A Quantum-Classical Hybrid Method for Image Classification and Segmentation

Sayantan Pramanik[†], M Girish Chandra[*], C V Sridhar[†], Aniket Kulkarni[†], Prabin Sahoo[+], Vishwa Chethan D V[†],
Hrishikesh Sharma[*], Ashutosh Paliwal[†], Vidyut Navelkar[†], Sudhakara Poojary[†], Pranav Shah[†], Manoj Nambiar[*]

[†]*TCS Incubation*   [*]*TCS Research*   [+]*TCS Manufacturing and Utilities*

{sayantan.pramanik, m.gchandra, sridhar.cv, aniket.k, prabin.sahoo, v.dv2, hrishikesh.sharma, ashutosh.paliwal,
vidyut.navelkar, sudhakara.poojary, pranav.shah, m.nambiar}@tcs.com

Abstract—Enormous activity in the Quantum Computing area has resulted in considering them to solve different difficult problems, including those of applied nature, together with classical computers. An attempt is made in this work to nail down a pipeline consisting of both quantum and classical processing blocks for the task of image classification and segmentation in a systematic fashion. Its efficacy and utility are brought out by applying it to Surface Crack segmentation. Being a sophisticated software engineering task, the functionalities are orchestrated through our in-house Cognitive Model Management framework.

Index Terms—quantum computing, variational quantum classifiers, q-means, data encoding, ansatz, measurement, quantum software development life cycle, computer vision, image classification, image segmentation, crack detection

I. INTRODUCTION

Image Segmentation and Classification is an extensively researched area with applications spanning across domains. Because of its importance, many classical approaches exist. With the emerging scenario of Quantum Computing, there is thrust to explore them in these tasks, either for a possible speed up or better some of the aspects, like, using a smaller number of training examples, more accuracy etc. One more aspect which is getting settled is the fact that even when improved quality and good enough sized quantum computers are available in the future, both quantum and classical computing work and "cooperate" together to solve useful real-life problems. Keeping this hybrid-architecture in mind, this paper attempts to suggest a possible pipeline of classical and quantum processing blocks to achieve image segmentation and further classification of these segments. In particular, we consider the classification and segmentation of Kaggle Surface Crack Data and in the process, suggest a rather generic pipeline to accomplish the task, taking into account the limitations of number of qubits and depth of the quantum circuits at present into account. The results obtained confirm the expected functionality of the proposal accompanied with quantitative metrics reinforcing its usefulness. Further, with different processing models (machine learning as well as others) of both quantum and classical variety working together, the software aspects involved in putting the pipeline into action are definitely complex. In order to negotiate this aspect and "productize" this software solution in the future, our exclusive Cognitive Model Management

Fig. 1: Generic pipeline for segmenting cracks in images, which involves the three steps highlighted in the simple flowchart.

(CMM) framework and OpSense tool are utilized to facilitate a systematic execution of the whole workflow.

In Section II, the big picture of the pipeline is briefed; Section III touches upon the data set considered; Section IV covers the details of various models in considerable detail, including segmentation, classification and optimization. Before providing comparative results, fully classical approaches are mentioned in Section V, followed by Section VI on results. There is a small section on Future Tasks (Section VII). Section VIII brings out our initiation into the Quantum Software Engineering pivoted on CMM and OpSense.

II. OVERALL PIPELINE

As shown in Fig. 1, the problem of crack detection and segmentation has been addressed in three steps:

1) The first step involves classifying the images to detect whether it contains a crack or not. If it is found not to contain any anomalies, then the subsequent pipeline may be terminated for that particular input image.
2) After classification, the images which are identified to have cracks in them are segmented into two different clusters. Depending on the size and resolution of the images, this could prove to be a time-consuming and computationally-intensive step, which need not be unnecessarily wasted on images which do not have any cracks. Hence the requirement of the first classification step. Further, clustering being an unsupervised algorithm, is expected to make the pipeline generic and transferable to other data sets for similar problems, without the requirement of retraining.
3) Once the clusters have been identified in an image, the different regions can be classified to check which of them actually correspond to cracks.

Different parts of the pipeline have been expounded upon in greater detail, along with more technical arguments, implementation steps and results, and subtleties in Sec. IV.

III. The Dataset

The data set in question, namely the Surface Crack data set, is a set of $40,000$ low-resolution images that are openly available-to-all via Kaggle [1]. Each image is 227×227, 3-channel RGB, accompanied with a binary label which identifies whether it contains a crack or not. The entire data set has been evenly-balanced between images with and without cracks, available as two folders named "Positive" and "Negative" with $20,000$ images in each. It must be noted that the images that fall under the "Negative" label do however contain dents, discolouration, and other forms of aberrations which are sometimes difficult to differentiate from actual cracks. Other surface patterns and textures exist in instances from both classes. Representative images, one from each folder, can be found in Fig. 2.

(a) An image without any cracks in it, labelled "Negative"

(b) An image with a crack

Fig. 2: Examples of images with and without cracks.

IV. Detailed Pipeline and Intermittent Results

An bried overview of the pipeline was presented in Sec. II, the components of which have been dissociated into more granular steps in Fig. 3. The process starts with the image classifier, for which features are extracted from the images using a pre-trained deep learning architecture (such as VGG16 [2] trained on the ImageNet data set, and stripped of its final classification layer). Considering the simplicity of the data set this approach is being tested on, it was concluded that using more complicated feature-extractors would be overkill. The VGG16 model extracts 4096 features from each image which require too many qubits to encode into a quantum circuit using angle-embedding, or too deep circuits if amplitude-embedding is used instead. Better classical-to-quantum data encoding techniques are presently being researched [3] to find an appropriate trade-off between circuit-depth and qubit-requirement.

