Using Quantum Convolutional Neural Networks to Classify Pulsars

Gabriel Duckworth Lima Pinto[1, *]

[1] *Department of Physics & Astronomy, University of Manchester, Manchester M13 9PL, United Kingdom*

In this paper, we implement and benchmark four quantum convolutional neural network (QCNN) architectures to classify pulsars from the HTRU2 dataset, which contains 17,898 labeled feature vectors. These QCNNs are designed from simplifying and modifying a benchmark QCNN, where their accuracy, specificity and recall were used to compare their performances. The QCNNs were also compared to classical machine learning models. Modifications, such as implementing parameters in the data embedding and encoding data in parallel with exponential coefficients, resulted in better metric performance relative to the benchmark. The Exponential Frequencies model performed the best with a training accuracy, specificity and recall of 0.966 ± 0.010, 0.979 ± 0.011, and 0.878 ± 0.018, respectively.

Contents

[*] gabriel.duckworthlimapinto@student.manchester.ac.uk

I. Introduction

Pulsars are highly magnetized, rotating neutron stars that periodically emit beams of electromagnetic radiation [1]. These rare neutron stars are of interest to astrophysicists, serving as compelling subjects for investigating gravitational theories [2, 3]. Identifying pulsars presents various challenges: the signals are often faint and exposed to noise, each pulsar shows unique traits (such as pulse periods and emission intensities) and the datasets for pulsar searches are massive, containing many spurious examples. The Square Kilometre Array (SKA) will be the largest radio telescope when launched in 2028 and is expected to return data in the order of exabytes [4, 5]. As the order of data increases, classical classification methods will eventually struggle. A proposition to this issue is through quantum machine learning (QML).
QML has emerged as a promising interdisciplinary field, leveraging the principles of quantum mechanics to enhance machine learning models. A successful approach to solving machine learning problems on current quantum hardware is through variational quantum circuits (VQC). *Quantum neural networks* (QNNs) arise from the appropriate treatment of VQCs, where by encoding data into a parameterised circuit and finding the expectation of an observable yields a real single value, analogous of a prediction in machine learning. In this paper, we explore the capability of different quantum convolutional neural networks (QCNNs), a subset of QNNs, in classifying pulsars from the HTRU2 dataset, a labelled set of data comprised of 17,898 feature vectors, containing 8 features and a binary classification label. One approach is to utilise hybrid quantum-classical algorithms where predictions are made by a quantum computer (in our case quantum simulations) and the gradient descent, parameter tweaking, data storing is done on a classical computer, figure 1.

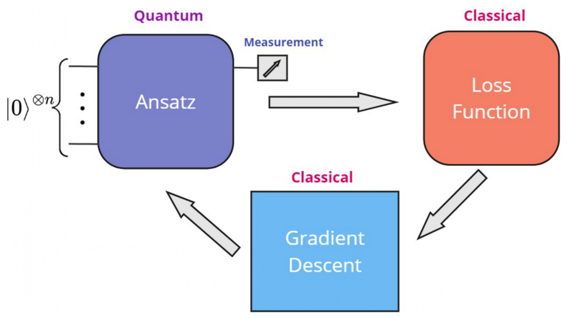


FIG. 1: Hybrid quantum-classical workflow: This diagram illustrates how a VQC and classical computers collaborate in a machine learning algorithm. The quantum component (Ansatz) processes the quantum data, which is then measured and passed to the classical component. The classical part computes the loss function and updates the parameters using gradient descent, feeding them back into the quantum circuit for iterative optimization.

II. Quantum Computing in a Nutshell

This introduction is a very brief overview of what is needed to follow along the project. If the reader wishes to study it more thoroughly, 'Quantum Computation and Quantum Information' by Isaac Chuang and Michael Nielsen [6] is a great place to begin.

A. States & Superposition

In quantum computing, the fundamental unit of computation is the quantum bit, or qubit, which serves a role analogous to that of the bit in classical computing. Unlike classical bits, which are confined to states representing either 0 or 1, qubits are represented by vectors in Hilbert space. Hilbert space is a complex vector space $\mathcal{H}$ equipped with an inner product $\langle \cdot, \cdot \rangle$ that associates each pair of vectors $\mathbf{u}, \mathbf{v} \in \mathcal{H}$ with a complex number. A single qubit is represented as a vector in a two-dimensional complex Hilbert space $\mathbb{C}^2$. The computational basis for this space includes the vectors $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, where the state of a single qubit $|\psi\rangle$ can be a superposition of these basis states,

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle , \tag{1}$$

where α,$\beta \in \mathbb{C}$, satisfying the normalization condition $|\alpha|^2 + |\beta|^2 = 1$. This superposition allows a qubit to represent multiple states simultaneously, a property that is not found in classical systems. The state of a single qubit can be visualized on the Bloch sphere, where any pure state $|\psi\rangle$ maps to a point on the sphere, figure 2. The state can be parametrized by the polar angle θ and the azimuthal angle ϕ, as follows:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\phi} \sin\left(\frac{\theta}{2}\right) |1\rangle. \tag{2}$$

Consider multiple qubits. Each qubit is represented as a vector in $\mathcal{H} = \mathbb{C}^2$ and the state space of a system of n qubits is the tensor product of their individual state spaces. For example, When two qubits ($|\chi\rangle = \alpha |0\rangle + \beta |1\rangle$,$|\phi\rangle = \gamma |0\rangle + \delta |1\rangle$) are considered, their combined state is

$$|\psi\rangle = |\chi\rangle \otimes |\phi\rangle = (\alpha |0\rangle + \beta |1\rangle) \otimes (\gamma |0\rangle + \delta |1\rangle) = \alpha\gamma |00\rangle + \alpha\delta |01\rangle + \beta\gamma |10\rangle + \beta\delta |11\rangle \tag{3}$$

Here, $|00\rangle , |01\rangle , |10\rangle , |11\rangle$ represent the four possible states of the two-qubit system and $\alpha\gamma, \alpha\delta, \beta\gamma, \beta\delta$ are complex coefficients satisfying the normalization condition.

A density matrix ρ is a more general representation of a quantum state, encompassing both pure and mixed states. It is defined as $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$, where p_i are probabilities such that $0 \le p_i \le 1$ and $\sum_i p_i = 1$, and $|\psi_i\rangle$ are the pure states of the system. Properties of the density matrix include: The trace of a density matrix is always equal to one, $\mathrm{Tr}(\rho) = 1$, and it is Hermitian, $\rho = \rho^\dagger$.

B. Entanglement

A quantum state is called *separable* if it can be written as a product of the states of individual systems. For example, consider two qubits where each qubit is in a definite state:

$$|\psi_{\text{sep}}\rangle = |0\rangle \otimes |1\rangle = |01\rangle. \tag{4}$$

This state represents a separable state where the first qubit is in the state $|0\rangle$ and the second in $|1\rangle$. No entanglement is present, as the state of each qubit can be described independently of the other. In contrast, an *entangled* state cannot be factored into a product of individual states. For instance, the Bell state $|\Phi^+\rangle$ is a well-known example of an entangled state:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{5}$$

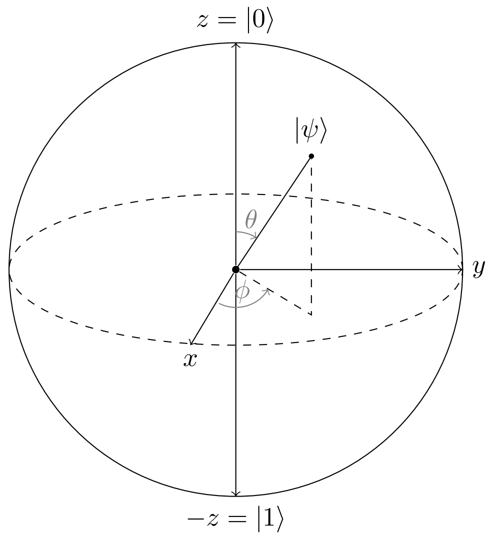


FIG. 2: The Bloch sphere representation of a single qubit state. The north and south poles represent the computational basis states $|0\rangle$ and $|1\rangle$, respectively. Any point on the surface of the sphere represents a pure state of the qubit, which can be expressed as a superposition $|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle$, where θ and ϕ are the polar and azimuthal angles.

