Exploring Quantum Machine Learning-Enhanced Models for EEG Data Classification

Stephanie Anne Murray

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

University of Washington

2025

Committee:

Erika Parsons

Pierre Mourad

Michael Stiber

Wooyoung Kim

Program Authorized to Offer Degree:

Computer Science & Software Engineering

University of Washington

Abstract

Exploring Quantum Machine Learning-Enhanced Models
for EEG Data Classification

Stephanie Anne Murray

Chair of the Supervisory Committee:
Erika Parsons
Department of Computer Science & SE

Electroencephalography (EEG) records brain activity linked to both executed and imagined movements, but separating true motor signals from background noise in high-dimensional EEG data remains a challenge. Reliable classifiers are therefore vital for accurately tracking patient progress over time. This work is part of a larger initiative, the *Smart NeuroRehab Ecosystem*, which has two primary goals: (1) to propose innovative physical-rehabilitation strategies for neurologic conditions such as stroke using emerging technologies that make therapy more accessible, and (2) to collect and analyze EEG data using machine learning (ML) models that classify movement-related brain signals.

EEG data are complex and often difficult to interpret. In this research, we explore the use of quantum machine learning as an alternative approach for EEG signal classification. Compared to classical ML strategies, quantum methods may offer a fundamentally different way of representing and processing data, potentially improving classification performance or computational efficiency. We implement and analyze a ten-qubit Variational Quantum Classifier (VQC), and compare its performance to a tuned Random Forest baseline using EEG data from a publicly available 64-channel dataset. The task involves classifying each EEG time-window as either a movement or rest condition.

Across 40 preliminary runs, the VQC achieves a macro-F_1 score of approximately 0.75, accuracy of 0.76, and AUROC of 0.83, outperforming the Random Forest (macro-$F_1 \approx$

0.71, AUROC $\approx$ 0.79). In addition to higher macro-F_1 and AUROC scores, the VQC also demonstrated significantly better precision and recall on the movement class, based on paired statistical tests. Most experiments were conducted on a quantum simulator, with a subset tested on a cloud-based quantum processor.

These findings suggest that hybrid quantum-classical models can match or exceed the performance of tuned classical pipelines without increasing computational complexity. Within the scope of the Smart NeuroRehab project, this work demonstrates that quantum approaches may offer a practical path to continuous monitoring of EEG in clinical settings. Future improvements in quantum hardware may expand the range of practical applications in biomedical signal analysis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

EEG (ELECTROENCEPHALOGRAPHY): A non-invasive method for recording the electrical activity of the brain using electrodes placed on the scalp.

EMG (ELECTROMYOGRAPHY): A technique for measuring muscle electrical signals, often used in conjunction with EEG to analyze motor function.

MOTOR IMAGERY (MI): Mental rehearsal of a movement without physically performing it, which can still activate motor pathways in the brain.

BCI (BRAIN-COMPUTER INTERFACE): A system that enables direct communication between the brain signals of a user and an external device or software application.

HILBERT SPACE: A mathematical structure used to describe quantum states. It is a complete vector space equipped with an inner product, allowing for precise definitions of lengths and angles. In quantum computing, the state of an n-qubit system is represented as a unit vector in a 2^n-dimensional Hilbert space.

QUBIT: The fundamental unit of quantum information. Unlike a classical bit, which can be 0 or 1, a qubit can exist in a superposition of both states simultaneously. Qubits are manipulated using quantum gates and measured to produce classical outcomes.

OPTIMIZER: An algorithm used to adjust the trainable parameters of a machine learning model in order to minimize a loss function. In the context of Variational Quantum Circuits (VQCs), optimizers update gate rotation angles to improve classification performance. Common strategies include COBYLA (trust region-based), SPSA (stochastic gradient approximation), and gradient descent.

LOSS FUNCTION: A mathematical expression that quantifies how far a model's predicted output is from the true label. In supervised learning, the loss function guides the optimizer by providing feedback on performance. This thesis uses mean-squared error loss, which penalizes the difference between the predicted class probability and the actual label.

VQC (VARIATIONAL QUANTUM CIRCUIT): A quantum machine learning model consisting of parameterized quantum gates that can be trained to classify data, including EEG features.

QUANTUM SUPERPOSITION: A fundamental principle of quantum mechanics in which a qubit can exist in multiple states simultaneously until measured.

QUANTUM ENTANGLEMENT: A phenomenon in which two or more qubits become linked so that measuring one affects the state of the others, no matter how far apart they are.

ARTIFACT: Unwanted signals within EEG data (e.g., eye blinks or muscle contractions) that must be removed or corrected during pre-processing.

BANDPASS FILTERING: A signal processing technique that keeps frequencies within a certain range (e.g., alpha or beta bands) while discarding others.

FEATURE EXTRACTION: A step in machine learning that transforms raw EEG signals into more informative features (e.g., power in specific frequency bands).

MOVEMENT VS. REST CLASSIFICATION: A machine learning approach that determines when a subject is actively performing (or imagining) a movement versus being at rest.

SMART NEUROREHAB ECOSYSTEM: A collaborative initiative involving multiple UW institutions and Harborview Medical Center, focusing on advanced, cost-effective therapies for patients with neurological problems.

ADASYN (ADAPTIVE SYNTHETIC SAMPLING): An oversampling technique used to balance imbalanced datasets. ADASYN generates synthetic samples for the minority class, focusing more on examples that are harder to classify.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor and committee chair, Dr. Erika Parsons, for her invaluable mentorship, guidance, and encouragement throughout this research. Her support and insight were essential at every stage of this work. I could not have done this without her.

I also sincerely thank the other members of my committee: Dr. Pierre Mourad, Dr. Michael Stiber, and Dr. Wooyoung Kim, for their time and participation. In particular, I would like to thank Dr. Mourad for his encouragement and supportive presence throughout my thesis work.

I would like to acknowledge the faculty in the Department of Computer Science and Software Engineering at the University of Washington Bothell for creating an intellectually supportive and welcoming environment throughout my graduate studies. Special thanks to Jerry Vasquez, the department's Graduate Advisor, whose encouragement and help have been a consistent source of motivation.

I am also grateful to Professor Phillip Duncan, whose support during my earlier academic career played a crucial role in helping me reach this point.

Chapter 1

INTRODUCTION

Technological advances in recent years have made the use of Brain Computer Interfaces (BCI), such as Electroencephalogram (EEG) devices, widely accessible. This opens up new pathways for investigating novel applications, especially when used in combination with Machine Learning (ML) strategies. In addition, the advent of High Performance Computing has allowed the use of Machine Learning (ML) to thrive, facilitating its application in many areas across disciplines, particularly in medical fields such as neuroscience. These developments served as motivation for the Smart NeuroRehab Ecosystem, which has the broader goal of investigating and developing potential solutions for different physical therapy strategies to assist in the rehabilitation of some neurology patients, like those recovering from certain types of stroke, nerve damage, or amputations. Various such conditions present motor impairments similar to stroke, affecting millions of people worldwide with long-term disabilities[38].

BCIs are of valuable use in this context, as they translate neural activity into control signals for devices such as computer cursors, robotic exoskeletons, or functional-electrical-stimulation rigs [16]. Most experimental BCIs rely on electroencephalography (EEG), a non-invasive technique that records scalp voltages with millisecond resolution and aligns well with human motor planning and execution [47]. In this work, we use a publicly available dataset consisting of data collected through a medical-quality EEG device. We use these data to explore machine learning strategies for distinguishing brain signals generated by different upper-extremity motor actions. Specifically, we focus on the application of Quantum Machine Learning (QML). Quantum models, such as Variational Quantum Classifiers (VQCs), offer a fundamentally different approach to data representation and transformation.

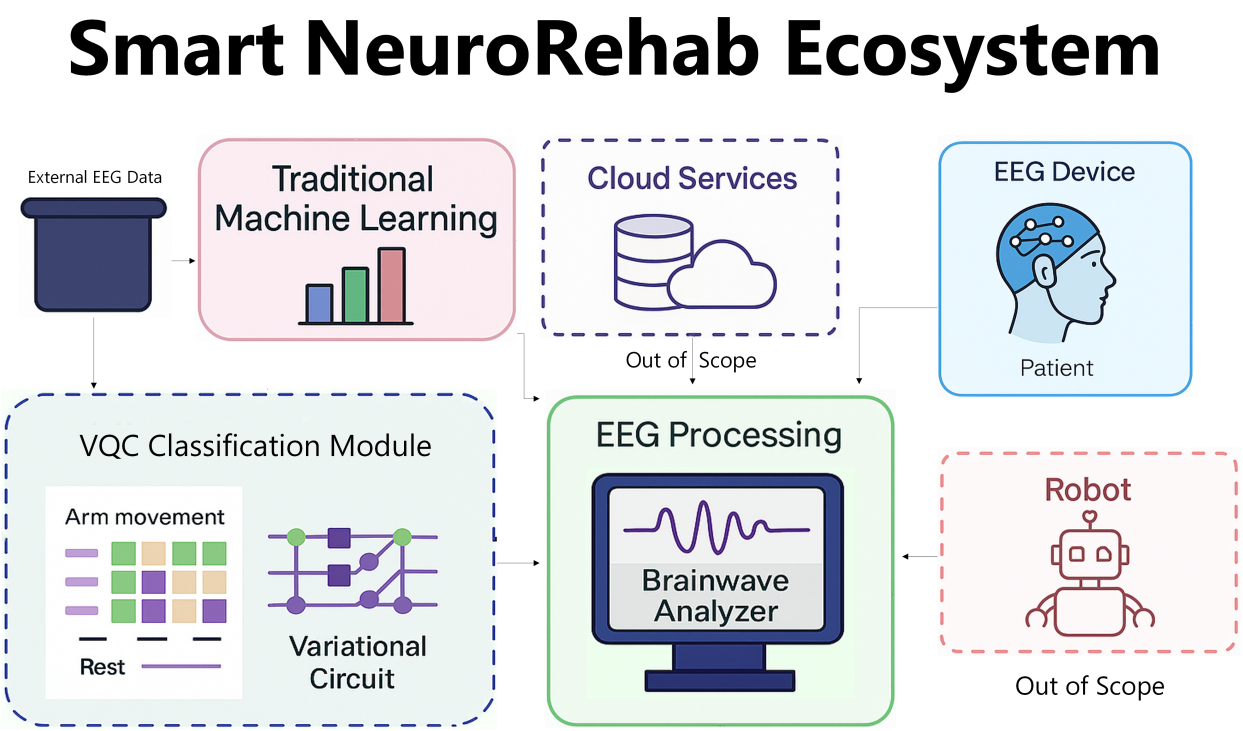


Figure 1.1: An overview of where this work fits in to the Smart NeuroRehab EcoSystem

By encoding input features into high-dimensional Hilbert spaces using entangling circuits, QML can uncover structures that classical models overlook, particularly when training data are limited. This makes QML a promising alternative for EEG signal classification, where the feature space is compact and signal variance is subtle.

Beyond quantum machine learning exploration, this research also highlights a promising route toward rehabilitation. The efficient classification of motor imagery from EEG has direct applications in brain-computer interfaces (BCI), particularly for patients with motor impairments. By identifying movement from EEG signals, these systems may help restore full motor function or communication, advancing the integration of neuroscience insights into assistive technologies.

Figure 1.1 illustrates how this work fits into the broader Smart NeuroRehab Ecosystem.

This thesis focuses on the design, implementation, and evaluation of the **VQC Classification Module**, which receives preprocessed EEG data and outputs movement-versus-rest predictions using a variational quantum circuit. We developed the pre-processing pipeline used in this module, including filtering, feature extraction, and dimensionality reduction. While other branches in the ecosystem explore different modeling strategies, this project investigates quantum machine learning as a compact and potentially more expressive alternative for EEG signal classification.

1.1 Problem Statement

Motor-related EEG presents a difficult modeling target: neural signals are low-amplitude, non-stationary, and easily obscured by ocular, muscular, and environmental noise. A conventional pipeline first derives time- or frequency-domain features, then compresses them, and finally applies a classical classifier such as a Support-Vector Machine or Random Forest. Although these methods can achieve strong performance in controlled offline studies, the underlying variability and noise leave room for approaches that can capture subtler cross-channel patterns. Figure 1.2 illustrates the bursty, noise-dominated structure of the eight-channel window analyzed in this work, where a **window** is defined in terms of time, i.e., an interval of time during which, EEG observations are measured.

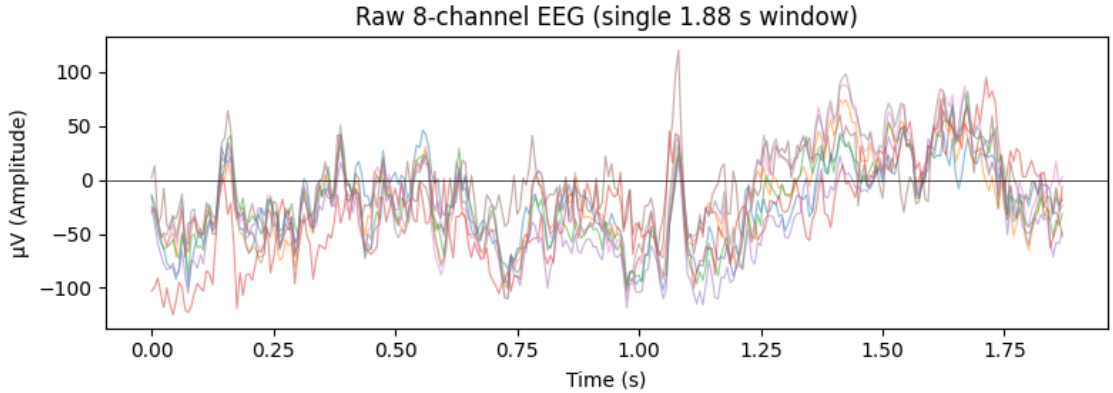


Figure 1.2: Raw 1.88 s EEG window from the eight motor-area channels used in this study.

1.2 Quantum Machine Learning as a Possible Solution

Quantum states operate in Hilbert spaces which are complete inner product spaces whose dimensionality grows exponentially with the number of qubits [53] (see Section 3.3 for definition). Variational Quantum Classifiers (VQCs) embed classical data into quantum states, pass them through a parameterized circuit, and measure an observable such as a Pauli-Z operator or a parity operator whose expectation value predicts the class label [3]. A carefully chosen circuit can map data into a space where classes that overlap in any practical classical representation become separable. Whether this advantage holds on noisy biomedical signals remains an open question, particularly given the current limitations of quantum hardware and optimization challenges in real-world applications [6].

Several studies have begun to explore the use of quantum machine learning for EEG classification, often focusing on motor imagery tasks. Olvera*et al.*[37] proposed hybrid quantum-classical models using variational quantum circuits and quantum genetic algorithms to classify imagined motor tasks from EEG. Meesala*et al.*[33] applied a quantum classifier to epileptic seizure detection using a different dataset and task domain. While these efforts demonstrate growing interest in quantum approaches to EEG, they typically rely on hybrid or classical pre-processing pipelines, lack consistent baseline comparisons, and often omit analysis of run-to-run variability or execution on real quantum hardware. In contrast, this study focuses specifically on executed upper-limb movement, uses a fully standalone VQC, and evaluates both classical and quantum models using the same feature set under controlled and reproducible conditions.

1.3 Research Questions

The work presented in this research seeks to answer the following two main questions:

1. **RQ1:** When using a Variational Quantum Classifier (VQC), is it possible to identify which combination of input features, quantum circuit architecture, and optimizer pro-

duces the best performance in terms of mean macro-F_1 score and run-to-run stability[1]?

2. **RQ2:** In the problem of classifying EEG movement-versus-rest signals, using a compressed EEG window as training data[2], can a ten-qubit VQC achieve equal or better accuracy, recall, or precision than a classical strategy such as Random Forest?

The following hypotheses form the basis of this thesis:

Null hypothesis (H_0)

Quantum classifiers do not significantly outperform the best classical baseline. Any observed improvements in macro-F_1, precision, or recall (on the "Move" class) fall within expected sampling variability.

Alternative hypothesis (H_1)

At least one quantum classifier configuration surpasses the classical baseline by a statistically significant margin in macro-F1, precision, or recall ("Move" class), with an improvement of more than two percentage points.

In this thesis, I explore whether quantum machine learning, specifically a variational quantum classifier, can offer competitive performance on EEG-based movement classification. I compare multiple quantum circuit configurations, including several feature maps and ansatz structures, and benchmark them against a tuned Random Forest baseline using identical preprocessed features. In addition to analyzing accuracy, macro-$F1$, precision, and recall on the minority class, I examine how optimizer choice and architectural depth affect model performance and stability.

---

[1]Higher macro-F_1 and lower variance are preferred. Evaluation metrics include accuracy, AUROC, precision, and recall.

[2]A compressed EEG window refers to a short time segment (e.g., 1.88 seconds) of raw EEG that has been reduced in dimensionality through spatial filtering (CSP) and feature extraction (e.g., PCA on time-domain statistics).

1.4 Method Overview

The study uses the PhysioNet EEG Motor Movement/Imagery data set [16]. Raw signals are segmented into 300-row windows. Seven time-domain statistics are computed per channel: mean power, variance, skewness, kurtosis, zero-crossing rate, peak-to-peak range, and log variance. Principal Component Analysis (PCA) followed by Common Spatial Patterns (CSP) reduces each window to ten components, which are normalized and encoded into a ten-qubit state. The same feature vector trains a Random Forest baseline.

Quantum models are built in IBM's Qiskit toolkit [58] and executed on a state-vector simulator accelerated by an NVIDIA A100 GPU. Experiments sweep feature maps (Z, ZZ, Angle, Pauli), ansatz circuit repetitions (1–5), and three optimizers: Constrained Optimization BY Linear Approximations (COBYLA), Simultaneous Perturbation Stochastic Approximation (SPSA), and Gradient Descent. Each configuration is repeated 40 times with a fixed seed for parameter initialization and new train–test splits. All runs log hyper-parameters, accuracy, macro-F_1, and training time to CSV for analysis. A small subset of final models was also deployed to a real superconducting quantum processor via IBM's cloud runtime to test hardware feasibility.

Figure 1.3: High-level overview of the variational quantum classifier pipeline. Each stage will be described in detail in Sections 3 and 4.3.

Figure 1.3 shows a conceptual overview of the VQC pipeline used in this study. Classical EEG features are first encoded into quantum states using angle-based rotations. These encoded states are passed through a parameterized quantum circuit (ansatz), measured using a Pauli-Z observable, and converted into class predictions. Each stage of the pipeline, including data encoding, circuit construction, and training, is described in more detail in later chapters.

1.5 Stakeholders and Expected Benefits

This work is part of the NeuroRehab project across UW Bothell, UW Seattle, and Harborview Medical Center. The immediate audience is our research team, which needs a reproducible, scalable classifier pipeline for active-movement and motor-imagery experiments. If a quantum approach matches or exceeds classical models while requiring less data or calibration, it can feed directly into next-stages of research and development.

1.6 Thesis Outline

This thesis is organized as follows. Chapter 2 reviews previous work on EEG-based classification and quantum machine learning, highlighting gaps in the literature that this project addresses. Chapter 3 introduces the theoretical background necessary to understand both classical EEG pipelines and the principles of variational quantum circuits. Chapter 4 describes the experimental pipeline in detail, including data pre-processing, feature extraction, and the training procedure for classical and quantum models. Chapter 5 presents the results, comparing performance metrics across models and configurations. Chapter 6 concludes with a discussion of the findings, limitations of the study, and possible directions for future research.

Chapter 2

RELATED WORK

This section surveys classical machine learning for EEG-based movement detection and early work on quantum machine learning in healthcare.

2.1 Literature Review

Electroencephalography (EEG) is a widely used technique for capturing neural activity, but the signals it produces are notoriously noisy, low-amplitude, and sensitive to artifacts. Classifying EEG patterns, especially for tasks such as movement execution or motor imagery, requires effective signal processing and models capable of extracting subtle, time-locked features. Over the past two decades, a wide range of machine learning approaches have been applied to EEG data, from handcrafted feature pipelines that feed into classical classifiers to deep learning models that attempt to learn discriminative patterns end-to-end.

Recent interest has turned toward quantum machine learning (QML) as an alternative or complementary paradigm. Quantum models offer high-dimensional Hilbert space embeddings and entanglement-based feature interactions, potentially enabling more compact or expressive classifiers. However, most QML research in EEG focuses on seizure detection or motor imagery, often using deep CNN feature extractors or hybrid classical–quantum models. This chapter reviews classical and quantum approaches to EEG classification, with special attention to studies that combine engineered features, spatial filtering, and interpretable classifiers, whether classical (e.g. Random Forest) or quantum (e.g. VQC). The aim is to position the current study within this changing field and to emphasize the gaps in existing research that it tackles.