Keeping the limitations of presently-available simulators and quantum processors in mind, the dimensionality of the features obtained from VGG16 was reduced from 4096 to the order of 10. Specifically, within the scope of this work, Principal Component Analysis (PCA) has been used to reduce the 4096 features to just 4, which were then passed on to the first quantum image classifier for training and inference purposes. This is similar to the Classical+Quantum approach proposed in [4]. Alternative approaches that use quantum computing for feature extraction and classification have also been proposed [4]–[6] that have been explored, but not implemented within the context of this work. Further, quantum classification can also be achieved through algorithms other than variational quantum classifiers, such as quantum versions of support-vector machines (QSVM) [7] post the feature-extraction stage.

Following image classification, the task of image segmentation has been attempted through a quantum counterpart of the k-means algorithm, known as q-means [8]. Additional methods of image clustering such as graph-cuts [9] which model the pixels of an image as vertices of a graph and edges formed by pixel intensities of n-nearest neighbours, and image segmentation by Quantum Hadamard Edge Detection (QHED) [10] also serve as interesting and viable approaches. As already discussed, all of the individual regions detected through image segmentation are passed on for another round of classification which uses the same techniques described for the aforementioned image classifier. The regions identified as cracks are highlighted as such in the original image, which concludes the process of crack segmentation [1].

A. Segmentation by Quantized k-means

Given a set of data points, each consisting of d-dimensional feature vectors, it is often necessary to segregate them into k different baskets, The ith data point is given by $v_i = \{v_i^1, v_i^2, \ldots, v_i^d\}$, $i \in [N]$; and the baskets are characterized by cluster centroids, which themselves are d-dimensional vectors, $c_j = \{c_j^1, c_j^2, \ldots, c_j^d\}$, $j \in [k]$. One starts with arbitrary guesses of the k centroid vectors and the pairwise absolute distance between each cluster centroid and data point is calculated: $|d(v_i, c_j)|$. The ith data vector is assigned to the cluster to which it is situated closest. Having allocated each data point to a cluster, the centroids themselves are updated by taking the average of corresponding features of each vector in the cluster. The process is repeated to convergence to obtain the final clusters and centroids.

1) q-means: A quantum equivalent of the k-means algorithm, aptly dubbed q-means, has been proposed in [8] which performs in poly-logarithmic complexity with respect to N, compared to being linear in N for the classical implementation. The algorithm, though, comes with the caveat the it employs the use of a $QRAM$ [11] to achieve the said improvement in performance. $QRAM$ has been an area of active research over the recent years, but is not something that is available for immediate use. In this paper, we have
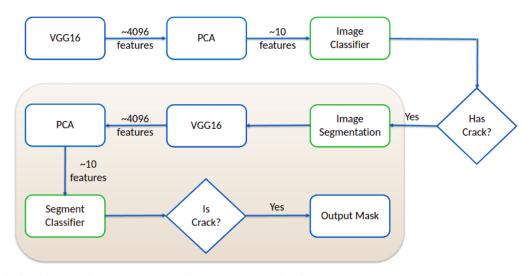
Fig. 3: Detailed pipeline to solve the problem of crack segmentation in images. The green-coloured boxes are steps for which quantum algorithms have been implemented and utilized. The steps in the shaded box correspond to image segmentation and classifying the detected regions.

attempted to circumvent this problem by using a somewhat naive strategy which works as a placeholder in the NISQ-era [12] until such memory units are available. Until such time, although a speed-up over classical techniques is not apparent, the exercise serves as a proof-of-principle which will be ready to provide benefit in the near future.

2) Distance Metrics: The cornerstone operation in classical k-means and quantum q-means algorithm is the calculation of distances between cluster centroids and the data vectors. The most common distance-metric of choice is the Euclidean distance D_E between two points in d-dimensional space. In the quantum-regime, Euclidean distance calculation is done by expressing the data vectors and centroids in the form of statevectors, and an affine transform of the modulus of inner products between them give the requisite distance values:

$$
\begin{aligned}
D_E^2(|x\rangle, |y\rangle) &= |(|x\rangle - |y\rangle)|^2 = ((\langle x| - \langle y|)(|x\rangle - |y\rangle)) \\
&= 2 - \langle x|y\rangle - \langle y|x\rangle \\
&= 2 - 2\langle x|y\rangle
\end{aligned}
\tag{1}
$$

where the entries of $|x\rangle$ and $|y\rangle$ being real-valued, $\langle x|y\rangle = \langle y|x\rangle$. $|D_E|$ and D_E^2 having the same monotonicity, the later can be used as a way to quantify the distance between data and centroid vectors without adversely affecting the performance of the algorithm [13].

In the subsequent subsections, the details of some quantum algorithms that help calculate D_E^2, along with their implementation details, advantages and disadvantages, have been spelt.

3) Swap Test: The Swap Test [14] is perhaps the most famous method of quantifying the overlap between two quantum states given by $|x\rangle$ and $|y\rangle$. The respective states are first prepared from $|0\rangle$ by the application of appropriate parametrized unitaries $E(x)$ and $E(y)$, respectively, which encode the points x and y into quantum states. The circuit in Fig. 4 gives an overview of the overall procedure.
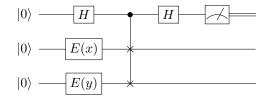

Fig. 4: Quantum Circuit to implement Swap Test.

The probability of measuring the first qubit in the state 0 in the computational basis is given by Eq. (2), can be used to determine the value of $\langle x|y\rangle$, which can be retrofitted into Eq. (1) to get D_E^2.

$$
P(0) = \frac{1}{2}\left(1 + |\langle x|y\rangle|^2\right)
\tag{2}
$$

However, the problem with using the Swap Test for determining an equivalent of Euclidean distance is multifold. The most obvious one is that Eq. (2) gives the value of $|\langle x|y\rangle|^2$, whereas Eq. (1) requires $\langle x|y\rangle$ to be evaluated. Further, the algorithm uses a controlled-swapping operation, which can prohibitively increase the depth of the circuit if the registers for $|x\rangle$ and $|y\rangle$ use multiple qubits.