This state cannot be expressed as a product of the states of the two qubits involved. The measurement of one qubit immediately determines the state of the other.

An advantage of entanglement is its connection to the exponential growth of the state space. In a classical system, the state space scales linearly with the number of bits. However, in a quantum system the state space grows exponentially with the number of qubits. For a system of n qubits, the state can be represented as:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i \,|i\rangle \tag{6}$$

where each $|i\rangle$ represents a unique state of the n-qubit system. This exponential expansion of the state space is one of the key features that potentially allows quantum computers to process and analyze vast quantities of data far more efficiently than classical computers. It enables quantum algorithms, such as Shor's algorithm for integer factorization and Grover's algorithm for database searching, to achieve significant speedups over the best-known classical algorithms.

C. Quantum Circuits and Gates

How is computation performed on a qubit? By exploiting the unitary evolution of closed quantum systems as described by the Time-Dependent Schrödinger Equation (TDSE),

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = H|\psi(t)\rangle. \tag{7}$$

H is the Hamiltonian defining the system's dynamics. Quantum gates are represented as $2^n \times 2^n$ unitary matrices, $U(t) = e^{iHt/\hbar}$, for an n-qubit system. These gates belong to the special unitary group $SU(2^n)$, that ensure $U^\dagger U = UU^\dagger = I$ and $\det(U) = 1$. Each quantum gate is therefore a transformation within this unitary framework, systematically manipulating the state of the qubit. This unitary matrix U can evolve the state by a suitable choice of the generating Hamiltonian. For example, if we wished to encode classical information, x, onto our qubit, one could define the generating Hamiltonian as $H = g\sigma_x$, where g is some parameter and σ_x is the Pauli-X gate. The result of this being $U(t) = e^{ig\sigma_x t/\hbar}$, where by a suitable change in variables yields $U(x) = e^{i\sigma_x x}$.

A quantum circuit is a model for quantum computation, where the computation is a sequence of gates acting upon the circuits qubits. Figure 3 shows an example of a quantum circuit. The qubits on the left are the initial qubits and the wires demonstrate the propagation of qubit through the circuit from left to right. The rectangles on the wires are single qubit gates, the vertical line connecting qubits 1 and 2 is a two qubit gate and block at the end of each wire is a measurement gate. Quantum gates manipulate qubit states and are represented by $SU(2^n)$ matrices. Important

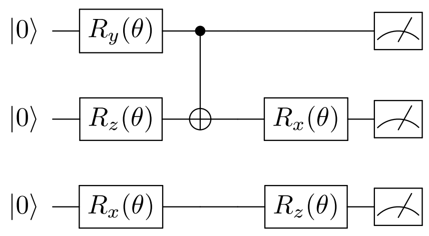


FIG. 3: A three qubit quantum circuit.

gates for this report include:

- Pauli-X, Pauli-Y, and Pauli-Z gates - these gates rotate the state of a qubit around one of the axes of the Bloch sphere by π radians. They are notated as $\sigma_n, n \in \{x, y, z\}$ and are all $\mathcal{SU}(2)$ gates:

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{8}$$

- Rotation gates are a more general Pauli gate, allowing each qubit to be rotated about an axis by an angle θ.

$$R_x(\theta) = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \quad R_z(\theta) = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix} \tag{9}$$

- The controlled-NOT or CNOT gate performs a conditional flip on the target qubit if and only if the control qubit is in the state $|1\rangle$. It can be seen as the vertical wire connecting wires 1 and 2 in figure 3, where the black dot is on the control qubit and the $\oplus$ symbol is on the target qubit. This two qubit gate belongs to the $\mathcal{SU}(4)$ group and takes the matrix form:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{10}$$

In this report, the term *universal* frequently appears in the context of describing circuits. This term refers to the ability of a set of quantum gates or operations to perform any possible unitary transformation on a quantum system's state space.

D. Measurement

To retrieve classical information from a quantum system, measurements must occur. Consider a quantum state represented as $|\psi\rangle = \sum_i c_i|i\rangle$, where c_i are the coefficients corresponding to the state amplitudes in the basis states $|i\rangle$. The process of measurement in quantum mechanics involves projecting this quantum state onto a specific basis state, $|j\rangle$. Upon performing a measurement, the probability p_k that the state $|\psi\rangle$ collapses to the state $|k\rangle$ is calculated Bohr's rule $p_k = |\langle k|\psi\rangle|^2$. $\langle k|\psi\rangle$ represents the amplitude of the component of $|\psi\rangle$ along $|k\rangle$. Substituting the expansion of $|\psi\rangle$, we have $\langle k|\psi\rangle = \langle k| \left(\sum_i c_i|i\rangle\right) = \sum_i c_i\langle k|i\rangle$. Since $\langle k|i\rangle$ is the Kronecker delta δ_{ki}, this simplifies to $\langle k|\psi\rangle = c_k$. Therefore, the probability p_k of the state collapsing to $|k\rangle$ upon measurement is:

$$p_k = |\langle k|\psi\rangle|^2 = |c_k|^2 \tag{11}$$

Observables O are mathematical operators that correspond to physical properties of a quantum system. Each observable is represented by a Hermitian operator, $O = O^\dagger$, ensuring that its eigenvalues are real. The average result or expectation value of an observable is given by

$$\langle O\rangle = \langle\psi|O|\psi\rangle. \tag{12}$$

III. Classical Machine Learning

Machine learning (ML) is a subfield of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computers to perform a task without using explicit instructions [7]. Instead, these models are trained on data, allowing them to improve their performance over time. The core idea behind machine learning is to enable machines to learn from data, identify patterns and make decisions with minimal human intervention [8]. This hot topic is emerging into various disciplines in physics, such as anomaly detection for particles [9], and into our everyday life, from the algorithms deciding your feed on social media to bots trading on the stock market [10].

A. Supervised Machine Learning

Supervised learning involves training models on labeled data to predict output labels for new, unseen inputs. Let us define the input space as X and the output space as Y. $x_i \in X$ represents the input features, which form a feature vector in an n-dimensional space, and $y_i \in Y$ represents the corresponding output or target which the model aims to predict. The primary goal in supervised learning is to find a function that effectively captures the relationship between inputs and outputs. This is mathematically expressed as

$$f_\theta : X \to Y, \tag{13}$$

where the function f_θ is trained to map any new input $\boldsymbol{x}$ from the space X to a predicted output $\hat{y} = f_\theta(\boldsymbol{x})$ in the space Y. A supervised ML model must be trained on data to be able to 'learn' how to be successful. The training dataset is a collection of data points where each data point consists of an input-output pair $(\boldsymbol{x}, y)$. The training dataset $\mathcal{D}_{train}$ can be represented as:

$$\mathcal{D}_{train} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\} \tag{14}$$

where m is the number of training examples, and is produced from randomly sampling the collected data. By randomly selecting data points from the true dataset, the training set is more likely to be

a representative sample of the entire data population and generalize better to the unseen test data. A testing dataset, $\mathcal{D}_{test}$ takes the same form as $\mathcal{D}_{train}$ and is produced from randomly sampling the true dataset, however, it is solely used for analysing how well the trained model performs to unseen data.
A ML model uses the training dataset to attempt to recreate the mapping of equation 13, although, it may take many attempts to get an acceptable approximation. A hypothesis class $\mathcal{H}$ is a set of mapping functions, with each function $h \in \mathcal{H}$ being a possible hypothesis that the learning algorithm can adopt to predict the output. This can be expressed as

$$\mathcal{H} = \{h : X \to Y\}. \tag{15}$$

Parametric models in ML result in the function h depending on a set of parameters denoted as $\boldsymbol{\theta}$. These parameters, or *weights*, are variables that the learning algorithm adjusts to fit the model to the training data. The hypothesis function h can thus be expressed more explicitly in terms of these parameters:

$$h(\mathbf{x}; \boldsymbol{\theta}) : X \to Y. \tag{16}$$

The choice and tuning of the parameters $\boldsymbol{\theta}$ directly influence the form and complexity of the mapping function h. As the parameters $\boldsymbol{\theta}$ are varied, the hypothesis h changes within the hypothesis class $\mathcal{H}$. This variability allows the learning model, which is effectively a specific instantiation of h with a chosen set of $\boldsymbol{\theta}$, to adapt to the nuances and patterns in the data.

