ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ στα
ΠΟΛΥΠΛΟΚΑ ΣΥΣΤΗΜΑΤΑ και ΔΙΚΤΥΑ
ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ
ΤΜΗΜΑ ΒΙΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΓΕΩΛΟΓΙΑΣ
ΤΜΗΜΑ ΟΙΚΟΝΟΜΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ
http://cosynet.auth.gr/

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Τίτλος Εργασίας

Κβαντικά Νευρωνικά Δίκτυα για Εφαρμογές στο Εγγύς Μέλλον

Μαρία Κοφτερού

Εγκρίθηκε από την Τριμελή Εξεταστική Επιτροπή την 17η Δεκεμβρίου 2019.

.....	.....	.....
Ιωάννης Αντωνίου Καθηγητής Α.Π.Θ.	Χαράλαμπος Μπράτσας Ε.Δ.Ι.Π. Α.Π.Θ.	Κωνσταντίνος Χατζησάββας Επιστημονικός Συνεργάτης, Α.Π.Θ.

Επιβλέπων:Ιωάννης Αντωνίου, Καθηγητής Α.Π.Θ.

Δεκέμβριος 2019- Τμήμα Μαθηματικών Α.Π.Θ.

……..
Μαρία Σ. Κοφτερού
Πτυχιούχος Μαθηματικός Α.Π.Θ.

INTER-FACULTY MASTER PROGRAM on
COMPLEX SYSTEMS and NETWORKS
SCHOOL of MATHEMATICS
SCHOOL of BIOLOGY
SCHOOL of GEOLOGY
SCHOOL of ECONOMICS
ARISTOTLE UNIVERSITY of THESSALONIKI
http://cosynet.auth.gr/

Master Thesis

Quantum Neural Networks focused on Near-Term applications

Maria Kofterou

Supervisor: Ioannis Antoniou

This dissertation is submitted for the Master Degree on the MSc. Complex Systems and Networks

December 2019- Department of Mathematics

……..
Maria S. Kofterou
Graduate Mathematical Department A.U.Th.

Declaration

This thesis has been submitted in November 2019 as the final part of my Master's degree in Aristotle University of Thessaloniki's Inter-faculty Master Program on Complex Systems and Networks. It does not aim to achieve innovative new contributions to the domain of Quantum Machine Learning but rather focuses on present, discuss and organize the existing literature 's proposals as well as contemplate the potentials and challenges of this newly developed and highly promising scientific domain.

Abstract

Machine Learning and Quantum computing are acknowledged to be among the state-of the-art technological developments of the 21st century. Both domains achieved huge advancements in the last decade, demonstrating very promising implementations for the progress of modern society. As a result of their success and progress, Quantum Machine Learning, a newly developed domain which comprise the combination of Machine Learning and Quantum Computing techniques, meets an explosive research interest in the recent years. Although the field is very young and thus fuzzy and immature compared to its parent fields, there is a variety of interesting proposals and approaches developed. This thesis consists an effort to define the domain of Quantum Machine Learning and identify the several schemes within the field as well as organize and summarize the existing literature in Quantum Machine Learning, with a focus on supervised learning, and follow the most recent developments. This effort particularly focuses on Quantum Neural Networks proposed in the scheme of Variational Quantum Circuits trained with hybrid methods. This choice follows the general spectrum of this thesis to concentrate on prospects of near-term applicable quantum algorithms for Machine Learning. Several Quantum Neural Network frameworks that appear prominent for near-term applications are presented and discussed in an effort to reveal the general picture of expected quantum enhancements brought by Quantum Machine Learning as well as the caveats these frameworks carry and the obstacles the domain has to overcome in order to demonstrate its first much-expected applications in the recent future.

Key Words: Quantum Machine Learning, Quantum Computing, Supervised Learning, Quantum Neural Networks, Variational Quantum Circuits, Hybrid-Learning, Near-Term applications

Acknowledgements

This thesis would not be completed without my supervisor Mr. Antoniou Ioannis who introduced me to the miraculous quantum world and inspired me to combine my pre-existent love for Machine Learning with the fields of Quantum Information and Quantum Computation. His guidance, assistance and support through this year's studies were critical. Thanks are also due to Dr Ch. Bratsas who introduced me to Machine Learning and knowledge processing and to Dr K. Chatzisavvas who clarified several issues on Quantum Computing. Special thanks to my family and friends for their continuous support and understanding during the course of my studies and particularly during the synthesis of this thesis. Finally, I own an acknowledgment to Francesco Petruccione and Maria Schuld, whose book Supervised Learning with Quantum Computers set the framework which inspired and guided this thesis.

CONTENT

List of Figures

List of Abbreviations and Acronyms

ANN: Artificial Neural Network

QNN: Quantum Neural Network

ML: Machine Learning

QML: Quantum Machine Learning

RUS: Repeat-Until-Success

QVE: Quantum Variational Eigensolver

QAOA: Quantum Approximation Optimization Algorithm

QCL: Quantum Circuit Learning

CM: Circuit Model

X_i: A one qubit Pauli operator on x-axis, σ_x applied to the i-th qubit

Y_i: A one qubit Pauli operator on y-axis, σ_y applied to the i-th qubit

Z_i: A one qubit Pauli operator on z-axis, σ_z applied to the i-th qubit

I: The identity matrix

PAC: Probably Approximate Correct

VC-dimension: Vapnik-Chervonenkis-dimension

CV: Continuous Variable

O: The "big-O" notation used to describe the limiting behaviour of a function when the argument tends towards a particular value or infinity. In this thesis is used for upper bounds. O(f(x)) means an upper bound polynomial to f(x), that is mf(x) for a real number m.

Ω: Equivalent to big-O notation, used for lower bound . Ω(f(x)) means a lower bound polynomial to f(x), that is nf(x) for a real valued n.

List of Tables

List of Appendices

Σύνοψη

Η ανάπτυξη των τομέων τόσο της Μηχανικής Μάθησης όσο και της Κβαντικής Υπολογιστικής θεωρούνται εξέχοντα τεχνολογικά επιτεύγματα του 21ου αιώνα, εφαρμογές των οποίων έχουν ήδη επιφέρει σημαντικές βελτιώσεις στην καθημερινή ζωή και έχουν αποτελέσει εφαλτήρια περεταίρω επιστημονικών ανακαλύψεων. Ως επακόλουθο των εξελίξεων, τις τελευταίες δύο δεκαετίες έχει αναπτυχθεί ένας καινούριος επιστημονικός κλάδος, η Κβαντική Μηχανική Μάθηση, που στοχεύει στον συνδυασμό μεθόδων και τεχνικών από τα προαναφερόμενα επιστημονικά πεδία για την ανάπτυξη αλγορίθμων μηχανικής μάθησης που εκμεταλλεύονται την υπολογιστική δυναμική των κβαντικών υπολογιστών και τις μοναδικές ιδιότητες των κβαντικών συστημάτων, όπως η υπέρθεση και η διεμπλοκή.

Ο νεοσύστατος αυτός κλάδος βρίσκεται ακόμη σε ένα ασαφές στάδιο και δεν η επιστημονική κοινότητα αδυνατεί να συμφωνήσει σε έναν ενιαίο ορισμό του περιεχομένου και των στόχων του. Ή ασάφεια αυτή έχει, σε μεγάλο βαθμό, κληρονομηθεί από τον κβαντικό χαρακτήρα του πεδίου και γεννάται από το γεγονός πως το hardware για την υλοποίηση κβαντικών υπολογισμών, και κατά συνέπεια κβαντικών αλγορίθμων, βρίσκεται ακόμη υπό κατασκευή. Ωστόσο, τα τελευταία χρόνια οι επιστημονικές εξελίξεις δείχνουν πως βρισκόμαστε όλο και πιο κοντά στην δημιουργία των πρώτων κβαντικών υπολογιστών ευρείας χρήσης, πέραν των μεμονωμένων κβαντικών μηχανών που έχουν ήδη δημιουργηθεί σε διάφορα ανά τον κόσμο εργαστήρια –D-Wave, IBM, Google, Microsoft–.

Η επανάσταση που αναμένεται να φέρει η υπολογιστική ισχύς των μηχανών αυτών, σε συνδυασμό με την εμπορική επιτυχία της Μηχανικής Μάθησης, έχει στρέψει το ενδιαφέρον τόσο της επιστημονικής όσο και της επιχειρηματικής κοινότητας προς την Κβαντική Μηχανική Μάθηση, με αποτέλεσμα ο τομέας να απολαμβάνει μεγάλου ερευνητικού ενδιαφέροντος τα τελευταία χρόνια και μια εκπληκτική ποικιλία προτάσεων και προσεγγίσεων συναντάται στη σχετική βιβλιογραφία. Στο 2ο Κεφάλαιο της εργασίας γίνεται μια προσπάθεια οργάνωσης και κατηγοριοποίησης των προτάσεων και προσεγγίσεων αυτών, επικεντρωμένη στην περίπτωση της Επιβλεπόμενης Μάθησης που είναι και η ηγετικότερη τάση μάθησης τόσο κλασικά όσο και κβαντικά.