4) Hadamard Test: Some of the difficulties faced in employing the Swap Test can be addressed through the use of Hadamard Test [15], instead, the circuit depiction of which can be found in Fig. 5. The latter also uses only about half the qubits compared to the former. A quick and simple analysis of the algorithm suggests that the Z expectation value of the first qubit is:

$$
\langle Z\rangle_0 = Re\left(\langle\psi|U|\psi\rangle\right)
\tag{3}
$$

The Hadamard test can be utilized to serve our purpose by defining $U = E(y)^\dagger E(x)$ and $|\psi\rangle = |0\rangle$, which simplifies

$Re\left(\langle\psi|U|\psi\rangle\right)$ to $\langle 0|E(y)^\dagger E(x)|0\rangle$. Similar to the Swap Test, the Hadamard analogue faces the potential challenge of running into higher circuit depth due to the presence of the CU gate. The Hadamard-Overlap test described in [16] partially bypasses the problem of controlled unitaries at the cost of using almost double the number of qubits in the circuit.
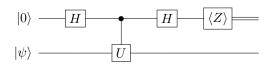

Fig. 5: Quantum circuit to calculate $Re\left(\langle\psi|U|\psi\rangle\right)$ using Hadamard Test.

5) Simple Overlap Calculation: The circuit in Fig. 6 show perhaps the simplest and most economical algorithm that can be used to calculate the similarity of two quantum statevectors. The method estimates $\langle y|x\rangle = \langle 0|E(y)^\dagger E(x)|0\rangle$ by consecutively applying the encoding unitaries $E(x)$ and $E(y)^\dagger$ to the same set of qubits, without the application being controlled by another qubit. The requisite value is then extracted by finding the probability of measuring all the qubits in the state 0 by running the circuit for multiple shots. Alternatively, the expectation value of the zero-projector, $|0\rangle_n\langle 0|_n$, also provides equivalent results. Owing to the shallow depth and low number of qubits required, this algorithm emerges as the most conducive choice to find D_E^2 in the implementation of q-means algorithm.
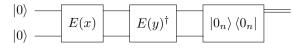

Fig. 6: Simple circuit to compute similarity between two quantum states.

The depicted circuits of all the algorithms described for distance estimation or similarity evaluation must be run for multiple shots to glean the requisite information from them. Naturally, higher the number of shots used, better are the results obtained. This acts as a NISQ-aligned alternative to the use of amplitude estimation [17], or variants thereof [18], [19], to extract the same information from the circuits, which comes at the cost of higher number of qubits required and much higher circuit depth, both of which are a bane of NISQ processors.

B. Image Segmentation

Image Segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics [20]. A lot of research has been done in the area of image segmentation using clustering. There are a plethora of methods and one of the popular ones is to use the k-means clustering algorithm. Although this algorithm was not originally developed specifically for image processing, it has been adopted by the computer vision community and is used up to these days [21]. The k-means algorithm requires the a priori knowledge of the number of clusters (k) into which the image pixels should be grouped. Each pixel of the image is repeatedly and iteratively assigned to the cluster whose centroid is closest to the pixel.

The q-means algorithm has been leveraged for the purpose of image clustering as follows: the images were converted to grayscale, as the pixel-intensity was found to be the qualifying feature to identify whether it belongs to a crack-region or not. The number of available qubits being limited in the NISQ-era, the images were also downscaled from 227×227 to a resolution of 50×50, post which a 5×5 Gaussian blurring filter was applied to get rid of noise and somewhat irrelevant high frequency components from the images. The new resolution was chosen arbitrarily and was visually found to retain the discriminating features between the cracks and background regions. The complexity of the images in the data set being low, the number of clusters has been set to two, i.e., $k = 2$; and the cluster-centroids are specified by 8-bit integers. Henceforth, the intensities of the pixels, which range between $[0, 255]$, have been treated as single-featured vectors of a data set.
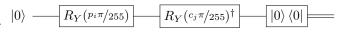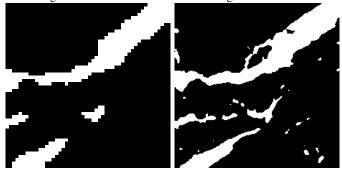

Fig. 7: Circuit to find distance of ith pixel-intensity from j^{th} cluster-centroid.

The simple overlap calculation method, described in Sec IV-A5 has been used to calculate the distances of the data vectors from the cluster-centroids. Angle embedding using R_Y gates has been used to encode the intensity and centroid information, post appropriate scaling, into a single qubit per pixel to preserve the simplicity of the procedure. If intensity of the ith pixel is p_i, and centroid of the jth cluster is c_j, then $E(p_i) = R_Y(p_i\pi/255)$ and $E(c_j) = R_Y(c_j\pi/255)$, resulting in a per-pixel circuit given in Fig. 7. Why this approach works for our distance calculations and how it compares with the classical calculations are captured in Appendix A; the results reinforce the usability of the suggested distance.