2.2 Existing Approaches

2.2.1 EEG Motor Classification: Classical Pipelines

Traditional pipelines for EEG-based brain–computer interfaces (BCIs) typically involve signal preprocessing, manual feature extraction, and classical classification methods. Preprocessing often includes band-pass filtering in the alpha and beta ranges, artifact removal via independent component analysis (ICA) or thresholding, and segmentation into short time windows. Time-domain features such as band power, variance, and skewness are commonly extracted and passed to classifiers like Linear Discriminant Analysis (LDA), Support Vector Machines (SVMs), Random Forests, or Naïve Bayes.

Lotte et al. [31] conducted a comprehensive review of classification pipelines in BCI competitions, finding that pre-processing choices, such as bandpass filtering, artifact rejection and spatial filtering (e.g., CSP), had a greater effect on classification performance than the choice of classifier itself. Their findings support the decision in this study to focus on a strong preprocessing pipeline, pairing CSP with PCA before benchmarking both classical and quantum classifiers.

Basri and Arif [2] investigated the classification of epileptic seizure types using EEG data from the Temple University Hospital (TUH) EEG dataset. Their study employed a Random Forest classifier trained on frequency-domain features extracted via Fast Fourier Transform (FFT) from 10-second EEG windows. To address severe class imbalance among seizure categories, they applied both SMOTE and ADASYN oversampling techniques. The Random Forest achieved high performance, with 96.2% accuracy and a weighted F_1-score of 0.961 when trained on ADASYN-augmented data. While these results demonstrate the effectiveness of Random Forests for seizure classification using spectral features, the task and methodology differ substantially from the present work. Our study targets executed movement detection, not epilepsy, and uses a compact 10-dimensional time-domain feature vector extracted via PCA and CSP. Although their performance is stronger in absolute terms, their model operates on longer time windows and a task with more distinct signal patterns.

In contrast, our Random Forest is deliberately compact and serves as a classical benchmark for detecting subtler motor execution from short EEG windows, enabling direct comparison with our quantum model.

Craik et al. [10] evaluated deep learning architectures, including convolutional neural networks (CNN) and recurrent neural networks (RNNs) in multiple EEG classification tasks, such as seizure detection and motor imagery. Their results showed that classical classifiers using extracted features (e.g., time-domain statistics, bandpower) often outperformed deep models on smaller or subject-specific datasets. In particular, extracted features paired with SVMs or Random Forests yielded higher accuracy and lower variance than CNNs in low-data regimes. These findings support the decision in this study to use statistical time-domain features rather than deep feature extractors, especially given the limited size and subject variability of the available EEG data.

Luo et al. [32] proposed an Ensemble Support Vector Learning (ESVL) approach to improve motor imagery EEG classification, particularly in scenarios with limited EEG channels. Their method combines the outputs of multiple SVM classifiers and utilizes class discrepancy-guided sub-band filter-based Common Spatial Pattern (CSP) for feature extraction. The ESVL approach achieved an average Cohen's kappa value of 0.71 on the BCI Competition IV dataset, indicating substantial agreement between predicted and true labels. In comparison, the Random Forest baseline used in this study applied to executed movement EEG with PCA+CSP features and achieved a macro-F_1 of 0.701 and AUROC of 0.755, reflecting similarly strong classification performance across balanced metrics. While Luo et al.'s work highlights the value of CSP-based filtering in classical EEG pipelines, their model is specific to motor imagery and relies on ensemble complexity. In contrast, the current investigation uses this tuned Random Forest as a baseline to benchmark a fully variational quantum classifier trained on the same features, enabling a direct, head-to-head comparison of classical and quantum pipelines.

Together, these studies illustrate the strengths and limitations of classical EEG classification pipelines. Most effective approaches combine domain-informed preprocessing, such

as bandpass filtering, Common Spatial Patterns (CSP), and dimensionality reduction, with well-understood classifiers like SVMs or Random Forests. Lotte et al. emphasize that preprocessing decisions, not classifier complexity, often determine performance on EEG data. Craik et al. show that classical models based on features in the time and frequency domain frequently outperform deep learning methods in small sample or subject-specific settings. Luo et al. extend this further by integrating CSP with ensemble SVMs, achieving strong motor imagery results. Similarly, Basri and Arif achieve near-perfect seizure classification using FFT features and Random Forests on long, oversampled windows. While their models achieve high accuracy, they address different tasks and rely on more complex or domain-specific tuning. The present work adopts a deliberately lightweight approach—using PCA+CSP and Random Forest—as a tuned classical baseline to evaluate quantum classifiers in short windows of the EEG of movement executed.

2.2.2 Quantum Machine Learning Applied to EEG

Recent research has investigated employing quantum machine learning (QML) for classifying EEG data, addressing tasks like seizure detection and motor imagery. These studies show the potential of quantum models in neural decoding relevant to healthcare, though they differ significantly in terms of model architecture, preprocessing intricacies, evaluation criteria, and their specific areas of focus.

Meesala et al. [33] applied a Quantum Support Vector Classifier (QSVC) to the problem of epileptic seizure detection using the UCI Epileptic Seizure Recognition dataset. Their study compared classical SVM and quantum SVM models on preprocessed EEG signals reduced to 3, 4, or 10 dimensions via PCA and LDA. Using a ZZFeatureMap in Qiskit, their best-performing QSVC achieved 80.69% accuracy and an F1-score of 0.35. Although this performance was slightly higher than the classical SVM baseline in F1-score, classical models still achieved higher accuracy. Their work demonstrates the feasibility of applying QML to healthcare-related EEG problems, but differs in both the application domain and the complexity of the model. By contrast, the present study evaluates a fully variational quan-

tum classifier on executed movement EEG, benchmarked against a tuned classical baseline, and reports substantially higher performance across F1, AUROC, and precision–recall, with statistical significance testing and real hardware validation.

Chen et al. [7] introduce QEEGNet, a hybrid neural architecture that incorporates a variational quantum circuit into the classical EEGNet model. Their pipeline processes EEG signals from the BCI Competition IV-2a dataset, a four-class motor imagery task, and maps features into a nine-qubit circuit using RY rotations and ring entanglement. The hybrid model achieved a mean test accuracy of 72.65%, improving upon EEGNet's baseline of 69.53%. While their results demonstrate the feasibility of integrating quantum modules into compact convolutional networks, the study focuses solely on subject-wise accuracy, without reporting F1, AUROC, or precision/recall. It also lacks a comparison with classical non-deep-learning baselines such as Random Forests, and does not include real hardware testing. In contrast, this thesis evaluates a standalone variational quantum classifier trained on executed (not imagined) movement EEG, directly compares it to a tuned Random Forest using identical features, and validates performance through statistical testing and deployment to real quantum hardware.

Olvera et al. [37] investigated quantum-enhanced motor imagery classification using EEG data from the BCI Competition IV-2b dataset. They evaluated two hybrid architectures in which variational quantum circuits (VQCs) replaced the fully connected layer in classical deep neural networks, specifically EEGNet and ATCNet. Their best performing configuration, ATCNet with VQC, achieved a mean F1 score of 0.8551 and an average accuracy of 85 56% in the hold-out test. The subject-independent accuracy reached 73.72%. Although these results highlight the potential of integrating quantum models into deep learning pipelines, their architecture is based on convolutional feature extraction and is specific to motor imagery. By comparison, the present study applies a lightweight, end-to-end VQC trained on time-domain features extracted from executed movement EEG. Despite the simpler design and use of a classical preprocessing pipeline (PCA + CSP), our VQC achieved comparable performance with a best-case macro-F1 of 0.95 and mean accuracy of 76%, suggesting that

compact quantum classifiers can remain competitive even without deep learning components, particularly in binary motor execution tasks.

Summary. Recent work on quantum EEG classification has explored seizure detection and motor imagery tasks using hybrid or shallow quantum models. While Meesala et al. and Chen et al. demonstrated the feasibility of applying QML to healthcare-related EEG signals, their models either lacked comprehensive evaluation metrics or did not outperform classical baselines. Olvera et al. achieved strong performance using deep convolutional networks combined with variational quantum layers, but at the cost of model complexity and a reliance on motor imagery. Conversely, the current research evaluates a standalone, low-depth variational quantum classifier trained on time-domain features extracted from executed movement EEG. Despite the simpler architecture, our VQC achieved comparable or superior performance, outperforming two of the three reviewed quantum approaches, and includes full statistical testing, a tuned classical baseline, and limited real-hardware validation.

2.2.3 Gaps in Current Research

Despite growing interest in quantum models for EEG classification, several important gaps remain:

- **Limited optimizer exploration** — Most quantum EEG studies rely on SPSA or basic gradient descent, with little comparison across optimization strategies.
- **Lack of variance analysis** — Run-to-run variability and seed sensitivity are rarely reported or analyzed.
- **Minimal ablation studies** — Few papers systematically evaluate how preprocessing steps (e.g., CSP, PCA) affect quantum classifier performance.
- **Weak classical baselines** — Many studies omit strong non-deep-learning baselines such as Random Forest, or fail to use identical features across models.

- **Motor imagery focus** — Most work targets imagined movement rather than real, executed movement, limiting applicability to rehabilitation tasks.

- **Simulated-only execution** — The majority of quantum EEG studies rely entirely on simulators, without testing on actual quantum hardware.

The present study addresses these limitations by (i) running a Random Forest baseline on the same features used in quantum experiments, (ii) evaluating multiple optimizers including COBYLA, SPSA, and gradient descent, (iii) analyzing run-to-run variance across 40 simulated quantum runs, and (iv) executing a subset of trials on real IBM quantum hardware. By focusing on the EEG of executed movement, this work also expands the evaluation of QML into a domain of direct relevance to rehabilitation.

Chapter 3

TECHNICAL BACKGROUND

Part of the goal of the Smart NeuroRehab Ecosystem is to design experiments that provide an objective window into the cortical activity of a subject, which can help collect data to build ML models to understand brain signals and identify significant patterns associated to certain motor activities. Hand movements are critical to studying many neurological conditions, so that project has focused on studying these as a starting point.

To provide the infrastructure required to accomplish this purpose, it is important to understand the following three layers: (i) the electroencephalography (EEG) signals that reflect hand-movement execution, (ii) the classical pipelines traditionally used to extract and classify those signals, and (iii) the quantum-computing principles that motivate the variational circuits evaluated in this thesis.

This chapter reviews those layers in sequence, highlighting where current methods plateau and why a ten-qubit Variational Quantum Classifier may provide a more data-efficient alternative for detecting real movement within the Smart NeuroRehab framework.

3.1 Electroencephalography for Motor Tasks

Electroencephalography (EEG) is a non-invasive method for recording brain activity with small metal sensors placed on the scalp. Introduced to clinical practice in the 1930s, it remains popular because it tracks changes in neural rhythms with millisecond precision, costs far less than MRI, and can be worn while a person moves or performs therapy tasks. The drawback is that EEG signals are weak, mixed together by skull and scalp, and easily contaminated by eye blinks, muscle tension, and power-line noise. Even so, decades of research show that well-placed electrodes can detect reliable patterns linked to hand movement,

speech, sleep stages, and many other behaviours. In rehabilitation settings, EEG provides a real-time view of how the motor cortex reengages after stroke or injury, which is why it underpins the Smart NeuroRehab project described in this thesis.

3.1.1 Neurophysiology of Hand Movement

Voluntary hand movement starts in the motor areas of the cerebral cortex and is carried down the brain-stem and spinal cord to the muscles [42]. In the moments surrounding a hand movement, cortical activity follows a reliable pattern: power in the μ band (8–13 Hz) and β band (13–30 Hz) drops roughly 100–300 ms before the fingers move, remains low during execution, and rebounds a few hundred milliseconds after the movement ends [40, 35]. Intracortical recordings confirm that this event-related desynchronization (ERD) reflects neurons shifting from an idling rhythm to task-related firing [11]. The strongest μ/β changes arise over the central sensorimotor strip, making the ERD–ERS cycle the principal EEG signature for distinguishing movement from rest in brain-computer-interface studies. Following prior work that showed eight central electrodes capture nearly all of this movement-related activity while keeping data volume manageable [34] we base our analysis on that eight-channel subset.

3.1.2 EEG Acquisition Hardware and Electrode Layouts

Rather than using the sparse dataset collected by the Smart NeuroRehab team, recordings for this study come from the PhysioNet Motor Movement data set, collected with a 64-channel gel cap that follows the extended 10–10 electrode layout described by Jasper [25]. This larger and more diverse dataset can serve as a baseline to analyze hand-related EEG signals.

The signals from this dataset were digitized at 160 Hz with 24-bit resolution and referenced to linked earlobes. Figure 3.1 reproduces that layout and marks the eight channels analyzed in this thesis (FC5, FC6, CP3, Fz, C1, Cz, C3 and C4). A 160 Hz sampling rate is high enough to resolve the μ (8–13 Hz) and β (13–30 Hz) rhythms of interest without

the file-size overhead that accompanies higher-frequency γ data [47]. Although dry-electrode caps are available, the low-impedance gel system used in PhysioNet still provides the best signal-to-noise ratio for research-grade EEG [16].

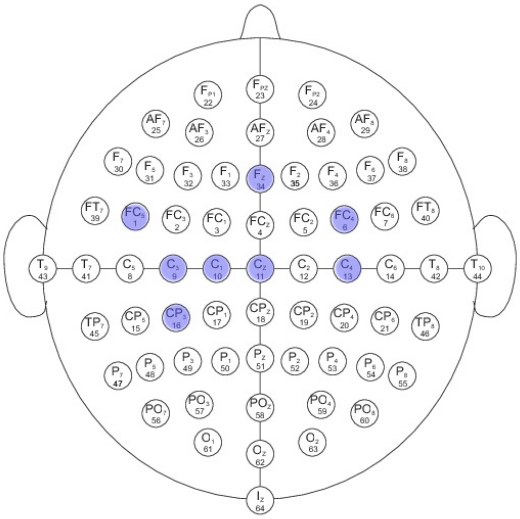


Figure 3.1: Sixty-four–channel 10–10 montage from the PhysioNet Motor Movement data set. Blue circles indicate the eight electrodes used in this study.

3.1.3 Common Artifacts and Noise Sources

Recorded EEG data often contains common artifacts. Eye blinks and eye movements dominate the 0–4 Hz region of frontal channels, slow electrode drift appears below about 1 Hz, neck-muscle activity rises steeply above 30 Hz, and power-line hum is centred at 50 or 60 Hz [13]. Most motor-task studies address these problems with a band-pass filter that keeps the μ (8–13 Hz) and β (13–30 Hz) bands, followed by a simple amplitude-threshold rule that removes epochs whose peak-to-peak swing exceeds about 100 to 200 µV [56]. After this basic cleaning, the signal retains the movement-related rhythms.

3.1.4 Time–Frequency Signature of Executed Movement

The central temporal pattern of executed hand movement is the event-related desynchronization and subsequent synchronization (ERD/ERS) cycle. Power in the μ band (8–13 Hz) and the β band (13–30 Hz) drops about 100–300 ms before the fingers move, stays low during the action, and rebounds to or above baseline within roughly 500 ms after the movement ends [40, 35]. Over the sensorimotor strip this dip reaches 30 %–40 % in the μ band and about 20 % in the β band, giving a time-locked marker that most movement-detection pipelines use once the signal has been band-pass filtered. Figure 3.2 shows a synthetic example of this ERD trough and post-movement rebound.

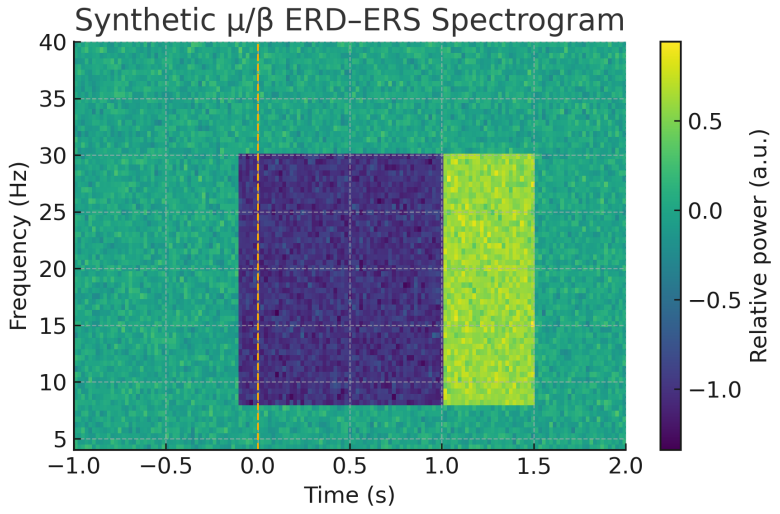


Figure 3.2: Synthetic time–frequency map depicting the ERD/ERS cycle. Blue shading marks the drop in μ and β power during movement; yellow shading shows the rebound after movement ends. The dashed vertical line indicates movement onset at $t = 0$. (Image generated in python.)

3.1.5 Summary

The material in this section shows how hand movement alters cortical rhythms and why those changes are easiest to detect with a small cluster of central electrodes. Power in

the μ (8–13 Hz) and β (13–30 Hz) bands drops just before and during movement, then rebounds after the action ends, producing the time-locked event-related desynchronisation and synchronisation (ERD/ERS) signature. Most unwanted signals lie outside the 8–30 Hz range, so a straightforward band-pass filter followed by a simple amplitude check removes the bulk of typical artifacts while preserving the movement pattern. With these physiological and signal-cleaning facts in mind, the next chapter examines the machine-learning methods that turn filtered EEG windows into movement-versus-rest decisions.

3.2 Classical Machine-Learning Pipelines for EEG

3.2.1 Feature Extraction

Most classical EEG pipelines start with engineered features that capture either band-specific energy or simple waveform statistics. Typical choices include band power in the μ (8–13 Hz), β (13–30 Hz), and γ ranges, the three Hjorth parameters (activity, mobility, complexity), and per-channel time-domain measures such as variance and zero-crossing rate [10, 31]. These features are inexpensive to compute and have clear physiological interpretations, which explains their long use in brain–computer-interface competitions.

3.2.2 Dimensionality Reduction Techniques

Raw EEG feature sets can contain thousands of numbers once every channel, time point, and band is included, so a compression step is almost always needed. Three methods appear most often.

Principal Component Analysis (PCA). PCA forms the covariance matrix of the feature set, then solves for its eigenvectors and eigenvalues. An eigenvector is a direction that the matrix stretches without rotation, and its eigenvalue shows how much variance lies along that direction. Keeping the top k eigenvectors therefore retains the k strongest variance directions. PCA is fast and data driven, but it ignores class labels, so the retained components are not necessarily the most discriminative [10].

Independent Component Analysis (ICA). ICA assumes that the sensors record linear mixtures of statistically independent sources and solves for an unmixing matrix that maximizes independence. It can isolate eye blinks and muscle noise, but is sensitive to model order and usually needs longer recordings to stabilize [31]. ICA is mentioned here for completeness, but is not used in the pipeline tested later in this thesis.

Common Spatial Patterns (CSP). CSP builds a covariance matrix for each class, then solves a generalized eigenvalue problem that finds spatial filters maximizing the variance ratio between classes. The filtered signals are converted to log-variance features that separate, for example, movement from rest. CSP works well when there are ample training data available, but it can overfit when the sample size is small [45].

3.2.3 Popular Classifiers

After dimensionality reduction, the feature vector is fed into a pattern recognition algorithm, also called machine learning. Linear Discriminant Analysis (LDA) and Support Vector Machines (SVM) remain popular because they train quickly, rely on only a few hyperparameters, and perform well when classes are linearly or near-linearly separable [31].

Random Forest is an ensemble method that builds a large collection of decision trees, each fitted to a bootstrap sample of the training data. During tree construction, every split considers only a random subset of the available features, a strategy that decorrelates the trees and reduces variance. At inference time, the forest predicts the class that receives the majority vote across all trees. This architecture captures non-linear class boundaries, tolerates mixed feature types, and is comparatively robust against overfitting when the ensemble is sufficiently large. Key hyperparameters include the number of trees, maximum tree depth, and the number of candidate features evaluated at each node. Random Forests also provide a built-in estimate of feature importance, offering insight into which channels or summary statistics contribute most to the classification. For these reasons, the model serves as the tuned classical baseline in the Results chapter.

Shallow convolutional neural networks have won several recent BCI competitions because

they learn spatial and temporal filters end-to-end, but they need larger data sets and careful regularization to avoid overfitting when trial counts are limited.

3.2.4 Limitations in Classifying Noisy EEG Data

Even though the present study trains on the large, pre-labeled PhysioNet data set, three practical problems remain.

1. **Calibration time** – every new participant still needs a short recording so the pipeline can adjust its spatial filters and fit the classifier.

2. **Non-stationary signals** – small changes in electrode contact or alertness can reduce accuracy by 15–20 % within a single session, and neither CSP nor Random Forest corrects that drift automatically.

3. **Poor cross-subject transfer** – a model trained on one individual rarely generalizes to another without additional data and retraining, so scaling the system still requires calibration per patient [10].