Η πιο γνωστή και αναγνωρισμένη επιτυχία της Μηχανικής Μάθησης στηρίζεται αναμφισβήτητα στα Τεχνητά Νευρωνικά Δίκτυα και την ανάπτυξη της λεγόμενης Βαθιάς Μάθησης (deep learning), που αποτελούν πολύπλοκα υπολογιστικά μοντέλα εμπνευσμένα από την βιολογική λειτουργία του εγκεφάλου και τους μηχανισμούς μάθησης που συναντώνται στη φύση. Η υπολογιστική τους πολυπλοκότητα είναι αυτή που χαρίζει στα Νευρωνικά Δίκτυα τις εξαιρετικές δυνατότητες προσέγγισης πολύπλοκων συναρτήσεων για εντοπισμό σύνθετων μοτίβων σε ένα σύνολο δεδομένων και την εξαγωγή συμπερασμάτων από αυτά. Ακόμη, η βαθιά μάθηση επιτυγχάνεται μέσω της κωδικοποίησης και επεξεργασίας της πληροφορίας σε διανυσματική μορφή και τανυστές.

Το μαθηματικό μοντέλο των Νευρωνικών Δικτύων επομένως φαίνεται ιδανικός υποψήφιος για υλοποίηση σε ένα κβαντικό σύστημα που επιτρέπει από τη φύση του την αποθήκευση και διαχείριση μεγάλων μιγαδικών διανυσμάτων και τανυστών και έχει υπολογιστική ισχύ εκθετικά μεγαλύτερη αυτής ενός κλασικού υπολογιστή. Ως φυσικό επακόλουθο, ο σχεδιασμός και η υλοποίηση Κβαντικών Νευρωνικών Δικτύων απασχολεί ένα πολύ μεγάλο κομμάτι της βιβλιογραφίας του τομέα της Κβαντικής Μηχανικής Μάθησης. Η παρούσα διπλωματική αποτελεί μια βιβλιογραφική μελέτη συνθετικής διάστασης που δε συναντάται στην μέχρι στιγμής διαθέσιμη βιβλιογραφία στον τομέα των Κβαντικών Νευρωνικών Δικτύων, με επίκεντρο τις εφαρμογές τους στο άμεσο μέλλον. Στόχος της είναι να προσφέρει μία ολιστική εικόνα στον τομέα των Κβαντικών Νευρωνικών Δικτύων με παραμετρικοποιημένα κβαντικά κυκλώματα, ώστε να διαμορφώσει μία σαφή εικόνα των δυνατοτήτων και των δυσκολιών που αντιμετωπίζει ο τομέας και να αποφανθεί τι δυνατότητες εκφράζονται για άμεσες μελλοντικές εφαρμογές. Μία αναλυτική ιστορική επισκόπηση όσων μοντέλων έχουν προταθεί ως Κβαντικά Νευρωνικά Δίκτυα από το 1995 όταν πρωτοεμφανίστηκε ο όρος και ως σήμερα βρίσκεται στο 1.2

Δεδομένου του ότι οι πρώτοι κβαντικοί υπολογιστές, που αναμένεται να είναι διαθέσιμοι τα επόμενα χρόνια, θα είναι μικρής διάστασης, δηλαδή θα αποτελούνται από 50-100 qubits -το κβαντικό αντίστοιχο των bits- επιζητούμε προτάσεις Κβαντικών Νευρωνικών Δικτύων υλοποιήσιμες με μικρό αριθμό qubits. Ακόμη, στην περίπτωση των κβαντικών υπολογιστών θα χρειαστούν αρκετά χρόνια μέχρι να αναπτυχθεί ένα σύστημα διόρθωσης σφάλματος αντίστοιχο αυτού των σημερινών κλασικών υπολογιστών, άρα είναι αναγκαίο τα μοντέλα αυτά να παρουσιάζουν ανοχή στον θόρυβο και τα σφάλματα. Επιπλέον, τα κβαντικά συστήματα χρειάζονται πολύ συγκεκριμένες συνθήκες απομόνωσης από το εξωτερικό

περιβάλλον για να διατηρήσουν τα μοναδικά κβαντικά χαρακτηριστικά τους όπως η υπέρθεση και η διεμπλοκή, και ακόμη και χωρίς εξωτερική παρεμβολή καταρρέουν σε μικρά χρονικά διαστήματα- η ανοχή τους εξαρτάται από το εκάστοτε φυσικό σύστημα που χρησιμοποιείται για την υλοποίηση. Επομένως, θέλουμε μοντέλα που μπορούν να πραγματοποιηθούν σε πολύ μικρά χρονικά διαστήματα, κάτι που στην περίπτωση κβαντικών υπολογιστών που βασίζονται στην υλοποίηση πυλών, στους οποίους επικεντρώνεται και η εργασία αυτή, σημαίνει περιορισμένο αριθμό πυλών. Τέλος, λόγω της δυσκολίας δημιουργίας μιας κβαντικής Μνήμης Τυχαίας Προέλευσης (qRAM), για να είναι ένα Κβαντικό Νευρωνικό Δίκτυο διαθέσιμο για υλοποίηση στους πρώτους κβαντικούς υπολογιστές πρέπει να μη στηρίζει τον σχεδιασμό του σε κάποια κβαντική μνήμη, τουλάχιστον αυτής της μορφής. Τα ιδιαίτερα χαρακτηριστικά που επιτάσσει η δυνατότητα υλοποίησης των αλγορίθμων αυτών σε άμεσο μέλλον συ ζητώνται εκτενώς στο 4ο Κεφάλαιο.

Με γνώμονα τα παραπάνω κριτήρια, η βιβλιογραφική έρευνα των προτάσεων που έχουν αναπτυχθεί για Κβαντική Μηχανική Μάθηση και παρουσιάζονται συνοπτικά στο Κεφάλαιο 2, κατέληξε πως τα παραπάνω χαρακτηριστικά είναι πιθανότερο να εντοπισθούν σε Κβαντικά Νευρωνικά Δίκτυα που σχεδιάζονται ως Παραμετρικοποιημένα Κβαντικά Κυκλώματα και εκπαιδεύονται με υβριδικές μεθόδους συνδυασμού κλασικών και κβαντικών υπολογισμών. Οι σημαντικότερες και πιο ολοκληρωμένες προτάσεις που έχουν γίνει σε αυτό το πλαίσιο παρουσιάζονται, αναλύονται και συζητούνται στο Κεφάλαιο 3.

Η Κβαντική Υπολογιστική έχει επιδείξει την υπεροχή της έναντι της κλασικής αρκετά χρόνια πριν με την απόδειξη κβαντικών αλγορίθμων όπως των Grover, Shor, Deutsch που καταφέρνουν να λύσουν προβλήματα που είναι απλησίαστα από κλασικούς υπολογιστές λόγω της πολυπλοκότητας τους, ή επιδεικνύουν εκθετικές επιταχύνσεις. Στο 4ο Κεφάλαιο της εργασίας αναλύεται ποια είναι η αναμενόμενη προσφορά των κβαντικών μοντέλων Μηχανικής Μάθησης, άρα και τον κβαντικών Νευρωνικών Δικτύων, αλλά και τα προβλήματα που πρέπει επιλύσουν τα συγκεκριμένα τόσο ως προς το σχεδιασμό όσο και ως προς την υλοποίησή τους.

Συνοπτικά, η παρούσα διπλωματική διαρθρώνεται ως εξής: Στο 1ο Κεφάλαιο αιτιολογείται η επιλογή ενασχόλησης με την Κβαντική Μηχανική Μάθηση, η εμβάθυνση στα Κβαντικά Νευρωνικά Δίκτυα για επιβλεπόμενη μάθηση και τέλος η επικέντρωση της συγγραφέως σε προτάσεις που, κατά τη δική της κρίση που επιβεβαιώνεται και βιβλιογραφικά, είναι πιθανότερο εφαρμόσιμες στο εγγύς μέλλον. Στο 2ο Κεφάλαιο κρίθηκε σκόπιμο να γίνει

μία ολιστική βιβλιογραφική ανασκόπηση των όσων έχουν προταθεί στον τομέα της Κβαντικής Μηχανικής Μάθησης, υπό το πρίσμα του ότι πολλές από τις τάσεις που έχουν διαμορφωθεί μπορούν να φιλοξενήσουν ένα Κβαντικό Νευρωνικό Δίκτυο και πολλοί αλγόριθμοι και εργαλεία Κβαντικής Μηχανικής Μάθησης να χρησιμοποιηθούν ως εργαλεία και σε αυτό. Στο 3ο Κεφάλαιο αποδεικνύεται πως υπό προϋποθέσεις τέτοια κβαντικά μοντέλα μπορούν να συντελέσουν μάθηση και 4 παραλλαγές Κβαντικών Νευρωνικών Δικτύων ως Παραμετρικοποιημένα Κβαντικά Κυκλώματα παρουσιάζονται και αναλύονται. Τέλος στο 4ο Κεφάλαιο συζητούνται τα πλεονεκτήματα μα και οι δυσκολίες τέτοιων κβαντικών μοντέλων, αναλύονται τα χαρακτηριστικά που απαιτεί μια άμεση υλοποίηση τους, και τα αναπάντητα ερωτήματα της έρευνας αυτής αλλά και του τομέα γενικότερα.

