( (0, 0, 0) \). This makes the pixel black. The classification problem is then to differentiate between images with and without a black pixel. The pixel can also be modified to have dark shades instead of absolute 0 values. Fig. 5 shows example images for different shades.

![2x2 Colour Images](image)

**TABLE I: Validation Set Accuracies using the VQC and AC on FRQI representation for BAS data**

| Training Set Size | Classifier | \( n \) |
|-------------------|------------|--------|
|                   | \( VQC \)  | 1      | 2      | 3      | 4      |
| 100               | 1.0        | 1.0    | 0.955  | 0.993  |
| 200               | 1.0        | 0.818  | 0.820  | 0.889  |
| 500               | 1.0        | 0.858  | 0.842  | 0.866  |

**TABLE II: Validation Set Accuracies using the VQC and AC on FRQI representation for BAS data**

- PennyLane [9] has been used to simulate the circuits. JAX [10] is combined with PennyLane as a high-performance simulator and to utilize the GPU. The optimization library optax [11] is used for optimizing the classifier. Adam optimizer [12] is used with a 0.1 step size for optimization for 250 epochs. 5 layers of the VQC and 1 layer of the AC are used. scikit-learn [13] is used for classical processing. All simulations were done with ‘None’ shots of the PennyLane device. This gives analytic results. Different sizes of training datasets are used with a fixed 1000 data points for the validation data. Any deviations from these settings are mentioned in the text.

**V. IMPLEMENTATION DETAILS**

**VI. RESULTS AND DISCUSSION**

Table I shows accuracies on the validation set using different training dataset sizes and \( n \) values for the BAS data. Table II
TABLE II: Validation Set Accuracies using the VQC on FRQI representation for BAS data with shots

| n     | Shots | 1  | 2  | 10 | 50 | 100 | 200 | 500 | 1000 |
|-------|-------|----|----|----|----|-----|-----|-----|------|
| 1     | 0.652 | 0.679 | 0.872 | 0.993 | 0.992 | 0.993 | 0.984 |
| 2     | 0.509 | 0.569 | 0.747 | 0.917 | 0.858 | 0.971 | 0.924 |
| 3     | 0.504 | 0.510 | 0.764 | 0.479 | 0.896 | 0.943 | 0.956 |

TABLE III: Validation Set Accuracies using the VQC and AC on FRQI representation for MNIST data

| Training Set Size | Classifier | n     | 1  | 2  | 3  | 4  | 5  | 10  |
|-------------------|------------|-------|----|----|----|----|----|-----|
| 100               | VQC        | 0.946 | 0.960 | 0.992 | 0.993 |
|                   | AC         | 0.922 | 0.905 | 0.917 | 0.811 |
| 200               | VQC        | 0.938 | 0.990 | 0.995 | 0.996 |
|                   | AC         | 0.904 | 0.877 | 0.793 | 0.691 |
| 500               | VQC        | 0.933 | 0.955 | 0.993 | 0.995 |
|                   | AC         | 0.904 | 0.950 | 0.925 | 0.826 |

TABLE IV: Validation Set Accuracies using the VQC and AC on FRQI representation for MNIST Corruption data

| Training Set Size | Classifier | Shot Noise | Impulse Noise | Glass Blur | Motion Blur | Shear |
|-------------------|------------|------------|---------------|------------|-------------|-------|
| 500               | VQC        | 0.996      | 0.998         | 0.994      | 0.990       | 0.997 |
|                   | AC         | 0.918      | 0.929         | 0.980      | 0.918       | 0.858 |
| 500               | Scale      | 0.988      | 0.993         | 0.993      | 0.965       | 0.993 |
|                   | Rotate     | 0.982      | 0.960         | 0.977      | 0.880       | 0.973 |
| 500               | Fog        | 0.988      | 0.995         | 0.997      | 0.992       | 0.989 |
|                   | Spatter    | 0.951      | 0.966         | 0.971      | 0.972       | 0.994 |
| 500               | Dotted Line| 0.951      | 0.966         | 0.971      | 0.972       | 0.994 |
|                   | ZigZag     | 0.951      | 0.966         | 0.971      | 0.972       | 0.994 |
| 500               | Canny      | 0.951      | 0.966         | 0.971      | 0.972       | 0.994 |
|                   | Edges      | 0.951      | 0.966         | 0.971      | 0.972       | 0.994 |

shows accuracies on the validation set using different numbers of shots and n values for the BAS data with a training set size of 100 and using 2 layers of VQC with 50 epochs.