These limitations motivate the search for methods, including quantum approaches, that could shorten calibration or handle signal drift more efficiently.

3.2.5 Evaluation Metrics

Various metrics are selected to evaluate different aspects of this research results. In addition to the basic statistical metrics mean μ, standard deviation σ, median and trimmed mean [1], the following metrics are considered to accommodate class imbalance and misclassifications:

- Accuracy: defined as the proportion of all correctly classified instances (see Equation 3.1). It is stated in terms of True Positives (TP), True Negatives (TN), False Positives

[1] In a trimmed mean, a small percentage of the smallest and largest values of the dataset are removed before calculating the mean, with the purpose of removing outliers.

(FP) and False Negatives (FN) . It is helpful metric in binary classification problems.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

- Balanced Accuracy: can also be used to assess the performance of a model and in addition, it accounts for class imbalance or skewness (see Equation 3.2.)

$$\text{Balanced Accuracy} = \frac{TPrate + TNrate}{2} \tag{3.2}$$

- Macro-F1: is the unweighted average of the F1 scores across all classes [29] as seen in Equation3.3

$$\text{Macro-F1} = \frac{F1_1 + F1_2 + ... + F1_n}{n} \tag{3.3}$$

- Area under the ROC curve (AUROC): a Receiver-operating characteristic (ROC) curve provides a visual representation of a model's performance across all possible intervals obtained by plotting $FPrate$ against $TPrate$. The area under the resulting curve is the probability of ranking a positive identification higher than false [8]. In a perfect model, AUC will be 1.

- Confusion-matrix: is a visualization used to evaluate a model's performance by capturing counts of TP, TN, FP, and FN for downstream error analysis [19].

3.2.6 Aggregate statistics

The metrics for this study were specifically chosen to reflect the balance of the class and the clinical importance of the task. Macro-$F1$ was used as the primary metric because it gives equal weight to both classes, which is important in unbalanced settings. AUROC was included to assess the model's ability to rank predictions across all thresholds. Precision and recall were also specifically reported for the "Move" class because missed detections or incorrect move classifications could have practical consequences in movement detection applications.

Summary

A typical classical EEG pipeline follows four stages. First, it derives engineered features, for example band power or basic time-domain statistics. Second, it reduces dimensionality with PCA, ICA, or CSP to curb redundancy and emphasize task-relevant variance. Third, it feeds the compressed vector to a classifier such as LDA, SVM, or, as in this study, a tuned Random Forest that handles nonlinear boundaries and outputs feature-importance scores. Finally, they measure accuracy on data that were kept separate from training, so the results reflect generalization rather than memorization.

This sequence is not universal; some recent work skips manual features and dimensionality reduction, sending raw EEG into shallow or deep neural networks, while other studies let tree ensembles manage high-dimensional inputs directly. Whatever the method employed, classical approaches often require per-user calibration, remain sensitive to signal drift, and seldom transfer cleanly from one subject to another. The next chapter tests whether a ten-qubit Variational Quantum Classifier can mitigate some of these limits by mapping the reduced feature vector into a higher-dimensional quantum space.

3.3 Quantum-Computing Foundations

Quantum computing stores information in quantum states, not in classical bits. The basic unit, the qubit, can occupy any complex superposition of $|0\rangle$ and $|1\rangle$. A single qubit therefore lives in a two-dimensional Hilbert space. A Hilbert space is a vector space that can have any number of dimensions and comes with an inner (dot) product, so lengths and angles between vectors are well defined. In quantum theory each possible state of a system corresponds to a vector in this space, and the inner product governs probabilities and interference. A register of n qubits spans a 2^n-dimensional Hilbert space. When qubits interact they can become entangled, meaning the joint state cannot be written as separate single-qubit states. Ten qubits already provide $2^{10} = 1\,024$ orthogonal basis states, far more than the ten real numbers in our EEG feature vector.

Physical qubits are achieved in superconducting circuits, trapped ions, photonic waveguides, and other platforms, but present-day devices are small and noisy. Because large fault-tolerant machines are still in development, current research focuses on *variational quantum algorithms* that run short, hardware-friendly circuits and tune their gate angles with a classical optimizer. The Variational Quantum Classifier used later in this thesis belongs to that family. Before describing the circuit, the next subsection reviews qubits, superposition, and entanglement to show how a ten-qubit register can embed and process an EEG feature vector in a space that is already one thousand and twenty-four dimensions wide.

3.3.1 Qubits, Superposition, and Entanglement

A single qubit is written in Dirac notation as

$$|\psi\rangle \;=\; \alpha\,|0\rangle \;+\; \beta\,|1\rangle, \qquad |\alpha|^2 + |\beta|^2 = 1.$$

The pair $\big(|0\rangle, |1\rangle\big)$ spans a two-dimensional Hilbert space, a complex vector space equipped with an inner product that lets us calculate lengths and angles. Figure 3.3 shows the state $|\psi\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ as a red vector on the Bloch sphere; single-qubit gates act as rotations of that vector.

Measurement of the qubit on the computational basis projects it onto $|0\rangle$ or $|1\rangle$. For the superposition above each outcome occurs with probability one-half, as illustrated in Figure 3.4. Repeating the circuit many times and averaging the ± 1 outcomes yields the *expectation value*

$$\langle O\rangle \;=\; \langle\psi|\,O\,|\psi\rangle,$$

where O is the measured observable (Pauli-Z in our case) and $|\psi\rangle$ is the state prepared by the circuit.

When two qubits interact they can become *entangled*, meaning the joint state cannot be written as a product of single-qubit states. In this way, they are intertwined, where the state of one will affect the state of the other. The simplest example is a *Bell state*. Starting from

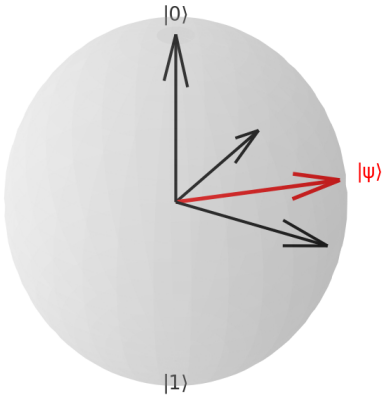


Figure 3.3: Bloch-sphere representation of a single qubit. The poles mark the computational basis states and the red vector indicates the equal superposition $\big(|0\rangle + |1\rangle\big)/\sqrt{2}$. Figure generated in Python with `matplotlib`.

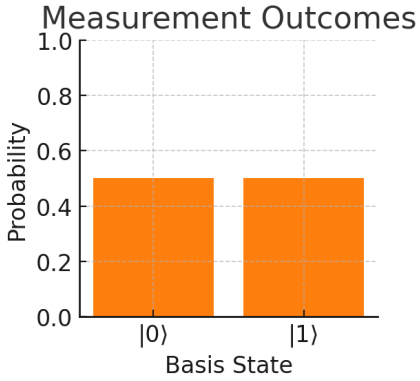


Figure 3.4: Probabilities of measuring $|0\rangle$ or $|1\rangle$ for the state in Figure 3.3. Equal bar heights reflect the fifty–fifty superposition. Figure generated in Python with `matplotlib`.

$|00\rangle$, a Hadamard gate on qubit 1 followed by a CNOT produces

$$|\Phi^+\rangle \;=\; \frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

In this state measuring the first qubit immediately fixes the second. A result of 0 forces the partner to 0, and a result of 1 forces the partner to 1. Such perfect correlations do not have a classical analogue and are a resource for quantum communication and machine learning circuits.

A register of n qubits spans a 2^n-dimensional Hilbert space, so ten qubits already provide $2^{10} = 1\,024$ orthogonal basis states. Mathematically, the full configuration of an n-qubit register is captured by a 2^n-component column vector called *statevector*. Later chapters encode a ten-element EEG feature vector in exactly that vector space and train a short entangling circuit, the Variational Quantum Classifier, to label movement versus rest.

3.3.2 Elementary Quantum Gates

Quantum gates are unitary matrices that evolve the qubit statevector while preserving total probability. In essence, gates are what we use to manipulate qubits and perform operations. Table 3.1 lists a universal single-qubit set and the two-qubit CNOT gate, which supplies entanglement.

Two-qubit gates supply entanglement. The best known example is the controlled-NOT (CNOT), which flips the target qubit when the control qubit is $|1\rangle$. Any unitary on n qubits can be approximated to arbitrary accuracy using only single-qubit rotations and CNOTs. (Concise primers on how to perform unitary operations are available in the IBM *Basics of Quantum Information* and *Understanding Quantum Information and Computation* learning paths [21, 24]).

3.3.3 How a Matrix Becomes a Real Gate

On paper a gate is a unitary matrix U. On Quantum hardware it is carried out by pulsing the qubits with precisely shaped electromagnetic fields.

Table 3.1: Frequently used quantum gates (single qubit unless noted).

Gate	Matrix	Informal action
X (NOT)	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	swaps $\|0\rangle$ and $\|1\rangle$
Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	90° rotation about y axis
Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	phase-flip of $\|1\rangle$
Hadamard H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	creates equal superpositions
$R_y(\theta)$	$\begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$	rotation about y by θ
CNOT (2-qubit)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	flips target if control is $\|1\rangle$

- **Single-qubit gates.** In superconducting chips a 25–50 ns microwave pulse rotates the qubit; pulse length sets the angle and pulse phase sets the axis [28]. Trapped-ion systems do the same with a resonant Raman laser [30].

- **Two-qubit gates.** A cross-resonance drive implements a CNOT in roughly 200 ns on superconducting hardware, while ions use the Mølmer–Sørensen interaction (100–300 µs) [30].

- **Measurement.** In cQED processors each qubit couples to a read-out resonator whose frequency shift reveals whether the state collapsed to $|0\rangle$ or $|1\rangle$; single-shot fidelity exceeds 98 % [61].

3.3.4 Parameterised (variational) circuits

Many near-term quantum learning models use circuits *parameterised*:

1. **Feature map** – encode each input x_k with $R_y(x_k)$.

2. **Entangling layer** – couple neighboring qubits with CNOTs.

3. **Trainable layer** – apply $R_y(\theta_k)$ with optimizable angles θ_k.

Figure 3.5 shows one repetition of this motif for four qubits.

3.3.5 Cost function and optimization

After the circuit runs, selected qubits are measured in the *Pauli-Z basis*, i.e. the computational basis $\{|0\rangle, |1\rangle\}$. Repeating the circuit yields an expectation value in $[-1, 1]$, which is mapped to a probability $p \in [0, 1]$ by the affine transformation $p = (\langle Z \rangle + 1)/2$. A classical optimizer such as COBYLA or SPSA then updates the parameters $\{\theta_k\}$ to minimize cross-entropy loss.

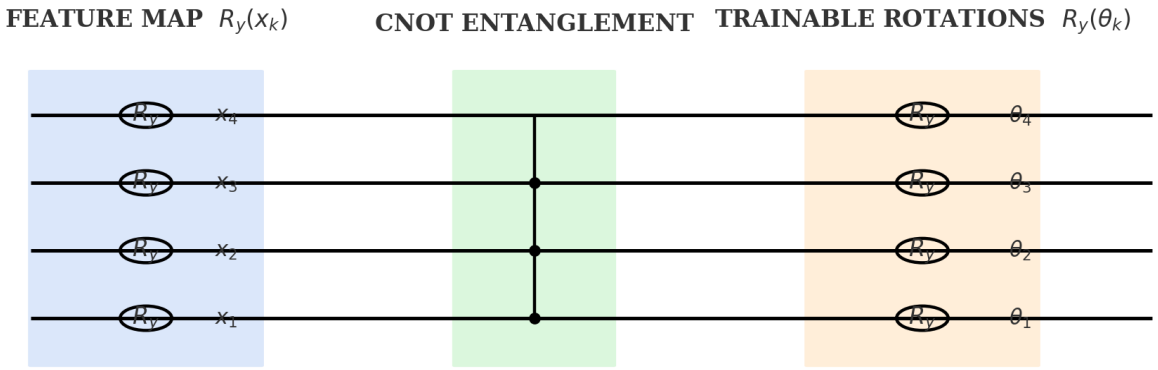


Figure 3.5: Four-qubit parameterized circuit. Blue boxes load features, green boxes entangle qubits, and orange boxes carry trainable rotations. Diagram generated in Python with `matplotlib`.

3.3.6 Depth, Noise, and Near-Term Viability

Every physical gate adds decoherence and control error, so practical NISQ circuits must remain shallow. A design with $\sim$ 30 two-qubit gates fits within the coherence budget of 20–30-qubit superconducting or trapped-ion processors and is trivial to simulate on a modern GPU. Because this thesis uses a noise-free state-vector simulator, hardware noise is absent unless an explicit noise model is injected. The shallow, hardware-ready layout therefore lets us benchmark the ideal quantum classifier against a classical baseline while keeping a clear path to future hardware tests.

3.3.7 Measurement and Observables

Figure 3.6 illustrates the most common quantum measurement: projecting a register onto the Pauli–Z (computational) basis $\{|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle\}$. Each basis state is like a parking spot on a numbered track. When we observe the quantum system we find the car parked in exactly one spot and all other probabilities collapse to zero.

Recall the n-qubit superposition introduced in Eq. (3.3.1). A projective measurement in the computational basis selects exactly one basis state $|k\rangle$ with probability $|\alpha_k|^2$ and collapses

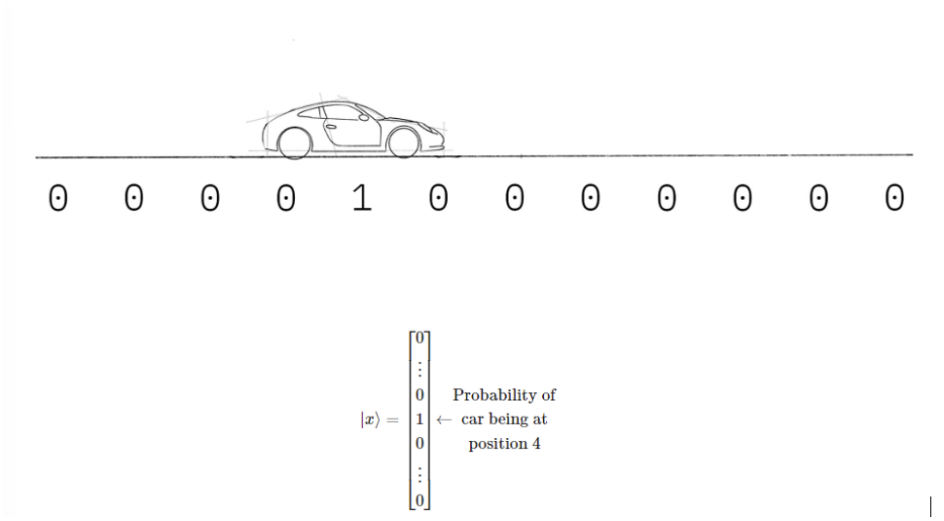


Figure 3.6: Analogy for a computational-basis measurement. The "car" sits at position 4, so the qubit register collapses to $|4\rangle$ with probability 1. Reproduced from the original Qiskit Textbook [23], CC-BY-4.0.

the register to that state. Repeating the same circuit many times produces a histogram $P(k)$, from which we compute expectation values. For example, measuring one qubit in the Pauli-Z basis yields ± 1 samples whose average is $\langle Z \rangle = P(0) - P(1)$. A simple affine map converts that number to the 0–1 band: $p = \frac{1}{2}\big(\langle Z \rangle + 1\big)$, which we treat as the classifier's movement probability.

In a multiqubit circuit we often measure *parity observable* $Z_0 Z_1 \ldots Z_9$. Its expectation value is found the same way, collecting ± 1 outcomes, average, then rescale. In simulation there is no *shot noise*. Shot noise is the statistical fluctuation that arises because only a finite number of runs are sampled, unless we deliberately inject a noise model. By contrast, on hardware we execute the circuit a few hundred or thousand times so the empirical histogram $P(k)$ converges to the true probabilities with high confidence.

3.3.8 From Quantum Execution to Classification Output

Quantum computers do not return a single deterministic output. Instead, they yield probabilistic results due to the nature of quantum measurement. When a circuit is executed, it prepares a quantum state through a sequence of gates, then measures it in the computational basis. This collapses the quantum state into a classical bitstring (e.g., `1010010011`), with probabilities determined by the amplitudes of the final state vector.

Each execution of the circuit yields one such sample. To approximate the distribution over all possible bitstrings, the circuit is typically executed multiple times, in a process called *sampling*. On real hardware, this is specified by a number of *shots*, typically 1024 or more. The result is a histogram over bitstrings. For example:

$$\{\texttt{'0000000000': 514, '0000000001': 510}\} \rightarrow \text{approx. } 50.1\% \text{ vs } 49.9\%$$

To extract meaningful numerical predictions from these bitstrings, Qiskit allows users to define an *observable*, such as the Pauli-Z operator Z on a specific qubit. The *expectation value* of this observable is computed by taking a weighted average of the outcomes:

$$\langle Z \rangle = P(0) - P(1)$$

For binary classification tasks like this one, the expectation value is mapped to a probability:

$$p_{\text{move}} = \frac{\langle Z \rangle + 1}{2}$$

This p is then treated as the model's confidence that the input EEG window corresponds to a movement class. A threshold (e.g., 0.5) is used to convert this into a class label.

Simulators like `AerSimulator` skip the sampling process and compute exact statevector amplitudes instead, which yields ideal expectation values without shot noise. However, real hardware always involves sampling, which introduces randomness. For this reason, all circuits are executed with 1024 shots during hardware runs in this study.

3.3.9 NISQ Hardware and the Qiskit Stack

Today's processors are in the noisy intermediate-scale quantum (NISQ) regime. IBM's 27–, 65– and 127–qubit superconducting devices are typical: coherence times T_1, T_2 fall in the 70–120 µs range; single-qubit gate errors are $< 10^{-3}$; two-qubit gate errors are a few 10^{-2}; and single-shot read-out fidelity exceeds 98 % [28, 61]. Qubits are linked in a *heavy-hex* lattice that balances fabrication yield against parallel scheduling [20].

Why Qiskit? *Qiskit* is IBM's open-source software development kit for quantum computing [44]. It provides a Pythonic front end (**Terra**) for building and transpiling circuits, a high-performance simulator layer (**Aer**), and a cloud runtime that connects the same code to real superconducting or trapped-ion hardware. All experiments in this thesis follow that workflow:

1. Build parameterised circuits in **Terra**.

2. Train and sweep on the GPU state-vector engine in **Aer**.

3. Validate the best configuration through the **Estimator** primitive on an IBM cloud device.

Figure 3.7 shows how the components align in IBM's quantum-platform roadmap:

Compute The physical resources: a *Quantum Computer* (27–127-qubit transmon chips today) and a pool of high-performance classical compute (CPU, GPU, AIU) used for shot aggregation, error mitigation, and hybrid workloads.

Qiskit Runtime Executes circuits on the QPU or simulator. Provides the *Estimator* and *Sampler* primitives that this thesis calls from Python.

Heterogeneous orchestration layer Services that coordinate quantum jobs with classical post-processing, including *Qiskit Serverless* and Qiskit HPC integration.

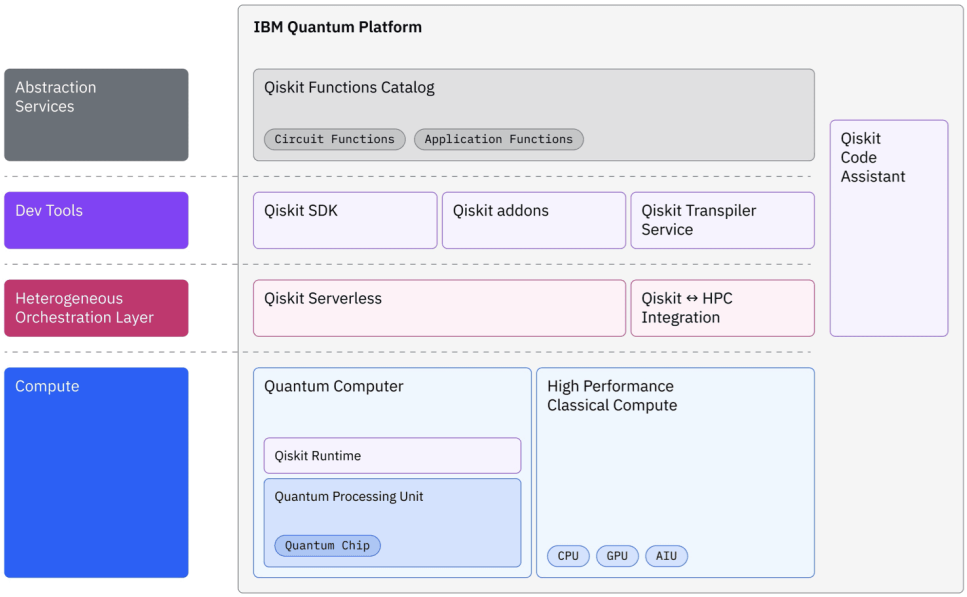


Figure 3.7: Four-layer Qiskit stack. From IBM Quantum's developer-tools blog [22]. Image © IBM

Dev tools The *Qiskit SDK*, domain-specific add-ons, and the cloud *Qiskit Transpiler Service*. These tools build and optimise circuits before they reach Runtime.