Για τη δημιουργία ενός αλγορίθμου μηχανικής μάθησης υπάρχουν δύο στρατηγικές. Η πρώτη ονομάζεται «Μεταφραστική» και αναφέρεται στην μετάφραση κλασικών αλγορίθμων μηχανικής μάθησης στην κβαντική γλώσσα, με στόχο την βελτίωση της απόδοσης, της υπολογιστικής ισχύς, του χρόνου απόκρισης του αλγορίθμου ή της εξάρτησης του από το πλήθος δεδομένων εκπαίδευσης. Τη στρατηγική αυτή ακολουθούν κβαντικά μοντέλα που έχουν σχεδιαστεί για να εξάγουν συμπεράσματα εφαρμόζοντας μία αντιστοίχιση εισόδου-εξόδου τύπου f(x)=y ή μια κατανομή δεσμευμένης πιθανότητας p(x|y). Τεχνικές σχεδιασμού τέτοιων κβαντικών μοντέλων συζητούνται στα υποκεφάλαια 2.2 και 3.1. Η δεύτερη στρατηγική ονομάζεται «Διερευνητική» και είναι η πιο μοντέρνα που στοχεύει στη δημιουργία κβαντικών μοντέλων μηχανικής μάθησης ικανών να γενικεύουν τα συμπεράσματά τους σε νέα δεδομένα. Αντί να επικεντρώνεται σε υπολογιστική επιτάχυνση, τα μοντέλα αυτά ξεκινούν από την κβαντική μηχανή στην οποία θέλουμε να εφαρμόσουμε κάποιο μοντέλο και χτίζουν τον αλγόριθμο βάση των δυνατοτήτων της μηχανής αυτής. Κατευθύνσεις για τον σχεδιασμό κβαντικών αλγορίθμων υπό αυτό το πρίσμα συζητούνται στα υποκεφάλαια 2.4 και 3.3. Τεχνικές εκπαίδευσης για τα κβαντικά μοντέλα που προκύπτουν από την κάθε κατεύθυνση παρουσιάζονται στο 2.3 και η τεχνική που επιλέγεται ως βέλτιστη για τις πρώτες εφαρμογές, η Υβριδική περίπτωση, αναλύεται στο υποκεφάλαιο 3.2.

Παρά την ασάφεια που επικρατεί ακόμα στον τομέα, η επιστημονική κοινότητα έχει συμφωνήσει στο ό,τι γενικά διακρίνονται 4 είδη αλγορίθμων. Αυτά που αφορούν κλασικά δεδομένα και κλασική επεξεργασία εμπνευσμένη από κβαντικά ανάλογα, την περίπτωση που κλασική μηχανική μάθηση χρησιμοποιείται για να βοηθήσει κβαντικές διαδικασίες, κβαντικά δεδομένα με κβαντική επεξεργασία και κλασικά δεδομένα επεξεργασμένα με κβαντικό τρόπο.

Οι περισσότερες προτάσεις εστιάζουν στην τελευταία περίπτωση, για διάφορους λόγους που αναλύονται στο 2.1.3. Στην περίπτωση αυτή, για να επεξεργαστεί ένας κβαντικός υπολογιστής κλασικά δεδομένα πρέπει αυτά να κωδικοποιηθούν σε κάποια από τις μεταβλητές του συστήματος με μία από τις 4 βασικές μεθόδους κωδικοποίησης που περιγράφονται στο κεφάλαιο 2.1.4. Ονομαστικά; Κωδικοποίηση στις καταστάσεις του συστήματος, Κωδικοποίηση στα πλάτη πιθανότητας, Κωδικοποίηση στην Χαμιλτονιανή ή qsamples, που είναι καταστάσεις του κβαντικού συστήματος που κωδικοποιούν τυχαίες μεταβλητές μιας κατανομής. Η υπολογιστική πολυπλοκότητα και το πλήθος των qubits και των πυλών που πρέπει να υλοποιηθούν για να πραγματοποιηθεί η κωδικοποίηση της πληροφορίας είναι πολλές φορές δυσβάσταχτα για ένα κβαντικό μοντέλο.

Οι προσεγγίσεις στις οποίες μπορεί να κατηγοριοποιηθεί το σύνολο της βιβλιογραφίας για κβαντικά μοντέλα για συμπεράσματα είναι τρεις. Γραμμικά μοντέλα, το αποτέλεσμα των οποίων υπό προϋποθέσεις μπορεί να θεωρηθεί ένα εσωτερικό γινόμενο δύο κβαντικών καταστάσεων ή η τελική κατάσταση ενός μικρού κβαντικού κυκλώματος, όπως αναλύεται στο 2.2.1. Αποδεικνύεται μάλιστα πως τέτοια κβαντικά μοντέλα μπορούν να αναπαρασταθούν γραφικά ως Νευρωνικά Δίκτυα χωρίς συναρτήσεις ενεργοποίησης, όπου κάθε κβαντική πύλη αντιστοιχεί σε ένα στρώμα νευρώνων. Μοντέλα βασισμένα σε συναρτήσεις πυρήνων, καθώς όπως εξηγείται στο 2.2.2 υπάρχει μία πολύ κομψή αναλογία ανάμεσα σε έναν feature space που ορίζεται από μια συνάρτηση πυρήνα και τον χώρο Χίλμπερτ που περιγράφει το κβαντικό σύστημα, λόγω της άμεσης σχέση των συναρτήσεων πυρήνα με τα εσωτερικά γινόμενα. Τέλος, πιθανοκρατικά μοντέλα τα οποία δεν θα μπορούσαν να λείπουν από το πλαίσιο της κβαντικής θεωρίας που είναι μια θεωρία πιθανοτήτων. Η τρίτη προσέγγιση παρουσιάζεται στο 2.2.3 και είναι άμεσα συνδεδεμένη με τις κβαντικές καταστάσεις qsamples που προαναφέρθηκαν.

Στη συνέχεια του 2^{ου} Κεφαλαίου παρουσιάζονται στρατηγικές εκπαίδευσης κβαντικών μοντέλων Μηχανικής Μάθησης για κάθε προσέγγιση. Τεχνικές εκπαίδευσης βασισμένες στις δυνατότητες μιας κβαντικής μηχανής να κάνει πράξεις γραμμικής άλγεβρας σε πολύ γρήγορους χρόνους, που απορρέουν από τον κβαντικό αλγόριθμο Harrow-Hassidim-Loyd για την επίλυση της γραμμικής εξίσωσης Ax=b -όπου A ένας πίνακας και b ένα διάνυσμα- και τις παραλλαγές του αναφέρονται στο 2.2.1. Στο 2.2.2. εξετάζονται μέθοδοι εκπαίδευσης που στηρίζονται σε ευριστικές μεθόδους αναζήτησης εμπνευσμένες από τον κβαντικό αλγόριθμου του Grover. Υβριδικές μέθοδοι που συνδυάζουν κλασικούς αλγορίθμους βελτιστοποίησης όπως Simplex, ή Gradient Descent με κβαντικούς υπολογισμούς συζητούνται στο 2.2.3 και

αναλύονται περαιτέρω στο 3.2 καθώς αποτελούν την ραχοκοκαλιά της εκπαίδευσης των Κβαντικών Νευρωνικών Δικτύων στα οποία επικεντρώνεται η εργασία. Τέλος, στο 2.2.4 γίνεται μία σύντομη αναφορά σε χρήση μεθόδων εμπνευσμένων από το Αδιαβατικό Θεώρημα για κβαντικά μοντέλα που αποβλέπουν σε υλοποίηση μέσω κβαντικής ανόπτησης.

Οι δύο παραπάνω κατηγορίες αλγορίθμων για εξαγωγή συμπερασμάτων και εκπαίδευση αποτελούν κυρίως κβαντικές μετάφρασης αντίστοιχων κλασικών μοντέλων και αλγορίθμων Μηχανικής Μάθησης. Στο 2.3 ακολουθείται μία διαφορετική τακτική, η «εξερευνητική» και παρουσιάζονται κβαντικά μοντέλα τύπου Ising για κβαντικά Hopfield Νευρωνικά Δίκτυα για Χεμπιανή μάθηση ή κβαντικές μηχανές Boltzmann για βαθιά μάθηση (2.3.1) ή Παραμετρικοποιημένα Κβαντικά Κυκλώματα ως Κβαντικά Νευρωνικά Δίκτυα ή Κατηγοριοποιητές (2.3.2). Κάποιες μεμονωμένες άλλες προσεγγίσεις αναφέρονται επίσης (2.3.3). Ο στόχος μοντέλων της «εξερευνητικής» προσέγγισης δεν είναι να επιτύχουν κάποια επιτάχυνση σε σχέση με κλασικά τους ανάλογα, άλλα είναι να εξερευνήσουν τις δυνατότητες μίας κβαντικής μηχανής να αναπτύξει τεχνικές μάθησης μέσα από την επεξεργασία ενός συνόλου δεδομένων.