The aforesaid process for image segmentation through q-means was carried out by iterating over each of the 2500 pixels in the downscaled image and over the two cluster-centroids. This implied the use of $2500 \times 2 = 5000$ qubits to find the Euclidean-equivalent distances per iteration of the q-means algorithm. Presently, this is way beyond the number of qubits that are available at our disposal through any vendor of quantum processors. Also of note is the fact that most popular vendors that provide the use of their quantum processors through the cloud, charge per circuit and the number of shots that the circuits are executed for, and not the overall width of the circuit. Thus, the most economical option is to use as many qubits per circuit as possible, a proposition which

(a) Original image from data set containing cracks

(b) Result of q-means on downscaled image

(c) Result of k-means on downscaled image

(d) Result of q-means on original image

Fig. 8: Results of k-means and q-means image segmentation on 50×50 and 227×227 versions of an image with cracks.

is not always advisable due to the reduced connectivity of the qubits in a processor which may lead to great circuit depths due to repeated swapping of the qubits. On fully-connected devices like IonQ's [22], however, this is easily amenable. Since the proposed overlap-calculation circuit does not use entanglement (which does not provide a quantum advantage, but again, this is a place-holder approach until such time a QRAM is available), the qubits are batched into groups of n for pixel-centroid distance estimation, where n is the number of qubits that are available. The results of q-means image segmentation on AWS Braket Simulator of an image with cracks (original resolution and downscaled) and comparison with classical k-means based segmentation have been portrayed in Fig. 8. It must be noted that the simulation times for the q-means algorithm acting on downscaled images ranged between $60 - 90$ seconds per image.

C. Quantum Classifier

Quantum machine learning has the potential for broad industrial applications, and the development of quantum algorithms for improving the performance of neural networks is of particular interest given the central role they play in machine learning today [23].

Feedforward neural networks play a key role in machine learning, with applications ranging from computer vision and speech recognition to data compression and recommendation systems. In a supervised learning scenario, a network is trained to recognize a set of labelled data by learning a hierarchy of features that together capture the defining characteristics of each label. Once trained, the network can then be used to recognize unlabelled (test) data.

Quantum algorithms, i.e., algorithms that can be executed on a quantum computer, have been investigated since the 1980s, and have recently received increasing interest all around the world. One of the main applications for quantum computing is the development of new algorithms for machine learning [24]. Variational quantum algorithms (VQAs) use classical optimizers to train parametrized quantum circuits and represent an advancement in quantum computing running on NISQ computers. Variational algorithms can be the basis for numerous applications, including the design of a quantum classifier.

In our proposed pipeline, we designed Variational Quantum Classifiers (VQC) to carry out the requisite classification at two places (see Fig. 3); the first classifier learns how to classify the whole input image to pass for crack detection or not. These images are segmented they are in turn tagged to cracks or not by the second classifier. A VQC itself consists of three stages:

1) State Preparation, which also encodes the classical data (feature vector) for further quantum processing,
2) A model circuit or Ansatz with optimizable parameters,
3) A measurement stage.

VQC generally consists of a number of one-qubit gates: the Hadamard gate H, and rotational gates; plus some entangling gates: the controlled-NOT and the controlled-Z. Very specifically, in our work we used angle encoding for the classical features and incorporated strongly entangling layers [25] in the Ansatz. The classical counterpart, completing the hybrid nature of VQC utilize the measurement outputs to evaluate a appropriately chosen cost function and further minimizing it either through the use of gradients or invoking non-gradient approaches (see also [26]).

Some of the advantages of Variational Quantum Classifiers include - A VQC can outperform a classical model using far less free parameters and, thus, being more efficient. Further, a complex classification task requires deeper quantum circuits, which nevertheless grow at a slower pace than the number of neurons needed in a Classical Neural Network for the same task [26]. Also, in some cases, VQCs can perform similar to (classical) classifiers with a lesser number of training samples.

1) Data Dimensionality Reduction: Large data sets are increasingly widespread in many disciplines. In order to interpret such data sets, methods are required to drastically reduce their dimensionality in an interpretable way, such that most of the information in the data is preserved. Many techniques have been developed for this purpose, but principal component analysis (PCA) is one of the oldest and most widely used. Its idea is simple—reduce the dimensionality of a data set, while preserving as much 'variability' (i.e.statistical information) as possible [27]. The latter translates into finding new variables that are linear functions of those in the original data set, that successively maximize variance and that are uncorrelated with

each other. Finding such new variables, the principal components (PCs), reduces to solving an eigenvalue/eigenvector problem [27].

As depicted in our pipeline, PCA is used as a dimensionality reducing preprocessing stage, whose output are used as "classical" features input to the VQC. The rationale is rather simple: since the dimension of the data corresponds to the number of qubits required in order to encode the data, and further, we can handle limited number of qubits both in the simulators and the NISQ computers, reduction in dimension is inevitable.

Needless to say, we have considered classical PCA for our purpose. (Classical) PCA involves a computational cost of $O(N^2)$, where, N is the size of the data vector, and the $n \times N$ covariance matrix of data is subjected to eigen-decomposition. It is useful to note that a Quantum PCA exists which has the computational cost of $O((\log N)^2)$ [28]. In future, when there is a progress in quantum computing hardware, including a form of QRAM, one can utilize the Quantum PCA for further speed up of the proposed pipeline execution.
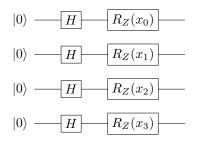

Fig. 9: R_Z encoding that encodes the features represented by x_is into the phases of quantum states.

2) Encoding: Of the myriad techniques available for classical-to-quantum data conversion and embedding them into circuits, the most simple method, i.e., angle-encoding was selected. More specifically, H gates followed by R_Z gates with the features as parameters have been used, which incorporates the said features as phases into the quantum states. The combination of H and R_Z, as in Fig. 9, was found to work slightly better than the other rotation gates available for angle-encoding, namely the R_X and R_Y gates. Basis-encoding would not have been conducive for embedding image features as it caters only to binary data. Amplitude encoding, on the other hand, would have been suitable for the task and advantageous due to the lower number of qubits required, but was neglected in favour of angle-encoding due to the overall difficulty and increased circuit-depth of the process.