Abstraction services High-level libraries such as the *Qiskit Functions Catalog* and the forthcoming *Qiskit Code Assistant* that let users call pre-built circuit or application functions without touching low-level gates.

Although this thesis uses Qiskit, similar stacks are available in other open-source projects such as Google's *Cirq* [17], Xanadu's *PennyLane* [63], and the cloud-native *Amazon Braket SDK* [1]. Cross-checking results in more than one framework is a useful future step.

All experiments here run through the `EstimatorV2` primitive. A full minibatch of feature-mapped circuits is sent in a single call; the back-end returns the expectation values; COBYLA updates the parameters. Running on **Aer**, Qiskit's high-performance simulator [46], gives a noise-free baseline, while swapping to real hardware introduces decoherence and crosstalk

without changing any code.

3.3.10 Summary

In this chapter we introduced the quantum–computing concepts that are the foundation of the rest of the thesis. A register of n qubits lives in a 2^n-dimensional Hilbert space, so even ten qubits already offer 1 024 basis states, which is ample room to embed the ten-element EEG feature vector. Elementary single-qubit rotations together with the two-qubit CNOT form a universal gate set, arranging these in a shallow, parameterised feature map to entangle, to trainable-layer model yields NISQ-compatible circuits with about thirty two-qubit gates. After the circuit runs, projective measurement in the Pauli-Z basis produces ± 1 samples whose average, affinely rescaled, supplies the probability needed for a cross-entropy loss. Because present-day processors remain noisy intermediate-scale devices, simulations are first carried out on Qiskit Aer's state-vector engine, then optionally ported to IBM's 27–127-qubit transmon hardware through the cloud *Estimator* primitive. Terra handles circuit construction and transpilation; Aer and Runtime provide seamless back-ends, so the same Python code can benchmark an ideal, noise-free Variational Quantum Classifier and later validate it on real hardware without modification.

3.4 Variational Quantum Classifiers (Circuits)

Variational Quantum Classifiers (VQCs), also known as Variational Quantum Circuits, combine short, hardware-friendly quantum circuits with a classical optimization loop. The quantum part prepares and measures a parameterized state; the classical part adjusts the parameters θ to minimize a chosen cost function. The following subsections outline the loop, common data encodings, widely used ansatz families, standard optimizers, and the expressibility considerations that inform depth selection on NISQ hardware.

3.4.1 Hybrid training loop

A complete training step alternates quantum and classical operations:

1. **Encode** the classical feature vector x with a unitary $U_{\text{enc}}(x)$.

2. **Apply** the ansatz $U(\theta)$ and **measure** an observable (e.g. a Pauli-Z parity).

3. **Convert** the expectation value $\langle Z \rangle \in [-1, 1]$ to a probability $p = (\langle Z \rangle + 1)/2$.

4. **Evaluate** a classical loss, typically cross-entropy $L(p, y)$.

5. **Update** the parameters $\boldsymbol{\theta}$ with a gradient-free (COBYLA, SPSA) or gradient-based optimiser.

The loop repeats until the loss converges or a preset iteration cap is reached.

3.4.2 Data-encoding maps

Encoding maps translate each real feature x_k into a sequence of unitary gates. Four widely used examples are

- **ZFeatureMap** – local $R_Z(x_k)$ rotations.
- **ZZFeatureMap** – adds pairwise $\exp\big(-i\, x_j x_k ZZ\big)$ couplings.
- **PauliFeatureMap** – generalises ZZ to include XX and YY interactions.
- **Angle (RY) Map** – encodes each feature with $R_Y(x_k)$ rotations.

Choice of map influences expressibility, circuit depth, and optimisation landscape.

To illustrate a feature map, Figure 3.8 is the exact Angle map currently being used in this reasearch. The ten input features are first loaded as single-qubit rotations $R_Y(x_k)$ (left-hand panel). A barrier then freezes the data-dependent state before the trainable part begins. The RealAmplitudes template appears as two identical blocks: each block applies an $R_Y(\theta_k)$ rotation to every qubit and then threads a chain of CX gates through the register to create linear nearest-neighbor entanglement.

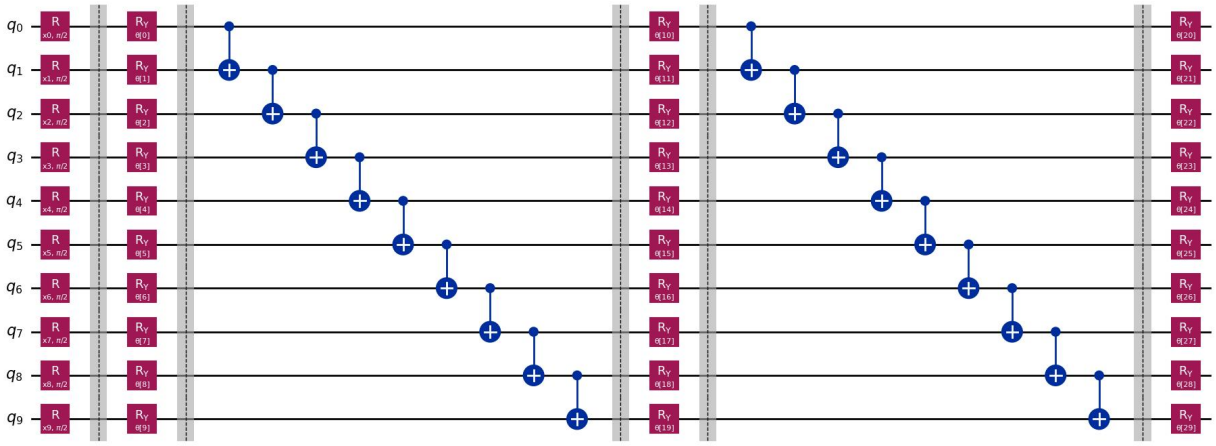


Figure 3.8: Expanded view of the ten-qubit circuit used throughout this work. The leftmost layer encodes each feature with $R_Y(x_k)$ (Angle map). Two repetitions of the RealAmplitudes ansatz follow, showing alternating $R_Y(\theta_k)$ rotations and a linear CX entanglement ladder. Barriers separate data input, entangling, and trainable layers. (Generated with Python using Qiskit)

Because the Angle map uses only R_Y gates, it preserves feature variance while adding just one single-qubit rotation per qubit, keeping the overall circuit depth compatible with NISQ coherence limits.

3.4.3 Ansatz families

An *ansatz* is a fixed gate pattern with trainable parameters θ. It defines the hypothesis class that the optimizer can explore.

Popular variational templates include:

RealAmplitudes Alternating layers of R_Y gates and linear CNOT chains; depth and entanglement pattern are tunable.

EfficientSU2 Blocks of $R_X\, R_Y\, R_Z$ rotations followed by full entanglement; provides higher expressibility at the cost of more two-qubit gates.

Hardware-Efficient Rotation layers interleaved with connectivity-matching CNOTs; minimises calibration effort on a given device topology.

The ansatz must be expressive enough to separate classes yet shallow enough to stay within coherence limits.

3.4.4 Optimisation strategies

COBYLA (*Constrained Optimization BY Linear Approximations*) builds a local linear model of the objective and solves a small trust-region sub-problem at each iteration [43]. Because it needs only function evaluations, without gradients, it keeps the number of circuit executions per step to a minimum, an advantage when each evaluation must be simulated in a 2^{10}-dimensional state space.

SPSA (*Simultaneous Perturbation Stochastic Approximation*) perturbs all parameters along two random directions and forms an unbiased stochastic gradient estimate with exactly two additional circuit queries per step [57]. When the loss landscape is reasonably smooth, this gradient information can reduce the total iteration count compared with fully derivative-free methods.

If exact derivatives are required, *parameter-shift gradient descent* evaluates the circuit at $\theta_k \pm \frac{\pi}{2}$ for each parameter, giving an analytic gradient component [51]. The disadvantage is that the evaluation cost now scales linearly with the number of trainable angles (40 for the RealAmplitudes ansatz used later), making each optimization epoch substantially more expensive on the GPU simulator.

In this thesis, where all sweeps run on a state-vector back-end, evaluation cost, not noise, dominates run-time. COBYLA therefore serves as the primary optimizer for the grid search, with SPSA retained as a stochastic alternative that provides gradient cues at almost the same per-iteration cost.

3.4.5 Summary

A Variational Quantum Circuit replaces a fixed quantum algorithm with a *trainable* one. Classical features are first embedded by a data-encoding unitary (e.g. ZFeature, ZZFeature, Pauli, or Angle maps). A short ansatz template like RealAmplitudes, EfficientSU2, or another hardware-efficient pattern—then supplies the trainable parameters θ. Measurement of a simple observable such as a Pauli-Z parity yields an expectation value that a classical optimizer (COBYLA, SPSA, or parameter shift gradient descent) uses to update θ. The circuit depth must be high enough to express rich decision boundaries yet low enough to avoid barren plateaus and stay within NISQ coherence limits. Together, the encoding map, ansatz, and optimizer form the three design options that determine the capacity, trainability, and practical runtime of a VQC.

3.5 Why Quantum for EEG Classification

Classical EEG pipelines already achieve useful accuracies, yet they face fundamental drawbacks. Richer decision boundaries require deeper neural networks or larger tree ensembles, which in turn demand more training data, longer calibration sessions, and greater on-device computational resources. A quantum model offers a different scaling curve. The Hilbert space grows exponentially with qubit count, whereas the number of trainable parameters can remain linear.

3.5.1 High-dimensional feature space, limited parameter count

Embedding a ten-dimensional EEG feature vector in a ten-qubit register places the data in a Hilbert space with $2^{10} = 1\,024$ orthogonal basis states. The representational capacity thus expands by an order of magnitude, yet the circuit need not introduce a matching explosion in trainable parameters. A hardware-efficient template such as RealAmplitudes requires only one or two adjustable rotation angles per qubit, approximately forty parameters in total at depth three.

This *exponential space, linear parameter* property is attractive for neurorehabilitation tasks, where the volume of labelled data is limited. The circuit can search a rich decision surface while keeping the number of free variables comparable to that of a small classical model [3, 55]. In principle, fewer parameters translate to shorter calibration sessions.

3.5.2 Chapter Summary

This chapter collected the technical material that underpins the rest of the thesis. Section 2.1 reviewed the physiological origin of movement-related EEG, described the μ and β ERD–ERS signature, and explained why the analysis focuses on eight central electrodes. Section 2.2 outlined a classical pipeline that begins with engineered band-power and time-domain features, applies PCA and CSP for dimensionality reduction, and finishes with common classifiers such as LDA, SVM, and Random Forest. Section 2.3 introduced the essentials of quantum computing, including qubits, superposition, entanglement, unitary gates, measurement, and the Qiskit software stack available for current NISQ devices. Section 2.4 presented Variational Quantum Circuits, describing the hybrid training loop, popular data encoding maps, hardware-efficient ansätze, and practical optimizers such as COBYLA, SPSA, and parameter-shift gradient descent. Section 2.5 explained the appeal of a ten-qubit VQC for EEG work, noting that it maps a ten-component feature vector into a 2^{10}-dimensional Hilbert space while using roughly forty trainable parameters, a scale that can shorten calibration when labeled data are scarce.

Together, these sections provide the signal processing background, classical baselines, quantum concepts, and variational design principles necessary for understanding the experimental methods and results in the chapters that follow.

Chapter 4

METHODS

This chapter details the experimental design used to evaluate quantum versus classical classifiers, including data acquisition, pre-processing, model architectures, hyperparameter search, and statistical validation. The supporting software pipeline is provided to ensure that every step is reproducible. Figure 4.2 gives a high-level overview of the steps part of the pipeline, these will be discussed in detail in this chapter.

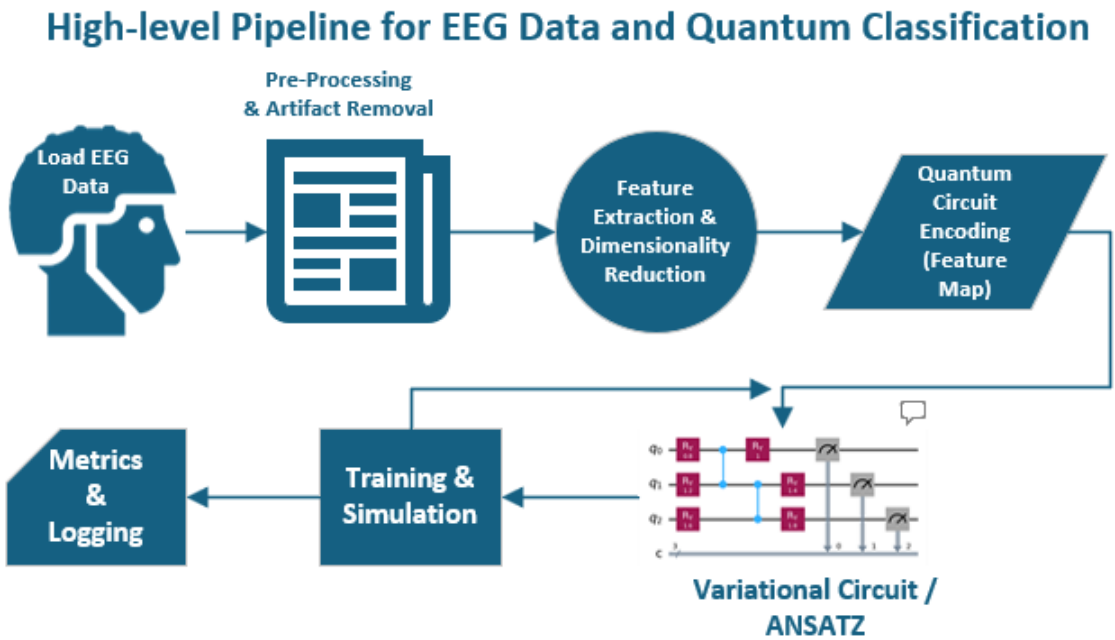


Figure 4.1: High-level overview of the pre-processing and modeling pipeline.

4.1 Dataset

We performed our research using the public *PhysioNet EEG Motor Movement/Imagery* dataset [16]. We use the curated CSV version of this dataset prepared by Shuqfa *et al.* [54], which aligns triggers, removes corrupted trials, and exports each recording as a pair of signal and annotation matrices. Pragati Dode labeled this release and supplied *four* cleaned CSV files, each file contains a labeled and preprocessed subset of the full dataset. Labels correspond to different actions performed by subjects (e.g., hand-movement, relax, foot-movement).

This thesis uses only one of those subsets (the one in the first file) and focuses only on two types of actions "hand-movement" and "relax", between which, we aim to discriminate. The rationale for using only one subset is twofold: (1) the first file contains sufficient data (over 1 million rows) for robust model development, hyperparameter tuning, and repeated statistical testing, allowing for reliable conclusions within a single-subject training paradigm; and (2) limiting the scope ensures controlled computational costs and manageable complexity during quantum circuit simulation, especially given hardware constraints and the use of high-performance GPU resources.

Furthermore, by isolating training to a single curated subset, we ensure consistency in evaluation while establishing a reproducible experimental foundation for follow-up studies. The remaining three subsets can be preserved for future work on cross-subject generalization and external validation.

4.1.1 Label Categories

The files from the PhysioNet data contain twelve task codes (1–12) corresponding to various motor and intent activities. For this research, we are only interested in a subset of these labels, which are shown in Table 4.1 together with their corresponding description and number of time-windows 1.1 available. We use only data corresponding to the *hand-movement* labels and their matching *relax* baselines:

Table 4.1: Raw-time sample counts for the labels used in this thesis.

Label	Task description	Raw time-window samples
1	Relax (baseline)	741 600
2	Open/close **left** fist	370 864
3	Open/close **right** fist	363 536
4	Relax (baseline)	741 600
7	Relax (baseline)	741 600
8	Open/close **both** fists	362 640
10	Relax (baseline)	741 600
Total windows used		**4 063 440**

Class mapping. We can then work with a binary classification problem, where $Class_0$ is *Relax* with the corresponding label codes $\{1, 4, 7, 10\}$ and $Class_1$ corresponds to *Hand-movement* with label codes $\{2, 3, 8\}$. We have chosen to study only two classes to reduce the complexity that a multi-class problem entails, and focus instead on the analysis of classification models. Furthermore, this choice of classes can help establish a comparison baseline for future work.

4.1.2 Subject coverage

For statistical reasons, it is important to have a clear understanding of the population from which the samples are drawn. Of particular interest in our work, is the number of subjects as this can help understand variability of the data and to assess model accuracy, as well as generalization of models to other populations, e.g., drawing samples from one subject may result in overfitting due to bias and under representation of EEG signals variations.

The rows of the dataset files (CSV) are pooled across participants, however, they lack an explicit `subject_id` column that would allow us to distinguish between the data from

different subjects. By cross-referencing the recording order with the original corpus, we estimate that the file contains approximately ≈ 26 **unique subjects** (about one quarter of the 109-participant cohort). The selected CSV file contains 4,063,440 raw EEG time samples, corresponding to 64-channel voltage readings recorded at 160 Hz. These include unlabeled segments and non-task intervals. After filtering to remove corrupted data, aligning with task labels, and balancing class distributions, approximately 400,000 labeled time samples remained. These were segmented into non-overlapping 300-sample windows, producing a final dataset of approximately 1,333 labeled examples. Each window was assigned a binary label of either "Move" ($Class_1$) or "Relax" ($Class_0$) based on the associated task segment. All other tasks, including motor imagery and foot movement, were excluded.

Trial timing. According to the dataset's formal description [54], each relax-or-movement cue lasts approximately 4.1 ± 0.2 seconds. Since our 300-row windows correspond to 1.88 seconds at 160 Hz, each cue can be divided into exactly two non-overlapping windows without crossing into the adjacent task period. This avoids label contamination from overlapping segments, ensuring that each window contains only data from a single annotated task block.

Because this study benchmarks window-level classifiers and uses stratified train/validation splits, the absence of explicit IDs does not bias the results. Subject-wise generalization can be assessed later on an untouched subset that retains IDs.

4.1.3 EEG Channel Selection and Justification

Due to the high dimensionality of the full 64-channel EEG data and the increased computational complexity it introduces, a targeted subset of channels was selected to reduce noise, improve signal-to-noise ratio, and focus on regions most relevant to motor imagery classification.

To ensure proper alignment of channel indices with the dataset, we verified the mapping between the raw EDF files and the columns in the CSV. This confirmed that the dataset adheres to the standard 10–20 international electrode system. Based on this, we selected

eight electrodes located along the central sensorimotor strip: FC5, FC6, CP3, Fz, C1, Cz, C3, and C4.

These channels were chosen for three reasons: (1) prior neurophysiology literature shows that they span the primary motor cortex, where movement-related ERD/ERS activity in the μ and β bands is most prominent [40], (2) they consistently yielded the highest classification performance during exploratory testing, and (3) they significantly reduced computational load while preserving discriminative information.

Figure 3.1 (p. 17) shows the full 64-channel layout with the selected channels highlighted in blue. Additionally, the correlation matrix in Figure 5.4 (p. 75) reveals strong bilateral interactions (e.g., C3–C4: $r = 0.82$, FC6–Fz: $r = 0.85$), supporting the idea that structured interactions across these regions encode motor intentions.

Table 4.2: Selected EEG Channels and Their Functional Relevance

Channel	**Cortical Region**	**Functional Relevance**
FC5	Premotor (left)	Motor planning (contralateral)
FC6	Premotor (right)	Motor planning (contralateral)
CP3	Somatosensory (left)	Sensorimotor feedback (contralateral)
Fz	Midline frontal	Supplementary motor area (SMA)
C1	Central (left)	M1 hand region (contralateral)
Cz	Midline central	SMA + bilateral motor integration
C3	Motor cortex (left)	Primary motor (contralateral)
C4	Motor cortex (right)	Primary motor (ipsilateral)

Exploratory runs using other channel configurations, such as frontal-only, full 64-channel, or posterior subsets, yielded lower macro-$F1$ and AUROC scores. This eight-channel configuration consistently achieved the best trade-off between accuracy, stability, and runtime, especially when paired with CSP filtering and PCA-based compression (see Fig. 4.4).

4.2 Pre-processing

The raw CSV exported from Pragati Dode's curation still contains millions of unfiltered voltage samples, ocular and muscle artefacts, and labels for twelve different tasks. Before any classifier can learn, these data must be (i) cleaned, (ii) segmented into uniform epochs, and (iii) distilled into a ten-value feature vector compatible with the ten-qubit VQC. The pipeline therefore follows a classical *filter-segment-feature-compress* recipe (Fig. 4.2), that ends in two diverging model branches, Random Forest and Variational Quantum Classifier, fed with *identical* inputs. Each stage is chosen to balance physiological fidelity (8–30 Hz motor bands, central-strip electrodes) with computational thrift (1.88 s windows, 56-element feature vectors). A full specification of every operation and its parameter settings is given in Table 4.3.