Στο 3ο Κεφάλαιο η εργασία επικεντρώνεται σε αυτό που τόσο η βιβλιογραφία, όσο και η προσωπική άποψη της συγγραφέως, δείχνει να είναι η οικογένεια Κβαντικών Νευρωνικών Δικτύων που μπορεί να φτάσει σε πρώιμες υλοποιήσεις με το πρώτο διαθέσιμο hardware. Η οικογένεια αυτή είναι τα Παραμετρικοποιημένα Κβαντικά Κυκλώματα, που αποτελούν κβαντικά κυκλώματα με κλασικές παραμέτρους και εκπαιδεύονται μέσω ενός υβριδικού σχήματος κβαντικού υπολογισμού του αποτελέσματος και κλασικής βελτιστοποίησης των παραμέτρων. Ο διαμερισμός των υπολογισμών σε κβαντικό και κλασικό υπολογιστή, σε συνδυασμό με την ικανότητα των μοντέλων αυτών να λειτουργούν με κλασικά αλλά και κβαντικά δεδομένα, καθώς και το ό,τι στηρίζονται σε κβαντικούς υπολογιστές που δουλεύουν με υλοποίηση πυλών είναι τα στοιχεία που κάνουν την οικογένεια αυτή την ιδανικότερη για άμεσες εφαρμογές. Στο 3.1 παρουσιάζονται δύο τέτοια μοντέλο για εξαγωγή συμπερασμάτων, το δεύτερο εκ των οποίων καταφέρνει να προσθέσει μια μη γραμμική συνάρτηση ενεργοποίησης χωρίς να διαταράξει τις κβαντικές αρχές του συστήματος, μέσω κωδικοποίησης της πληροφορίας του σταθμισμένου αθροίσματος των νευρώνων ενός στρώματος σε γωνίες περιστροφών στο σύστημα.

Στο 3.2 αναλύονται οι τεχνικές υβριδικής εκπαίδευσης ενός Κβαντικού Νευρωνικού Δικτύου, ταξινομημένες σε τεχνικές χωρίς παραγώγιση, η μέθοδος Nelder-Mead συγκεκριμένα (3.2.1), τεχνικές με νουμερική προσέγγιση παραγώγων (3.2.1) και τεχνικές με αναλυτικό υπολογισμό των παραγώγων (3.3.3) μιας ορισμένης αντικειμενικής συνάρτησης. Στην τρίτη περίπτωση μάλιστα αποδεικνύεται πως αυτός υπολογισμός σε ορισμένες περιπτώσεις μπορεί να γίνει εύκολα κβαντικά, και μόνο ο υπολογισμός της τιμής των ανανεωμένων παραμέτρων ανατίθεται στην κλασική μηχανή. Τέλος , στο 3.3.1 παρατίθεται η απόδειξη του κβαντικού ανάλογου του Θεωρήματος Καθολικής Προσέγγισης, που υποδεικνύει την δυνατότητα μάθησης και προσέγγισης κάθε δίττιμης μεταβλητής από ένα Κβαντικό Νευρωνικό Δίκτυο αυτής της μορφής. Αφού αποδεικνύεται αυτή η δυνατότητα μάθησης, 4 μοντέλα που αποτελούν παραλλαγές του ίδιου σχήματος παρουσιάζονται και συζητούνται. Και τα 4 βασίζονται στο σχεδιασμό παραμετρικοποιημένων κβαντικών συστημάτων με κλασικές παραμέτρους, όπου διαφοροποιείται το είδος των κβαντικών πυλών που αποτελούν το σύστημα. Ανάλογα με το είδος των κβαντικών πυλών αναπτύσσονται και διαφορετικοί τρόποι αναλυτικού υπολογισμού των παραγώγων της συνάρτησης κόστους, και κατ' επέκτασιν του ίδιου του κυκλώματος. Μία πιο ειδική περίπτωση αποτελεί το μοντέλο που παρουσιάζεται στο 3.3.5, που στηρίζεται σε κβαντική υπολογιστική συνεχών μεταβλητών. Πιο συγκεκριμένα, η πληροφορία δεν διακριτοποιείται για να αποθηκευτεί σε qubits, αλλά αξιοποιούνται μεταβλητές του κβαντικού συστήματος που είναι συνεχείς, όπως η ορμή και η θέση ενός σωματιδίου, και οι πράξεις αναπαρίστανται σε Γκαουσιανούς και μη Γκαουσιανούς πίνακες που δημιουργούν ένα πλήρες σύνολο τελεστών. Σημαντικές συμπληρωματικές τεχνικές που χρησιμοποιούν αυτά τα μοντέλα παρατίθενται στο τέλος της εργασίας σε μορφή βοηθητικών υπομνημάτων (Appendices).

Στο σύνολο τους τα μοντέλα αυτά συγκλίνουν σε επίπεδα πολυπλοκότητας και απαιτήσεις στο πλήθος των qubits και των προς υλοποίηση κβαντικών πυλών, διαφέροντας σε σημεία αρχιτεκτονικής και του είδους των λογικών μοναδιαίων κβαντικών πυλών στα οποία μπορεί να αναλυθεί το παραμετρικοποιημένο κύκλωμα. Οι απαιτήσεις αυτές προσπαθούν να κρατηθούν στο ελάχιστο δυνατό από τους ερευνητές, χωρίς ωστόσο να καταφέρνουν να αγγίξουν την χαμηλή πολυπλοκότητα που απαιτείται από την επικρατούσα κατάσταση διαθέσιμου κβαντικού hardware. Ιδιαίτερες κβαντικές ιδιότητες όπως η υπέρθεση του συνόλου δεδομένων για παράλληλη επεξεργασία τους είναι εφικτές στα πλαίσια σχεδιασμού των Κβαντικών Νευρωνικών Δικτύων αυτών, αλλά δεν αξιοποιούνται όταν το σύστημα υπόκεινται στους περιορισμούς που επιβάλλουν τα διαθέσιμα μέσα. Επιπλέον, όλες οι προτάσεις που

αναλύονται αποτελούν γενικότερα πλαίσια ανάπτυξης Κβαντικών Νευρωνικών Δικτύων και όχι εξειδικευμένα μοντέλα, καθώς το είδος των πυλών ενός κυκλώματος και η στρατηγική επιλογής της αρχιτεκτονικής του δεν έχουν αναλυθεί ακόμη από την παρούσα βιβλιογραφία. Ακόμη, συμπεραίνουμε πως ιδιαίτερη μελέτη πρέπει να γίνει και για τον τρόπο επιλογής των αρχικών παραμέτρων ενός τέτοιου μοντέλου ανάλογα με το πρόβλημα που θέλουμε να επιλύσει και το είδος των δεδομένων. Τέλος, αξίζει να σημειωθεί πως στην περίπτωση κλασικών δεδομένων τα οποία πρέπει στο πρώτο στάδιο ενός μοντέλου να κωδικοποιηθούν σε κβαντική μορφή μέσα σε αυτό, η πολυπλοκότητα αυξάνεται δραματικά.

Στο 4ο κεφάλαιο κλείνουμε με μία σύνοψη συμπερασμάτων, τόσο από αυτά που παρουσιάστηκαν στα προηγούμενα κεφάλαια, όσο και μελετώντας ανάλογη βιβλιογραφία. Στο 4.1 αναλύονται οι αναμενόμενες βελτιώσεις που παρουσιάζει ένα Κβαντικό Νευρωνικό Δίκτυο, και ένας κβαντικός αλγόριθμος μηχανικής μάθησης γενικότερα -όπως προαναφέραμε η κατηγοριών των δύο εννοιών είναι μεγάλη- είτε σε σύγκριση με τα υπάρχοντας κλασικά μοντέλα, είτε και ως νέα κβαντικά εμπνευσμένα μοντέλα μάθησης. Για να επιτευχθεί μία πολύπλευρη ανάλυση, τα πιθανά επιτεύγματα κατηγοριοποιούνται στην περίπτωση βελτιώσεων σε υπολογιστική πολυπλοκότητα, πολυπλοκότητα αναγκαίου δείγματος εκπαίδευσης και πολυπλοκότητα του μοντέλου. Στην πρώτη περίπτωση αναγνωρίζονται τρία είδη πιθανών υπερβάσεων από κβαντικά μοντέλα, με παραδείγματα κβαντικών αλγορίθμων για κάθε περίπτωση. Στην δεύτερη περίπτωση αποδεικνύεται πως με όποιο τρόπο και έχει πρόσβαση στο σύνολο εκπαίδευσης ένας κβαντικός αλγόριθμος, έχει γραμμική σχέση με τις ανάγκες πλήθους παραδειγμάτων ενός κλασικού αλγορίθμου και δεν αναμένεται κάποια μεγάλη προσφορά από το QML για το πρόβλημα αυτό. Τέλος, λόγω της πολυπλοκότητας υπολογισμού της VC-διάστασης ενός μοντέλου -ιδεί ενός πολυεπίπεδου μοντέλου όπως τα Νευρωνικά Δίκτυα- τόσο στην κλασική όσο και στην κβαντική περίπτωση, η προσπάθεια εξαγωγής κάποιου συμπεράσματος κρίνεται άκαρπη.