Binary classification is a type of supervised learning where the output space Y consists of two classes, typically labeled as 0 and 1. The goal is to find a mapping function $f_\theta : X \to \{0, 1\}$ that accurately predicts the class label for any input $\boldsymbol{x}$.

B. Loss Functions

A *loss function* or *cost function* [11], quantitatively measures how well a specific model performs with respect to its ability to predict the expected outcome. It quantifies the discrepancy between the predicted outputs of the model and the actual outcomes from the data. This discrepancy is needed for training models as it guides the learning algorithm in adjusting the model parameters. The loss function $\mathcal{L}$ is typically expressed as a function of the actual labels y from the dataset and the predictions $\hat{y}$ made by the model, $\mathcal{L}(y, \hat{y})$. A common loss function is the mean square error (MSE),

$$\mathcal{L}_{MSE}(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{17}$$

where n is the number of samples in the dataset. It can be seen that the MSE loss penalizes larger errors more severely than smaller ones, encouraging the model to adjust its parameters to minimize the sum of the squared errors across all predictions.

For binary classification problems, where $y_i \in \{0, 1\}$, a common loss choice is the cross-entropy (CE) loss function,

$$\mathcal{L}_{\mathrm{CE}}(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]. \tag{18}$$

The function imposes a heavy penalty when predictions are both incorrect and made with high confidence: if the actual class is 1 ($y = 1$) and the model predicts a probability near zero ($\hat{y} \approx 0$), the loss approaches infinity, reflecting a severe penalty for confident, incorrect predictions. Conversely, correct predictions with high confidence ($\hat{y}$ close to y) yield minimal loss, enhancing the model's ability to adjust towards more accurate and confident predictions over training iterations. This property encourages not only correctness in classification but also confidence in the probability estimates.

C. Training & Gradient Descent

The process of training a ML model is to identify the optimal set of parameters $\boldsymbol{\theta}^*$ that minimizes a loss function $\mathcal{L}(\boldsymbol{\theta})$,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}). \tag{19}$$

The learning model is a subset of the hypothesis class in that it represents a specific hypothesis chosen based on the training data. It is instantiated by fixing $\boldsymbol{\theta}$ at $\boldsymbol{\theta}^*$, effectively selecting one function from the set $\mathcal{H}$. This model is then used to make predictions or perform other tasks on new, unseen data, ideally generalizing well beyond the examples it was trained on.

The 'learning' in machine learning arises from differentiating the loss function. The loss functions, when plotted with respect to the models trainable parameters, produce a loss landscape. For the model to learn, it must navigate this landscape to find the parameters that minimize the loss. Gradient descent is a optimization method used to minimize the loss function $\mathcal{L}(\boldsymbol{\theta})$ by iteratively updating the parameters $\boldsymbol{\theta}$ in the opposite direction of the gradient of the loss function with respect to $\boldsymbol{\theta}$. The update rule in gradient descent is defined as follows:

$$\boldsymbol{\theta}_{\mathrm{t}+1} = \boldsymbol{\theta}_{\boldsymbol{t}} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}_{\boldsymbol{t}}}, \tag{20}$$

where α is the learning rate, a hyperparameter that controls the size of the steps taken towards the minimum of the loss function. The gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$ points in the direction of the steepest ascent in the loss landscape, and subtracting it moves the parameters towards the direction of steepest descent [12]. To compute the gradient, the total loss $\mathcal{L}$ is considered over all training examples:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i(y_i, h(\boldsymbol{x}_i; \boldsymbol{\theta})), \tag{21}$$

where $\mathcal{L}_i$ is the loss corresponding to the i-th data point, computed as the discrepancy between the actual label y_i and the prediction $h(\boldsymbol{x}_i; \boldsymbol{\theta})$.

1. Adam Optimiser

The Adam (Adaptive Moment Estimation) optimizer extends gradient descent by adapting learning rates for each parameter. It combines the benefits of AdaGrad and RMSProp, using running averages of both gradients and the second moments of gradients to adjust learning rates. Adam's update rules are:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{22}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \tag{23}$$

where g_t is the gradient at timestep t, m_t and v_t are estimates of the first moment (mean) and second moment (uncentered variance), respectively. The decay rates β_1 and β_2 are typically close to 1. To correct initialization bias, Adam uses:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{24}$$

The parameters are updated as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t, \tag{25}$$

where η is the learning rate and ϵ ensures numerical stability, making Adam effective for sparse gradients and varied parameter scales [13].

D. Evaluating Model Performance

1. *Confusion Matrix*

A confusion matrix is a specific table layout that allows visualization of the performance of an algorithm; the matrix compares the actual target values with those predicted by the machine learning model [14].

- **True Positives (TP)** are instances where the model correctly predicts the positive class. It represents the number of times the model correctly identified a positive case.
- **True Negatives (TN)** are instances where the model correctly predicts the negative class. It represents the number of times the model correctly identified a negative case.
- **False Positives (FP)** occur when the model incorrectly predicts the positive class for a negative instance.
- **False Negatives (FN)** occur when the model incorrectly predicts the negative class for a positive instance.

		Predicted	
		Positive	**Negative**
Act.	**Positive**	TP	FN
	Negative	FP	TN

TABLE I: Confusion matrix comparing the actual target values with the predicted values. The matrix entries are used to evaluate the performance of a binary classification model.

The confusion matrix is used to calculate various performance metrics, including accuracy and specificity. Accuracy is defined as the proportion of correct predictions (both true positives and true negatives) out of the total number of predictions. It is calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{26}$$

Specificity, also known as the True Negative Rate (TNR), measures the proportion of actual negatives that are correctly identified. It is calculated as:

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{27}$$

Recall, or the True Positive Rate (TPR), measures the proportion of actual positives that are correctly identified. It is particularly important in scenarios where identifying all positive instances is crucial. Recall is calculated as:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{28}$$

These metrics help in understanding how well the model performs in terms of correctly identifying both positive and negative instances. Accuracy provides an overall performance measure, specificity focuses on the model's ability to identify negative cases, and recall emphasizes the detection of positive cases.

2. ROC & AUC

The Receiver Operating Characteristic (ROC) curve is a graphical plot used to evaluate the performance of a binary classifier as its discrimination threshold is varied [15]. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. The FPR is defined as the ratio of false positive predictions to the total actual negatives. Example of this plot can be seen in section VII. The Area Under the Curve (AUC) is a scalar measure used to quantify the overall performance of a binary classifier. The AUC measures the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1). An AUC of 1.0 indicates a perfect model; an AUC of 0.5 suggests a model with no discriminative ability, equivalent to random guessing.

E. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep neural networks, most commonly applied to image recognition and processing.The architecture of a CNN typically consists of three types of layers: convolutional, pooling and fully connected layers [16]. **Convolutional layers** use a set of learnable filters that apply convolution operations as they scan over the input image. Each filter activates certain features at certain spatial positions in the input. **Pooling layers** reduce the dimensionality of each feature map but retain the most important information. **Fully connected layers** connect every neuron in one layer to every neuron in the next layer, similar to traditional neural networks. The information from previous layers, now transformed into high-level features, is combined in the fully connected layers to make final decisions.

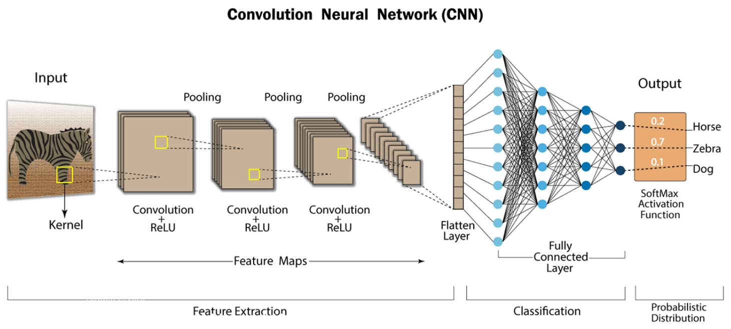


FIG. 4: Shematic on how CNNs work, features including convolution, pooling and fully connected layers - image taken from [17].