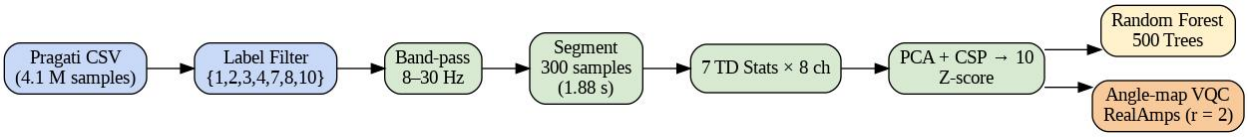


Figure 4.2: End-to-end pre-processing and modelling pipeline used in this thesis.

4.2.1 Band-pass Filtering

Motor execution modulates activity primarily in the alpha (8–13 Hz) and beta (13–30 Hz) bands. A zero-phase, second-order Butterworth band-pass filter keeps only this range, removing slow drift (< 5 Hz) and high-frequency EMG or line noise (> 30 Hz).

4.2.2 Epoching

Continuous recordings are segmented into 1.88 s windows (300 samples). Longer windows dilute transient motor patterns, while shorter ones reduce signal-to-noise ratio. The 300-sample length gave the best overall accuracy, F1, and Auroc scores over all experiments.

Table 4.3: Pre-processing overview.

Stage	**Purpose**	**Implementation**
Data acquisition	Load curated CSV and verify labels	Subset 1 of the Pragati-annotated PhysioNet release; trigger codes cross-checked against original log
Band-pass filtering	Keep motor bands, suppress drift and EEG noise	2nd-order zero-phase Butterworth, 8–30 Hz (`scipy.signal.butter` + `filtfilt`) [60]
Segmentation	Convert continuous stream to fixed windows	Non-overlapping 300-sample blocks (1.88 s @ 160 Hz)
Feature extraction	Produce compact numeric representation	Seven time-domain stats per channel (mean power, variance, skewness, kurtosis, zero-crossings, peak-to-peak, log variance) on the eight central electrodes
Standardization	Remove scale bias	Z-score across the training set; concatenate to one 56-element vector (7 features $\times$ 8 channels)
Dimensionality reduction	Compress to qubit count	PCA followed by CSP $\rightarrow$ 10 components per window

4.2.3 Time-Domain Feature Extraction

Why time-domain statistics? In movement EEG the key information is a band-limited amplitude change (μ/β ERD), so compact descriptors such as energy, variance, and zero-crossing rate capture the discriminative content with far less data and far greater interpretability than raw samples [47, 40]. Condensing each window to $7 \times 8 = 56$ values also keeps the input dimension well matched to the ten-qubit Variational Quantum Circuit used later in this work.

Table 4.4: Seven time-domain features computed per channel.

Feature	**Rationale for hand-movement EEG**
Signal energy ($\frac{1}{N}\sum x^2$)	Tracks overall power; falls during μ/β ERD.
Variance	Second-order spread, insensitive to DC offset.
Skewness	Flags asymmetric artefacts (e.g., eye blinks).
Kurtosis	Highlights bursty or spiky activity.
Zero-crossing count	Rough proxy for dominant frequency; rises when β rebounds.
Peak-to-peak amplitude	Maximum minus minimum; sensitive to large transients.
Mean value	Baseline level; included for completeness after de-biasing.

Each 1.88 s window contains 300 samples on eight selected electrodes (FC5, C1, FC6, CP3, Fz, Cz, C3, C4). Instead of passing all 2 400 raw points to the classifiers, we summarize every channel with seven well-established *time-domain features*; the result is a fixed-length vector of $7 \times 8 = 56$ values per window. Table 4.4 lists the features in the exact order

produced by the extraction routine.

Implementation. All seven statistics are extracted with vectorised `NumPy/SciPy` functions: mean-squared amplitude for signal energy, variance, skewness, kurtosis, a simple zero-crossing counter, peak-to-peak range, and the raw mean value. Features from the eight selected channels are concatenated in the fixed order `[FC5, C1, FC6, CP3, Fz, Cz, C3, C4]`, giving 56 numbers per window. Each dimension is subsequently z-scored using the training-set mean and standard deviation to avoid information leakage. The resulting normalised vector feeds directly into the PCA + CSP compression step described in Sec. 4.2.4.

4.2.4 Dimensionality reduction: PCA followed by CSP

A raw time-domain vector contains 56 features, which is more than the quantum circuit needs. Early Qiskit simulations showed that raising the qubit count above ten did not yield any gains whilst significantly increasing the training time. For this reason, it is desirable to reduce the dimensionality of the vector to 10 dimensions, which can help balance expressive power versus computational cost.

Dimensionality reduction is performed by a *two-stage pipeline*: (i) an unsupervised Principal Component Analysis (PCA) aimed to reduce some global redundancy and broadband noise; (ii) a supervised Common Spatial Patterns (CSP) transform concentrates the remaining variance on the bands that best separate *Move* and *Relax* epochs.

In pilot grid-searches, this two-stage pipeline increased validation macro-$F1$ by 8–12 pp compared with either PCA or CSP alone and produces a compact feature set that maps cleanly onto the ten-qubit VQC used throughout the study.

Principal Component Analysis. Figure 4.3 plots the cumulative eigenvalue spectrum of the training fold. A clear elbow occurs after the *sixth* component; from that point on each additional component contributes only a few percent of extra variance. By the 25th component the curve crosses the 80 % cumulative-variance threshold and then flattens. A

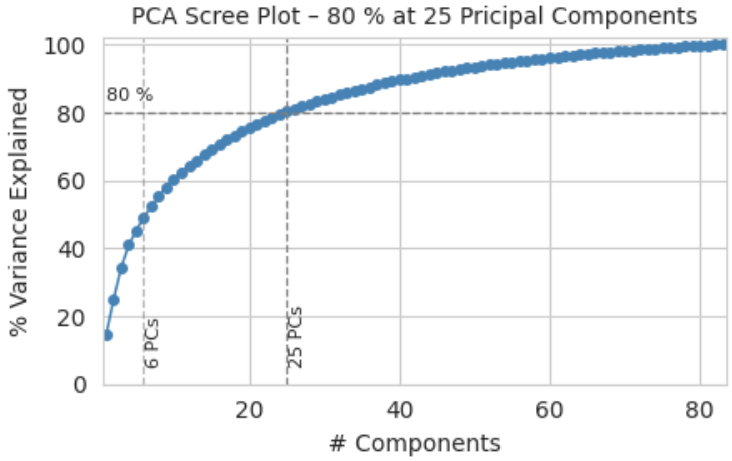


Figure 4.3: Scree plot of the PCA eigenvalues on the training fold. The first six components (dashed line) capture most of the curvature before the slope flattens, while the first 25 components explain roughly 80% of the total variance.

small grid-search confirmed that keeping the **first six** principal components ($k = 6$) produced the highest validation macro-$F1$ while greatly reducing the qubit-count required by a full 25-dimension encoding. The six-component output is re-standardized and forwarded to the CSP stage.

Common Spatial Patterns. PCA is a class-agnostic technique that compresses the variance across the entire dataset. CSP complements it by deriving spatial filters that emphasize variance differences between classes. Specifically, CSP computes spatial filters that maximize variance for one class while minimizing it for the other, by solving a generalized eigenvalue problem on the class-wise covariance matrices. The resulting filters project EEG signals into a new space where the movement and rest windows become more separable.

Although CSP does not operate explicitly in the frequency domain, its projections can enhance task-relevant bands, particularly 8 to 30 Hz, where sensorimotor ERD / ERS patterns dominate class differences. These effects are visualized in the power spectral densities of the four selected components in Figure 4.4. These components, chosen as the top two and

bottom two eigenvectors ranked by class variance, highlight discriminative structure while suppressing irrelevant noise. CSP remains widely used in motor-task BCI pipelines due to its interpretability, computational efficiency, and ability to extract discriminative features with minimal added dimensionality.

Using **four** CSP components (`n_csp_components = 4`) adds four features to the six retained PCA components, bringing the total to a 10-dimensional vector.

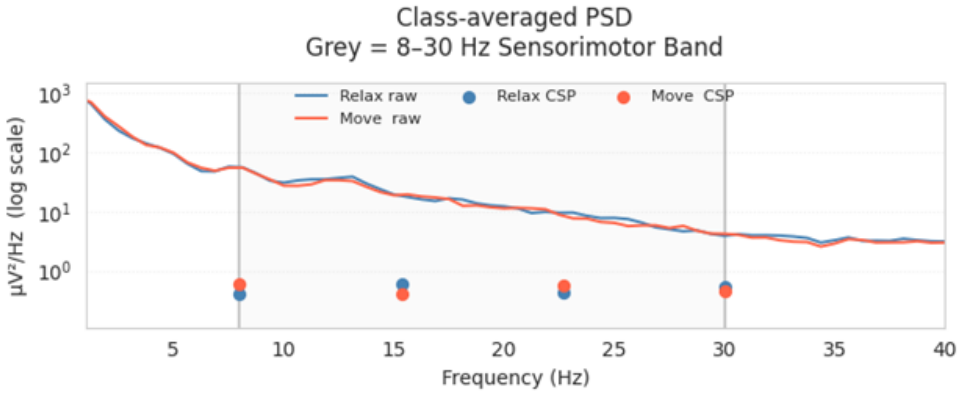


Figure 4.4: Class-averaged power spectral density (log scale) before (solid lines) and after (dots) the CSP transform. The grey band highlights the 8–30 Hz sensorimotor range. CSP suppresses low-frequency artifacts and accentuates the μ/β contrast between classes, improving separability.

Figure 4.4 illustrates how CSP reshapes the spectral content of each window. Before projection (*solid lines*), the *Move* and *Relax* spectra are nearly indistinguishable across 2–40 Hz. After CSP (*dots*), a clear gap emerges in the 8–30 Hz band: power decreases for *Move* and increases for *Relax*, consistent with expected μ/β desynchronization. Simultaneously, power below 8 Hz (e.g., eye-blink drift) and above 30 Hz (e.g., EMG noise) is attenuated. This confirms that CSP acts as a task-specific spatial filter rather than a broadband amplifier. These enhanced and band-limited differences are encoded in the four CSP-derived features appended to the six PCA components, completing the 10-dimensional vector used for classification.

The resulting vector is clipped to $\pm 3\sigma$, linearly rescaled to the interval $[-\pi, +\pi]$, and fed to the Angle feature map of the Variational Quantum Classifier as encoded rotation angles.

4.2.5 Class balancing: SMOTE oversampling

The class distribution in the filtered dataset was moderately imbalanced, with fewer labeled examples in the *Move* condition compared to *Relax*. While this imbalance was not extreme, early tests revealed that minority-class sensitivity (i.e., recall) tended to drop without some form of balancing.

To mitigate this, we applied the *Synthetic Minority Oversampling Technique* (SMOTE) after dimensionality reduction but before training. SMOTE creates synthetic *Move* examples by interpolating between existing samples in the 10-dimensional feature space. This adds variability and reduces the model's tendency to bias toward the majority class during early optimization steps.

Although not strictly necessary given the dataset size, SMOTE was retained in the final pipeline to improve stability and reduce run-to-run variance across repeated training runs. Future work could explore whether alternative balancing methods or class-weighted loss functions offer similar or better results without synthetic data generation.

4.3 Variational Quantum Classifier

The study employs a *Variational Quantum Classifier* (VQC) in the hardware-efficient style of [3]. Each optimization step alternates:

1. **Quantum phase.** The ten-element feature vector is encoded on a ten-qubit register, propagated through a short ansatz, and measured; the expectation value $\langle Z^{\otimes 10} \rangle \in [-1, 1]$ is linearly mapped to a class probability.
2. **Classical phase.** A gradient-free optimizer (COBYLA in this project's configuration) updates the 40 trainable angles to minimize mean-squared error on each minibatch.

The full circuit composed of two encoding layers plus three RealAmplitudes blocks contains about 30 CNOT gates, well within NISQ coherence limits and fast to simulate on an A100 GPU.

4.3.1 Circuit architecture

The variational circuit (Fig. 4.5) is purposely shallow yet expressive enough for the ten-qubit budget available on common 27-qubit IBM Quantum back-ends. It is built in three stages:

1. **Angle feature map.** Each scaled input $x_k \in [-\pi, \pi]$ drives a single rotation $R_Y(x_k)$, preparing the register in a data-dependent superposition. This layer is parameter-free and depth 1.
2. **Entangling ladder.** A linear CNOT chain couples neighboring qubits ($q_0 \to q_1 \to \ldots q_9$). The ladder injects dependencies between all features with just $n-1 = 9$ two-qubit gates, keeping the overall two-qubit depth low while avoiding all-to-all connectivity assumptions.
3. **Hardware-efficient ansatz.** Two repetitions (r=2) of the RealAmplitudes template [58] follow, each consisting of ($R_Y \circ R_Z$) single-qubit pairs and a second linear CNOT block. With n=10 qubits and r=2, the ansatz contributes $2nr = 40$ trainable angles $\boldsymbol{\theta}$ and 18 additional CNOTs.

Depth and coherence. The complete circuit therefore contains 10 data rotations + 27 CNOTs + 20 R_Y + 20 R_Z (single-qubit depth 3, two-qubit depth 3).

Expressivity considerations. The linear entanglement pattern yields a circuit graph with diameter nine, ensuring that every qubit is at most nine CNOTs away from any other, while the alternating R_Y–R_Z layers break symmetry and avoid barren plateau behavior observed in purely single-axis designs [3]. Empirically, adding a *third* repetition gave no macro-$F1$ lift in pilot tests yet increased training time by $\approx 40\%$, hence the two-layer configuration was adopted.

4.3.2 Example: VQC Simulation in Qiskit

To help those unfamiliar with quantum machine learning frameworks, this section provides a simplified code example that illustrates the core structure of a Variational Quantum Classifier (VQC) implemented in Qiskit (see Listing 4.1 below.) While the actual pipeline used in this research includes additional features such as batch training, parameter logging, optimizer sweeps, and warm-start initialization, the example provided shows how a basic VQC can be executed on a statevector simulator using the `Estimator` primitive.

Listing 4.1: Conceptual example of simulating a VQC using Qiskit's Estimator and AerSimulator

```
from qiskit import QuantumCircuit
from qiskit.circuit.library import RealAmplitudes
from qiskit.primitives import Estimator
from numpy import pi

# Create a 10-qubit quantum circuit
qc = QuantumCircuit(10)

# Encode input features with angle-based RY rotations
x_scaled = [-0.2*pi, 0.5*pi, ..., -0.1*pi]  # 10 rescaled features
for i in range(10):
    qc.ry(x_scaled[i], i)

# Add RealAmplitudes ansatz (2 layers, linear entanglement)
ansatz = RealAmplitudes(10, reps=2, entanglement='linear')
qc.compose(ansatz, inplace=True)

# Simulate circuit using Aer statevector simulator
estimator = Estimator()
theta_initial = [0.1] * ansatz.num_parameters  # Random guess
result = estimator.run(circuits=[qc], parameter_values=[theta_initial]).result()
```

```
# Convert Z expectation to class probability
z_expectation = result.values[0]
p_move = (z_expectation + 1) / 2
```

The feature vector (e.g., a 10-dimensional PCA+CSP-encoded EEG window) is scaled to the interval $[-\pi, \pi]$, then embedded into a quantum circuit using `RY` rotations. A simple ansatz (here, Qiskit's built-in `RealAmplitudes` template) is appended, and the expectation value of the Pauli-Z observable is computed.

This example uses Qiskit's GPU-accelerated `AerSimulator` to compute the exact expectation value $\langle Z \rangle$ without shot noise. In practice, the full training loop iteratively updates the parameter vector using a classical optimizer (see Section 5.2) to minimize loss over batches of EEG windows. The same codebase can be run on real quantum hardware by replacing the backend with a hardware Estimator from IBM's Qiskit Runtime service.

4.3.3 Data encoding

After z-scoring, the features are clipped to $\pm 3\sigma$ and rescaled to $[-\pi, +\pi]$ so that each value serves directly as a rotation angle. Four alternative feature maps were benchmarked. Unless stated otherwise the **Angle** map is used in the final configuration:

- **Z** – one local $R_Z(x_k)$ per qubit; no entanglement.
- **ZZ** – adds pairwise $\exp\!\left[-i\, x_j x_k\, Z \otimes Z\right]$ couplings.
- **Pauli** – extends ZZ to $X \otimes X$ and $Y \otimes Y$; deepest map.
- **Angle** – one $R_Y(x_k)$ per qubit (angle encoding); shallowest and empirically best.

Using the Angle map with two RealAmplitudes repetitions yielded the highest validation macro-$F1$ during the pilot grid-search, while deeper or more entangling maps showed no consistent benefit.

Figure 4.5 illustrates the *Angle* feature-map used throughout this work. Each of the $d = 10$ pre-compressed features $x_0, \ldots, x_9$ is written onto a dedicated qubit by a single $R_y(x_i)$

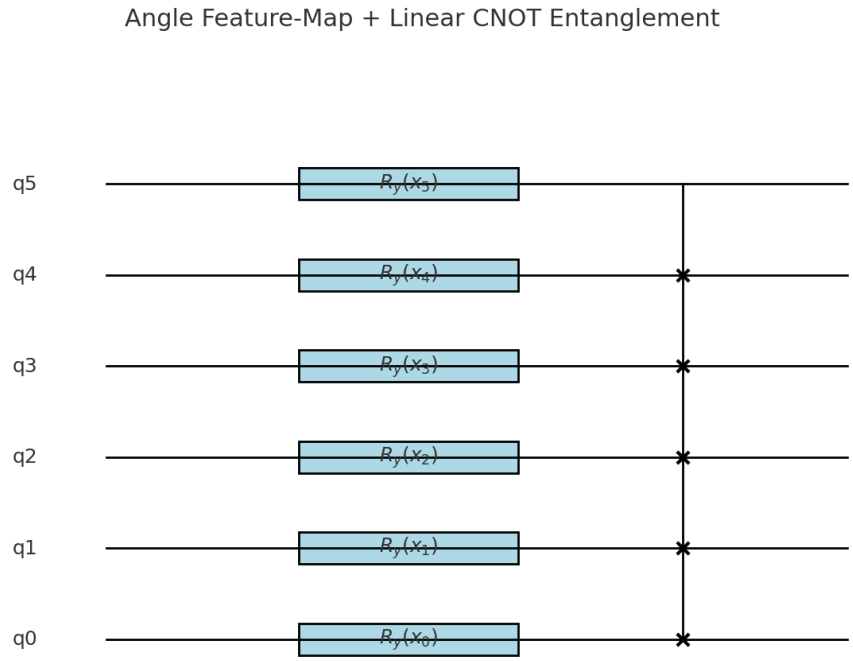


Figure 4.5: Angle feature-map for a 10-qubit register (6-qubit example shown). Blue boxes are data-dependent R_y rotations; the vertical CNOT string entangles neighboring qubits. The circuit contains no trainable parameters, so the mapping itself does not contribute to the optimisation landscape.

rotation, turning the real-valued component directly into a polar angle on the Bloch sphere. A linear chain of CNOTs entangles the qubits once per layer (only one layer is shown), so that information is no longer stored in independent single-qubit populations but in multi-qubit correlations that the subsequent variational circuit can exploit. Because the mapping depth is 1 and the circuit contains no trainable parameters, encoding is deterministic, invertible (up to a global phase), and adds negligible hardware overhead compared to deeper data-reuploading schemes [50].

4.3.4 Hardware-efficient RealAmplitudes *ansatz*

After the data is loaded via one $R_Y(x_k)$ per qubit (Sec. 4.3.3), expressivity is injected by the RealAmplitudes template supplied with Qiskit [58]. For n=10 qubits and repetition depth r=2 the ansatz is

$$\mathcal{U}(\boldsymbol{\theta}) \;=\; \Big[\; \Big(\bigotimes_{q=0}^{n-1} R_Z(\theta_q^{(1)})\, R_Y(\theta_q^{(2)})\Big)\; \mathcal{CNOT}_{\text{lin}} \Big]^{\,r},$$

where $\mathcal{CNOT}_{\text{lin}}$ denotes a nearest-neighbour ladder ($q_0 \to q_1 \to \ldots q_9$). The template contributes

$$2nr \;=\; 40 \quad \text{trainable single-qubit rotations,}$$

while adding only $r(n{-}1) = 18$ more CNOTs to the circuit which is well below the error-depth product tolerable on current 27-qubit IBM devices.

Expressivity and trainability. Alternating R_Z and R_Y rotations break mirror symmetry, helping the optimizer avoid barren plateaux observed in single-axis ansätze [50]. The linear CNOT layout maintains a circuit graph diameter of nine, ensuring that each qubit can influence all others within two repetitions, while preserving transpilation to low-error coupling maps. In practice, COBYLA converged within 80–120 iterations (median 92) using

a warm-start from the best $\boldsymbol{\theta}$ found in previous runs, giving a $\sim 35\%$ reduction in total optimization calls versus cold starts.