Το 4.2 είναι αφιερωμένο στην ανάλυση των προκλήσεων που αντιμετωπίζει ο τομέας της Κβαντικής Μηχανικής Μάθησης, οπού επιβεβαιώνεται αυτό που αναφέρεται καθ’ όλη τη διάρκεια της παρούσας Διπλωματικής, πως όλα τα προβλήματα πηγάζουν από την έλλειψη διαθέσιμου κβαντικού hardware για να υποστηρίξει την υλοποίηση κβαντικών μοντέλων και πρακτικό πειραματισμό. Το θέμα αυτό παρουσιάζεται ολιστικά στο 4.2.1, ενώ στο 4.2.3 εμβαθύνουμε στην έννοια της qRAM, η ανάγκη της οποίας έχει τονισθεί πολλές φορές κατά τη διάρκεια της εργασίας, κυρίως γιατί στην ύπαρξη της στηρίζονται σχεδόν όλοι οι κβαντικοί

αλγόριθμοι, μηχανικής μάθησης και μη, που αξιοποιούν την κβαντική υπέρθεση. Το θέμα των πολύ μικρών χρόνων συνεκτικότητας ενός Κβαντικού Συστήματος και η ευαισθησία του σε εξωτερικές παρεμβολές ή την πάροδο του χρόνου και τα προβλήματα που αυτό δημιουργεί θίγονται στο 4.2.2.

Συνεχίζοντας, στο 4.3 γίνεται η σύνοψη των απαραίτητων προϋποθέσεων που πρέπει να καλύπτει ένα μοντέλο Κβαντικής Μηχανικής Μάθησης για να μπορέσει να υλοποιηθεί στους πρώτους διαθέσιμους κβαντικούς υπολογιστές. Χωρίς την ανάπτυξη μιας qRAM, η οποία θα αργήσει όπως αναφέρεται στο 4.2.3, η παράλληλη επεξεργασία που χαρίζει φυσικά σε ένα κβαντικό μοντέλο η υπέρθεση θα πρέπει να μείνει αναξιοποίητη. Τόσο το πλήθος των διαθέσιμων qubits όσο και το βάθος των κβαντικών κυκλωμάτων, δηλαδή το πλήθος των κλασικών πυλών που πρέπει να υλοποιηθούν- οφείλουν να είναι περιορισμένα σε 50-100 qubits και 1000-3000 πύλες. Ακόμα, η επιτυχία των μοντέλων βασίζεται στην ανοχή τους στο θόρυβο, καθώς θα πρέπει να δουλεύουν χωρίς σύστημα αυτόματης διόρθωσης λαθών, κάτι που είναι αρκετά πιο δύσκολο να αναπτυχθεί για έναν κβαντικό υπολογιστή απ' ότι ήταν για τον κλασικό υπολογιστή, λόγω του κβαντικού Θεωρήματος μη Κλωνοποίησης, που απαγορεύει τη δημιουργία αντιγράφων των κβαντικών καταστάσεων. Επιπλέον, η πιθανοκρατική φύση των κβαντικών συστημάτων σε συνδυασμό με το γεγονός πως η υλοποίηση των πυλών από τους πρώτους κβαντικούς υπολογιστές είναι περιορισμένης ακρίβειας και απαιτεί αλλεπάλληλες πραγματοποιήσεις και μετρήσεις του συστήματος για την αξιόπιστη προσέγγιση του αποτελέσματος, υπογραμμίζουν την ανάγκη της εύκολης και γρήγορης αναπαραγωγικότητας των κβαντικών μοντέλων μηχανικής μάθησης.

Η εργασία αυτή ολοκληρώνεται με την παράθεση των σημαντικότερων ανοιχτών ερωτημάτων στον τομέα της Κβαντικής Μηχανικής Μάθησης, στο 4.4, όπου συναντώνται ερωτήματα θεμελιώδους σημασίας που υπογραμμίζουν το σε πόσο πρωταρχικό στάδιο είναι ακόμη ο τομέας αυτός. Στο τελευταίο υποκεφάλαιο, 4.5, παρατίθενται τα συμπεράσματα της συγγραφέως κατόπιν αυτής της συνθετικής βιβλιογραφικής μελέτης. Τα συμπεράσματα αυτά συνοψίζονται σε τρία σημεία. Πρώτων, ο τομές των Κβαντικών Νευρωνικών Δικτύων βρίσκεται στο στάδιο που βρισκόταν η έρευνα των Τεχνητών Νευρωνικών Δικτύων τη δεκαετία του 80, όταν ελλείψει της διαθέσιμης υπολογιστικής ικανότητας για την υλοποίηση τους ήταν απλά ασαφείς ιδέες στις οποίες λίγοι πίστευαν. Τα Κβαντικά Νευρωνικά Δίκτυα θα συνεχίσουν να μας απασχολούν σε θεωρητικό επίπεδο μέχρι το διαθέσιμο hardware επιτρέψει την υλοποίηση τους, και όταν αυτή η στιγμή έρθει και ο πειραματισμός σε ευρύ σκέλος είναι

εφικτός, θα συντελέσουν μία υπολογιστική επανάσταση. Αυτό διότι, όταν η παράλληλη επεξεργασία των δεδομένων μέσω της υπέρθεσης και η χρήση κβαντικών ευριστικών αλγορίθμων βασισμένων στο πλάτος πιθανότητας όπως του Grover γίνουν δυνατές, η εύρεση των απολύτως βέλτιστων βαρών χωρίς επαναληπτικές διαδικασίες που είναι ικανά να υλοποιήσουν είναι κάτι που δεν έχει ούτε μπορεί να αποκτήσει κλασικό ανάλογο.

Δεύτερον, η οικογένεια Κβαντικών Νευρωνικών Δικτύων που παρουσιάζεται στο κεφάλαιο 3 είναι η καλύτερη πρόταση που υπάρχει αυτή τη στιγμή με προοπτικές άμεσης υλοποίησης. Και αυτό γιατί, αν περιοριστεί στην περίπτωση δίτιμων συναρτήσεων, επιτρέπει την ικανή προσέγγιση τους με σχετικά μικρής πολυπλοκότητας κβαντικά κυκλώματα, υλοποιήσιμα από τους διαθέσιμους κβαντικούς υπολογιστές που στηρίζονται σε υλοποίηση κβαντικών πυλών. Αυτά τα κυκλώματα μπορούν να επιτύχουν ικανοποιητική προσέγγιση χωρίς να στηρίζονται σε κβαντική υπέρθεση, άρα και την ύπαρξη μιας qRAM, ακόμα και όταν υπόκεινται σε αυστηρούς περιορισμούς σχετικά με τα χαρακτηριστικά των πυλών αυτών για να μειώσουν την πολυπλοκότητα του μοντέλου στο ελάχιστο δυνατό. Τέλος, η χρήση κλασικών παραμέτρων επιτρέπει την αποθήκευση και επαναχρησιμοποίηση τους, ενώ η κλασική τους επεξεργασία διευκολύνει την διαδικασία της εκπαίδευσης. Εξάλλου, από τα τέσσερα είδη εκπαίδευσης που συζητήθηκαν στο 3.2, η υβριδική εκπαίδευση είναι η μόνη που δε στηρίζεται σε πολύπλοκες κβαντικές υπο-ρουτίνες και υπέρθεση δεδομένων.

Τρίτον, προσωπική άποψη της συγγραφέως είναι πως η έρευνα στον τομέα των Κβαντικών Νευρωνικών Δικτύων θα έπρεπε για το επόμενο διάστημα να σταματήσει να αφορά αποκλειστικά την ανάπτυξη νέων θεωρητικών και αδύνατο να τεκμηριωθούν προτάσεων για μοντέλα και να επικεντρωθεί στην δημιουργία μιας ολιστικής πρότασης. Με επίκεντρο τα Κβαντικά Νευρωνικά Δίκτυα που σε συνδυασμό με υβριδικές τεχνικές εκπαίδευσης φέρονται οι βέλτιστοι υποψήφιοι για άμεσες εφαρμογές, τα σοβαρά ερωτήματα του πως δομείται η αρχιτεκτονική ενός πραμαετρικοποιημένους κβαντικού κυκλώματος, πως αρχικοποιούνται βέλτιστα οι παράμετροι του, τι συμβαίνει με τις δυνατότητες μάθησης και της υπερμοντελοποίησης (overfitting) και πώς μπορεί να αυξηθεί η ανοχή τους στο θόρυβο και τα σφάλματα είναι κάποιες από τις βασικότερες κατεύθυνσεις προς διερεύνηση για την κατασκευή ενός πλήρους Κβαντικού Νευρωνικού Δικτύου.

