Individual Coursework Submission Form

Specialist Masters Programme

Surname: Samireddi	**First Name: Sri Saikamal Priyank**
MSc in: Business Analytics	**Student ID number: 210040118**
Module Code: SMM799	
Module Title: Applied Research Project	
ARP tutor name: Dr Rui Zhu	**Submission Date: 2nd September 2022**

Declaration:

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the coursework instructions and any other relevant programme and module documentation. In submitting this work I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

We acknowledge that work submitted late without a granted extension will be subject to penalties, as outlined in the Programme Handbook. Penalties will be applied for a maximum of five days lateness, after which a mark of zero will be awarded.

Marker's Comments (if not being marked on-line):

Deduction for Late Submission:

Final Mark: **%**

JPyramid Quantum Computing for BA

(Climate Change)

By Sri Saikamal Priyank Samireddi

September 2, 2022

This report is submitted as a part of requirements for the award of the

MSc Business Analytics

Bayes Business School

Abstract

Deforestation has many negative effects on forest ecosystems, which in turn have a negative impact on climate change. To combat this, detection and analysis methods should be improved, as they have been in the past, and it is now clear that deep learning neural networks, particularly convolutional neural networks, can efficiently perform image classification tasks. This study implements a well-performing CNN on a dataset of forest classification images and then implements the same through a quantum approach, comparing the metrics of both approaches to assess the advantage of a quantum approach, It has been demonstrated that, although quantum provides computational benefits and shorter training durations, quantum pre-processing the data requires a considerable amount of time and is not practicable for image classification jobs with high quality pictures and large data sets.

Acknowledgement

I would like to acknowledge and thank Dragomir Kovacevic, Founder of Jpyramid, and Dr. Ahmad Abu-Khazneh, my supervisor, and my tutor Dr, Rui Zhu for all their support and knowledge, assisting me throughout the project from day one and providing me with all the experience and technical knowledge they possess, allowing me to successfully conduct my research and complete the Applied Research Project.

Table of Contents

Abstract. 1

Acknowledgement 2

1. **Introduction**. 6

2. **Literature Review** 7

3. **Methodology**. 10

3.1. Data Description 10

3.2. Data Preparation and analysis 10

3.2.1. Data Pre-processing for CNN Model. 11

3.2.2. Data Pre-processing For Transfer Learning. 13

3.3. Handling Imbalanced Datasets. 14

3.3.1. SMOTE for Class Imbalance. 14

3.3.2. Data Augmentation. 16

3.4. Network Implementation. 16

3.4.1 Q – C Approach. 16

3.4.2 C – Q Approach. 17

3.5. Evaluation Metrics 18

4. **Computations and Results**. 21

4.1. Evalution of Q – C Approach. 21

4.2. Evalution of C – Q Approach. 24

5. **Conclusion** 29

Future Work. 30

References. 31

Appendices. 34

1. **Appendix A** 34
2. **Appendix B** 35
3. **Appendix C**. 38

Table of Tables

Table 1: AUC Score Grading. 20

Table 2: Comparison of Accuracies for Q-C approach. 21

Table 3: Comparison of Evalution metrics for Q-C Approach. 22

Table 4: Confusion Matrix of Classical Model (Q-C Approach) 23

Table 5: Confusion Matrix of Hybrid Model (Q-C Approach) 24

Table 6: Accuracies and loss comparison (ResNet18) 25

Table 7: Evaluation Metrics Comparison (ResNet18) 25

Table 8: Accuracies and Loss Comparison (VGG16) 26

Table 9: Evaluation Metrics Comparison (VGG16) 26

Table 10: Accuracies and Loss Comparison (Densenet) 27

Table 11: Evaluation Metrics Comparison (Densenet) 27

Table 12: Accuracies and Loss Comparison (Alexnet) 28

Table 13: Evaluation Metrics Comparison (Alexnet) 28

Table of Figures

Figure 1: Pennylane Transfer Learning. 9
Figure 2: Sample Dataset for Classical Model. 12
Figure 3: Sample visualization of Quantum Pre-processed Dataset. 13
Figure 4: Directory Structure of a Pytorch Dataset. 14
Figure 5: Before SMOTE vs After SMOTE. 15
Figure 6: CNN For Image Classification. 16
Figure 7: Hybrid vs Classical Accuracy Plot (Q-C Approach) 22
Figure 8: Hybrid vs Classical Loss Plot (Q-C Approach) 23

Chapter 1

Introduction

Since 1990, the world has lost approximately a billion acres of forest, largely in continents of Africa and South America. Around 1/6th of the Amazon has been destroyed during the last fifty years. Forests are important to humans for a range of reasons, not the least of which is their ability to slow climate change. (*Deforestation and Its Effect on the Planet*, 2022). Machine learning and artificial intelligence could help predict climate change in various ways, such as detecting the amount of deforested area of patches of forests to regrow or increase the forest area significantly. Scientists have been using machine learning to improve their climate change projections. Image classification is critical in many fields of science and technology, including medical diagnosis and prognosis, face identification, and multiple object detection for self-driving cars, as well as climate change. Convolutional Neural Networks can tackle this problem using classic models. Deep learning models are very accurate and do not require a lot of setups. In a very short period, quantum computing has gained public attention and curiosity, and it is also being widely researched by scientists. Quantum computing leverages quantum mechanics capabilities such as entangling and superposition, which guarantees excellent computational power and has been proven to be consistent in recent research. Despite the expanding number of models, there is still a gap in their testing on real-world data, such as Amazon data. The objective of this study is to compare the performance of quantum machine learning models with traditional machine learning models by applying them to a forest deforestation image classification task, to determine whether a quantum approach can be used to improve accuracy in detecting deforestation in forests.

Chapter 2

Literature Review

The CNNs mainly consists of convolutional layers, pooling layers and fully connected layers which often are known as hidden layers (Watanabe, Sumi and Ise, 2020). The output of a CNN is evaluated by running several operations on the network's input and then passing it through an activation function, often a SoftMax function, for multi-label classification. The complete input and output of artificial neurons may be expressed mathematically as follows where w_i is the weight of the replicated justifying the how well the neurons are connected, x^i is input variable and *b* being the threshold (Watanabe, Sumi and Ise, 2020).

$$u = \sum_i w_i x_i$$

(i)

$$z = f(u + b) = f(\sum_i w_i x_i + b)$$

(ii)

Where

u - total input

z - output

x - input variables

w - weights

b - bias

Sigmoid, hyperbolic, or rectified linear function, function, are known to be non-linear functions, Relu is used as an activation function here.