IV. Variational Quantum Circuits

Variational Quantum Circuits (VQCs) are quantum circuits that have been parameterized, meaning the circuit, which is made up of a sequence of quantum gates, contains gates with a dependence on a tunable parameter. An application of VQCs are to construct machine learning models – this is approached by using quantum gates to encode classical information $x_i \in X$ and trainable parameters $\boldsymbol{\theta}$ onto the quantum circuit then measuring the expectation of some observable, yielding a single value that anagolous to a prediction in ML. This prediction can be expressed as

$$f_\theta(\boldsymbol{x}) = \langle\phi|U_\theta^\dagger(\boldsymbol{x})MU_\theta(\boldsymbol{x})|\phi\rangle, \tag{29}$$

where $U_\theta(\boldsymbol{x})|\phi\rangle$ is a unitary set of gates dependent on $\boldsymbol{\theta}$ and $\boldsymbol{x}$, acting on the qubit in state $|\phi\rangle$ [18].

VQCs can take any structural form of quantum gates, however, some structures are better at solving specific problems than others. There is no 'shoe that fits all' in relation to the best VQC, meaning educated guesses are made for which sequence(s) of gates would perform the best, commonly known as an *ansatz*. The choice of ansatz is crucial for how the quantum neural network will perform - having the capability to experience ML phenomena such as over-training and under-training.

A. Data Encoding

Data encoding - the embedding of classical data into quantum states - is very important for VQCs; the choice of how data is encoded into a quantum system can significantly influence the performance and applicability. Common encoding methods include:

- Basis encoding: This is where classical data is directly mapped to the basis states of a quantum system. For instance, a binary string 0101 could be represented by the corresponding quantum state $|0101\rangle$.

- Amplitude encoding: This method encodes data into the amplitudes of a quantum state. For example, a normalized classical vector $[x_1, x_2, \ldots, x_N]$ could be represented as a quantum state $\sum_{i=1}^{N} x_i|i\rangle$. Amplitude encoding is highly space-efficient, allowing N features to be encoded using only $\log_2(N)$ qubits.

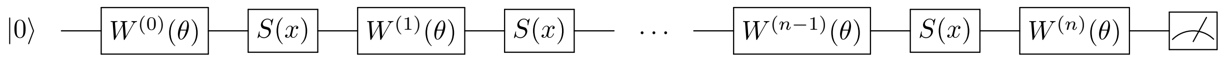
- Angle encoding: This technique encodes data into the angles of rotational quantum gates. For instance, a feature x might control the rotation angle θ in a gate like $R_x(x)$ or $R_y(x)$. This method is particularly versatile and easier to implement with current quantum technologies. VQCs that encode data via angle encoding realise a truncated Fourier series [19],

$$f_\theta(x) = \sum_{\omega\in\Omega} c_\omega(\theta)e^{i\omega x}, \tag{30}$$

where $c_\omega \in \mathbb{C}$, deriving from the trainable gates, and the frequency ω arises from the choice of encoding. $e^{i\omega x}$ are orthogonal basis functions as $\omega \subset \mathbb{Z}^N$.

1. Partial Fourier Series Proof

The following derivation is from following the elegant method outlined in [19]. Consider the basic example of a single qubit circuit, where the W's and S' are a set of trainable and encoding



gates, respectively. The result of this circuit yields,

$$f(x,\theta) = \langle 0|W^{(1)\dagger}(\theta)S^{\dagger}(x)W^{(2)\dagger}(\theta)\cdots M \cdots W^{(2)}(\theta)S(x)W^{(1)}(\theta)|0\rangle. \tag{31}$$

We can simplify the notation by including all of the processing gates into $U(x,\theta)$.

$$f(x,\theta) = \langle 0|U(x,\theta)^{\dagger}(\theta)MU(x,\theta)|0\rangle \tag{32}$$

Lets analyse one half of the expression and ignore the variable dependencies on each gate.

$$[U(x)|0\rangle]_i = \sum_{j_0,\ldots,j_T}^{d} W^{(L+1)}_{i,j_T} S_{j_T,j_{T-1}} \cdots W^{(2)}_{j_3,j_2} S_{j_2,j_1} W^{(1)}_{j_1,j_0} |0\rangle \tag{33}$$

The index j_0 can be fixed to 1 by absorbing $|0\rangle$.

$$W^{(1)}_{j_1,j_0}|0\rangle = \begin{array}{cc} & j_0 \rightarrow \\ j_1 \downarrow & \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \end{array} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} = W^{(1)}_{j_1,1} \tag{34}$$

Substituting into equation 33:

$$[U(x)|0\rangle]_i = \sum_{j_1,\ldots,j_T}^{d} W^{(L+1)}_{i,j_T} S_{j_T,j_{T-1}} \cdots W^{(2)}_{j_3,j_2} S_{j_2,j_1} W^{(1)}_{j_1,1}. \tag{35}$$

The data encoding blocks S can be expressed as $S(x) = \exp(-iHx)$ where the generating Hamiltonian H is a Hermitian operator. As H is hermitian it means that $\exp(-i\mathcal{H}x)$ is unitary. A Hermitian matrix can be decomposed into its eigenvalues and eigenvectors, such that $H = V\mathcal{G}V^{\dagger}$, where V is a unitary matrix whose columns are eigenvectors of H and $\mathcal{G}$ is a diagonal matrix containing the eigenvalues of H. In the following equation we will let the W blocks absorb V,$V^{\dagger}$ to result in

$$[U(x)|0\rangle]_i = \sum_{j_1,\ldots,j_T}^{d} W^{(L+1)}_{i,j_T} e^{-i\mathcal{G}_{j_T,j_{T-1}}x} \cdots W^{(2)}_{j_3,j_2} e^{-i\mathcal{G}_{j_2,j_1}x} W^{(1)}_{j_1,1}. \tag{36}$$

We can further simplify this expression be exploiting that $e^{-i\mathcal{G}x}$ is a diagonal matrix containing the eigenvalues of $\mathcal{G}$. Using,

$$e^{-i\mathcal{G}_{j_2,j_1}} = e^{-i\lambda_{j_1}}\delta_{j_2,j_1}, \tag{37}$$

equation 36 becomes,

$$[U(x)|0\rangle]_i = \sum_{j_1,\ldots,j_L}^{d} e^{-i(\lambda_{j_1}+\cdots+\lambda_{j_L})x} W^{(L+1)}_{i,j_L} \cdots W^{(2)}_{j_2,j_1} W^{(1)}_{j_1,1}. \tag{38}$$

where the total number of indices have been halved, $\frac{j_T}{2} = j_L$. Multiplying 38 by its adjoint, we retrieve

$$f(x,\theta) = \sum_{\mathbf{j},\mathbf{k}} \sum_{i',i} e^{-ix[(\lambda_{j_1}+\cdots+\lambda_{j_L})-(\lambda_{k_1}+\cdots+\lambda_{k_L})]} W^{(L+1)\dagger}_{1,k_1} \cdots W^{(1)\dagger}_{j_1,1} M_{i',i} W^{(L+1)}_{i,j_L} \cdots W^{(1)}_{j_1,1}. \tag{39}$$

Here the measurement matrix added brings a summation of indices i,i' and the multi-index $\mathbf{j} = \{j_1, ..., j_L\}$, $\mathbf{k} = \{k_1, ..., k_L\}$. Finally, by denoting $a_{\mathbf{k},\mathbf{j}} = \sum_{i',i} W^{(L+1)\dagger}_{1,k_1} \cdots W^{(1)\dagger}_{j_1,1} M_{i',i} W^{(L+1)}_{i,j_L} \cdots W^{(1)}_{j_1,1}$ and $\Lambda_{\mathbf{q}} = \lambda_{q_1} + \cdots + \lambda_{q_L}$,

$$f(x, \theta) = \sum_{\mathbf{j},\mathbf{k}} e^{i(\Lambda_{\mathbf{k}} - \Lambda_{\mathbf{j}})x} a_{\mathbf{k},\mathbf{j}}. \tag{40}$$

A level of degeneracy is present in the orthogonal basis functions $e^{i\omega x}(= e^{i(\Lambda_{\mathbf{k}} - \Lambda_{\mathbf{j}})x})$ that have the same Fourier frequency, ω. The frequency spectrum is the set of all frequencies that the QNN has access to $\Omega = \{\Lambda_{\mathbf{k}} - \Lambda_{\mathbf{j}}\}$. Which allows to simplify to

$$f(x, \theta) = \sum_{\omega} c_\omega e^{i\omega x} \tag{41}$$

Fourier accessibility space refers to the range and diversity of the Fourier frequencies that a quantum model can access and utilize in representing data. A greater Fourier accessibility space implies that the quantum model can use a wider variety of frequency components to construct the output function. The *expressivity* of a quantum model, or how well it can approximate complex functions, directly correlates to the number of Fourier terms used.