Ελληνικό Γλωσσάριο

Adiabatic: Αδιαβατικός

Adjacency matrix: Πίνακας Γειτνίασης

Amplitude: Εύρος/Πλάτος

Annealing: Ανόπτηση

Artificial Intelligence: Τεχνητή Νοημοσύνη

Artificial Neural Networks: Τεχνητά Νευρωνικά Δίκτυα

Classification: Κατηγοριοποίηση

Clustering: Ομαδοποίηση

Coherence: Συνεκτικότητα

Continuous Variable: Συνεχής Μεταβλητή

Density Matrix: Πίνακας Πυκνότητας

Eigenvalue: Ιδιοτιμή

Eigenstate: Ιδιοδιάνυσμα

Entanglement: Διεμπλοκή

Gradient descent algorithm: Αλγόριθμος σύγκλισης με ελάττωση της παραγώγου

Hamiltonian: Χαμιλτονιανή

Hermitian matrix: Ερμιτιανός Πίνακας

Inner Product: Εσωτερικό Γινόμενο

Information Encoding: Κωδικοποίηση Πληροφορίας

Ion-Traps: Παγίδες ιόντων

Ising-Type Models: Μοντέλα τύπου Ising

Kernel: Πυρήνας

Markov chain: Μαρκοβιανή αλυσίδα

Machine Learning: Μηχανική Μάθηση

Oracle: Χρησμός

Overfitting: Υπερμοντελοποίηση

Principal Component Analysis: Ανάλυση Κύριων Συνιστωσών

Probability Distribution: Κατανομή Πιθανότητας

Quantum Computing: Κβαντική Υπολογιστική

Quantum Gate: Κβαντική Πύλη

Quantum Machine Learning: Κβαντική Μηχανική Μάθηση

Quantum Neural Networks: Κβαντικά Νευρωνικά Δίκτυα

(Quantum) Random Walk: (Κβαντικός) Τυχαίος Περίπατος

Regression: Παλινδρόμηση

Reinforcement Learning: Ενισχυτική Μάθηση

Singular Value Decomposition: Ανάλυση σε Ιδιάζουσες Τιμές

Superposition: Υπέρθεση

Supervised Learning: Επιβλεπόμενη Μάθηση

Support Vector Machines: Μηχανές Διανυσμάτων Υποστήριξης/ Μηχανές Εδραίων Διανυσμάτων

Symplectic matrix: Συμπλεκτικός Πίνακας

Unitary gate: Μοναδιαίος Πίνακας

Unsupervised Learning: Μάθηση χωρίς επίβλεψη

Variational Quantum Circuits: Παραμετρικοποιήσιμα Κβαντικά Κυκλώματα

1 Introduction

1.1 Scope of this thesis

This thesis' purpose is to provide a holistic overview of the literature research in Quantum Neural Networks aiming in near-term applications based on Variational Quantum Circuits trained via Hybrid methods. It constitutes a synthetic review of the present literature in the domain, and in order to be in parallel with the explosive number of publications in the last two years, was updated several times. It should be pointed out that, at the author's knowledge, there is no such equivalent in the available literature at this moment. It should not be claimed, of course, that it is absolutely complete as the developments and the scientific interest in this subject are constant and new proposals are published every week. The choice to focus on Quantum Neural Networks represented by Variational Quantum Circuits was made after several months of studying the field of Quantum Machine Learning and Quantum Neural Networks in particular, in October 2018, motivated by the author's perception that this scheme presents several potentials for an application in the first quantum hardware available. This belief was strengthened in the following months by the rise of the researcher's interest in this specific type of models and the exponential increase in the number of publications. It will also be proven based on facts in the rest of this thesis.

At this point, it should probably be pointed out that this thesis started in the scope of a straightforward comparison of an Artificial Neural Network (from now on referred as ANN) and a Quantum Neural Network (QNN) through simulation of both models on the same task, trained with the same dataset. The reasons this idea was abandoned were several, and I would like to mention the three most important of them; first, the naivety of the idea that simulating a quantum neural network on a classical computer would be a trivial task, which was indicative of my personal ignorance of the domain. Second, in order to analyse a QNN as a Variational Quantum Circuit (VQC), to train the model, choose the specific architecture and design the gates, even encode the classical data -given that there is no guidance available in the literature so far- is a task so demanding that would overpass the time available to submit a Master Thesis.

And although my supervisor was kind enough to offer me both support and time, the third reason this idea was abandoned was my personal belief, which was constantly reinforced through my research, that simulating an extremely small-scaled QNN for a very specific task, in a specific dataset, under much strict constraints imposed by software, hardware and complexity limitations and compare it to an ANN would have nothing to offer in the domain. And that is because, no generalization can be achieved and no absolute conclusions can be drawn from such a comparison, as I would end up comparing two different things. It is not fair to compare a QNN which comes with no tools for the optimal choice of initial parameters and has the form of a "toddler" idea and compare its results with an ANN which under years of intensive hands-on research has reached a premium stage. Moreover, as it will be explained, it is generally accepted that there is no benchmark comparison of an algorithm in Machine Learning (ML) and one in QML, let alone ANN and QNN which are of a much complex nature than simple ML algorithms. Finally, it is of the author's belief that the true enhancements QML has to offer will come from an approach that enables quantum machines to develop learning techniques on their own and by doing so revolutionize the ML domain, instead of using the quantum machines as computing devices and try to translate classical ML algorithms to quantum formulism.

Thus, the scope of this thesis changed to meet the current status in the domain of Quantum Machine Learning (QML) and especially QNN, which is theoretical in principal. As it will be evident in the process, the newly founded domain faces a quite fuzzy and immature state and undergoes a transition from a depended to an independent domain; the lack of hardware to support and realize complex quantum models, such as QNN, makes this effort no easier. The author of this thesis believes that an holistic and organized presentation of the several approaches formed so far in the domain of QML along with a synthetic review of the developments in the most promising type of QNNs, focused on the most recent ones and detached from the abstract and unfounded ideas which ruled the literature till 2015, would be a work of real value for the domain. All these with the hope that the present work can enable the reader to obtain a global view of what exists in the domain of QML and a guideline for structured research, as well as a realistic description of the advantages and challenges to expect. Finally, the originality of this work stems from the fact that this holistic analysis of the literature on the most prominent QNN type for near-term applications can lead to some conclusions about the state of the field, the realistic challenges and how they may be overcome as well as the

properties that must be carried to be near-term applicable, which cannot be drawn by an unstructured research in the scatter present literature.

This thesis is organised as follows; In the rest of this Section, the importance of QML and especially QNN will be pointed out, as a justification of this thesis subject of interest. Also, the need for near-term applications will be highlighted and historical overview of the literature on QNN- which cannot be found at this depth in the literature so far- will be adduced. Section 2 presents and briefly discusses all approaches in QML that have been met so far, grouped in quantum models for inference, training techniques and learnable quantum models. As Artificial Neural Networks are a subdomain of ML, the same applies to QNN and QML. Therefore, it is considered important to review the "tool-box" of practices and quantum routines developed for QML in general, before moving towards QNNs in particular. After all, there is a significant overlap between these two subjects and to support our choice of focus as a prominent for near-term applicable scheme, the rest of available schemes should be explained and discussed. In Section 3 we will centre our attention on QNN models, focusing on the approach that presents to be the best candidate for near-term applications. Several QNN models based on Variational Quantum Circuit architectures and Hybrid training methods will be presented. Finally, in Section 4 the expected quantum advantages, as well as the challenges QML struggles with, will be discussed and remaining open questions will be gathered. Finally, the conclusions that can be drawn from this literature research will be discussed.

1.2 Quantum Machine Learning

Machine Learning (ML) is a very popular subset of Artificial Intelligence (AI) based on algorithms and statistical models used by computer systems to perform specific tasks, like regression or classification for example, without explicit instructions. They are data-driven algorithms designed to fit mathematical models to data and use these models to derive decisions (inference). If these decisions can be accurately generalized to new, unseen by the algorithm, data then the model is declared learnable. Machine learning witnesses tremendous progress in the last decade, mainly thanks to the technological advancements of the 21st century that made computational tasks abundant and cheap as well as data collection, storage and manipulation easy.

Quantum Machine Learning (QML) on the other hand is a much younger field which is still struggling to find its definition and mainly consists of ideas and theories rather than

actual applications. Under the scope of this thesis, QML is defined as the implementation of quantum algorithms and techniques which combine the powers of quantum computing with the existing classical ML algorithms and theories targeting to runtime speedups, better accuracy or even tackle problems intractable from classical ML. QML models can be designed as entirely quantum models inspired from an ML task, or as enhancements to already existing models. This thesis will focus on the first case.

Unfortunately, only two paragraphs into this and the problems begin. QML comes to a quite blurry framework and the relevant literature lacks consistency and organisation. It also witnesses a roller-coaster trajectory inherited by its quantum nature. On the one hand, the hardware to create a quantum computer and test the efficiency of the QML algorithms compared to classical ones is yet to come. Of course, there do exist quantum simulators, the "quantum analogue computers", which is the most powerful quantum data analysis technique obtained in our days. Quantum simulators are basically classical systems whose dynamics can be programmed to match the dynamics of a desired quantum system. However, as in the classical ML case, lots of theories had to wait for the era of digital technology to come and make digital computers available in order to be implemented and tested. The exact same applies to QML and that is why in our days it remains a "deeply theoretical advanced technical domain".