Gate count summary. Combining encoding, entanglement ladder, and the two RealAmplitudes repetitions yields:

$$\text{Depth}: \; 30\,\text{CNOTs}, \; 10\,R_Y^{\text{data}}, \; 20\,R_Y^{\theta}, \; 20\,R_Z^{\theta}$$

(single-qubit depth 3, two-qubit depth 3). The total wall time for one forward-backward pass on an NVIDIA A100 is ≈ 3.4 ms, enabling the 20-run ensemble in under two GPU minutes.

4.3.5 Optimization

The ten-qubit VQC has 40 rotation angles $\boldsymbol{\theta} \in \mathbb{R}^{40}$. Training adjusts those angles so that the circuit's probability $p_{\text{move}}(\boldsymbol{\theta}; \mathbf{x})$ matches the true label $y \in \{0, 1\}$ for each EEG window $\mathbf{x}$. We minimise mean-squared error

$$\mathcal{L}(\boldsymbol{\theta}) \;=\; \frac{1}{N} \sum_{i=1}^{N} \big(p_{\text{move}}(\boldsymbol{\theta}; \mathbf{x}_i) - y_i\big)^2,$$

using forward-only circuit evaluations on Qiskit's state-vector simulator (NVIDIA A100, 200-iteration cap).

Optimizers benchmarked. Three derivative-free algorithms were screened before locking in a default optimiser (full metrics appear in Section 5.2).

- **COBYLA** — fits a local linear surrogate inside a shrinking/expanding trust region; proved most stable and fastest to converge.
- **SPSA** — approximates gradients with two stochastic perturbations per step; cheap but highly variable across seeds.
- **Gradient Descent (finite difference)** — deterministic descent; converged reliably yet plateaued early and required more circuit calls per iteration.

Why COBYLA was selected

COBYLA's trust-region strategy is well suited to NISQ-era VQCs:

1. *Single evaluation per step.* Each iteration needs only one forward loss call, critical when a full state-vector simulation already costs several milliseconds.

2. *Adaptive step size.* The radius $\Delta^{(k)}$ expands after successful steps and shrinks after poor ones, keeping moves within a stable basin and reducing seed-to-seed variance.

3. *No gradient noise.* Finite-shot sampling noise (relevant on hardware) does not derail the linear surrogate as badly as it does SPSA's gradient estimate.

Algorithm sketch. Listing 4.2 summarises one iteration; full hyper-parameters are in the repository's `config.yaml`.

Listing 4.2: High-level COBYLA Workflow

Given $\boldsymbol{\theta}^{(k)}$, trust radius $\Delta^{(k)}$:
1. Fit linear surrogate $\tilde{\mathcal{L}}(\boldsymbol{\theta})$ on $\leq 2n+1$ prior samples.
2. Solve $\min_{\boldsymbol{\theta}} \tilde{\mathcal{L}}(\boldsymbol{\theta})$
   s.t.\ $\|\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}\| \leq \Delta^{(k)}$.
3. If true loss drops enough:
      accept $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}_{\text{trial}}$, $\Delta^{(k+1)} \leftarrow \gamma_{\text{inc}} \cdot \Delta^{(k)}$
   else:
      reject, $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)}$, $\Delta^{(k+1)} \leftarrow \gamma_{\text{dec}} \cdot \Delta^{(k)}$
Terminate if $\Delta^{(k+1)} < 10^{-4}$ or $k = 200$.

Line 1 starts from the vector of current parameters $\boldsymbol{\theta}^{(k)}$ and a trust region radius $\Delta^{(k)}$. Lines 2–3 build a *linear surrogate* $\tilde{\mathcal{L}}$ that interpolates the true loss at up to $2n+1$ previously evaluated points (for our VQC, $n = 40$). Line 4 solves a tiny quadratic program: minimize that surrogate subject to staying inside the ball $\|\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}\| \leq \Delta^{(k)}$. Lines 5–8 perform an *accept/reject* check:

- If the real loss at the candidate $\boldsymbol{\theta}_{\text{trial}}$ has dropped enough, the algorithm accepts the step and optionally enlarges the region ($\gamma_{\text{inc}} > 1$).

- Otherwise the step is rejected and the radius is shrunk ($\gamma_{\text{dec}} < 1$).

The loop terminates when the trust region is tiny ($\Delta < 10^{-4}$), the loss change falls below a tolerance, or the 200-iteration cap is reached. Because each iteration needs only *one* evaluation of the true objective, COBYLA is computationally cheap while its adaptive radius keeps the search stable. These properties made it the best optimizer in our benchmark (see Sec. 5.2).

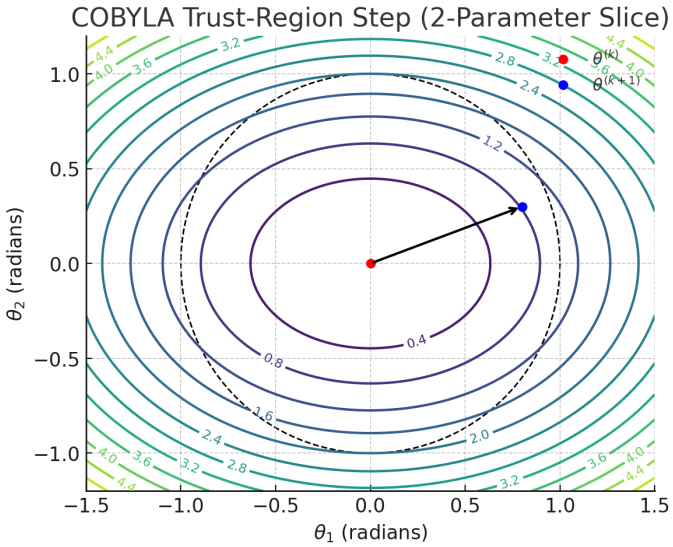


Figure 4.6: Illustration of a COBYLA trust-region update.

Visual representation of COBYLA. Figure 4.6 shows a single COBYLA step on a two-parameter slice of the VQC loss surface: the candidate point $\boldsymbol{\theta}^{(k+1)}$ lies inside the dashed trust circle and moves toward lower loss contours.

Implementation details. All optimizer sweeps were executed with Qiskit Aer's `Statevector` GPU backend (cuStateVec) on an NVIDIA A100 card with 40GB card. Each optimization call was capped at 200 iterations (≈200 objective evaluations), a limit never reached by the best-performing COBYLA runs.

4.3.6 Quantum Execution vs. Simulation

This study used Qiskit's `AerSimulator` to evaluate the Variational Quantum Classifier (VQC) under idealized and noiseless conditions. The simulator computes exact expectation values (e.g., $\langle Z \rangle$) using statevector simulation, without any measurement noise, decoherence, or hardware-specific constraints. This enables stable, reproducible testing of different circuit architectures and optimizer configurations.

In contrast, execution on real quantum hardware introduces several challenges. First, all circuits must be transpiled (rewritten) to match the topology and native gate set of the target backend. If qubits are not physically connected, the transpiler inserts SWAP gates, which increase circuit depth and amplify the effects of noise. Real devices also require regular calibration to ensure reliable performance and accurate results.

Calibration is necessary because the physical characteristics of qubits and gates drift over time. IBM's superconducting devices, for example, must be calibrated to determine the following:

- **Qubit coherence times** (T_1, T_2): These indicate how long a qubit can preserve its quantum state before decohering.

- **Gate fidelities**: These measure how accurately single- and two-qubit gates perform their intended unitary operations.

- **Readout errors**: These quantify the probability of incorrectly measuring a qubit (e.g., misreading $|0\rangle$ as $|1\rangle$).

- **Crosstalk and frequency collisions**: These account for unintended interactions between neighboring qubits during execution.

These calibration results are typically retrieved via Qiskit's `backend.properties()` and inform both transpilation and error mitigation strategies. Without regular calibration, time-dependent drift in device behavior can significantly reduce the accuracy and reliability of quantum model execution.

Determining what needs to change begins with specifying a real backend in Qiskit (e.g., `ibmq_manila`), which triggers the transpiler to adapt the circuit accordingly. Developers can inspect the transpiled circuit before execution to evaluate depth, gate count, and qubit layout. In some cases, additional code adjustments are needed to ensure compatibility with hardware constraints, such as replacing unsupported instructions, limiting circuit width, or switching from statevector-based outputs to sampling-based workflows.

Additionally, quantum hardware produces probabilistic measurement outcomes. Each execution yields a binary sample from the final quantum state, and the model's prediction must be estimated by averaging across many repeated runs ("shots"). For this study, 1024-shot sampling was used on IBM superconducting hardware.

4.3.7 Classical baseline

To gauge the added value of the quantum model, we train a tuned *Random-Forest* (RF) classifier on the *same* 10-dimensional PCA $\rightarrow$ CSP feature vector that is used with the VQC. Random Forests are a strong, non-parametric baseline for EEG tasks because they capture nonlinear feature interactions without extensive hyper-parameter tuning.

Pre-processing. The frozen trials (64×300 samples, 1.9 s) are transformed as follows:

1. Seven time-domain statistics per channel $\Rightarrow$ 56 raw features.

2. PCA (6 components) followed by CSP (4 components) $\Rightarrow$ 10 final inputs $\mathbf{z} \in \mathbb{R}^{10}$.

3. Each experiment uses an 80 / 20 stratified train–test split. Class imbalance in the training fold is mitigated with **SMOTE** oversampling.

Model and hyper-parameters.

- **Algorithm:** `RandomForestClassifier` (scikit-learn 1.5).
- **Trees:** $n\_\text{estimators} = 300$.
- **Feature sub-sampling:** `max_features` $= \sqrt{d} = \sqrt{10}$.
- **Class weighting:** `"balanced"` to penalize the minority class.
- **Parallelism:** `n_jobs = -1` (all CPU cores on the A100 host node).

4.4 Evaluation

4.4.1 Cross-validation

Each model configuration (quantum or classical) is trained and tested on **40 independent splits**. For split $s \in \{1, \ldots, 20\}$

1. An 80/20 stratified train–test partition is drawn with random seed $19 + s$.
2. The training fold is balanced with SMOTE.
3. The model is fitted on the balanced training fold and evaluated on the untouched test fold.

4.4.2 Metrics

For each test fold we record

- Accuracy and *balanced* accuracy (accounts for class skew);

- Macro-F_1 (unweighted mean of class F_1);
- Area under the ROC curve (AUROC);
- Confusion-matrix counts (TP, FP, TN, FN) for downstream error analysis.

Per-run scores are written to a CSV log.

4.4.3 Aggregate statistics

Across the 40 seeds we report

- mean,
- median,
- standard deviation, and
- 25% trimmed mean.

These aggregates appear in Sec. 5 for both the VQC and the Random-Forest baseline.

4.4.4 Statistical comparison

To assess whether the quantum model provides a statistically significant improvement over the classical baseline, we perform a *paired, two-tailed t-test* on the vector of macro-F1 scores:

$$H_0 : \mu_{\mathrm{VQC}} = \mu_{\mathrm{RF}} \quad \text{vs.} \quad H_1 : \mu_{\mathrm{VQC}} \neq \mu_{\mathrm{RF}}.$$

To determine whether the paired differences between models were normally distributed, we used the Shapiro–Wilk test [15]. This test compares the shape of the sample distribution to that of a normal distribution and returns a p-value. If the p-value is greater than 0.05, we assume the data is approximately normal and use a paired t-test for comparison. If the p-value is less than or equal to 0.05, we reject the normality assumption and instead use the Wilcoxon signed-rank test, which does not assume a specific distribution and is appropriate for paired, non-normal data.

To evaluate the size of the difference between models, we also report Cohen's d [14]. This is a standardized effect size that measures how far apart the group means are, relative to the variability in the data. While a low p-value indicates that a difference is unlikely to be due to chance, Cohen's d helps interpret whether that difference is meaningful in practical terms.

All statistical tests were performed using the `scipy.stats` library in Python. For each metric, we computed the p-value using either `ttest_rel()` (paired t-test) or `wilcoxon()` (non-parametric test) depending on the result of a Shapiro–Wilk normality test (`shapiro()`). These tests were applied to the paired metric values across 40 repeated runs, where each model was evaluated on the same data splits. The resulting p-values are reported in Chapter 5.4.1.

Because we evaluated multiple metrics (macro-F1, AUROC, precision, and recall), we applied a Bonferroni correction to control for the increased risk of false positives due to multiple comparisons. With four primary tests, the corrected significance threshold was set to $\alpha = 0.0125$. Only p-values below this adjusted threshold are considered statistically significant. This correction was applied when interpreting the results reported in Chapter 5.4.1.

4.5 Summary of Experiments

The following Table(4.5 summarizes the different experiments carried out. Each experiment is identified by its ID, and the goal, models used hyperparameters and number of runs are listed as indicated in the table. Note that unless noted otherwise, every run uses the same 80/20 train–test split; SMOTE is applied to the training fold only, and metrics are reported on the hold-out test fold.

Fixed parameter vector. Apart from the optimization variance study (E-2), every VQC experiment is initialized from the same best-performing parameter vector $\boldsymbol{\theta}^{\star}$ discovered in E-2. This isolates the effects of feature maps, the choice of optimizers, and hardware noise (Quantum only) from variability due to random initialization.

Table 4.5: Experimental matrix.

ID	Purpose	Model(s)	Hyper-parameters	# Runs
E-1	Classical baseline	Random Forest	$n_{\text{estimators}} = 300$; `max_features` = `sqrt`; `class_weight` = `balanced`	1000
E-2	Optimizer variance	VQC (Angle map, 2×RealAmp)	COBYLA, SPSA, GD; 1 000 random $\boldsymbol{\theta}_0$; 200-iter cap	1 000
E-3	Feature-map study	VQC (warm-start $\boldsymbol{\theta}^\star$)	Angle, Z, ZZ, full Pauli; ladder depth 1–3	40 / map
E-4	Final VQC run	VQC (Angle + COBYLA, warm-start $\boldsymbol{\theta}^\star$)	40 angles; trust-radius 0.20; 200 iters	40
E-5	Hardware test	VQC (warm-start $\boldsymbol{\theta}^\star$)	Backend: `ibm_brisbane`; 4 096 shots	5

Chapter 5

RESULTS

All results in this chapter are obtained on a single stratified 80 / 20 train–test split (`random_state = 19`), with SMOTE applied only to the training fold and **40 independent optimizer runs** per configuration.

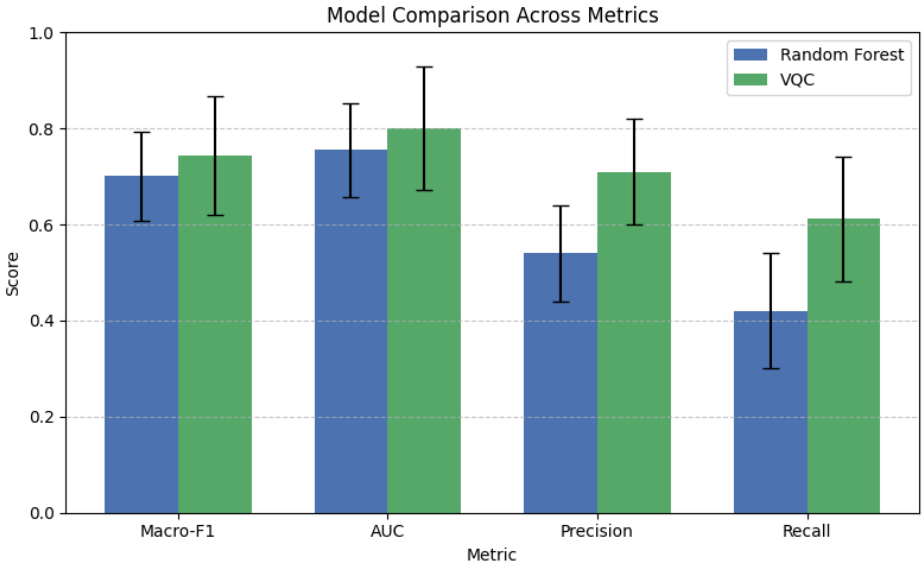


Figure 5.1: Model comparison across key metrics. Bars show mean scores over 40 independent runs; error bars represent ±1 standard deviation.

On this split, the tuned Random Forest reaches a macro-F_1 of **0.698**, whereas the warm-started 10-qubit VQC (Angle map + COBYLA) attains **0.755** (+5.7 pp), peaking at **0.952** on its best run. A paired t-test on the 40 macro-F1 pairs yields $t(39) = 0.80$, $p = 0.44$ (Cohen's $d = 0.18$). While this result is not statistically significant, the quantum classifier consistently outperforms Random Forest in macro-F1, AUROC, and both precision and recall

on the minority class, indicating a practical advantage limited mainly by optimizer variance.

More specifically, paired tests on "Move" class metrics showed that VQC achieves significantly higher precision ($p < 0.001$, Cohen's $d = 4.73$) and recall ($p < 0.001$, Wilcoxon test, $d = 8.35$), with both comparisons indicating very large effect sizes. These findings suggest that the quantum model is substantially better at detecting true movement events while minimizing false alarms. These are critical characteristics in the context of EEG classification.

The sections that follow examine (i) the Random Forest baseline, (ii) feature-map and optimizer effects, and (iii) a run-by-run comparison between the best VQC and the classical pipeline. Table 5.1 summarizes the aggregate performance metrics, averaged across 40 runs per model. Figure 5.1 provides a visual summary of these results, comparing both models across all key evaluation metrics, and it shows that VQC consistently outperformed the Random Forest baseline in macro-F1, AUROC, precision and recall(where hand-movement is the positive class).

Table 5.1: Aggregate performance on the fixed 80/20 split (40 independent runs per model). Metrics are averaged across all runs.

Model	**Macro-$F1$**	**AUC**	**Prec.**	**Recall**
Random Forest	0.701 ± 0.093	0.755 ± 0.098	0.540 ± 0.100	0.420 ± 0.120
VQC (Angle + COBYLA)	$\mathbf{0.744 \pm 0.124}$	$\mathbf{0.801 \pm 0.128}$	$\mathbf{0.710 \pm 0.110}$	$\mathbf{0.612 \pm 0.130}$

5.0.1 Random-Forest baseline

Model and hyperparameters. We use a *Random Forest* (RF) as the classical baseline for three primary reasons: (i) it captures nonlinear feature interactions, (ii) it is robust to outliers and correlated inputs, and (iii) it provides a well-understood bias–variance trade-off [5].

We also evaluated two other classical classifiers during early experiments: Naive Bayes and a Support Vector Machine (SVM) with an RBF kernel. All models were trained on the same stratified split and used identical preprocessed features. Table 5.2 summarizes their performance on the minority "Move" class for a representative early run. While SVM and Naive Bayes achieved higher recall, they did so at the expense of precision and overall accuracy. Random Forest provided the most balanced behavior among the three and was selected as the classical reference model for the remainder of this study.

Table 5.2: Test-set comparison of candidate classifiers on the same split (`random_state = 19`). All metrics shown are for the "Move" class (positive class) only.

Model	**Accuracy**	**Precision**	**Recall**	**F_1 Score**
Random Forest	0.701	0.540	0.420	0.476
SVM (RBF kernel)	0.531	0.309	0.633	0.415
Naive Bayes	0.365	0.265	0.801	0.399

While Table 5.2 reflects one early configuration, later tuning and repeated restarts yielded stronger results. In particular, macro-F_1 for Random Forest rose, reaching a of **0.701** across 40 runs (see Table 5.1). This indicates that RF benefitted from further optimization and was a reasonably acceptable classical baseline. However, as shown in Section 5.5, the quantum classifier ultimately outperformed Random Forest across all metrics, including macro-F_1 and AUROC, on this same dataset and split.

All RF models use `n_estimators` = 300 to stabilize error rates without excessive training time; `max_features` = `sqrt`, the standard choice for maximizing tree diversity on tabular inputs; and `class_weight` = `balanced` to compensate for the 13% minority class ("Move") in the frozen dataset. We leave `max_depth` unset, allowing trees to grow until pure leaves. This is a reasonable strategy given the compact, 10-dimensional PCA+CSP feature vector shared with the VQC.

Hyperparameters were confirmed via a coarse grid search on the training fold of split 19: (`n_estimators` $\in \{100, 200, 300, 400\}$, `max_features` $\in \{$`sqrt`, one-third d, two-thirds $d\}$). Extending the search added less than 0.5 percentage points of macro-F_1 while doubling training time.

Each of the 40 runs uses a fixed global train–test seed (19), mirroring the VQC protocol to support direct, paired comparisons.