B. Optimization and Gradients

The loss functions are dependent on the prediction of the quantum circuit and must be optimized to find the corresponding weights that minimize the loss function. The adjoint method provides an efficient approach to computing gradients by leveraging the reversible nature of unitary quantum operations. This method consists of two primary phases in a quantum circuit:

1. **Forward pass:** The quantum circuit is run normally to compute the output state or the expectation value of an observable.

2. **Backward pass:** Instead of differentiating the circuit's operations directly, the adjoint of the entire sequence of quantum gates is applied. Mathematically, the gradient of an expectation value $\langle O \rangle$ with respect to a parameter θ_i can be formulated as:

$$\frac{\partial \langle O \rangle}{\partial \theta_i} = 2\mathrm{Re}\left(\langle 0| U^\dagger(\theta) M U(\theta) \frac{\partial U}{\partial \theta_i} |0\rangle \right) \tag{42}$$

   where $U(\theta)$ is the unitary operation dependent on the parameter θ, M is an observable, and Re denotes the real part of the complex number [20]. This approach computes the gradient by exploiting the inherent reversibility of quantum operations, thereby significantly reducing the computational overhead compared to methods that necessitate forward simulation of the circuit for each parameter.

The adjoint method is particularly beneficial in quantum systems where memory and computational resources are limited, as it requires the storage of only two quantum states—the forward and the adjoint state—and facilitates an efficient gradient calculation by sequentially applying the conjugate transpose of each gate encountered in the forward pass.

C. Shots & Classification

By executing the quantum circuit multiple times, each measurement acts as an independent trial, sampling from the underlying probability distribution defined by the qubit's amplitudes. Consider a single qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $|\alpha|^2$ and $|\beta|^2$ represent the probabilities of measuring the qubit in the $|0\rangle$ and $|1\rangle$ states, respectively. By increasing the number of measurements, known as *shots*, we obtain a frequency distribution of the outcomes that more accurately approximates the true probabilities $|\alpha|^2$ and $|\beta|^2$. Thus, multiple executions help us statistically infer the likelihoods of the qubit's state amplitudes and their corresponding quantum states.

The binomial distribution can be used to model the probability of obtaining a specific number of successes in a fixed number of trials, with each trial having the same probability of success. Consider calculating the probability of measuring the single qubit $|\psi\rangle$ in the state $|1\rangle$. If N denotes the total number of shots and $p = |\beta|^2$ denotes the probability of measuring the qubit in $|1\rangle$, then the probability of measuring a qubit in this state is given by:

$$P(k; N, p) = \binom{N}{k} p^k (1-p)^{N-k}, \tag{43}$$

where k denotes the number of times the qubit is observed in the $|1\rangle$ state out of N shots.
We leverage the probabilistic phenomena of quantum measurements to test our quantum model. The quantum circuit will be executed n_{shots} times for each embedded feature. The quantum simulation is stochastic, meaning each measurement is probabilistic. After each measurement, the expectation value of the observable is stored on a classical computer where it then predicts the classification (pulsar or non-pulsar) and stores the result in an array. Once all shots have been executed, the classical computer makes the final classification decision based on the majority of predictions. This final classification prediction is cross-checked with the features actual label, contributing an entry to the confusion matrix.
But **how does one classify a pulsar from a measurement?** As mentioned in Section II D, the measurement of a qubit returns a single expectation value. Consider a circuit measuring the single qubit $|\psi\rangle$ in the Pauli-Z basis. This measurement yields:

$$\langle\sigma_z\rangle = \langle\psi|\sigma_z|\psi\rangle = |\alpha|^2 - |\beta|^2 = P(|0\rangle) - P(|1\rangle) \tag{44}$$

where, using the sum of all basis state probabilities, $P(|0\rangle) + P(|1\rangle) = 1$, one can rearrange this equation to find:

$$P(|1\rangle) = \frac{1 - \langle\sigma_z\rangle}{2}. \tag{45}$$

Equation 45 shows how the measurement relates to the probability of the qubit's final state being $|1\rangle$. By introducing a threshold at 0.5, the final state's probability can be classified into either $|0\rangle$ or $|1\rangle$ depending on whether the result lies below or above the threshold, respectively.

D. Barren plateaus

Barren plateaus refer to regions in parameter space where the gradient of the loss function vanishes. This phenomenon leads to a significant challenge as the optimization algorithm relies on the gradient of the loss function leading to the loss's non-optimal convergence. VQCs can be expressed as a partial Fourier series, a summation of sines and cosines, and because the loss function is dependent on this, the periodic nature of these functions means that the average gradient over a complete cycle tends towards zero.

$$\mathbb{E}\left[\frac{\partial \mathcal{C}}{\partial \theta_i}\right] \approx 0, \tag{46}$$

The variance of these gradients diminishes exponentially with increasing system size or circuit depth:

$$\mathrm{Var}\left(\frac{\partial \mathcal{C}}{\partial \theta_i}\right) \propto e^{-\alpha n}, \tag{47}$$

where α is a positive constant and n is the number of qubits or the depth of the circuit.

V. HTRU2 Dataset

The HTRU2 dataset is derived from the High Time Resolution Universe Survey (HTRU), a large-scale survey of the sky specifically designed to search for pulsars [21]. Pulsars are highly magnetized, rotating neutron stars that emit beams of electromagnetic radiation. These stars are observed as pulsating sources of radio emission, which are highly periodic. The unique signature of these emissions allows researchers to distinguish pulsars from other celestial sources of radiation, which is a challenging task due to the presence of various noise and interference sources, such as radio frequency interference and other celestial phenomena. The HTRU2 dataset specifically consists of 17,898 total examples, where each example represents a potential pulsar candidate derived from analysis of pulsar surveys. Out of these, 1,639 are real pulsars, and the remaining 16,259 are spurious examples caused by noise and interference. Each candidate in the dataset is described by eight continuous features and one class label that identifies whether the candidate is a real pulsar or not.
The eight continuous features are derived from two measurements of each pulsar candidate: the **integrated pulse profile** and the **dispersion measure signal-to-noise ratio (DM-SNR) curve** [22]. Each of these measurements contributes four features to the dataset, which summarize the key aspects of the signals:

1. **Mean**: Represents the average value of the signal, i.e measure of the signal's intensity.
2. **Standard Deviation**: Captures the variability or dispersion from the mean, which reflects the spread of the pulse or SNR values.
3. **Excess Kurtosis**: Measures the tailedness of the signal distribution relative to a normal distribution, indicating heavy tails that are typical in pulsar profiles and SNR curves.
4. **Skewness**: Quantifies the asymmetry of the signal distribution around its mean, characteristic of the rotational and dispersion properties unique to pulsar signals.

VI. QCNN Models

A quantum convolutional neural network (QCNN) is a quantum adaptation of a CNN - adopting concepts such as convolutional and pooling layers within the quantum circuit. The convolutional layer involves the application of parameterized quantum gates on wires, which manipulate the qubits in ways that depend on the input data. These gates can be seen as analogs to the filters

in classical CNNs, designed to capture patterns and features from quantum data encoded in the qubits. The pooling layers achieve dimensionality reductions by entangling qubits and performing measurements that effectively reduce the quantum state space. QCNNs are a special case of the variational circuits, i.e., QCNNs are a particular class of ansatz. This emphasises by realising the QCNN's ansatz can be deconstructed into a sequence of W and S blocks, IV A 1. All the quantum models in this paper use angle encoding to encode data.
Last semester we investigated the performance of a universal single qubit model on the HTRU2 dataset. The quantum circuit was inspired by [23] and achieved an accuracy of 0.914 ± 0.006 and specificity of 0.908 ± 0.006, with 54 trainable parameters.

A. Benchmark

The benchmark QCNN circuit, taken from [24], is shown in figure 5. This 8 qubit circuit is comprised of an initial data encoding, three convolution and pooling layers. Features, $\mathbf{x}$, are encoding in the circuit by the following encoding block S,

$$S(\mathbf{x}) = \bigotimes_{i=1}^{8} e^{i\sigma_y(x_i)}. \tag{48}$$

The convolutional layers are made of CNOT and RY gates, with unique parameter on each rotation. The pooling layer halves the number of qubits used for computation until only the final qubit remains, where a measurement then occurs. There are a total of 26 parameters to optimize within this model.