3) Ansatzes: An ansatz is a block of gates with optimizable parameters that selects a basis of measurement appropriate for the problem at hand by iterating over the training inputs for multiple epochs and trying to match the post-processed measurements with the ground-truth values corresponding to each example. Similar to the bias-variance conundrum in classical machine and deep learning, an ansatz may provide a better expressibility or may choose to increase the entan-

glement capability of the qubits in a circuit [29]. Similarly, an ansatz may be constructed to be efficiently implementable on processors with a certain kind of qubit-connectivity, or hardware-agnostic versions of it can be created with just the complexity of the problem in mind.

To address the problem of image and segment classification, the three ansatzes (only single layers of which are displayed in Fig. 10) have been utilized. The ansatz in Fig. 10a, has the highest circuit depth of the three and uses three parametrized gates per qubit, per layer of the ansatz [25]. It is widely known that any single-qubit unitary can be expressed as a combination of $R_Y R_Z R_Y$ gates, up to a global phase. The rotation gates are followed by multiple $CNOT$ gates, the target qubit of each gate is determined using the relation $t = (c + l + 1) \bmod n$, where c and t are the control and target qubit numbers, l is the ansatz-layer being constructed and n is the number of qubits the ansatz operates on. An L-layer ansatz avails $3nL$ optimizable parameters.

The ansatz in Fig. 10b is similar in nature, but uses only an R_Y gate per qubit, which reduces the number of parameters to just nL. In contrast to the previous two, the third ansatz in Fig. 10c (employed in [30]) is constructed using a fixed topology of entangling gates. Since the $CNOT$ gates appear before the rotation ones in each layer, this ansatz is incompatible with phase-encoding through R_Z gates, and hence has been used in conjunction with angle-encoding in the form of R_Y rotations, instead.

4) Measurement: Post the application of the encoding and the ansatz layers, the quantum information must be brought back to the classical world through the process of measurement. In an otherwise linear manipulation of data by the various gates in the circuit, measurement provides the requisite non-linearity to the VQC [4]. Various measurement and post-processing techniques have been to glean information from the circuits - using the parity of qubit-readout [7], measuring the expectation values in a certain basis and passing them along to another classical neural network for inference [4], etc., are among the popular approaches.

The expectation value with respect to the Pauli-Z operator of the first qubit, which lies in the range of $[-1, 1]$, has been used in this work as the feature that segregates each data point into either of the two classes in the binary-classification task. If the measured value is beyond a certain threshold, that is determined by the classical optimizer, then the data point is assigned to a particular class; otherwise, the other class is chosen for the given input.

5) Classical Optimizer: To accrue some benefit out of Noisy Intermediate-Scale Quantum computers that are plagued with limited number of qubits, noisy and decoherence, the use of hybrid quantum algorithms has been suggested. In such algorithms, certain tasks that are amenable on quantum are offloaded to quantum processors, and classical and quantum counterparts work in tandem to solve a larger problem. In variational quantum algorithms, more often than not, the optimization of parameters is entrusted to the classical processors.

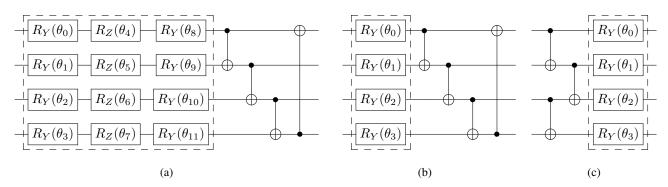Within the scope of this work, the gradient-free $COBYLA$

(a) (b) (c)

Fig. 10: (a) First layer of the Strongly Entangling Layers ansatz. The subsequent layers feature parametrized $R_Y R_Z R_Y$ gates followed by $CNOT$ gates with skip connections, with the skip-level depending on the layer-number. (b) First layer of the Basic Entangling Layers ansatz. Similar to the Strongly Entangling ansatz, the subsequent layers feature parametrized R_Y gates followed by $CNOT$ gates with skip connections, with the skip-level depending on the layer-number. (c) A single layer of the ansatz used in the Pennylane tutorial. The dashed boxes include rotational gates with optimizable parameters.
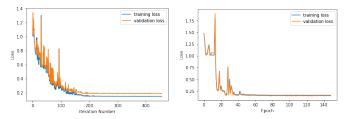
optimizer from the Scipy python package has been employed. The calculation of quantum gradients using methods like the parameter-shift rule [31] was avoided to keep the number of shots and circuit simulation times in check. To the best of the authors' knowledge, AWS Braket SDK does not provide an interface to TensorFlow, PyTorch, or any of the popular deep learning libraries. This hinders the creation of computational graphs from the circuits, rendering the advanced optimizers from the aforesaid libraries unusable. The classical optimizer was tasked with minimizing the Mean Squared Error (MSE) between the ground-truth values and the predictions from the VQC.

D. Image Classification

As illustrated in Sec. II, a classifier is run on each image to qualify whether it contains a crack or not. The rest of the inference pipeline is executed only if it is determined that the image does indeed have crack(s) in it. Over the following subsections, the training process for such a pipeline has been described in detail.

1) Data set and Preprocessing: The data set described in Sec. III is a balanced one, i.e., it contains an even $20,000$ images for each category. However, in an industrial setting, anomalous data is more difficult to come by. With an exterme dearth of data for a particular task, the training of efficient classifiers is a precarious affair. This is a peril for which quantum classifiers seem to provide an anticipation of a remedy [32]. A similar impediment has been simulated by selecting a subset of the data with a modest imbalance. 1000 images were randomly chosen from the overall data set, with the precondition that only 25% of which would be images that have cracks in them. The dimensionality-reduced VGG16 features of the 1000 images were further preprocessed by scaling them to the range of $[0, \pi]$. 25% of these reduced and scaled feature vectors were set aside for testing, and of the remaining 75%, a further 25% were utilized only for the purpose of validation. As a result, the training was performed on only about 563 images.