The convolution layer scans the information of all the channels of an image. To avoid the uncertainty in results, the image is stripped of any channels to have only one channel before it is fed into the network to produce accurate results. A pooling layer scans portions of the output image from a convolutional layer to select the maximum or mean of the features, which is done by adding maxpooling and avgpooling layers. A SoftMax or a sigmoid function is used to generate the output target variable. (Watanabe, Sumi and Ise, 2020).

In this work, we use the functions given by pennylane to turn the model into a hybrid quantum-classical model. Quanvolutional neural networks is a method proposed by pennylane for converting a traditional model to quantum, in which data is pre-processed through a quantum circuit and then fed to the traditional model, which is known to be a hybrid approach using state-of-the-art classical computational algorithms and gaining a quantum edge.

A conventional convolutional neural network with filter sizes of f1, f2, f3,... is to be created. and a kernel size of k x k with a Relu activation function and fully linked layers with n1, n2, n3... hidden units to form a deeper neural network. A simple quanvolutional neural network is formed with a single quanvolutional layer, the input transformation layer, which pre-processes the data, and the model structure is built as a Q-C hybrid model (Henderson et al., 2022).

Further transfer learning is used to implement a CQ – Hybrid model. The pre-trained models are models that have been shown to be extremely accurate and good for a certain use case and may be utilised to our advantage to retrain the trainable layers, freezing most of the layers, which can be effectively employed to tackle new issues (Quantum transfer learning — PennyLane, 2022).

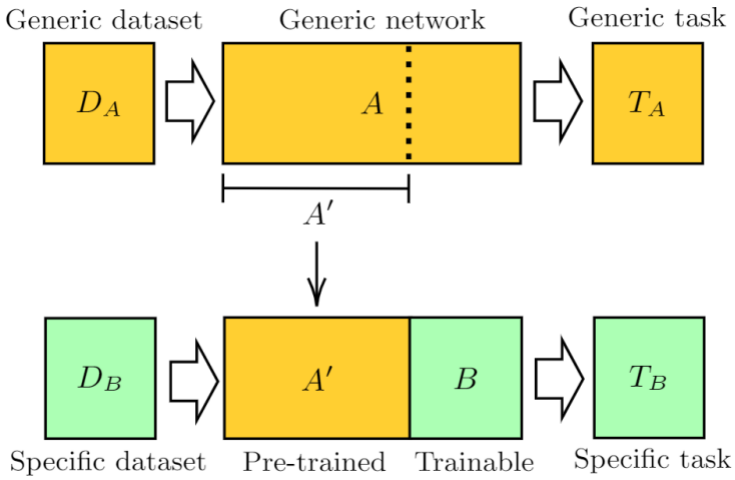

Figure 1: Pennylane Transfer Learning

Source: (Pennylane.ai, 2022)

In the above figure, A can be any pretrained network, such as resnet18, resnet50, FC-densenet, U-net, and others, and the last fully connected layers of the pretrained network are replaced by a dressed quantum circuit B, which has 4 qubits. This circuit is a quantum layer, which is replaced by the layer just above the final output layer of the pretrained network. The hybrid model is trained on the forest dataset (planet-dataset) including photos of Amazon forests while maintaining A′ constant (Quantum transfer learning — PennyLane, 2022).

Chapter 3

Methodology

3.1 Data Description

The planets-dataset was scrapped from the Kaggle using Kaggle API, a full frame of planet was used to extract the images which were captured by 4-band satellites. The photos are in GeoTiff format and have four data bands namely RGB and near-infrared, these images were sourced from the complete geographic area of the amazon basin. These photographs were processed with Planet's visual product processor and saved as jpg files (*Planet: Understanding the Amazon from Space | Kaggle*, 2022).

These images are multi-labelled with different common and unusual characteristics of a landscape. Agriculture, selective_logging, artisinal_mine, conventional_mine, cultivation, road, and slash_burn, hazy, cloudy, clear, primary, water, bare_ground, blooming, habitation, and partly_cloudy are among the tags assigned to the images. These are conveniently provided by images mapped to their classes in a csv file.

3.2 Data Preparation and Analysis

The data is extracted using the Kaggle API using a key and then the data is altered to be used with the study being conducted, as the research tries to perform analysis on a single-label classification problem the multi-labelled dataset must be modified to have only two known classes that are “Detected_Deforestation” and “Forest”.

From the tags mentioned above in the data description, the tags which are considered to signs of deforestation are namely “agriculture”, “selective_logging”, “artisanal_mine”, “conventional_mine”, “cultivation”, “road” and “slash_burn. Hence tags column of the csv file provided is scanned for the string amongst the above tags and if found are replaced by a “Detected_Deforestation” tag and similarly all other tags which resemble a forest are replaced by the tag “Forest” and the images which are tagged as “cloudy”, “partly_cloudy” and “haze” are removed from the dataset.

3.2.1 Data pre-processing for CNN Model

For the 1st approach in order to get the filtered data from the above process a for loop is used to create a balanced dataset from images, which is by using an if else statement to append equal number of images per classes to a list of images which are correctly mapped to another list of classes having them converted into binary data of 1’s and 0’s where “1” corresponds to the tag “Detected_Deforestation” and “0” corresponds to the tag “Forest”. Both the lists are converted to NumPy arrays using np. array and further the datatype of the elements in the array is changed to float.

The images for the hybrid model are pre-processed by a quantum circuit, which turns the photos into multi-channel images with four channels; the process of pre-processing will be detailed more in Chapter 4 Network implementation.