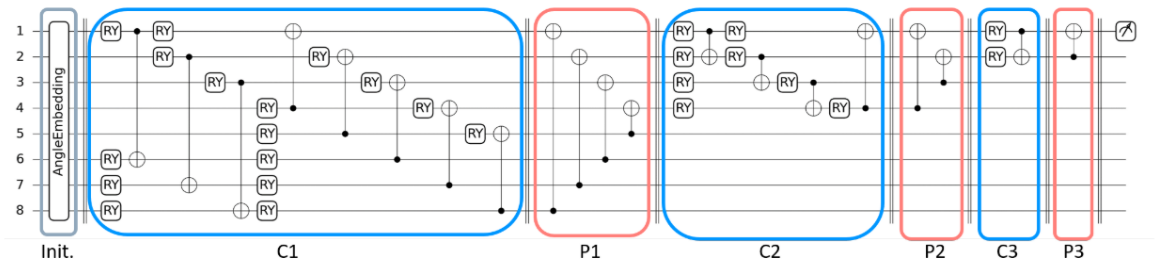


FIG. 5: Benchmark QCNN circuit, taken from [24]. 'C' and 'P' blocks indicate convolutional and pooling layers, respectively. Initial angle embedding RY gates, 48.

B. Simplified Benchmark

The motivation behind finding redundancies within the QCNNs ansatz came from investigating the gradients of the parameters. Every 10 epochs, each of the parameter's gradient was absolute valued and summed its subsequent gradients from the other epochs, to see which parameters contributed more to the model's training. We saw that the summed evolution of seven parameter's gradients were of the order 10^{-16}, i.e, of a negligible contribution. This led us onto investigating the ansatz of the circuit to search for simplifications that result in the negligible parameters becoming redundant. The following steps explain our method used for the simplification.
The first step, shown on the circuit in green, displays two consecutive CNOTs, thus they are to be

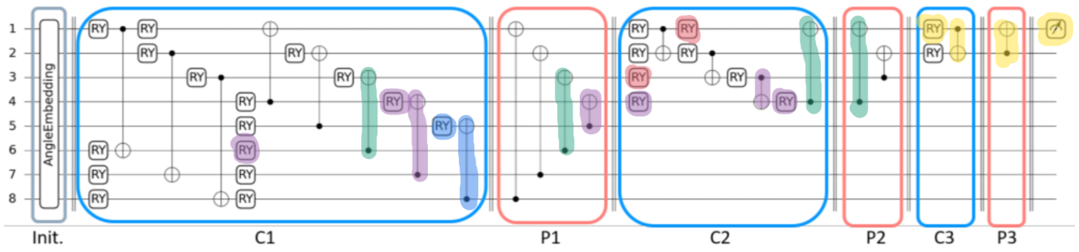


FIG. 6: Colour coded Benchmark QCNN to explain steps for simplification.

made redundant[1]. From the removal of two CNOTs (at the end of C2 and beginning of P2), the last place qubit 4 contributes to the computation is the CNOT$_{4,1}$, meaning all the gates past this point do not contribute to the models reslts. The second step, highlighted in purple, removes all gates on or acting on qubit 4. Additionally, from the removal of the CNOTs$_{6,3}$, the second RY on qubit 6 is not used for processing, allowing it to be removed. Due to the changes, the third step, highlighted in blue, removes two gates on qubit 5. This is because these gates do not contribute to a qubit whose information is propagated to the measurement. Step four, red, removes one of the consecutive RY gates on the respective qubit. Step five, yellow, simplifies the two consecutive flipped CNOTs. Consider isolating final CNOTs for the top qubits and their measurement (bottom left of figure 7).

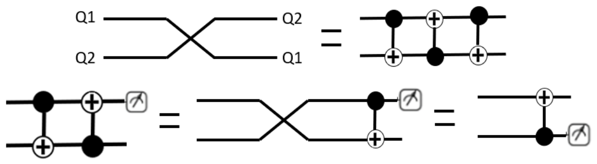


FIG. 7: Top: Two qubits can be swapped with three CNOTs. Bottom: Using the swap on the CNOTs highlighted in yellow, circuit 6.

The measurement analytics of this isolated example can be expressed as,

$$\langle \sigma_z \rangle = Tr[CNOT_{2,1}\rho_{1,2}CNOT_{2,1}(I \otimes \sigma_z)] = Tr[CNOT_{2,1}(I \otimes \sigma_z)CNOT_{2,1}, \rho_{1,2}], \tag{49}$$

where the cyclic property of a trace has utilised. We can write the CNOT gate in terms of tensor products,

$$CNOT_{2,1} = I \otimes |0\rangle\langle 0| + \sigma_x \otimes |1\rangle\langle 1|. \tag{50}$$

Using this CNOT definition, we can simplify the inside of the trace

$$CNOT_{2,1}(I \otimes \sigma_z)(I \otimes |0\rangle\langle 0| + \sigma_x \otimes |1\rangle\langle 1|) = CNOT_{2,1}(I \otimes |0\rangle\langle 0| - \sigma_x \otimes |1\rangle\langle 1|)$$

$$= I \otimes |0\rangle\langle 0| - I \otimes |1\rangle\langle 1| = I \otimes \sigma_z \tag{51}$$

Plugging the result into 50 yields,

$$Tr[\rho_{1,2} I \otimes \sigma_z]. \tag{52}$$

[1] CNOT$^2 |\psi\rangle = \mathbb{I}^4 |\psi\rangle$

By measuring in the σ_z basis, the block simplifies to the removal of both CNOTs highlighted in yellow where the final RY on qubit 1 can too be made redundant. Finally, as the angle embeddings are in the RY basis, we can absorb the first eight trainable RYs into the data encoding,

$$S(\mathbf{x}) = \bigotimes_{i=1}^{8} e^{i\sigma_y (x_i + \theta_i)}. \tag{53}$$

There are a total of 18 parameters to be optimised.

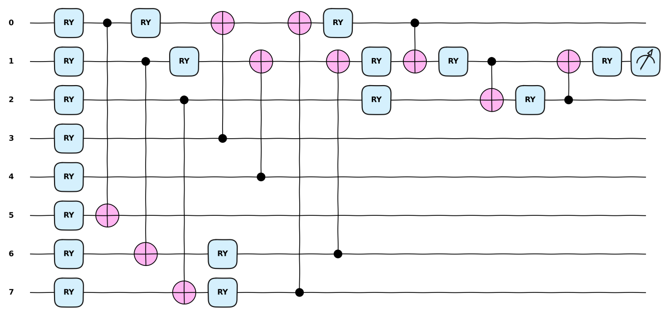


FIG. 8: Simplified benchmark QCNN circuit.

C. Simplified QCNN with Trainable Frequencies

The authors of [25] have shown that by allowing the QCNN to adjust the scaling of input data dynamically, the network can learn to prioritize certain features over others during the training process. A simple yet effective modification is to be made to the Simplified Benchmark QCNN, where the encoding is expressed as

$$S(\mathbf{x}) = \bigotimes_{i=1}^{8} e^{i\sigma_y \cdot (\theta_i x_i + \phi_i)}, \tag{54}$$

where ϕ_i denotes another trainable parameter. This modification introduces another 8 parameters to the Simplified model, totalling to 26.

D. Exponential Frequencies

This QCNN ansatz was ispired by [26], where they showed by encoding a single feature in parallel with exponential coefficients, the degree of the partial Fourier series increases exponentially (in turn exponentially increasing the models expressivity)[2]. The design of the eight qubit quantum model now embeds the same feature twice in parallel, with a rotation angle of x and 3x, where the two equal features pass through a universal 2 qubit trainable block W, figure 9.
The ansatz for this QCNN is shown in figure 10, where after embedding the feature vector the Simplified QCNN ansatz is applied.