2) Training: The variational quantum model for image classification was trained with the data set described above using phase-encoding, the three Ansatzes in Fig. 10 individually with $n = 4$, $L = 3$, 1000 shots per circuit, and the MSE loss was reduced using the COBYLA optimizer. Without gradient-calculation and with 36, 12 and 12 parameters, each, the loss from the circuits with the three ansatzes converged in about 90, 15 and 15 minutes, respectively. In keeping with the number of parameters to be optimized, the first circuit ran for about 450 epochs before convergence, while the other two required only about 150 iterations, each. A typical plot of the training and validation losses against the corresponding iteration numbers can be found in Fig. 11. Again, in the absence of gradient-based optimizers, the optimizer blindly bounces around in the loss landscape, taking larger steps initially. Gradually, it converges to a value, but continues to run further iterations before finally terminating the optimization procedure. Having fewer parameters to optimize significantly reduces the jitter in loss, along with the number of epochs and time required to converge.
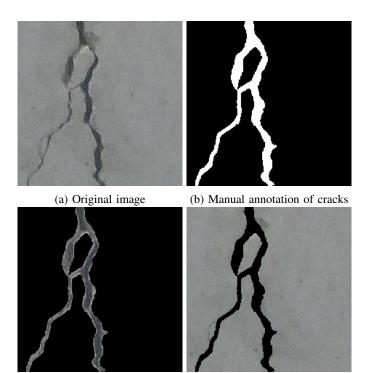

(a) Losses for circuit with the ansatz in Fig. 10a. (b) Typical losses for circuits with ansatzes from Figs. 10b and 10c.

Fig. 11: Plot of training and validation loss per iteration encountered while training the image classification circuit.

The testing accuracy metrics for the VQC models with the three different types of ansatzes for the test data set have been reported in table I. The number of misclassifications although being low, were mostly false negatives, which are detrimental because the subsequent pipeline does not get triggered for

those images. This is an artefact of the artificial imbalance introduced in the data set, the effect of which can be offset through the use of more sophisticated loss functions, such as **Focal Loss**, that are more sensitive to imbalanced data.

E. Segment Classification

The images that are adjudges by the image classification model to contain cracks are passed to the segmentation algorithm in Sec. IV-B. The segmented images exemplified in Fig. 8 have multiple regions that may either be cracks, or may belong to the other aberrations discussed about in Sec III and shown in Fig. 2a. Each such region is passed along to another variational quantum classification model that has been dubbed segment classifier.

(a) Original image   (b) Manual annotation of cracks

(c) Image with only region 1   (d) Image with only region 2

Fig. 12: Figures show the original image considered, the masks created manually for the crack-regions, a blank image containing only the crack-region, and another blank image with only the background.

1) Data set and Preprocessing: To create the data set to train the segment classifier, a corpus of 150 images with cracks was created through random selection and the cracked regions were manually annotated. The different regions of the annotated images were identified through contour detection, and each region (including the background) were extracted out separately, as shown in Fig. 12. The VGG16 features of all such region-images was obtained and stored along with their labels to identify whether the regions correspond to cracks or not. Further preprocessing in the form of dimensionality-reduction, scaling, and splitting for training, validation and testing were carried out as described before, considering the same percentages.

2) Training: Training for the segment classifier was carried out with the same circuits and methods described in Sec. IV-D2. The segment classifiers with the ansatzes in Fig. 10, being trained on a more balanced data set, and on a considerably lesser number of examples, took only about 20, 5 and 5 minutes, each. The plots of loss vs epochs also were similar to those in Fig. 11 for the three ansaetze. The segment classification accuracy metrics for the three trained models on the test data set are captured in table I.

3) Post-processing: Unlike the image classifier, the segment classifier is plagued with the problem of false-positives. Many regions of the images which are not cracks, get misidentified as such. Fig. 13a shows an image where although the crack is segmented properly, many additional regions are also denoted as cracks by the classifier. The difficulty may be remedied by enclosing the regions identified as cracks in oriented bounding boxes and finding the aspect ratio of the latter. A plot of the aspect ratios for crack and non-crack regions can be found in Fig. 13b. The fairly-separable distribution of ratio-values in the plot suggests that a simple classical or quantum classifier can be trained to filter out the non-crack regions getting identified as cracks, though at the cost of increased false-negatives.
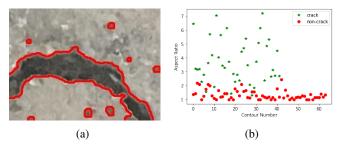

(a)   (b)

Fig. 13: (a) Result of segment classification on an image with false-positives. (b) Plot of aspect ratios of oriented bounding boxes for crack and non-crack regions.

V. CLASSICAL APPROACHES

To benchmark the performance of the quantum approaches discussed to solve the problem of crack-detection, it was compared against two classical solutions. One of the solutions was devised by replacing the quantum blocks (i.e., the quantum classifier and the q-means algorithm) in the pipeline detailed in Sec. IV with their direct classical counterparts. The 4-dimensional PCA-reduced VGG16 features from an image were passed to a simple fully-connected layer to get binary, categorical output, and the binary cross entropy loss was minimised using the $Adam$ optimizer. The other approach involved trials with a home-brewed deep learning model called SCNet [33] built specifically for surface crack detection. The models were trained, validated and tested against the same set of data as their quantum analogues, although for SCNet, the images had to be augmented. The performance metrics of the classical approaches have also been briefed in table I.

VI. OVERALL RESULTS AND ACCURACY

A further 150 images were annotated to test the accuracy of the end-to-end pipeline, which was quantified using Intersection over Union (IoU) of the segmented cracks with the ground-truth annotations. Along with the accuracy value of various classical and quantum classification models (without downscaling of images), the respective IoU scores of the overall pipeline have also been included in table I. The segmentation results from some of the approaches for the image in Fig. 8a are shown in Fig. 14.