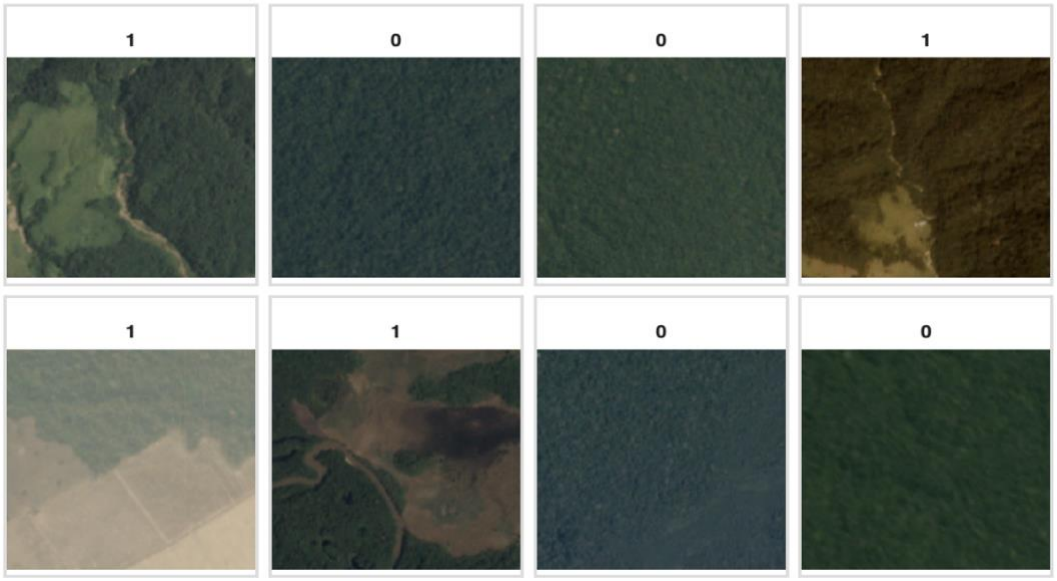

Figure 2: Sample Dataset for Classical Model

The images for the hybrid model are pre-processed by a quantum circuit, which turns the photos into multi-channel images with four channels; the process of pre-processing will be detailed more in Chapter 4 Network implementation.

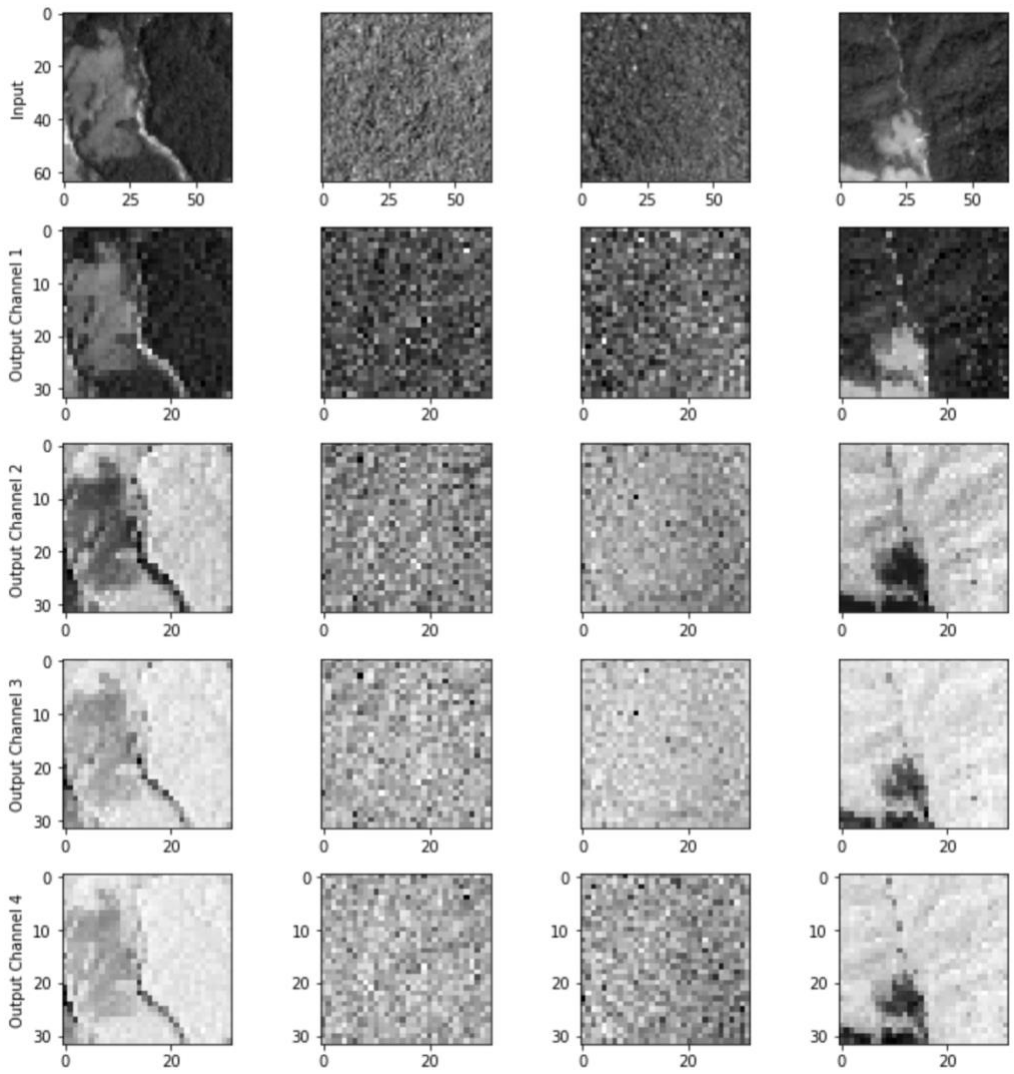

Figure 3: Sample visualization of Quantum Pre-processed Dataset

3.2.2 Data pre-processing for Transfer learning

Transfer learning necessitates the transformation of datasets to pytorch datasets, which can only be accomplished if the data directory follows a specific structure.

To be more specific, the data folders must be divided into train, val, and test folders, which are then further divided into class folders.

Figure 4: Directory Structure of a Pytorch Dataset

This can be accomplished by first creating a folder with images divided into two sub folders of classes, and then using a package split-folders, splitting the data into the required ratio of train, val, and test or just train and val.