[2] See Appendix for more information

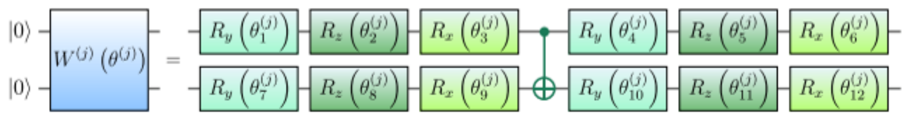


FIG. 9: Two qubit trainable block used in 10 – circuit taken from [26].

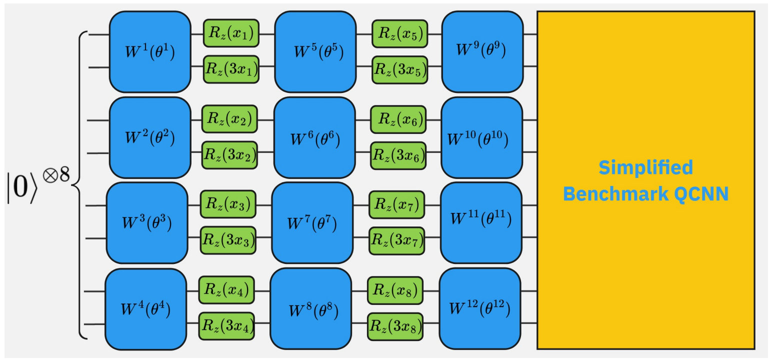


FIG. 10: Exponential Frequencies QCNN. Trainable block W found in figure 9 and the data encoding gates used are RZ. Once the feature vector is encoded, it is followed by the Simplified QCNN ansatz found in figure 8.

VII. Results

The HTRU2 dataset was imported and each feature was normalised between (0,2π) due to the rotation encoding. We randomly sampled from the normalised dataset to produce a train and test set of the dimensions (5, 200) and (5, 400), respectively. The first dimension, 5, arises from the amount of times we want to evaluate the model, in order to calculate uncertainties. Initial weights were randomly sampled between (0,2π). The QCNNs were trained on the train set for 300 epochs, for each of the 5 repetitions. The circuits were simulated using Pennylane's *lightning.qubit* device, where the adjoint method was used to evaluate the gradients, and the Adam optimizer was employed to perform gradient descent on the cross-entropy loss. Once the weights had been optimized, the QCNN was run again with these weights and the test set for inference, where the results were evaluated as seen in IV C. The 5 repetitions meant that the mean of the QCNNs testing metrics were calculated with their uncertainty being the standard deviation. Training and testing plots can found in A, and the test results can be found in table II.

Models:	# Parameters	Train Time[a] (s)	Accuracy	Specificity	Recall
Benchmark	26	454 $\pm$ 9	0.966 $\pm$ 0.008	0.980 $\pm$ 0.006	0.833 $\pm$ 0.047
Simplified Benchmark	18	360 $\pm$ 6	0.967 $\pm$ 0.005	0.982 $\pm$ 0.009	0.810 $\pm$ 0.052
Trainable Frequencies	26	1034 $\pm$ 10	0.966 $\pm$ 0.010	0.984 $\pm$ 0.008	0.792 $\pm$ 0.066
Exponential Frequencies	154	3536 $\pm$ 8	0.969 $\pm$ 0.010	0.979 $\pm$ 0.011	0.878 $\pm$ 0.022
Classical CNN	8481	Negligible	0.980 $\pm$ 0.002	0.993 $\pm$ 0.001	0.854 $\pm$ 0.018
CSVM[b]	9	Negligible	0.979 $\pm$ 0.001	0.995 $\pm$ 0.001	0.821 $\pm$ 0.009

[a] per 300 epochs
[b] Classical Support Vector Machine

TABLE II: Test results for various quantum and classical models. All models were trained on 200 samples and tested on 400. The train time refers to how long it takes to train one set of data.

VIII. Discussion

The **Benchmark QCNN** achieved an accuracy of 0.966 ± 0.008 and a specificity of 0.980 ± 0.006, surpassing the Single Qubit model explored last semester, which had significantly more parameters (54) a but lower performance. Straight away it demonstrates the promise of the QCNN ansatz in effectively utilising multi-qubits and entanglement to better represent the test data for classification. The **Simplified Benchmark QCNN** performed similarly to the Benchmark QCNN, with an accuracy of 0.967 ± 0.005 and a specificity of 0.982 ± 0.009 – this was expected as the two models are equally as expressive. The reduced parameter count (18) likely contributed to slightly better performance by reducing the risk of overfitting. Fewer parameters also meant faster training times. The **Trainable Frequency QCNN** achieved a maximum accuracy of 0.966 ± 0.010 and specificity of 0.984 ± 0.008. Despite having the same number of parameters as the Benchmark QCNN (26), this model revealed the highest specificity upper bound. The **Exponential Frequencies** recorded the highest metrics among the QCNNs, with an accuracy of 0.969 ± 0.0104 and a specificity of 0.979 ± 0.011. However, this model has the greatest number of parameters (154), suggesting a higher likelihood of overfitting the training data. Despite this, it generalised well to the test data. Accounting for the challenges in pulsar detection, a successful model is one with a high specificity as it minimises the misclassification of spurious examples. Therefore, the Exponential Frequencies model is best suited[3] for the classification of pulsars. Both classical models, CNN and SVM, outperformed all the quantum models.
This paper empirically demonstrates that simple modifications of the VQC's ansatz can result in a more expressive model and yield better classification results. Although the quantum models do not perform as well and take longer to train, the theoretical benefits of seizing a quantum model's exponential state space for processing is justifiably for further research in VQCs, hopefully closing the performance gap between quantum and classical.
How would these quantum models perform on today's quantum hardware? Currently, there are quantum computers with 10 times the amount of qubits used in each QCNN [27], however, [28, 29] show that barren plateaus are a significant challenge for models with a large number of qubits and parameters. The Exponential Frequencies will most likely encounter this phenomenon.
Future work: a natural extension would be to investigate parameterising the encoding in the Exponential Frequencies QCNN. Also, it would be interesting to see the effect of strongly penalising the loss function for each incorrect classification.

[1] N. Roberts and D. R. Lorimer, Handbook of pulsar astronomy, Fancy Journal **1**, 1 (2005).
[2] J. M. Lattimer and M. Prakash, The physics of neutron stars, Science **304**, 536 (2004).
[3] B. P. Abbott, R. Abbott, T. D. Abbott, *et al.*, Gw170817: Observation of gravitational waves from a binary neutron star inspiral, Physical Review Letters **119**, 161101 (2017).
[4] BBC News, Ska: Construction to begin on world's biggest telescope, `https://www.bbc.co.uk/news/science-environment-63836496` (2022), accessed: 2024-05-09.
[5] T. An, Science opportunities and challenges associated with ska big data, Science China Physics, Mechanics & Astronomy **62**, 10.1007/s11433-018-9360-x (2019).
[6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010) higher Education from Cambridge University Press.
[7] Wikipedia, Machine learning (2022), accessed on 9 May 2024.
[8] K. P. Murphy, *Machine Learning: A Probabilistic Perspective* (n.d.) [online] Available at:.
[9] V. Belis, P. Odagiu, and T. K. Aarrestad, Machine learning for anomaly detection in particle physics, Reviews in Physics **12**, 100091 (2024).

[3] Best suited out of the QCNNs presented in this paper.

[10] M. Sadiku, T. J. Ashaolu, A. Ajayi-Majebi, and S. Musa, Artificial intelligence in social media, International Journal Of Scientific Advances **2** (2021).

[11] S. Mangini, Variational quantum algorithms for machine learning: theory and applications (2023), arXiv.org.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016) `http://www.deeplearningbook.org`.

[13] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, International Conference on Learning Representations (ICLR) (2015).

[14] Y. Liu, Y. Zhou, S. Wen, and C. Tang, A strategy on selecting performance metrics for classifier evaluation, International Journal of Mobile Computing and Multimedia Communications **6**, 20 (2014).

[15] A. P. Bradley, The use of the area under the roc curve in the evaluation of machine learning algorithms, Pattern Recognition **30**, 1145 (1997).

[16] R. Yamashita, M. Nishio, R. K. G. Do, *et al.*, Convolutional neural networks: an overview and application in radiology, Insights Imaging **9**, 611 (2018).

[17] N. Shahriar, What is convolutional neural network — cnn (deep learning) (2023).

[18] R. Kumar and A. Raina, Generating probability distributions using variational quantum circuits (n.d.), accessed 10 May 2024.