(a)   (b)
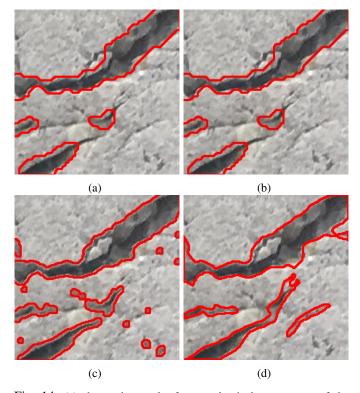
(c)   (d)

Fig. 14: (a) shows the results from a classical counterpart of the same pipeline used for quantum. (b) and (c) show the results obtained from the quantum models with the Strongly Entangling Layers ansatz, without any post-processing. For (a) and (b), the original image was downscaled to speed up the process of image segmentation. (d) portrays the results from the in-house classical crack-segmentation model, SCNet.

	Image Classification	Segmentation Classification	IoU
Ansatz in Fig. 10c	97.6	98.7	72.76
Basic Entangling Layers	94.8	97.4	73.14
Strongly Entangling Layers	99.2	98.7	73.33
Classical pipeline	100	100	75.21
SCNet	N/A	N/A	91.36

TABLE I: Accuracy values for image and segment classification along with IoU scores for various methods.

The low accuracy values reported for the quantum models may be due to the drastic reduction in dimensionality to fit the contemporary quantum simulators and hardware. Of course, further research is also possible in terms of data encoding, etc. It is also noteworthy that the explored quantum models use around $12 - 36$ parameters and get trained in a fraction of the time compared to SCNet which depends on about *30 million* trainable parameters!

VII. FUTURE WORK

The work carried out for this paper opened up many threads for explorations. As mentioned in the previous section, one immediate thing to consider is arrive at other suitable data encoding schemes, say, dense angle encoding which facilitates encoding two features per qubit [3]. Since we have carried out only simulations in AWS, running on different hardware is another task, including comparative performance and benchmarking.

Within the context of classification and segmentation, extending the research to multi-class scenario is another useful and challenging direction. In this paper, we considered a data set with a number of examples; in case, if only few images are available, generating a sized number of images [34] for training is a practical and an important issue to address.

Going tangentially off, since the Deep Learning based technique of [33] performed well for segmentation, ways to "quantize" it can be considered as well.

VIII. QUANTUM SOFTWARE ENGINEERING

The increasing complexity involved in the solutions developed using quantum computing has mandated the formulation and formalisation of a Quantum Software Development Life Cycle [35], which is formed by the interweaving of quantum, machine learning and software engineering life cycles. None of the components can be considered independently, and a holistic view of the system is necessary for proper functioning. Such a life cycle involves all of the machine learning and software engineering life cycle steps, in addition to certain quantum specific parts such as selection of data-encoding strategies, quantum-models etc., among others.

While experimenting with Quantum Machine Learning models (which are invariably a mix of classical and quantum techniques), keeping tabs on all the classical options is a daunting task in itself, which is augmented by the plethora of quantum options, rendering manual tracking of the models, parameters, metrics, performance indicators, version-tracking, and the permutations and combinations thereof, an unsurmountable feat. Further, quantum computing being at a nascent stage, is nowhere close to being productionized. As a result, most practitioners train, validate and test their developed models within the same file or jupyter notebook. The possibility of saving quantum models for future inferencing and use is rarely considered or discussed.

Trained QML models which consist mostly of the optimized weights, along with the helper functions, in the form of objects, can be easily stored in the form of *pickle* or *dill* files if the quantum SDK being used is python-based. Additionally, to address non-functional aspects of the problem like performance and experiment tracking, reproducibility, orchestration, distributed execution etc., Cognitive Model Management (CMM) was used, which is a framework, intended to help the

user manage the lifecycle of the quantum and (or) classical models. CMM provides modular API-based architecture to obtain maximum agility and scalability along with orchestration facility where the user can orchestrate workflows from tasks and execute them in a distributed and clustered environment. Finally, when the models were trained, another self-developed tool called OpSense was leveraged to track the experiments and deploy the model that had the best metrics.

CONCLUSION

Set out to solve the image classification and segmentation, which can be of great use across domains, the paper brought out a generic hybrid quantum-classical pipeline. Each quantum block was worked out in a systematic manner and the entire pipeline was assessed for its functionality and performance by considering Kaggle Surface Crack Detection data. Comparison with other fully classical pipelines and the associated metrics were captured. In order to systematically handle the software-engineering aspects, we also adopted the Cognitive Model Management framework as well as another tool called OpSense, and successfully put into action the entire work flow involved.