3.3 Handling Imbalanced Datasets

3.3.1 SMOTE for Class Imbalance

The Synthetic Minority Oversampling Technique (SMOTE) is a machine learning technique that helps to cope with unbalanced datasets by oversampling the minority class of the dataset with synthesized samples of the existing pictures, ensuring that the model does not overfit. Because we have a class imbalance in our dataset, with class 1 being the minority class, smote is used to overcome this for both datasets before training. Figure 5 shows that the minority class 1 is oversampled from before and after SMOTE representation.

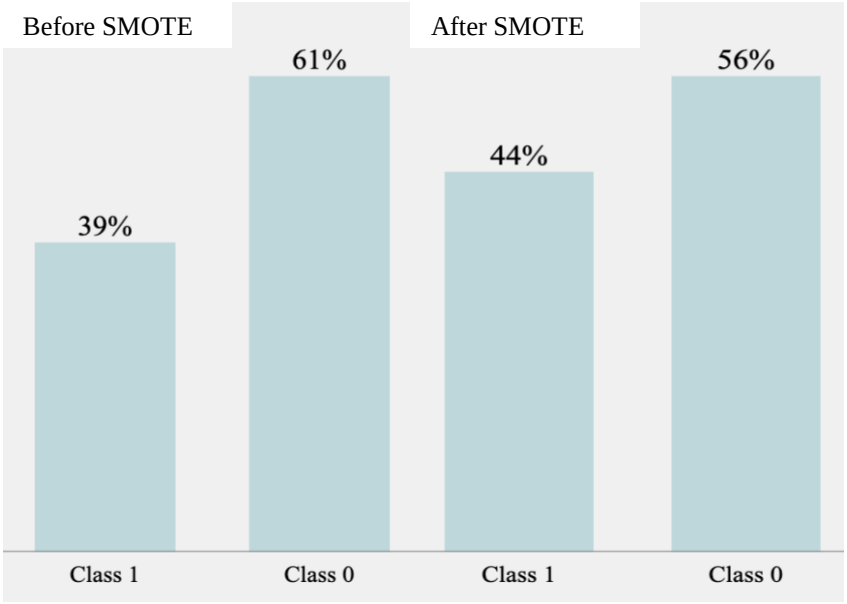

Figure 5: Before SMOTE vs After SMOTE

3.3.2 Data Augmentation

Data Augmentation is a way to deal with unbalanced datasets by adding more samples that are different from the ones already in the dataset. The images may be flipped horizontally or vertically, and they can also be rotated by a factor, with each image at a different rotation angle added to the dataset. We use data augmentation to even out an unbalanced dataset so that it can be used for transfer learning.

3.4 Network implementation

3.4.1 Q-C Approach

The constructed model has a total of 832,305 parameters, 448 of which are untrainable. The Conv2D layer consists of a kernel of size 3x3 with padding, other hidden layers include a maxpooling layer with a pool size and strides of 2x2, a batch normalisation layer, and a dropout layer. The output layer is added at the end after the final fully connected dense layer, which is a layer outputting the probabilities using a sigmoid activation function, a detailed figure of the constructed model can be seen in figure 6. Further, Images from a dataset must be pre-processed through a quantum circuit, which processes small regions of the input image in parallel and produces an output image of the same pixel but assigned to different channels. In this case, the output image from the circuit is a four-channel image.

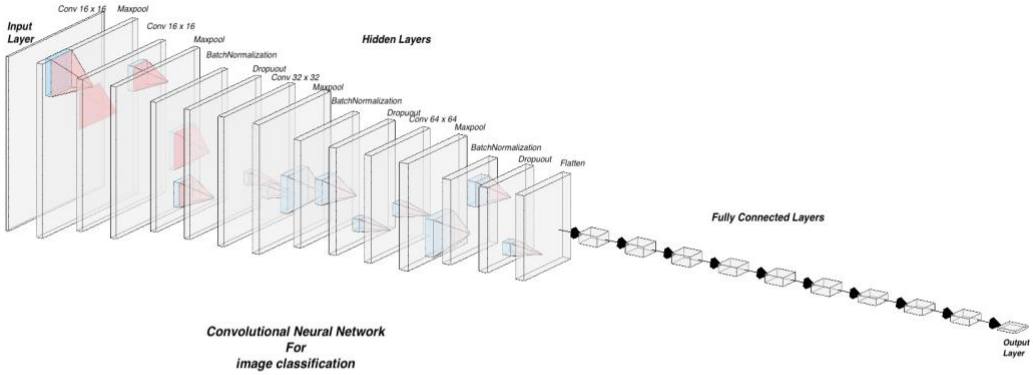

Figure 6: CNN for Image Classification

First, a new empty axis is added to all the images in the dataset to accommodate the additional channels that are going to be added to the image. The pennylane package creates a default quantum device. A function is defined to pre-process the images through the quantum circuit, The python code to the process is given in Appendix B. The images are shrunk to 64x64 for the convenience of pre-processing the basic images using a quantum circuit. Both datasets of normal images and pre-processed images consist of 1000 images because the quantum circuit takes a long time to pre-process images and even though the traditional method can accommodate several thousand images to be trained on the network, it is done to keep it fair to compare both model performances to each other. Both the models are compiled with Adam as optimiser and mean squared error as loss.

3.4.2 C – Q Approach

Transfer learning is used in this strategy, which uses models that have already been trained to do a job. *Resnet18*, *Vgg16*, *Densenet*, and *Alexnet* were chosen in this research since they are among the top pretrained models shown to be efficient for image categorization. As was already said, a quantum layer, which is a dressed quantum net, takes the place of the FC layer or classifier in these models. A standard A qubit device *dev* is initialised with pennylane, and a torch device *cuda* is initialised with torch module. Additionally, a layer with single Hadamard gates with four wires is defined, as the number of qubits in this case is four; and a quantum entangling layer with four qubits is defined. All the previously mentioned functions are put together to make a new function for a quantum network. The weights are reshaped so that qdepth and qubits are the x and y dimensions, respectively. In addition, a class called DressedQuantumNet has been developed, which includes pre-processing, parametric, and post-processing layers.