However, great advancements towards a universal quantum computer have been made in the last five years since the domain drew besides academic also industrial attention. Technological giants like D-Wave, Google, Microsoft, and IBM have dedicated departments for quantum mechanics with the latest leading the race -at least till the very moment these lines are written-. In May 2016 IBM made history by launching *IBM Q Experience*. IBM Q Experience is an online platform through which anyone can have access via the cloud to two five-qubit and one sixteen-qubit processors made by superconducting transmon qubits. In March 2017 IBM Q Experience was enhanced by Qiskit, an open-source framework for quantum computing. It provides tools for creating and manipulating quantum programs and running them on prototype quantum devices and simulators. These developments gave rise to a new generation of QML literature, more mature and realistic. Thus, more well defined and compact theories started to form. At least 72 academic papers have been published using IBM Q Experience via cloud for their experiments and approximately one-third of them were QML related.

QML started to grow and draw more and more interest from both academics as well as industries. ML has formed a very profitable and dominating market in the last 5 years, hence with the expected quantum speedups, both in runtime as well as in memory capacity, QML forms a seductive domain. In the last two years the first gatherings of QML research community started to take place, 3 books have been published, tenths of start-ups related to QML were founded and yet no strict definition has been given to the field. The current status in the domain can be described by the following facts about QML from one of the very first official meetings of the QML research community, the Workshop "Quantum Machine Learning and Biomimetic Quantum Technologies", which took place from the 19th to the 23rd of March, 2018 at The Paraninfo of the University of the Basque Country in Bilbao, Spain (speeches can be found in youtube):

- There are huge fights about what the QML field actually is
- QML researchers are divided into two main categories; the ones that believe that QML is too detached from the research in classical ML and should try to adapt and learn from it, and those who believe that in an effort to mimic classical algorithms and strategies, QML loses its full potentials
- The American Statistics Association recently created a *Special Interest Group on Quantum Computing*
- There is a QML group in LinkedIn which currently has 719 members
- There are currently 3 books published for QML and two more are confirmed to be on their way, but "*only the latest of these three actually worth reading*"
- Till the moment these lines are written there are little less than 400 papers published on the topic and almost all of them are published in Physics magazines and not magazines related to Artificial Intelligence or Machine Learning
- Although most quantum-enhanced ML papers promise some kind of speedup, they still harvest with low-hanging fruits
- Only a maximum of 5% of a quantum computer's power is actually exploited by the usage of QML in real problems
- There are already dozens of start-ups that in one way or another try to design and implement QML algorithms
- All the technology companies in Silicon Valley have hired a QML or Quantum Computing consultant, in order to be prepared for the quantum computers era

- There are so many ideas impatiently waiting to be tested that some interfaces and programming tool-boxes like dedicated machine learning platforms for quantum computers have already immerged

These were March 2018 and now is November 2019 and not a lot of things have changed. We still don't actually have quantum computers, at least not of the scale required to test QML theories. However, in the last couple of years one thing certainly changed. A great part of QML community realized that the majority of QML theories so far will remain just theories for a long time to come. Industry strives for near-term applications of QML and for those it is willing to invest. And quantum community in general needs investments to build a quantum computer.

The first quantum computers, the "near-term" ones, are currently being developed in a variety of hardware platforms such as ion traps, annealers, superconducting materials and photonic chips. Reality indicates that they are to be small-scaled in terms of qubits, non-fault-tolerant in terms of error-correction techniques deployed as well as gate fidelity and quite expensive to use. These near-term devices offer the testbed for QML algorithms. Therefore, from a bullish yet pragmatistic point of view, a near-term QML algorithm is considered to be realized in some thousands of unitary operations, applied on 50-100 qubits, and demonstrate certain robustness to errors. Many of those small-scale algorithms appeared in recent literature either in an effort to demonstrate quantum superiority over classical ones or, disregarding the need to prove superiority, focused on offering practical solutions and having useful real-life applications. In the second approach of near-term QML models this thesis will focus on.

1.3 Quantum Neural Networks

Let's turn back to the classical ML. The most state-of-the-art subcategory of ML that brought the field to a renaissance period is Artificial Neural Networks (ANN), recently forming the domain of deep learning. ANN and deep learning enjoy an enthusiastic reception both scientifically as well as commercially with numerous applications in tasks including complex pattern recognition, forecasting, document intelligence, recommendation systems, signal processing and others. The breakthroughs of ANN and deep learning where boosted by the technological evolutions in hardware and processing units available in digital computers, as these models perform highly complex computational tasks. Their computational and

generalization power stems from the fact that instead of conventional bit registers, deep learning techniques are based on continuous vectors and tensors transformed in high dimensional spaces. As a digital computer is limited in bit registers, ANN currently settles with approximations of continuous computations. The applications of the domain are so vast that the scientific community is already on the way to create specialized hardware dedicated to deep learning techniques, the Neuromorphic electronic systems, which is fundamentally analog in nature[1,2].

The computational complexity of ANNs is what makes them perfect quantitates for implementation on a quantum computer. As quantum computing and quantum information processing, a closely related field to QML, are currently undergoing their transition from academic to industrial subjects of interest, one can easily detect the growing interest in combining one state-of-the-art technological advancement with another. In fact, quantum mechanics naturally carry properties that enable the representation and storage of large complex-valued vectors, tensors and matrices. As naturally they also perform linear operations on such vectors with a computational power that meets no classical equivalent. Therefore, it is legitimate to assume that a quantum equivalent to ANN, a Quantum Neural Network (QNN) will demonstrate an exponential increase in memory storage or an exponential decrease in algorithm's runtime. In fact, before an applicable QNN available yet, several ideas have already made their appearance for possible applications[3–6]. Due to the author's personal interest on ANNs as well as all the exciting possibilities a QNN demonstrates, this thesis is mainly devoted to the QML models related to possible QNN implementations.

The first reasonable thing to do in order to explore the potentials of a QNN is to define what a QNN is. And this is where the troubles begin. Although there is an extensive variety in QML literature concerning QNNs, starting from the early 90's, the field endures a confused state of further blurriness and hasn't reach the maturity of other QNN fields yet. The majority of papers till 2014 followed a "mimic" approach, trying to translate classical ANN architectures in a quantum computer and recreate ANN strategies in a quantum formulation aiming to accelerate their performance. More recent approaches though turned their focus on creating quantum models that can perform the same tasks an ANN can perform without concerning must about the resemblance of these models to their classical counterparts.

In the first complete and coherent systematic review in the domain of QNN research[7], the authors argue that variety and diversity in the proposed QNN models forbids a strict definition at the moment, and argued that in a general framework a QNN should satisfy three basic requirements. First, the initial state of the quantum system should be able to encode any binary string of length N. Moreover, the QNN should reflect one or more basic neural computing mechanisms and, finally, system's evolution must be based on quantum effects, such as superposition, entanglement and interference, and be fully consistent with quantum theory.

This last one requirement is extremely crucial in order for a QNN to actually be quantum but it also points out the first fundamental difference between QNN and ANN. In ANNs non-linearities that come in the form of activation functions in the neurons play a crucial role and provide the ANN with its power to detect complex patterns in data. On the contrary, quantum mechanics are in principle linear. Even if some techniques have been proposed to resolve this issue and add non-linearities to a quantum model through "back-channels", there are still other obstacles to overcome especially when it comes to QNN applied to classical data.

The dissipative nature of the deterministic ANN, where once you move to the next iteration and the weight values are updated, there is no way to turn back, is incompatible with the time symmetry unitary operators ensure. Moreover, ANN's training is based on the difference between its output and the target value (supervised learning). The detection of this difference requires a measurement. A measurement's effect on a quantum system, however, would cause the system's superposition to collapse, basically transforming it to a classical system.

In their present form, the closest a QNN comes to an ANN is probably in the level of neurons. And although it has been demonstrated that the capacity of a QNN in most cases is approximately the same with an ANN[8]'s, numerical experiments indicate that a QNN can demonstrate greater generalization performance and tolerance to noise[9]. All these issues will be analysed extensively in Sections 3 and 4. In one last stop, before dive deeper, let's take a look at the history of QNN.

1.4 The History of QNN

The first vision of a QNN came from Kak[10] in 1995, followed by Narayanan and Menneer's[11] neural network inspired by quantum processes. Chrisley[12] followed with the third publication of the year about QNN which was also, totally theoretical. It took three years for the first comprehensive study of a QNN mode which was finally done by Menneer for the needs of his master thesis.

In 2000 Ezhov and Ventura published a Hopfield QNN[13] with quantum associative memory inspired by Everett's parallel universes interpretation of the quantum world, which theoretically gains an exponential speedup by deploying Grover's algorithm in the recall process. The same year Narayanan and Menneer[9], inspired by the double-slit experiment, also considered the notion of a set of one-parameter MLPs combined to a QNN whose weights are superpositions of the weights of all perceptrons existing in parallel universes. They also suggested the 4 groups of QNN architectures which will be discussed in Section 2 and survive until today. Mitja Perus[14] compared the neuron's input-to-output function of weighted sums, without an activation function or a threshold, with Green's function that describes the Time-evolution of a quantum state.

The first complete proposal of a QNN came in 2000 from Elizabeth Behrman[15] and her coworkers, who developed Perus idea. It is a Time-array Neural Network of one "quron" -a quantum neuron - that consists of 1 qubit propagated. Green's function formulated by Feynman path integral forms the input-to-output function of the quron and the weights are engineered by the system's interaction with the environment and updated based on a common backpropagation with gradient descent rule. The physical implementation proposed for this model is a quantum dot interacting with photons.