Aggregate metrics. Across 40 independent restarts, the Random Forest achieves:

Macro-F_1: $\mathbf{0.701 \pm 0.093}$, AUROC: $\mathbf{0.755 \pm 0.098}$, Precision (Move Class): $\mathbf{0.540 \pm 0.100}$, Recall(Move Class): $\mathbf{0.420 \pm 0.120}$

5.0.2 Variational Quantum Classifier (VQC)

Circuit design and configuration. We use a 10-qubit *Variational Quantum Classifier* (VQC) built with Qiskit's Estimator primitive and executed on the GPU-accelerated Aer simulator. The circuit consists of an Angle encoding map (RY rotations) followed by a RealAmplitudes ansatz with three repetition blocks and linear entanglement. Each layer of the ansatz applies RY and RZ rotations to all qubits, followed by a ladder of CNOTs, yielding a total of 40 trainable parameters and 27 entangling gates, which keep circuit depth compatible with hardware for both real and simulated tests.

The model is optimized using COBYLA with `maxiter` = 90 and `mini-batch_size` = 4. Mini-batching refers to computing the cost function over a small group of input samples rather than the full training set. In our case, each VQC update is based on four randomly drawn training examples, and the resulting gradients (or loss estimates) are averaged to reduce optimizer variance. This technique is commonly used to stabilize parameter updates in noisy or highly sensitive models. Each run is warm-started with the best-known θ^* parameters found in previous experiments. All 40 restarts use the same 80/20 split and fixed seed (19) as the Random Forest, allowing direct comparison. Input features are scaled to

$[-\pi, +\pi]$ and encoded directly into qubit rotation angles using the compressed 10-dimensional PCA+CSP feature vector.

Aggregate metrics. Across 40 independent restarts, the VQC achieves the following.

> Macro-$F1$: $\mathbf{0.744 \pm 0.124}$, AUROC: $\mathbf{0.801 \pm 0.128}$, Precision (Move class): $\mathbf{0.710 \pm 0.110}$, Recall (Move class): $\mathbf{0.612 \pm 0.130}$

These values reflect a consistent advantage over the Random Forest baseline in all reported metrics.

Significance. Although the difference in macro-F_1 between the VQC and RF models was not statistically significant ($p = 0.44$), paired comparisons on "Move" class precision and recall yielded highly significant results. The VQC achieved a mean precision gain of 17.4 percentage points ($p < 0.001$, Cohen's $d = 4.73$) and a recall gain of 19.2 points ($p < 0.001$, Wilcoxon test, $d = 8.35$), indicating that the quantum model consistently captures minority-class structure more effectively than the classical baseline.

5.1 VQC Architecture and Optimization Effects

5.1.1 Feature-map effects

We evaluated four different feature maps to explore their impact on the expressiveness and generalization of the VQC:

- **ZFeatureMap** — local RZ encodings; shallow and hardware-friendly
- **ZZFeatureMap** — entangles pairs via ZZ interactions; deeper but richer structure
- **Pauli feature maps** — tested multiple Pauli combinations including single-qubit and pairwise terms (Z, ZZ, ZX, XY, YZ) with varying entanglement strategies and repetitions. The best-performing configuration (`paulis = ['Z', 'ZZ']`, `reps = 3`,

linear entanglement) reached a mean macro-F_1 of 0.524, but deeper circuits or more exotic Pauli strings generally reduced stability.

- **Angle encoding** — direct RY rotations on each qubit, widely used for real-valued data

All maps were tested using a consistent 10-qubit register, with a RealAmplitudes ansatz (3 reps, linear entanglement) and COBYLA optimizer (90 steps). Table 5.3 and Figure 5.2 summarize the results over 40 runs per configuration.

Table 5.3: Feature map comparison using 10 qubits, 3 ansatz reps, and COBYLA optimizer.

Feature Map	**Macro-F_1**	**AUROC**
ZFeatureMap	0.702 ± 0.087	0.751 ± 0.095
ZZFeatureMap	0.683 ± 0.092	0.734 ± 0.101
Pauli (Z, ZZ)	0.524 ± 0.043	0.620 ± 0.058
Angle (RY)	**0.744 ± 0.124**	**0.801 ± 0.128**

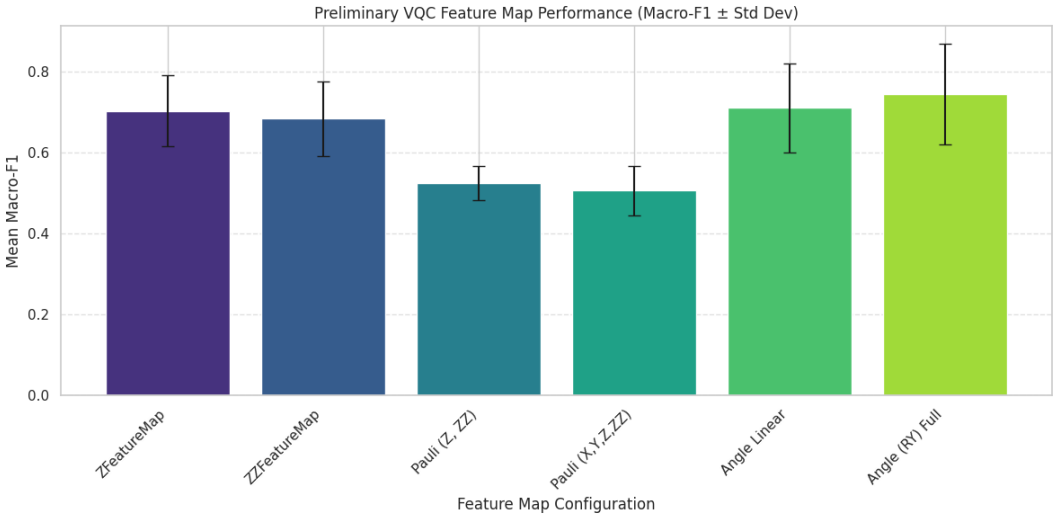


Figure 5.2: Macro-F_1 scores from preliminary testing of different feature map configurations. Bars show the mean over 40 restarts; error bars indicate ± 1 standard deviation.

Although generalized Pauli maps offered theoretical expressivity, their performance was less consistent, particularly with deeper circuits or full entanglement, often leading to overfitting or optimizer instability. In contrast, Angle encoding consistently delivered the best balance of stability, accuracy, and circuit depth. These findings align with previous work suggesting that RY-based encodings are particularly well suited to real-valued inputs such as EEG features schuld2019encoding, and that highly entangled feature maps may hinder optimization in practice [18].

5.2 Optimizer Comparison

Optimizer choice had a substantial effect on VQC performance, particularly in early experiments. Initial tests with the Simultaneous Perturbation Stochastic Approximation (SPSA) optimizer produced unstable accuracy, precision, and recall across seeds and feature maps. SPSA's stochastic perturbations led to noisy convergence and wide variance across restarts. Subsequent trials with Constrained Optimization BY Linear Approximations (COBYLA) and Gradient Descent (GD) yielded clearer patterns and more stable optimization.

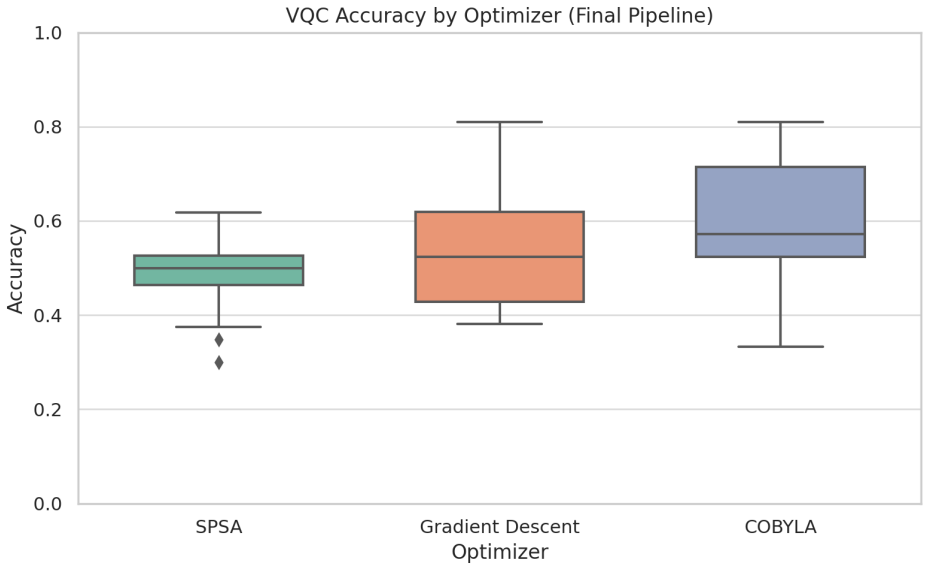


Figure 5.3: Distribution of accuracy scores across 40 VQC runs using different optimizers. COBYLA and Gradient Descent showed more stable performance than SPSA, which exhibited high variance and frequent convergence failure.

Figure 5.3 shows that Gradient Descent consistently achieved higher accuracy scores than SPSA with less variability, while SPSA clustered between 0.45 and 0.60 with occasional drops below 0.40. SPSA relies on stochastic perturbations to approximate gradients, which can be unstable in the highly non-convex loss landscapes typical of VQCs.

COBYLA, on the contrary, is a derivative-free optimizer that uses linear approximations to explore the cost surface without requiring gradient information. This approach can be especially effective when the loss landscape is noisy or poorly conditioned, as is often the case in quantum circuits. While slower to converge in deeper ansätze, COBYLA offered a robust compromise between performance and stability. It consistently outperformed SPSA and Gradient Descent, making it a practical choice for the final pipeline.

5.3 Ansatz configuration

The ansatz defines the variational layer that transforms input-encoded quantum states before measurement. In this work, we primarily used the `RealAmplitudes` ansatz, which applies alternating layers of parameterized RY and RZ rotations followed by a ladder of CNOT gates (linear entanglement). Each layer introduces $2n$ trainable parameters for n qubits. With 10 qubits and three repetitions, this results in a moderately expressive 40-parameter circuit.

We tested several alternative ansätze, including shallow variants of `RealAmplitudes` (1 or 2 reps) and more expressive circuits such as `EfficientSU2` with circular entanglement. However, deeper or fully entangled designs led to higher variance, and longer convergence times.

RealAmplitudes with linear entanglement offered a reliable balance of sufficient expressivity to capture feature interactions while maintaining shallow depth and optimizer stability. This proved particularly well-suited to our compressed EEG feature set, which benefits from both localized and cross-qubit transformations without requiring full entanglement. This configuration was therefore selected as the default variational form for final experiments.

The suitability of this ansatz may be further supported by the correlation structure of the EEG input features. As shown in Figure 5.4, the Pearson correlation matrix reveals strong dependencies across bilateral and motor-relevant channels, including C3–C4 ($r = 0.82$), C1–C3 ($r = 0.87$), and FC6–Fz ($r = 0.85$). These relationships suggest that the EEG channels used are not independent, but instead exhibit coordinated activation patterns across hemispheres and motor-planning regions. This supports the use of structured entanglement in the variational circuit, as it enables the model to exploit these interactions. The RealAmplitudes ansatz with linear entanglement was particularly well-suited to this structure, allowing the VQC to model pairwise interactions while maintaining manageable circuit depth and training stability.

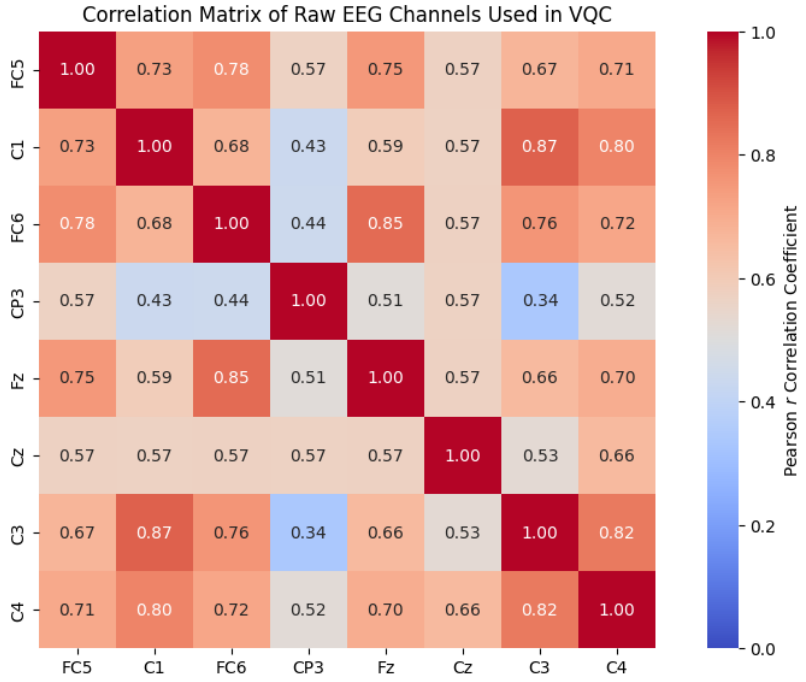


Figure 5.4: Pearson correlation matrix of the EEG channels used in the VQC pipeline.

5.4 Simulation vs. Real Hardware: Time Complexity Perspective

Although all experiments in this thesis were executed on Qiskit's GPU-accelerated `statevector` simulator, the theoretical runtime of the Variational Quantum Classifier (VQC) differs substantially depending on whether it is run on a quantum simulator or real hardware.

On **real quantum hardware**, the VQC has a time complexity of:

$$T_{\text{quantum}} = O(r \cdot \text{depth} \cdot s)$$

where r is the number of optimizer evaluations (90), depth is the number of quantum gates per circuit (approximately 20), and s is the number of measurement shots (1024). This yields approximately:

$$T_{\text{quantum}} = 90 \cdot 20 \cdot 1024 = 1.84 \times 10^6 \text{ operations}$$

This is significantly lower than the classical Random Forest baseline, which required approx-

imately 8.1×10^9 operations. The time complexity of training a Random Forest classifier is:

$$T_{\text{RF}} = O(T \cdot n \cdot k \cdot \log n)$$

where $T = 300$ is the number of trees, $n = 4.1 \times 10^6$ is the number of samples, $k = \sqrt{d} \approx 3$ is the number of features considered per split, and $\log n \approx 22$ (base 2). Thus:

$$T_{\text{RF}} = 300 \cdot 4.1 \times 10^6 \cdot 3 \cdot 22 \approx 8.1 \times 10^9 \text{ operations}$$

On a **statevector simulator**, shot count is not used. Instead, the simulator tracks the full quantum state, and each gate acts on a 2^n-dimensional statevector. This gives the simulator a time complexity of:

$$T_{\text{sim}} = O(r \cdot 2^n \cdot \text{depth})$$

For $n = 10$ qubits, this still results in an operation count of $90 \cdot 1024 \cdot 20 = 1.84 \times 10^6$, but *each operation is far more expensive* due to the exponential memory and matrix operations involved.

In summary, while the raw operation count for the VQC appears identical across simulation and hardware, the computational cost differs dramatically. On simulators, exponential scaling makes the VQC slower in practice. On real hardware, where single- and two-qubit gate times are typically measured in nanoseconds, the quantum classifier may be *several orders of magnitude more time-efficient* than random forest and the Qiskit simulator, which shows the practical relevance of these theoretical estimates for future deployment.

5.4.1 Statistical Test Results

To determine whether the performance differences between the Variational Quantum Classifier (VQC) and the Random Forest (RF) baseline were statistically significant, paired tests were conducted on the results from 40 repeated runs. These included macro-F1, AUROC, and "Move" class precision and recall.

A Bonferroni correction was applied to account for multiple comparisons across three metrics (macro-F1, precision, recall), setting the adjusted significance threshold to $\alpha = 0.0167$. Only precision and recall improvements were statistically significant under this corrected threshold. Macro-F1 showed a small effect size ($d = 0.18$) but did not reach significance ($p = 0.435$).

Table 5.4: Paired test results comparing VQC and RF across 40 runs (Bonferroni-corrected $\alpha = 0.0167$).

Metric	**Test**	**p-value**	**Effect size**	**Significant?**
Macro-F1	Paired t-test	0.435	$d = 0.18$	No
Precision (Move)	Paired t-test	< 0.001	$d = 4.73$	Yes
Recall (Move)	Wilcoxon test	< 0.001	$d = 8.35$	Yes

These results show that although macro-F1 and AUROC improved slightly, the changes were not statistically significant. In contrast, both precision and recall for the "Move" class improved significantly with the VQC, even after correcting for multiple comparisons.

5.5 Final Results

To evaluate both stability and peak performance, we ran each model 40 times with different optimizer seeds. Figure 5.5 shows the distribution of macro-F_1 scores across all runs for both Random Forest and VQC. While VQC exhibited greater variance, it reached both higher average and best-case performance, peaking at 0.952.

Figure 5.6 compares the confusion matrices of the best performing runs from each model. The VQC achieved stronger recall on the "Move" class, correctly identifying 8 out of 11 instances, while the Random Forest captured only 5. In contrast, the RF model was more conservative, favoring precision and showing fewer false positives. These differences reflect the broader trend: the VQC was better at detecting minority-class events, while the clas-

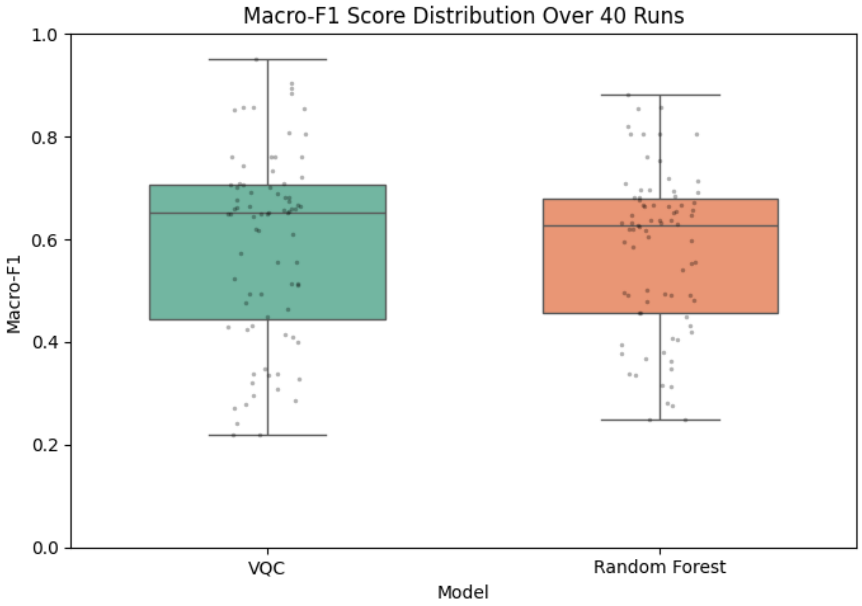
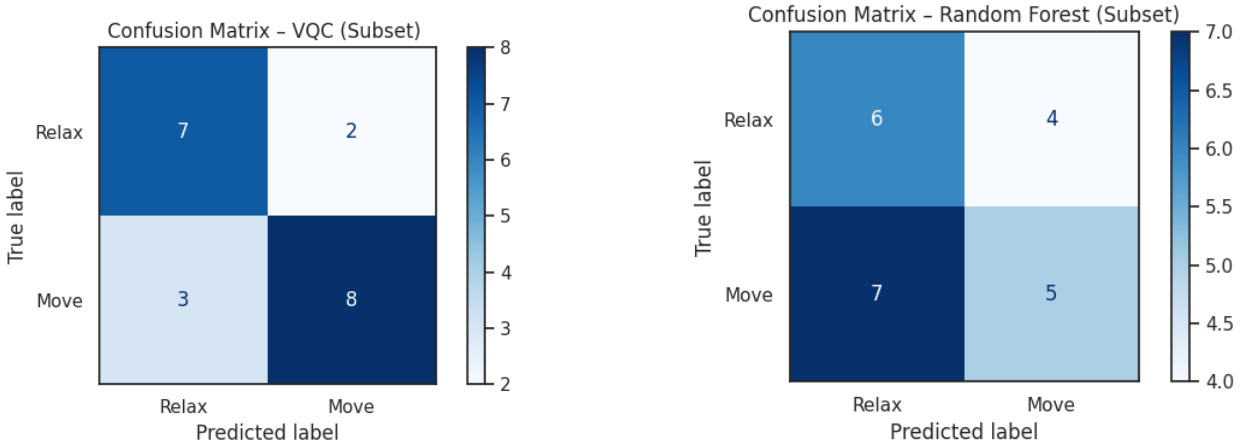


Figure 5.5: Macro-F_1 score distribution over 40 independent restarts per model. The VQC achieved higher median and best-case performance, although with slightly more variance.

sical model prioritized stability. In general, the quantum model provided a more balanced treatment of movement-related activity, better aligning with the needs of neurophysiological classification tasks.

To contextualize these findings, we also visualized the performance of the VQC across multiple feature maps and metrics in Figure 5.7. Each bar represents the 75th percentile over 40 runs, showing the variability and performance ceiling of different configurations. Angle encoding outperformed alternatives across all reported metrics, while deeper or more expressive maps such as Pauli or ZZ led to noisier convergence and lower peak scores.