After obtaining the model from Pytorch, the parameters and weights to be trained are frozen, and only the fc layer, which is now a quantum layer, is trained with the image dataset using a cross entropy loss and an Adam optimizer with a learning rate of 0.0004 and is scheduled to decrease by a factor of 0.1, i.e., gamma, every 10 steps using a lr scheduler. Both the pretrained models and the quantum layer replacement pretrained models are trained on the picture dataset for 30 epochs with a batch size of 4 before being assessed and compared.

3.5 Evaluation Metrics

In this study Accuracy, F1 score, Recall, precision, and Area under the ROC curve (AUC) and Confusion matrix are some measures used to compare the performance of Hybrid models to Traditional Machine Learning models.

Accuracy

For classification issues, accuracy is a popular assessment parameter. It is the proportion of correct forecasts to total predictions.

$$Accuracy = \frac{Total\ Correct\ Forecasts}{Total\ Predictions}$$

Confusion Matrix

A confusion matrix is a frequently used performance indicator for validating a model's overfitting or underfitting when evaluated on test data. The four alternative outcomes for our situation are shown below.

True Positive (TP): Suggests that the Predicted "Deforestation" and is Deforestation in reality.

True Negative (TN): Suggests that the Predicted "Forest" and Forest in reality.

False Positive (FP): Suggests that the Predicted "Deforestation" and Forest in reality.

False Negative (FN): Suggests that the Predicted "Forest" and Deforestation in reality.

Precision and Recall

When there is a class imbalance, accuracy can become an untrustworthy metric for gauging performance. To compare metrics of such models, precision and recall are relevant. Both precision and recall range from 0 to 1, 1 being the perfect.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

F1 score

The F1 score can range between 0 and 1, with 0 being the worst and 1 being perfect. It is the harmonic mean of recall and precision. The harmonic mean tends to be influenced by the smaller numbers, so it ignores the large outliers, resulting in a fair evaluation metric when a dataset is imbalanced.

$$F1\ Score = 2 * \left(\frac{Precision * Recall}{Precision + Recall}\right)$$

AUC

The AUC is the graph having a curve representing true positive versus false positive rates, is a very commonly used evaluation metric while assessing machine learning models.

$$Sensitivity(True\ Positive\ Rate) = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$Specificity(True\ Negative\ Rate) = \frac{True\ Negatives}{True\ Negatives + False\ Positives}$$

$$Auc\ score = \frac{Sensitivity + Specificity}{2}$$

AUC Score	Grading
0.9 < Score < 1	Perfect
0.8 < Score < 0.9	Good
0.7 < Score < 0.8	Poor
Score < 0.7	Worst

Table 1: AUC Score Grading

Chapter 4

Computations and Results

4.1 Evaluation of Q – C Approach

The hybrid model seems to compute faster, resulting in shorter training times, and it also appears to converge faster than the classical model. Nonetheless, pre-processing the dataset for the hybrid model takes a lot of time.

	Classical	Hybrid
Training Accuracy	**94.31%**	**93.21%**
Testing Accuracy	**83%**	**84%**

Table 2: Comparison of Accuracies for Q-C approach

However, there is no substantial difference in either the training or testing accuracies, since the training and testing accuracies of the Hybrid model are 93% and 84%, respectively, which is relatively comparable to the training and testing accuracies of the classical model, which are 94% and 83%. Also, the other metrics shown in table 3 below do not appear to have a significant difference except for precision, which is higher for the hybrid model, which may indicate that the hybrid model generalises well on the test dataset, and the AUC score of both models is the same and lies between 0.8 and 0.9, which is considered good.

	Classical	Hybrid
Precision	**0.7843**	**0.8571**
Recall	**0.8695**	**0.7826**
F1 score	**0.8247**	**0.8181**
AUC Score	**0.8329**	**0.8357**

Table 3: Comparison of Evalution metrics for Q-C Approach

Figures 6 and 7 compare the accuracy and loss plots of both models. The graphs indicate that the hybrid model is significantly more stable than the classical one and has a consistent rise and drop in accuracy and loss when compared to the classical model.

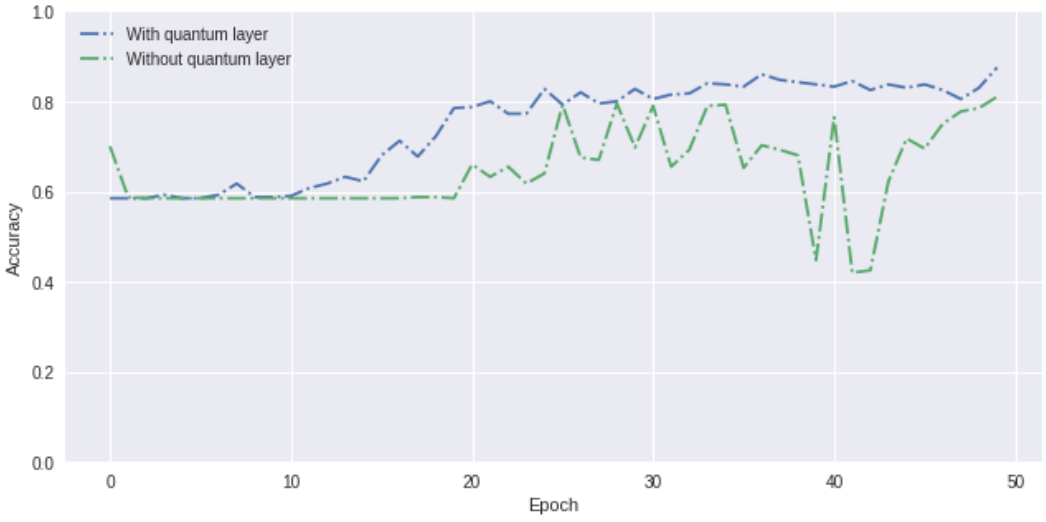

Figure 7: Hybrid vs Classical Accuracy Plot (Q-C Approach)