[19] M. Schuld, R. Sweke, and J. J. Meyer, The effect of data encoding on the expressive power of variational quantum machine learning models (2020), arXiv preprint arXiv:2008.08605.

[20] C. Lee, Adjoint differentiation (2021), accessed 10 May 2024.

[21] M. P. I. for Radio Astronomy, High time resolution universe, Monthly Notices of the Royal Astronomical Society **506** (2021).

[22] U. Irvine, Htru2 dataset (2017).

[23] M. Kordzanganeh, A. Utting, and A. Scaife, Quantum machine learning for radio astronomy, arXiv preprint arXiv:2112.02655 (2021).

[24] D. Slabbert, M. Lourens, and F. Petruccione, Pulsar classification: Comparing quantum convolutional neural networks and quantum support vector machines (n.d.), accessed 10 May 2024.

[25] B. Jaderberg, A. A. Gentile, Y. A. Berrada, E. Shishenina, and V. E. Elfving, Let quantum neural networks choose their own frequencies, Physical Review A **109**, 10.1103/physreva.109.042421 (2024).

[26] M. Kordzanganeh, P. Sekatski, L. Fedichkin, and A. Melnikov, An exponentially-growing family of universal quantum circuits (n.d.), accessed on 10 May 2024.

[27] Record-breaking quantum computer has more than 1000 qubits.

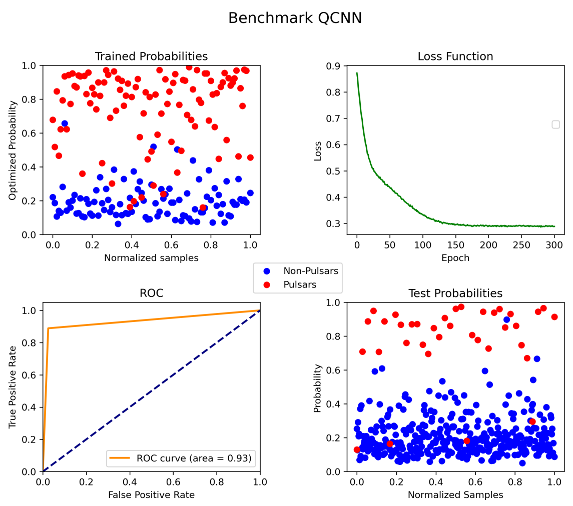
[28] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, Nature Communications **9**, 10.1038/s41467-018-07090-4 (2018).

[29] M. Kordzanganeh, M. Buchberger, B. Kyriacou, M. Povolotskii, W. Fischer, A. Kurkin, W. Somogyi, A. Sagingalieva, M. Pflitsch, and A. A. Melnikov, Benchmarking simulated and physical quantum processing units using quantum and hybrid algorithms, Advanced Quantum Technologies 10.1002/qute.202300043 (2023).

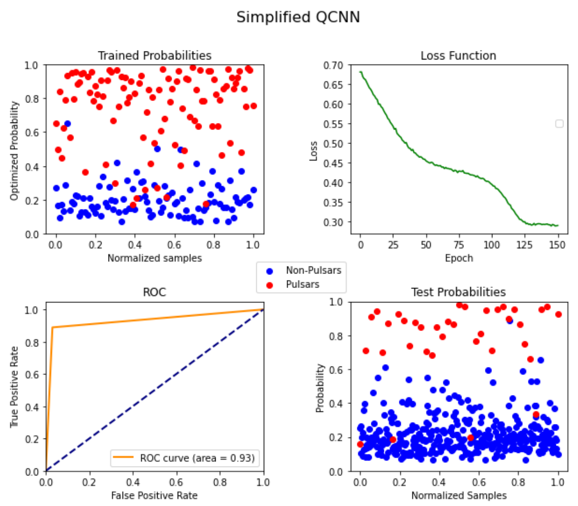
[30] G. Pinto, Semester 2 project code (2024), `https://github.com/gquantym/QCNN`.

A. Appendix - Model Plots

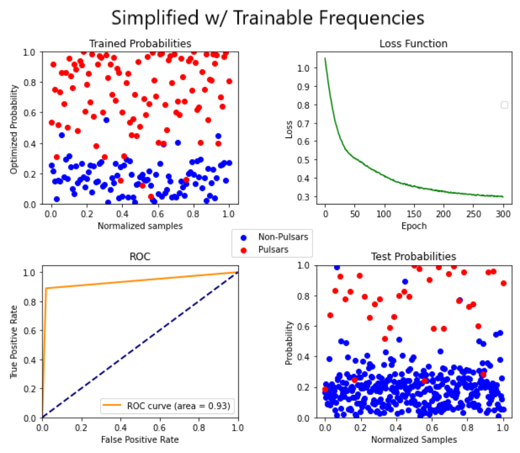
This appendix shows figures of the training classification, loss, test classification and ROC curve, for each quantum model. Each model was ran with the same initial conditions - i.e, epochs, train and test set, initial weights.



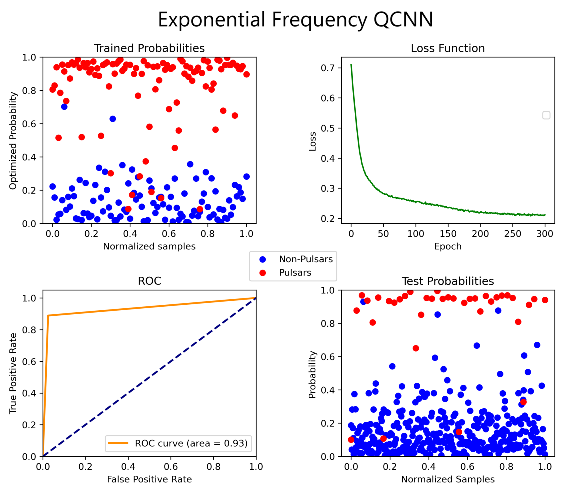
(a) Benchmark QCNN



(b) Simplified Benchmark QCNN



(a) Simplified Benchmark QCNN with Trainable Frequencies



(b) Exponential Frequencies QCNN

B. Appendix - Exponentially Expressive Proof

The generator used for encoding in the computational basis can be generally expressed as shown in Equation B1, and its implementation on a quantum circuit is depicted in Figure 13.

$$\mathcal{G}^{exp} = \sum_{i=1}^{n-1} I^{\otimes(i-1)} \otimes 2^i \sigma_z \otimes I^{\otimes(n-i)} + I^{\otimes(n-1)} \otimes (2^{n-1}+1)\sigma_z \tag{B1}$$

The eigenvalues of $I^{\otimes n}$ are all 1, and the eigenvalues of σ_z are $\{1, -1\}$. Using the property of the tensor product, we have:

$$\sigma_z \otimes I = \{1, -1\} \times \{1, 1\} = \{1, 1, -1, -1\}$$

From this, it can be deduced that the eigenvalues of $\mathcal{G}^{exp}$ are:

$$\{1, -1\} + 2^0\{1, -1\} + 2^1\{1, -1\} + \cdots + 2^{n-3}\{1, -1\} + 2^{n-2}\{1, -1\} + (2^{n-1} + 1)\{1, -1\}$$

This series produces 2^n distinct eigenvalues, forming a set of exponentially growing Fourier bases. The inclusion of terms such as $2^i\sigma_z$ for i ranging from 0 to $n - 2$, and the final term $(2^{n-1} + 1)\sigma_z$, leads to an exponential scaling of the eigenvalues. This construction ensures that the eigenvalues cover a wide range, supporting the exponential growth in the expressiveness of the quantum circuit.

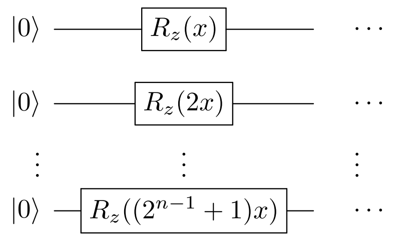


FIG. 13: Quantum circuit with rotations

C. Appendix - Code

Project code can be found here [30].

Acknowledgments

A thank you to the following: my supervisor T. Elliott and friend K. Kugabalan for making this project fun, to the authors of papers I reached out to D. Slabbert & M. Khordzanganeh, to M. Thomson for proof reading & S. Pinto for answering my ML questions.