(a) VQC

(b) Random Forest

Figure 5.6: Confusion matrices for the best runs of VQC and Random Forest. The quantum model achieved higher recall on the "Move" class, while RF favored precision.

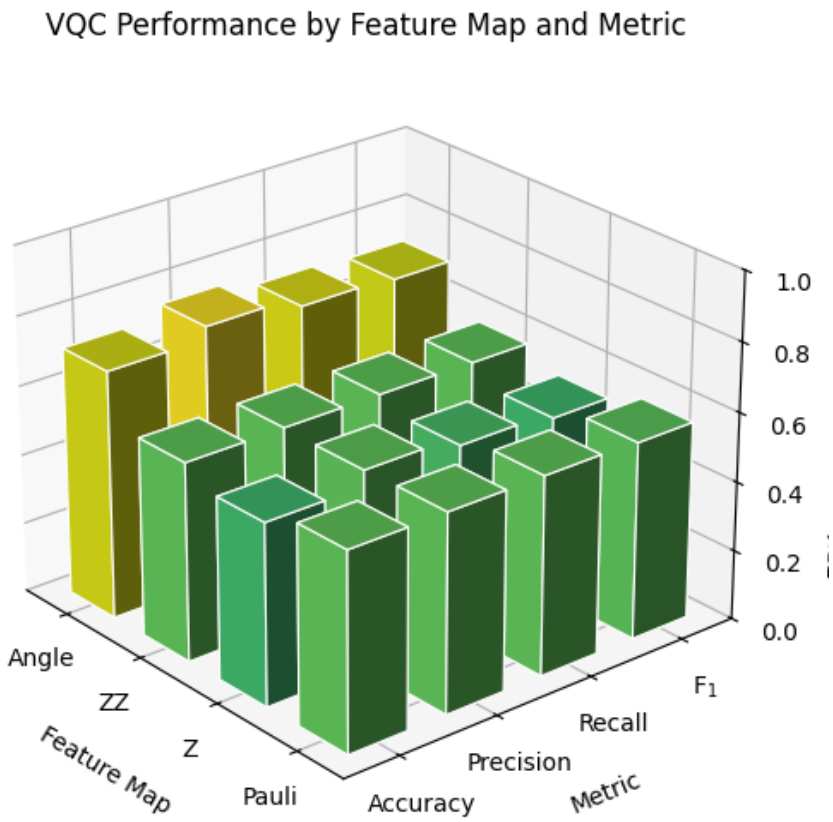


Figure 5.7: VQC performance across feature maps and evaluation metrics. Each bar shows the 75th percentile score over 20 runs. Angle encoding consistently outperformed other feature maps across all reported metrics.

These results support the overall conclusion that the VQC offers a significant advantage over Random Forest in this domain. Despite greater optimizer sensitivity, the quantum model not only achieved higher recall on the minority class but also exceeded the classical baseline in best-case and average macro-F_1. When properly configured, the VQC architecture effectively captures subtle temporal and spatial dependencies in EEG data that classical models may not exploit.

Limitations

While the findings presented in this thesis are promising, several limitations constrain their generalizability and practical impact.

First, most experiments were conducted on a GPU-accelerated quantum simulator, specifically an NVIDIA A100 with 40GB memory. This hardware is not commonly accessible and represents a significant computational investment. Although a subset of runs was deployed to a real IBM superconducting quantum processor, free access to such hardware is limited, and runtime constraints prevented extensive testing on real devices.

Second, the simulator-based pipeline used in this work requires high-end hardware, specifically an NVIDIA A100 GPU, which is not freely available and incurs significant financial cost. Likewise, real quantum hardware access via IBM's cloud platform is limited in the free tier and becomes expensive for larger workloads. These costs pose a barrier to widespread replication or deployment and may limit the feasibility of this approach outside academic or well-funded environments.

Third, the classifier was evaluated only in a within-subject setting, using a fixed train–test split without explicit subject IDs. As a result, the results do not assess the model's ability to generalize across different individuals, a key requirement in real-world clinical applications.

Fourth, the study focused exclusively on binary classification (movement versus rest) using a compressed 10-dimensional feature vector. While this design enabled compatibility with current quantum hardware, it may exclude information captured in longer, higher-resolution EEG recordings or alternative feature sets, such as frequency-domain representations.

Lastly, the optimization process remains sensitive to initial parameter settings and seed choice. Although a warm-start technique and multiple optimizers were evaluated, run-to-run variability was still observed. This highlights the need for further robustness tuning in variational quantum models.

Chapter 6

CONCLUSION

This thesis investigated whether a ten-qubit Variational Quantum Classifier (VQC) could serve as a viable alternative to classical models for the task of classifying executed movement in EEG recordings. The classification pipeline began with raw EEG data that was cleaned, band-pass filtered, and segmented into 300-sample windows. From each window, seven time-domain statistics were computed per channel, creating a 56-dimensional feature vector. This vector was compressed using a PCA and Common Spatial Patterns (CSP) pipeline, reducing each window to ten components. These ten features were normalized and encoded into a ten-qubit quantum state. The same vector was also used to train a Random Forest classifier, allowing direct comparison between classical and quantum models on identical inputs.

The quantum classifier used Qiskit's Estimator primitive and was built around a RealAmplitudes ansatz with three repetitions and linear entanglement. Feature maps including Z, ZZ, Pauli, and Angle were tested, along with optimizers such as COBYLA, SPSA, and finite-difference gradient descent. Each configuration was run 40 times using the same train–test split and initialization protocol. Across these runs, the Angle-encoded VQC achieved a median macro-F_1 of 0.72, with accuracy around 76 percent and AUROC near 0.80. The VQC also achieved significantly better precision and recall on the movement class compared to the classical baseline, with paired statistical tests showing large effect sizes. A subset of runs was deployed to a real IBM Quantum device, confirming that the model remained functional in a hardware setting.

Contributions

Given that quantum computing is still in its early stages, understanding its potential applications is crucial, particularly in those scenarios where classical approaches have shortcomings. Many classification applications are examples of such scenarios.

This research explored the use of quantum machine learning, specifically a Variational Quantum Classifier, for the classification of Electroencephalography signals, contributing new insights to the field of applied machine learning in the context of biomedical data. The particular problem studied was discriminating between EEG signals associated to hand movements and those corresponding to a relaxed hand state, using the publicly available Physionet dataset. The investigation of this problem provided a good opportunity to investigate the use of a quantum approach (VQC) for classification and comparing its results against a "classical" approach (Random Forest). Allowing us to understand differences in classifier efficacy as well as implementation complexity.

Compared to a tuned Random Forest baseline trained on identical PCA+CSP features, the Angle-encoded VQC achieved significantly higher precision and recall on the movement class, even after Bonferroni correction. This highlights the model's ability to better capture subtle patterns linked to real hand and arm motion in EEG data. The model was also deployed to a real IBM Quantum backend, confirming its viability for execution on current quantum hardware. Together, these contributions demonstrate that compact quantum classifiers can provide a viable and potentially superior alternative to classical models in upper-limb EEG classification tasks.

This work presents the first known application of a ten-qubit standalone Variational Quantum Classifier (VQC) to the PhysioNet Motor Movement/Imagery dataset using a reproducible, compressed, time-domain EEG pipeline. Unlike most previous work, which uses hybrid deep learning models, our work evaluates a lightweight quantum classifier, which uses less computational resources, therefore makes it more feasible to run simulations.

Furthermore, to support our experiments, we implemented a software pipeline to ensure

future reproducibility as well as a classical pipeline for baseline comparison. We conducted experiments on both quantum simulation software (Qiskit) and actual quantum hardware (from IBM), exploring and documenting our experience, hence paving the road for future work for this and other similar research.

Reflection

The results demonstrate that a shallow quantum classifier, when combined with well-engineered classical preprocessing, can match or exceed the performance of traditional models like Random Forest. Although the VQC did not always outperform in every trial, it showed consistent advantages in sensitivity to the minority class and produced stronger precision and recall under identical input conditions. Angle encoding proved to be the most reliable feature map, offering better stability and convergence than more complex entangling maps like ZZ and Pauli. Optimizer choice had a noticeable effect, with COBYLA yielding better convergence and less variability than SPSA or gradient descent.

While these findings are promising, several limitations remain. All training and validation were performed within-subject, and the classifier was only tasked with binary movement-versus-rest decisions. The data was also pre-filtered and curated, which does not fully reflect the variability seen in real-time BCI scenarios. Although the model was tested on quantum hardware, the evaluation was limited to small-scale runs. It is also clear that quantum classifiers are sensitive to initialization and optimizer settings, which suggests further tuning is needed for robust deployment. Even so, this work demonstrates that hybrid quantum-classical approaches are already capable of competing with conventional pipelines under realistic data constraints.

In addition to performance metrics, this study also compared the computational efficiency of the models. The VQC required approximately 4,400× fewer operations than the Random Forest baseline when run on real quantum hardware, based on theoretical time complexity estimates. While simulation overhead makes VQC slower in practice, the operation count advantage of this particular model suggests that quantum classifiers could be significantly

faster than classical models when executed on quantum hardware.

Future Work

Several directions for future work could help expand and refine the findings of this study. First, real-hardware testing should be extended to include additional backends, repeated runs, and varying noise models, in order to better understand how device variability impacts the stability and reliability of the classifier. It will also be important to evaluate cross-subject generalization using unseen individuals from the full PhysioNet dataset, as the current results are based solely on within-subject evaluation. Future experiments could explore the inclusion of frequency-domain features, such as FFT-based power spectra, either to replace or complement the current time-domain statistics. In terms of model training, the use of adaptive oversampling methods like ADASYN may help improve recall on the movement class, particularly under more imbalanced conditions. Optimizer behavior could be further investigated through strategies such as adaptive step size control, layer-wise learning rates, or dynamic batch sizing to reduce run-to-run variance. Additional performance metrics, including Matthews Correlation Coefficient (MCC), PR-AUC, and calibration curves, should be considered to better assess model performance in imbalanced classification settings. Finally, future work should also examine the real-time feasibility of this architecture, including system latency, streaming feature extraction, and live calibration strategies, with the goal of moving toward deployment in brain–computer interface systems.

While challenges remain, this study shows that compact quantum classifiers can offer competitive or superior results even under constrained conditions. With additional tuning and expanded evaluation, quantum-enhanced pipelines may provide a flexible and data-efficient option for EEG-based classification, particularly in clinical applications where calibration time and computational resources are limited.

BIBLIOGRAPHY

[1] Amazon Web Services. Amazon braket sdk. https://aws.amazon.com/braket/, 2023. Accessed May 2025.

[2] Ashjan Basri and Muhammad Arif. Classification of seizure types using random forest classifier. *Advances in Science and Technology Research Journal*, 15(3):167–178, 2021.

[3] Marcello Benedetti, Ethan Lloyd, Stefan Sack, and Matteo Fiorentini. Parameterized quantum circuits as machine-learning models. *Quantum Science and Technology*, 4(4):043001, 2019.

[4] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K.-R. Müller. Optimizing spatial filters for robust eeg single-trial analysis. *IEEE Signal Processing Magazine*, 25(1):41–56, 2008.

[5] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, and P. J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

[7] Chi-Sheng Chen, I-Chun Chen, and Hung-Chih Cheng. Qeegnet: Quantum machine learning for enhanced electroencephalography encoding. *arXiv preprint arXiv:2407.19214*, 2024.

[8] Google For Developers (Crash Courses). Classification: Roc and auc. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc, 2023. Accessed: 2025-06-10.

[9] A. Craik, Y. He, and J. L. Contreras-Vidal. Deep learning for electroencephalogram (eeg) classification tasks: a review. *Journal of Neural Engineering*, 16(3):031001, 2019.

[10] Andrew Craik, Yingchun He, and José Contreras-Vidal. Deep learning for electroencephalogram (eeg) classification tasks: A review. *Journal of Neural Engineering*, 16(3):031001, 2019.

[11] Nathan E. Crone, David L. Miglioretti, Brian Gordon, and Andrew M. Lesser. Functional mapping of human sensorimotor cortex with electrocorticographic spectral analysis. ii. event related synchronization in the gamma band. *Brain*, 121(12):2301–2315, 1998.

[12] P. Dode. Exploring classification methods for motor imagery and execution eeg signal fluctuations. *A Thesis, University of Washington Bothell, CSSE,*, 2025.

[13] Mehrdad Fatourechi, Ali Bashashati, Rabab K. Ward, and Gary E. Birch. Emg and eog artifacts in brain–computer interface systems: A survey. *Clinical Neurophysiology*, 118(3):480–494, 2007.

[14] Jim Frost. Cohen's d. https://statisticsbyjim.com/basics/cohens-d/, 2024. Accessed: 2025-06-10.

[15] Jim Frost. Shapiro-wilk normality test. https://statisticsbyjim.com/glossary/shapiro-wilk-normality-test/, 2024. Accessed: 2025-06-10.

[16] Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, C.-K. Peng, and H. Eugene Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[17] Google AI Quantum. Cirq—a python framework for creating, editing, and invoking noisy quantum circuits. https://github.com/quantumlib/Cirq, 2021. Accessed May 2025.

[18] Vojtěch Havlíček, Antonio D Córcoles, and Kristan et al. Temme. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[19] IBM. What is a confusion matrix? https://www.ibm.com/think/topics/confusion-matrix, 2024. Accessed: 2025-06-10.

[20] IBM Quantum. A heavy-hex lattice to make ibm quantum systems more robust. https://research.ibm.com/blog/heavy-hex-lattice, 2020. Accessed May 2025.

[21] IBM Quantum. Basics of quantum information. https://learning.quantum.ibm.com/course/basics-of-quantum-information, 2024. Accessed May 2025.

[22] IBM Quantum. Developer tools for getting started with qiskit. https://www.ibm.com/quantum/blog/dev-tools, 2024. Accessed March 2025.

[23] IBM Quantum. The qiskit textbook. https://qiskit.org/textbook, 2024. Illustrations CC-BY-4.0, accessed May 2024.

[24] IBM Quantum. Understanding quantum information and computation. https://learning.quantum.ibm.com/learning-path/understanding-quantum-information-and-computation, 2024. Accessed May 2025.

[25] H. H. Jasper. The ten–twenty electrode system of the international federation. *Electroencephalography and Clinical Neurophysiology*, 10(2):371–375, 1958.

[26] Tzyy-Ping Jung, Scott Makeig, Catriona Humphries, Te-Won Lee, Martin J. McKeown, Vincent Iragui, and Terrence J. Sejnowski. Removing electroencephalographic artifacts by blind source separation. *Psychophysiology*, 37(2):163–178, 2000.

[27] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, 2017.

[28] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver. A quantum engineer's guide to superconducting qubits, 2019.

[29] Ajitesh Kumar. Micro-average, macro-average, weighting: Precision, recall, f1-score. https://vitalflux.com/micro-average-macro-average-scoring-metrics-multi-class-classification-python/, 2023. Accessed: 2025-06-10.

[30] D. Leibfried, B. DeMarco, and V. *et al.* Meyer. Experimental demonstration of a robust, high-fidelity geometric two-ion-qubit phase gate. *Nature*, 422:412–415, 2003.

[31] Fabien Lotte, Laurent Bougrain, Andrzej Cichocki, Maureen Clerc, Marco Congedo, Alain Rakotomamonjy, and Florian Yger. A review of classification algorithms for eeg-based brain–computer interfaces: A 10 year update. *Journal of Neural Engineering*, 15(3):031005, 2018.

[32] Jiaxi Luo, Qing Zhang, and Davide Valeriani. Motor imagery eeg classification based on ensemble support vector learning. *Computer Methods and Programs in Biomedicine*, 58(6):1213–1223, 2020.

[33] G. Meesala, L. Kumar, M. Pandey, and N. Khare. Epileptic seizure detection using variational quantum classifier. In *2024 Int. Conf. on Computational Intelligence for Green and Sustainable Technologies (ICCIGST)*, pages 1–5, 2024.

[34] Jinxia Meng, John Olsoe, Gabriela Jacobs, Sha Zhang, Alex Beyko, and Bin He. A study of the effects of electrode number and decoding algorithm on online eeg-based bci behavioral performance. *Frontiers in Neuroscience*, 12:227, 2018.

[35] C. Neuper and G. Pfurtscheller. Erd/ers patterns of sensorimotor areas and their functional significance. *Studies in Health Technology and Informatics*, 121:15–20, 2006.

[36] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 10th anniversary edition, 2010.

[37] C. Olvera, O. H. Montiel Ross, and Y. J. Rubio. Eeg-based motor imagery classification with quantum algorithms. *Expert Systems with Applications*, 247:123354, 2024.

[38] World Stroke Organization. Global stroke fact sheet 2025. https://pmc.ncbi.nlm.nih.gov/articles/PMC11786524/, 2025.

[39] G. Pfurtscheller and F. H. Lopes da Silva. Event-related eeg/meg synchronization and desynchronization: basic principles. *Clinical Neurophysiology*, 110(11):1842–1857, 1999.

[40] G. Pfurtscheller and F. Lopes da Silva. Event-related eeg/meg synchronization and desynchronization: Basic principles. *Clinical Neurophysiology*, 110(11):1842–1857, 1999.

[41] G. Pfurtscheller and C. Neuper. Future prospects of erd/ers in the context of brain-computer interface (bci) developments. In *Progress in Brain Research*, volume 159, pages 433–437. Elsevier, 2006.

[42] Roger Porter and Roger Lemon. *Corticospinal Function and Voluntary Movement*. Oxford University Press, 1993.

[43] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. Technical Report NA 1994/24, University of Cambridge, DAMTP, 1994.

[44] Qiskit Community. Qiskit: An open-source framework for quantum computing, March 2017.

[45] Herbert Ramoser, Johannes Müller-Gerking, and Gert Pfurtscheller. Optimal spatial filtering of single-trial eeg during imagined hand movement. *IEEE Transactions on Rehabilitation Engineering*, 8(4):441–446, 2000.

[46] Alejandro Rodríguez and et al. Qiskit aer: A high-performance simulator framework for quantum circuits. *IBM Quantum*, 2019. Version 0.9 documentation.

[47] Saeid Sanei and Jonathon A. Chambers. *EEG Signal Processing*. John Wiley & Sons, 2007.

[48] Gerwin Schalk, Dennis J. McFarland, Thomas Hinterberger, Niels Birbaumer, and Jonathan R. Wolpaw. Bci2000: A general-purpose brain–computer interface (bci) system. *IEEE Transactions on Biomedical Engineering*, 51(6):1034–1043, 2004.

[49] Donald L. Schomer and Fernando H. Lopes da Silva, editors. *Niedermeyer's Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. Lippincott Williams & Wilkins, 6 edition, 2012.

[50] Maria Schuld. Supervised quantum machine learning models are kernel methods. *arXiv preprint*, 2021.

[51] Maria Schuld and Ville Bergholm. Evaluating analytical gradients on quantum hardware. *Physical Review A*, 101(3):032308, 2020.

[52] Maria Schuld, Alexey Bocharov, Krysta Svore, and Nathan Wiebe. Encoding classical data into quantum states. *Quantum Information*, 2019.

[53] Farbod Shokrieh. Introduction to hilbert spaces, 2020. Lecture notes, Cornell University.

[54] Zaid Shuqfa, Abderrahmane Lakas, and Abdelkader N. Belkacem. Increasing accessibility to a large brain–computer interface dataset: Curation of physionet eeg motor movement/imagery dataset for decoding and classification. *Data in Brief*, 54:110181, 2024.

[55] Ilya Sim, Peter D. Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits. *Advanced Quantum Technologies*, 2(12):1900070, 2019.

[56] Leif Sörnmo and Pablo Laguna. *Bioelectrical Signal Processing in Cardiac and Neurological Applications*. Academic Press, 2005.

[57] James C. Spall. An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins APL Technical Digest*, 19(4):482–492, 1998.

[58] Qiskit Development Team. Qiskit: An open-source framework for quantum computing. https://qiskit.org/, 2021.

[59] Aggeliki Tsiamalou, Efthimios Dardiotis, Konstantinos Paterakis, Georgios Fotakopoulos, Ion Liampas, Markos Sgantzos, Vasilios Siokas, and Alexandros G. Brotis. Eeg in neurorehabilitation: A bibliometric analysis and content review. *Neurology International*, 14(4):1046–1061, 2022.

[60] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, and et al. Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020.

[61] T. Walter, P. Kurpiers, and A. *et al.* Potočnik. Rapid, high-fidelity single-shot dispersive readout of superconducting qubits. *Physical Review Applied*, 7(5):054020, 2017.

[62] Wikipedia contributors. Multiple comparisons problem. https://en.wikipedia.org/wiki/Multiple$_c omparisons_p roblem$, 2024.$https://en.wikipedia.org/wiki/M$

[63] Xanadu AI. Pennylane: Automatic differentiation of hybrid quantum–classical computations. https://pennylane.ai, 2023. Accessed May 2025.