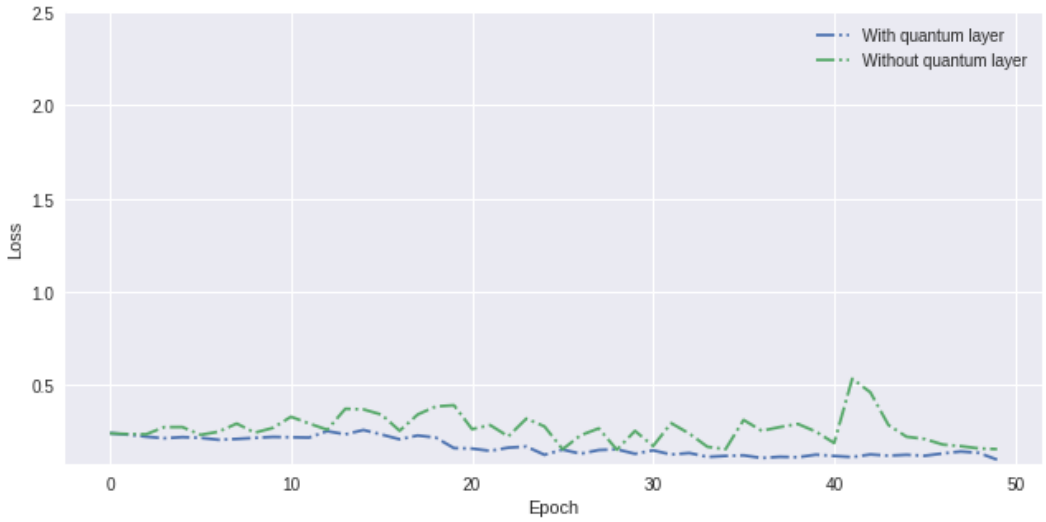

Figure 8: Hybrid vs Classical Loss Plot (Q-C Approach)

Confusion Matrix Q-C Approach (Classical Model)		Actual	
		Forest	Deforestation
Prediction	Forest	**43**	**11**
	Deforestation	**6**	**40**

Table 4: Confusion Matrix of Classical Model (Q-C Approach)

Both the confusion matrices from table 4 and 5 show similar characteristics and proves that the model doesn't overfit or underfit the data.

Confusion Matrix Q-C Approach (Hybrid Model)		Actual	
		Forest	Deforestation
Prediction	Forest	**48**	**6**
	Deforestation	**10**	**36**

Table 5: Confusion Matrix of Hybrid Model (Q-C Approach)

4.2 Evaluation of C – Q Approach

In this section, we will compare the performance of ResNet18, VGG16, Densenet, and Alexnet when the fc layer is replaced by a dressed network and when it is not. When their fully connected layers were replaced with a quantum dressed network, all the models took longer to train, regardless of the pretrained model. Pretrained models required 2 to 5 minutes to train, while hybrid models took 25 to 30 minutes to train each model.

Resnet18

Resnet's training accuracy seems to have reduced from 89% to 81% when the FC layer is replaced by a dressed quantum network. The hybrid model has much less validation accuracy than its training accuracy. The model's training loss has also grown by over 45% which is undesirable. Also, the evaluation metrics seem to be

similar, except for the recall, which is much lower for hybrid resnet18, indicating that the model's true negative rate is lower and it is unable to determine if the image is a forest.

	ResNet18	
Type	**Classical**	**Hybrid**
Training Acc	**0.8965**	**0.8097**
Validation Acc	**0.8903**	**0.8504**
Training Loss	**0.2860**	**0.4469**
Validation Loss	**0.2982**	**0.3815**

Table 6: Accuracies and loss comparison (ResNet18)

	ResNet18	
Type	**Classical**	**Hybrid**
Precision	**0.8370**	**0.8376**
Recall	**0.9085**	**0.7865**
F1 Score	**0.8713**	**0.8113**
AUC Score	**0.8930**	**0.8405**

Table 7: Evaluation Metrics Comparison (ResNet18)

VGG16

When compared to the classical counterpart, the hybrid VGG seems to perform well; training accuracy has risen from 86% to 90%, as has validation accuracy; training and validation losses have been reduced. Overall, the evaluation metrics seem to have improved, notably the AUC score, which is positive given the dataset's class imbalance.

	VGG16	
Type	**Classical**	**Hybrid**
Training Acc	**0.8664**	**0.9015**
Validation Acc	**0.8329**	**0.8703**
Training Loss	**0.3485**	**0.2888**
Validation Loss	**0.4210**	**0.3369**

Table 8: Accuracies and Loss Comparison (VGG16)

	VGG16	
Type	**Classical**	**Hybrid**
Precision	**0.7342**	**0.8333**
Recall	**0.9268**	**0.8536**
F1 Score	**0.8194**	**0.8433**
AUC Score	**0.8473**	**0.8677**

Table 9: Evaluation Metrics Comparison (VGG16)

Densenet

The hybrid densenet and its classical counterpart do not seem to exhibit much change; even though the model's training accuracy has decreased from 85% to 81%, the validation accuracy has largely remained the same; all other evaluation metrics indicate no variation overall.

	Densenet	
Type	**Classical**	**Hybrid**
Training Acc	**0.8531**	**0.8114**
Validation Acc	**0.8603**	**0.8678**
Training Loss	**0.3603**	**0.4468**
Validation Loss	**0.3735**	**0.3831**

Table 10: Accuracies and Loss Comparison (Densenet)

	Densenet	
Type	**Classical**	**Hybrid**
Precision	**0.8253**	**0.8627**
Recall	**0.8353**	**0.8048**
F1 Score	**0.8303**	**0.8328**
AUC Score	**0.8565**	**0.8581**

Table 11: Evaluation Metrics Comparison (Densenet)

Alexnet

The Alexnet also does not appear to show much difference; both accuracies and losses are similar, and the evaluation metrics f1 score and AUC score are similar for both, but the precision and recall seem to decrease and increase respectively for the hybrid model, indicating that the model can now classify forests well while also performing well.