Table III shows accuracies on the validation set using different training dataset sizes and n values for the MNIST data. For AC 100 epochs for n < 3 with MNIST data have been used. Table IV shows accuracies on the validation set for the MNIST Corruption data for n = 4. Table V shows the results for Multiclass classification between 0, 1 and 2 digit images using the VC. For this, the ez results for Multiclass classification between 0, 1 and 2 digits using the VQC with different training set sizes and different n values using the one-vs-rest strategy for a validation set size of 5000. For one-vs-rest, instead of measuring ez, the probability of getting the first qubit in the computational basis states is measured. The probability of measuring |0⟩ is taken as the positive class, and the probability of |1⟩ is interpreted as the rest class. Then the log-loss is used to train the three classifiers corresponding to the three digits and then the validation set is labelled according to the classifier which returns the max probability of |0⟩ state.

Table VII shows the accuracy on the validation set using different training dataset sizes and shade values for the 2 × 2 colour image data. As expected, the accuracy decreases as the shade is increased (see Fig. 6). This is because higher shade values imply smaller difference between the classes, with a value of 255 implying no difference between the two classes. When using autoencoder with MCQI, (3) is not used, and
TABLE VIII: Validation Set Accuracies using Combined Classifier for FRQI representation of BAS data

| Training Set Size | n (Number of Data Qubits) | 1 (1) | 2 (3) | 3 (3) |
|-------------------|---------------------------|-------|-------|-------|
| 150               | 1.0                       | 1.0   | 0.769 |
| 200               | 1.0                       | 1.0   | 0.712 |
| 500               | 1.0                       | 1.0   | 0.906 |
| 1000              | 1.0                       | 1.0   | 0.804 |

TABLE IX: Validation Set Accuracies using Combined Classifier for FRQI representation of MNIST data

| Training Set Size | n (Number of Data Qubits) | 1 (1) | 2 (3) | 3 (3) |
|-------------------|---------------------------|-------|-------|-------|
| 150               | 0.947                     | 0.961 | 0.995 | 0.996 |
| 200               | 0.938                     | 0.962 | 0.996 | 0.995 |
| 500               | 0.930                     | 0.957 | 0.995 | 0.992 |
| 1000              | 0.921                     | 0.949 | 0.993 | 0.994 |

TABLE X: Validation Set Accuracies using Combined Classifier for MCQI representation of $2 \times 2$ Colour Image data

| Shade | Training Set Size |
|-------|-------------------|
|       | (I) | (3) | (I) | (3) | (I) | (3) | (I) | (3) |
| 0     | 0.0  | 0.979 | 0.975 | 0.983 | 0.982 | 0.999 | 0.998 | 0.999 | 0.997 |
| 10    | 0.0  | 0.989 | 0.993 | 0.991 | 0.997 | 0.997 | 0.996 | 0.996 | 0.997 |
| 20    | 0.0  | 0.969 | 0.971 | 0.992 | 0.998 | 0.995 | 0.983 | 0.983 | 0.976 |
| 30    | 0.0  | 0.968 | 0.975 | 0.952 | 0.953 | 0.965 | 0.969 | 0.974 | 0.978 |
| 50    | 0.0  | 0.662 | 0.859 | 0.856 | 0.886 | 0.816 | 0.877 | 0.836 | 0.883 |
| 100   | 0.0  | 0.321 | 0.645 | 0.705 | 0.745 | 0.626 | 0.756 | 0.669 | 0.694 |
| 200   | 0.0  | 0.509 | 0.533 | 0.506 | 0.589 | 0.523 | 0.610 | 0.489 | 0.587 |
| 255   | 0.0  | 0.496 | 0.520 | 0.511 | 0.494 | 0.501 | 0.510 | 0.496 | 0.491 |

instead, the pixel values are used directly.

Table VIII and IX show the results of using a combined classifier on BAS and MNIST data, respectively, where first the autoencoder is used to compress the state and then the VQC is applied to the data qubits for classification. 5 layers have been used for both the autoencoder and the VQC with a single data qubit for $n = 1$ and three data qubits for $n > 1$. When using a single data qubit, one layer of the VQC is one general rotation gate. Table X shows the results of the combined classifier on $2 \times 2$ colour image data. As $n = 1$ for this dataset, both 1 and 3 data qubits have been used.