ACKNOWLEDGEMENT

REFERENCES

[1] Surface crack detection dataset
[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv:1409.1556.
[3] R. LaRose and B. Coyle, "Robust data encodings for quantum classifiers," Physical Review A, vol. 103, number 3, September 2020.
[4] A. Mari, T. R. Bromley, J. Izaac, M. Schuld and N. Killoran, "Transfer learning in hybrid classical-quantum neural networks," Quantum, vol. 4, 2020, p. 340.
[5] I. Cong, S. Choi and M. D. Lukin, "Quantum convolutional neural networks," Nat. Phys. 15, 1273–1278 (2019).
[6] M. Henderson, S. Shakya, S. Pradhan and T. Cook, "Quanvolutional neural networks: powering image recognition with quantum circuits," arXiv:1904.04767.
[7] V. Havlicek, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow and J. M. Gambetta, "Supervised learning with quantum enhanced feature spaces," Nature, vol. 567, March 2019.
[8] I. Kerenidis, J. Landman, A. Luongo and A. Prakash, "q-means: a quantum algorithm for unsupervised machine learning," arXiv:1812.03584.
[9] L. Tse, P. Mountney, P. Klein and S. Severini, "Graph cut segmentation methods revisited with a quantum algorithm," arXiv:1812.03050.
[10] X. Yao, et al., "Quantum image processing and its application to edge detection: theory and experiment," Phys. Rev. X, vol. 7, issue 3, September 2017.
[11] V. Giovannetti, S. Lloyd and L. Maccone, "Quantum random access memory," Physical Review Letters, vol. 100, number 16, April 2008.
[12] J. Preskill, "Quantum computing in the NISQ era and beyond," Quantum, vol. 2, August 2019.
[13] S. Johri, S. Debnath, A. Mocherla, et al. "Nearest centroid classification on a trapped ion quantum computer," npj Quantum Inf 7, 122 (2021).
[14] H. Buhrman, R. Cleve, J. Watrous and R. de Wolf, "Quantum fingerprinting," Physical Review Letters, vol. 87, number 16, September 2001.
[15] D. Aharonov, V. Jones and Z. Landau, "A polynomial quantum algorithm for approximating the jones polynomial," Algorithmica, vol. 55, number 3, 2009.
[16] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio and P. J. Coles, "Variational quantum linear solver," arXiv:1909.05820.
[17] G. Brassard, P. Hoyer, M. Mosca and A. Tapp, "Quantum amplitude amplification and estimation," Quantum Computation and Information, p. 53-74, 2002.
[18] D. Grinko, J. Gacon, C. Zoufal, et al. "Iterative quantum amplitude estimation," npj Quantum Inf 7, 52 (2021).
[19] Y. Suzuki, S. Uno, R. Raymond, et al. "Amplitude estimation without phase estimation," Quantum Inf Process 19, 75 (2020).
[20] https://morioh.com/p/dcdc9957bc1c
[21] J. Gaura, E. Sojka and M. Krumnikl, "Image segmentation based on k-means clustering and energy-transfer proximity," 7th International Symposium, ISVC2011.
[22] K. Wright, K. M. Beck, S. Debnath, et al. "Benchmarking an 11-qubit quantum computer," Nat Commun 10, 5464 (2019).
[23] J. Allcock, C. Hsieh, I. Kerenidis, and Shengyu Zhang, "Quantum algorithms for feedforward neural networks," arXiv:1812.03089.
[24] I. Griol-Barres, S. Milla, A. Cebrián, Y. Mansoori and J. Millet, "Variational quantum circuits for machine learning: an application for the detection of weak signals," MDPI Applied Sciences, July 2021.
[25] M. Schuld, A. Bocharov, K. Svore and N. Wiebe, "Circuit-centric quantum classifiers," Physical Review A, vol. 101, number 3, March 2020.
[26] E. M. G. Fuster, "Variational quantum classifier," http://diposit.ub.edu/dspace/bitstream/2445/140318/1/GIL FUSTER Elies Miquel.pdf
[27] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," DOI: 10.1098/rsta.2015.0202
[28] S. Lloyd, M. Mohseni and P. Rebentrost, "Quantum principal component analysis," Nat. Phys. 10, 631 (2014).
[29] S. Sim, P. D. Johnson and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," Advanced Quantum Technologies, vol. 2, number 12, October 2019.
[30] Quantum Transfer Learning
[31] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac and N. Killoran, "Evaluating analytic gradients on quantum hardware," Physical Review A, vol. 99, number 3, March 2019.
[32] S. L. Wu, et al., "Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the LHC," Physical Review Research, vol. 3, number 3, September 2021.
[33] H. Sharma, P. Pradhan, P. Balamuralidhar, "SCNet: a generalized attention-based model for crack fault segmentation," arXiv:2112.01426.
[34] M. S. Rudolph, N. B. Toussaint, A. Katabarwa, S. Johri, B. Peropadre, A. Perdomo-Ortiz, "Generation of high-resolution handwritten digits with an ion-trap quantum computer," arXiv:2012.03924.
[35] B. Weder, J. Barzen, F. Leymann and D. Vietz, "Quantum software development lifecycle," arXiv:2106.09323.
[36] B. Eastin and S. T. Flammia, "Q-circuit tutorial," arXiv:quant-ph/0406003v2.

APPENDIX A
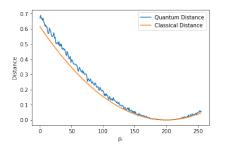## CLASSICAL VS QUANTUM DISTANCE CALCULATION

Fig. 15: Comparison of quantum and classical distance estimations for $c_j = 200$ and p_i ranging from 0 to 255, at unit intervals.

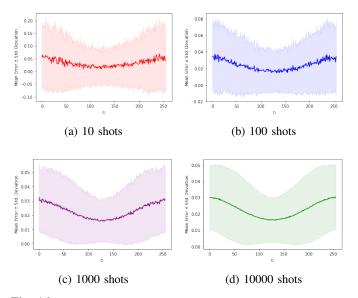(a) 10 shots

(b) 100 shots

(c) 1000 shots

(d) 10000 shots

Fig. 16: The average and standard deviation of errors in Euclidean distance calculation for various number of shots, with various values of c_j on the x-axis.

The suitability of the distance-calculation approach can be verified by the plot in Fig. 15, which compares the distances estimated using simulated quantum circuits against scaled classical distances $(p_i - c_j/255)^2$. The number of shots considered for this validation exercise was 1000 per circuit. A comparative study of the results against the number of shots can be found in the plots of Fig. 16, where both p_i and c_j were varied from 0 to 255 and the average error per value of c_j has been shown as the baseline, surrounded by $\pm$ standard deviation of errors. As expected, the curves become smoother and error reduces as the number of shots is increased, due to better estimation of expectation values.