	Alexnet	
Type	**Classical**	**Hybrid**
Training Acc	**0.8982**	**0.8881**
Validation Acc	**0.8778**	**0.8628**
Training Loss	**0.2620**	**0.3371**
Validation Loss	**0.3505**	**0.3745**

Table 12: Accuracies and Loss Comparison (Alexnet)

	Alexnet	
Type	**Classical**	**Hybrid**
Precision	**0.8248**	**0.7766**
Recall	**0.8902**	**0.9329**
F1 Score	**0.8563**	**0.8476**
AUC Score	**0.8797**	**0.8736**

Table 13: Evaluation Metrics Comparison (Alexnet)

Chapter 5

Conclusions

The Quantum - Classical Approach produces similar results as traditional CNN models but appears to be more stable and has shorter computing times, which could be advantageous when training the model on large datasets. However, the pre-processing of the datasets takes a long time and is not feasible for large datasets with high resolution images, this contradicts the whole premise of using quantum computing to anticipate climate change, which requires speedy real-time processing and dealing with massive information.

For the Classical – Quantum Approach With the exception of the VGG16 pretrained network, all pretrained models perform similarly and exhibit a fall in recall when the FC layer is replaced with a dressed quantum network, indicating actual negative rates. Furthermore, models with quantum layers need more training time than conventional models, in this case the classical pretrained models for image classification tasks are proven to be better.

To conclude, hybrid models with quantum layers could be trained on vast amounts of data with either poor or equivalent results, or they can be trained on smaller amounts of quantum pre-processed data with more accurate predictions. There is no ideal equilibrium between the two, and quantum computing must find a perfect balance.

Future work

Image segmentation and semantic segmentation will be implemented in the future for the purposes of analysing the advantages of quantum in this computer vision task. A residual U-Net must be built from the ground up for this. An encoder, a bridge layer, a decoder, and an output layer comprise a residual U-Net. Each layer is made up of many convolutional layers. An encoder is required to encode the input picture to its mask, and then a decoder segments certain regions of the image, which might be a patch of area recognised as a forest and a patch of area detected as deforestation. Because the photos utilised here are high resolution 4 band views of the Atlantic Forest received from SENTINEL and LANDSAT photographs, this is a tremendously complicated process that takes a lot of computer power. Because image segmentation is the most challenging work of all, it requires extensive study, and the residual-U-net is being changed into a hybrid quantum model to demonstrate the benefits of quantum in image segmentation tasks. Some images of the implemented Residual U-net are shown in Appendix A.

References

(2022) Arxiv.org. Available at: https://arxiv.org/pdf/2011.02831.pdf (Accessed: 26 August 2022).

Chauhan, N. (2022) Model Evaluation Metrics in Machine Learning - KDnuggets, KDnuggets. Available at: https://www.kdnuggets.com/2020/05/model-evaluation-metrics-machine-learning.html (Accessed: 26 August 2022).

Deforestation and Its Effect on the Planet (2022) Environment. Available at: https://www.nationalgeographic.com/environment/article/deforestation (Accessed: 26 August 2022).

Hello Tomorrow—I am a Hybrid Quantum Machine Learning (2022) Medium. Available at: https://andisama.medium.com/hello-tomorrow-i-am-a-hybrid-qml-b70751e36142 (Accessed: 26 August 2022).

Image Segmentation: Part 1 (2022) Medium. Available at: https://towardsdatascience.com/image-segmentation-part-1-9f3db1ac1c50 (Accessed: 26 August 2022).

Models and pre-trained weights — Torchvision main documentation (2022) Pytorch.org. Available at: https://pytorch.org/vision/stable/models.html (Accessed: 26 August 2022).

Planet: Understanding the Amazon from Space | Kaggle (2022) Kaggle.com. Available at: https://www.kaggle.com/competitions/planet-understanding-the-amazon-from-space/data (Accessed: 26 August 2022).

The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks (2022) Medium. Available at: https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb (Accessed: 26 August 2022).

The role of machine learning in helping to save the planet (2022) World Economic Forum. Available at: https://www.weforum.org/agenda/2021/08/how-is-machine-learning-helping-us-to-create-more-sophisticated-climate-change-models#:~:text=Scientists%20are%20using%20machine%20learning,model%20future%20climate%20change%20outcomes (Accessed: 26 August 2022).

Using Convolutional Neural Network for Image Classification (2022) Medium. Available at: https://towardsdatascience.com/using-convolutional-neural-network-for-image-classification-5997bfd0ede4 (Accessed: 26 August 2022).

Transfer Learning for Computer Vision Tutorial — PyTorch Tutorials 1.12.1+cu102 documentation (2022) Pytorch.org. Available at: https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html (Accessed: 26 August 2022).

PyTorch ImageFolder for Training CNN Models - DebuggerCafe (2022) DebuggerCafe. Available at: https://debuggercafe.com/pytorch-imagefolder-for-training-cnn-models/ (Accessed: 26 August 2022).

Henderson, M., Shakya, S., Pradhan, S. and Cook, T. (2022) Quanvolutional Neural Networks: Powering Image Recognition with Quantum Circuits, arXiv.org. Available at: https://arxiv.org/abs/1904.04767 (Accessed: 26 August 2022).

Watanabe, S., Sumi, K. and Ise, T. (2020) "Identifying the vegetation type in Google Earth images using a convolutional neural network: a case study for Japanese bamboo forests", BMC Ecology, 20(1). doi: 10.1186/s12898-020-00331-5.