One advantage of using image representation methods to encode the images into quantum states over more direct methods is the reduced number of qubits required. This allows the classifier also to use fewer qubits. This can be useful in cases where the image itself is stored in Quantum Random Access Memory (QRAM), but when it is needed for processing, it is first compressed to reduce the qubit count. Another futuristic aspect of the qubit efficient representation of the images is for transmitting them through quantum internet.

VII. CONCLUSIONS AND FUTURE WORK

Encouraging results on benchmark datasets have been obtained with both VQC and AC for binary and multiclass image classification. The work can be expanded to classify more involved images, which can be larger and more complex. The ansatz used for VQC was straightforward. More research can be done to find a better ansatz that can improve performance while using fewer epochs. Some preliminary results of using a small number of shots have also been provided. A more thorough study can be performed on the effect of the number of shots. The number of qubits for the classifier can be further reduced by using an autoencoder to compress the state.

As multiclass classification is an important Machine Learning task, how to improve its performance further than what is obtained needs to be studied. More expressive models might help in this regard. Another issue is getting multiple outputs from a quantum circuit through different observables, for instance, is also worth pursuing.

The impact of noise in the data on performance can also be studied. One can also train the model with regular MNIST data and then use it to classify the corrupted data. This study might shed some light on how noise and distortion affect the model. Comparison of the states for distorted and regular images might also throw some light on how to deal with noise and other artifacts in the data. To compress further effectively in noisy scenarios, denoising autoencoders can also be considered. Further, one can look into other Image Processing tasks like filtering on the representations/encodings. While the results were obtained from simulators in the present study, how the models will perform on evolving noisy quantum computing hardware is another essential thread to follow.

REFERENCES

[1] R. LaRose and B. Coyle, “Robust data encodings for quantum classifiers,” Phys. Rev. A, vol. 102, p. 032420, Sep. 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.102.032420
[2] P. Q. Le, F. Dong, and K. Hirota, “A flexible representation of quantum images for polynomial preparation, image compression, and processing operations,” Quantum Information Processing, vol. 10, no. 1, pp. 63–84, Feb. 2011. [Online]. Available: https://doi.org/10.1007/s11128-010-0177-y
[3] B. Sun, A. M. Iliyasu, F. Yan, F. Dong, and K. Hirota, “An rgb multi-channel representation for images on quantum computers,” J. Adv. Comput. Intell. Intell. Informatics, vol. 17, pp. 404–417, 2013.
[4] A. Khandelwal, M. G. Chandra, and S. Pramanik, “On classifying images using quantum image representation,” 2022. [Online]. Available: https://arxiv.org/abs/2206.11509
[5] J. Romero, J. P. Olson, and A. Aspuru-Guzik, “Quantum autoencoders for efficient compression of quantum data,” Quantum Science and Technology, vol. 2, no. 4, p. 045001, 2017.
[6] S. Mangini, A. Maruzzo, M. Piantanida, D. Gerace, D. Bajoni, and C. Macchiavello, “Quantum neural network autoencoder and classifier applied to an industrial case study,” Quantum Machine Intelligence, vol. 4, no. 2, p. 13, Jun 2022. [Online]. Available: https://doi.org/10.1007/s42484-022-00070-4
[7] Y. LeCun, C. Cortes, and C. Burges, “Mnist handwritten digit database,” AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist, vol. 2, 2010.
[8] N. Mu and J. Gilmer, “Mist-c: A robustness benchmark for computer vision,” arXiv preprint arXiv:1906.02337, 2019.
[9] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri, K. McKiernan, J. J. Meyer, Z. Niu, A. Száva, and N. Killoran, “Pennylane: Automatic differentiation of hybrid quantum-classical computations,” 2018. [Online]. Available: https://arxiv.org/abs/1811.04968

[10] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, “JAX: composable transformations of Python+NumPy programs,” 2018. [Online]. Available: http://github.com/google/jax

[11] M. Hessel, D. Budden, F. Viola, M. Rosca, E. Sezener, and T. Hennigan, “Optax: composable gradient transformation and optimisation, in jax!” 2022. [Online]. Available: http://github.com/deepmind/optax

[12] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.