Appendices

Appendix A

Encoder:

```
"""# **Residual-Unet**

***Encoder***
"""

input = tf.keras.Input(shape=(input_size,input_size,num_channels))
x = input
x1 = tf.keras.layers.Conv2D(filters=num_filters[0],
                            kernel_size=3,
                            strides=1,
                            padding='same',
                            kernel_initializer = 'he_normal')(x)
    #x1 = tf.keras.layers.BatchNormalization()(x1)
x1 = tf.keras.layers.Activation('relu')(x1)
x1 = tf.keras.layers.Conv2D(filters=num_filters[0], kernel_size=3,strides=1, padding='same', kernel_initializer = 'he_normal')(x1)

x = tf.keras.layers.Conv2D(filters=num_filters[0],kernel_size=1,strides=1,padding='same',kernel_initializer = 'he_normal')(x)
x = tf.keras.layers.Add()([x, x1])
    #x = tf.keras.layers.BatchNormalization()(x)

encoder_output = [x]
for i in range(1, len(num_filters)):
    layer = 'encoder_layer' + str(i)
            #x1 = tf.keras.layers.BatchNormalization()(x)
    #x1 = tf.keras.layers.Activation('relu')(x1)
    x1 = tf.keras.layers.Activation('relu')(x)
    x1 = tf.keras.layers.Conv2D(filters=num_filters[i], kernel_size=kernel_size, strides=strides[0], padding='same', kernel_initializer = 'he_normal')(x1)
    #x1 = tf.keras.layers.BatchNormalization()(x1)
    x1 = tf.keras.layers.Activation('relu')(x1)
    x1 = tf.keras.layers.Conv2D(filters=num_filters[i], kernel_size=kernel_size,strides=strides[1], padding='same', kernel_initializer = 'he_normal')(x1)

    x = tf.keras.layers.Conv2D(filters=num_filters[i],kernel_size=1,strides=strides[0],padding='same',kernel_initializer = 'he_normal')(x)
    x = tf.keras.layers.Add()([x, x1])

    encoder_output.append(x)

    encoder_out = encoder_output
```

Bridge Layer:

```
"""***Bridge Layer***"""

#x1 = tf.keras.layers.BatchNormalization()(x)
    #x1 = tf.keras.layers.Activation('relu')(x1)
x1 = tf.keras.layers.Activation('relu')(encoder_out[-1])
x1 = tf.keras.layers.Conv2D(filters=num_filters[-1]*2, kernel_size=3, strides=strides[0], padding='same', kernel_initializer = 'he_normal',name="bridge"+'_1')(x1)
    #x1 = tf.keras.layers.BatchNormalization()(x1)
x1 = tf.keras.layers.Activation('relu')(x1)
x1 = tf.keras.layers.Conv2D(filters=num_filters[-1]*2,kernel_size=3,strides=strides[1], padding='same', kernel_initializer = 'he_normal',name="bridge"+'_2')(x1)

x = tf.keras.layers.Conv2D(filters=num_filters[-1]*2,kernel_size=1,strides=strides[0],padding='same',kernel_initializer = 'he_normal',name="bridge"+'_shortcut')(x)
    #x = tf.keras.layers.BatchNormalization()(x)

x1 = tf.keras.layers.Add()([x, x1])
```

Decoder:

```
"""**Decoder**"""

for i in range(1, len(num_filters)+1):
    layer = 'decoder_layer' + str(i)
    target_size = encoder_out[-i].shape[1:3]
    x_resized = upsample(x1, target_size)
    x = tf.keras.layers.Concatenate(axis=-1)([x_resized, encoder_out[-i]])
    x1 = tf.keras.layers.Activation('relu')(x)
    x1 = tf.keras.layers.Conv2D(filters=num_filters[-i], kernel_size=3,  strides=1, padding='same',  kernel_initializer = 'he_normal')(x1)
    #x1 = tf.keras.layers.BatchNormalization()(x1)
    x1 = tf.keras.layers.Activation('relu')(x1)
    x1 = tf.keras.layers.Conv2D(filters=num_filters[-i], kernel_size=3,strides=1, padding='same', kernel_initializer = 'he_normal')(x1)

    x = tf.keras.layers.Conv2D(filters=num_filters[-i], kernel_size=1,strides=1,padding='same',kernel_initializer = 'he_normal')(x)
    #x = tf.keras.layers.BatchNormalization()(x)

    x1 = tf.keras.layers.Add()([x, x1])

x1.shape
```

Appendix B

The Python code for all the results analysed and displayed is provided in the file “Quantum_ARP.ipynb” . This project was completed on Google colab. To download a dataset through the Kaggle API, a key ”Kaggle.json” is required. PREPROCESS Must be set to False if .npy file of quantum pre-processed images is available.

Table of Contents for Python Notebook

Installing ALL Required Packages

Importing All Necessary Packages

Uploading Kaggle.json file key to access kaggle API

Downloading Planets dataset

Data Pre-Processing

Q - C Approach

Train-Test-Split for Classical Model

Building CNN Model From Scratch

Compile & Fit the training data to the model

Calculating all Evaluation metrics

Hybrid Quantum Classical Model

Quantum Preprocessing of the Dataset

Intializing The Quantum Device

Defining a quantum circuit as shown in Pennylane

Training Classical AI Model with Quantum Preprocessed Data

Calculating all Evaluation Metrics

C - Q Approach

Transfer Learning

Classical Transfer Learning

Resnet18

VGG16

densenet

Alexnet

Quantum Transfer Learning

Resnet18

VGG16

Densenet

Alexnet

Appendix C

The Below are the confusion matrices of the transfer learning methods both hybrid and normal.

Confusion Matrix ResNet18 (Classical Model)		Actual	
		Forest	Deforestation
Prediction	Forest	**208**	**29**
	Deforestation	**15**	**149**

Confusion Matrix ResNet18 (Hybrid Model)		Actual	
		Forest	Deforestation
Prediction	Forest	**212**	**25**
	Deforestation	**35**	**129**

Confusion Matrix VGG16 (Classical Model)		Actual	
		Forest	Deforestation
Prediction	Forest	**182**	**55**
	Deforestation	**12**	**152**

Confusion Matrix	Actual

VGG16 (Hybrid Model)		Forest	Deforestation
Prediction	Forest	**209**	**28**
	Deforestation	**24**	**140**

Confusion Matrix Densenet (Classical Model)		Actual	
		Forest	Deforestation
Prediction	Forest	**208**	**29**
	Deforestation	**27**	**137**

Confusion Matrix Densenet (Hybrid Model)		Actual	
		Forest	Deforestation
Prediction	Forest	**216**	**21**
	Deforestation	**32**	**132**

Confusion Matrix Alexnet (Classical Model)		Actual	
		Forest	Deforestation
Prediction	Forest	**206**	**31**
	Deforestation	**18**	**146**

Confusion Matrix	Actual

Alexnet (Hybrid Model)		Forest	Deforestation
Prediction	Forest	**193**	**44**
	Deforestation	**11**	**153**