In 2000 and 2002 Trugenberger[16,17] brought back Menner's Hopfield QNN by introducing the decoding of hamming distance between an input state and each memorized pattern in the Hamiltonian time evolution and utilizing the measurement to add non-linearities to the model. In the meantime, Altaisky[18] envisaged the first quantum perceptron, in a quantum formalism that tries to mimic the mathematical description of the classical one and was the first one to state the problem of non-linearity in QNNs. Later Fei et al.[19] improved this idea by

introducing the delta rule as a learning rule for the quantum perceptron, although they also failed to suggest an activation function.

In 2002 Gupta and Zia[8] utilized the Deutsh algorithm in quantum circuits proposed as QNN dedicated to quantum computations. Their model is computationally powerful and opens the door for a family of QNN models based on quantum circuits, yet this QNN carries no ANN attributes and no learning can be achieved. A year later an abstract yet pioneer proposal of a QNN came from Ricks and Ventura[20] . The authors considered a QNN that consists of several MLPs in superposition and uses Grover's algorithm for learning.

In 2006 Zhou et al.[21] developed a quantum perceptron based on the quantum phase adequately that was the first quantum perceptron proposed which could compute the XOR problem, a problem intractable by the classical perceptron. However, this approach inherited the iterative nature of its classical counterpart, which at the time was considered a big disadvantage in the race for quantum speedups. Not long after that, Oliveira, Silva et al.[22] developed a QNN model based on weightless NNs which stipulates the existence of a quantum random access memory (qRAM) followed by Sagher and Metwally[23]' s QNN based on entanglement, where the information transferred from a neuron to the other is based on original teleportation protocol. They also discuss a theoretical application in the prediction of virus spread.

As soon as the new decade came, the first suggestion about non-linear operators appeared by Panella and Martinelli[24]. They suggested general non-linear quantum operators to build a feedforward QNN with successively entangled qubits. In 2013 an autonomous quantum perceptron[25] with the weights encoded in the amplitudes of the state , mathematically, proved able to solve the XOR problem, also requiring iterations to give a solution. The same year offered a quantum MLP with 3 layers based on data encoded in the quantum basis and weights in the amplitude of the quantum state, which uses Grover's algorithm to detect the optimal weights[26]. Although this idea comes in a nice mathematical framework, this design requires an immersive number of qubits and gates to be realized, therefore the idea was abandoned.

2014 was an important year for QML and QNNs. Peter Wittek published the very first book devoted to QML[27] and Schuld et. al [7] published the first systematic approach on the QNN research, followed by a second one by Altaisky[28], although the second one focuses more on

quantum Hopfield NN and comes with a quantum Dot NN proposal. The same year quantum deep learning made its first appearance in literature by Wiebe et al.[29] who developed two algorithms to efficiently train Random Boltzmann machines based on amplitude amplification and quantum Gibbs sampling. In 2015 an abstract idea for a QNN based on Hamming NN for unsupervised learning was published by Zidan[30]. Later on, the idea of a quantum perceptron is enhanced by introducing unitary weights[31] (Seow Behrman and Steck). Boltzmann machines and quantum annealing techniques for deep learning were first introduced by Maxwell and Adachi[32].

The first compact and mature review of QML so far was published by Biamonte et. al[33] at 2017. The very same year the second book regarding QML was published by Dunjko and Briegel[34] followed by the third one by Maria Schuld[35], which among three QML books the literature has so far that, quoting Peter Wittek is "the one worth reading". A probabilistic QNN is proposed by Chen and Wang[36], but the proposal comes with the admission of a universal quantum computer of large scale required to implement this proposal. A general framework for the quantum adaption of a feedforward MLP using quantum circuit models is given by Wan[37].

At the end of the year, Cao and partners[38] proposed the first strategy that employs angle and amplitude encoding to form a strategy to add non-linear activation functions to quantum qurons that could potentially be combined to form an ANN. This strategy was improved in 2018 by Wei Hu[39] who added an alternation of the above strategy to build an activation function for a quantum neuron that approximates the ReLu function.

Returning to 2015 McClean et. al[40] set the framework for variational hybrid learning algorithms. A QNN designed in this framework based on a variational quantum circuit and focused on near-term applications was proposed by Farhi and Neven[41] in 2018 and in the same work, the quantum analogous to Universal Approximation Theorem was proved for QNNs of this architecture. In 2019 Mitarai ae al[42]. as well as Schuld et. al.[43] published their work regarding learnable QNN models in quantum circuits for regression and classification tasks respectively, trained with gradient-based methods.

Along with these, 2018 was a year rich in publications regarding QNN. A Quantum Generative Adversarial Network[44] based on the Variational Circuit Model was introduced by Seth Lloyd at the end of the year. The idea of a quantum Hopfield NN for quantum Hebbian

learning was revisited by Redentrost et al.[45] based on amplitude encoding and defining the quantum system's energy function as a cost function. It also introduced classical techniques to control the complexity. In a nice review of coherent QML algorithms with firm mathematical foundations and provable speedups Biamonte, Wittek et. al[33] discuss in length the potentials of quantum deep learning in a Boltzmann machine, underling that there is a significantly higher chance to apply these techniques before too long if restricted to quantum data.

The first idea for QML in feature Hilbert spaces proposed by Schuld and Kiloran[46] in May 2018 and then two months later by Schuld, Kiloran and associates[47] a proposal was published about a QNN based on variational quantum circuit architecture. This idea was revisited, improved and experimentally tested in a quantum computer simulator by Dendukuri and his partners[48] in May of 2019. In the same year came an interesting alternative to what has been proposed so far, a quantum-assisted Helmholtz machine than can provide assistance and speedups by deploying classical deep learning to extract low-dimensional representations of data, suitable for quantum computing. This approach is referenced as quantum-assistant ML[49].

In march 2019 Tacchino and his team[50] successfully implemented a classical perceptron on a quantum processor, using IBM's cloud 5-qubit computer. Finally, Verdon et. al[51], improved the initial study of parameter initialization in variational quantum algorithms published by Guerreschi and Smelaynskiy[52], and developed a case of hybrid quantum-classical ML. Based on Quantum Approximate Optimization Algorithm, they trained a Recurrent Neural Network to assist in heuristic parametrization of a quantum model for ML, such a QNN or in simpler cases, as the example given by the authors, a Variational Classifier. And that leads us to today.

2 An Introduction to Quantum Machine Learning so far

In Section 2 QML in general will be discussed. In the beginning, some introductory concepts and definitions shall be given. The remaining three subsections are dedicated to approaches to design quantum models for Inference and Learning and possible methods to optimize and train these models. Section 2 is inspired by the book "*Supervised Learning with Quantum Computers"*[35] by Maria Schuld and Francesco Petruccione , published in October of 2018. This book played a crucial role in the progress of this thesis as it consists of the first concrete and mature book about Quantum Machine Learning that gathers together the scattered literature of more than 20 years' research in the domain. All the existent algorithms and techniques for supervised QML are classified into three groups; quantum inference, quantum training and quantum learning. Within each group, the various approaches are furtherly distinguished based on the different quantum interpretations, schemas and properties they use. Since this is the first attempt to organize the field in subdomains and there is no alternative approach to consider so far, Section 2 will follow closely the structure and content of book's respective chapters and thus it could be considered an outline or a brief summary of the book.

2.1 Basic Concepts and Definitions

2.1.1 Types of Learning

In ML and QML "learning" is the process of generalization in an algorithm-the ability to perform accurately a task, trained by a set of *N* examples, to all the possible other data of a similar type as the given example. Learning frameworks started as deeply mathematical algorithms, inspired by human learning techniques and statistics. In classical ML there is now an inspiring variety including Transfer Learning, Online Learning, Supervised Learning, Unsupervised Learning, Reinforcement Learning, Semi-supervised Learning, PU Learning,

One-shot Learning, Hebbian Learning, Deep Learning, Imitation Learning, Multi-Label Learning and more. They all stem from the basic three categories; Supervised, Unsupervised and Reinforcement Learning. Not so much variety is met in the QML domain, yet there has been progress in the three leading learning types.

Unsupervised learning is self-organised learning in the sense that the train data provided to the algorithm carry no specified information about their labels or their within relationship and the model tries to detect an underlying structure in the data, correlations, patterns or tendencies. Unsupervised learning allows probability densities of the inputs to be modelled and can be seen as a type of Hebbian learning. Examples of unsupervised learning are clustering tasks, anomaly detection and signal separation. In QML unsupervised learning techniques proposed so far include a q-means algorithm[53], which is a quantum equivalent to k-means clustering algorithm, quantum Principal Component Analysis[54], hybrid clustering algorithms[55], quantum anomaly detection[56,57] and more. Yet the majority of unsupervised techniques are naturally more intriguing, due to the lack of previous knowledge for the data, thus such techniques are not expected to be the first applications of QML to be realized in a quantum computer.

In reinforcement learning, which belongs to the semi-supervised family, the learning process is assisted by an exogenous agent that provides feedback, positive or negative, to the algorithm to penalize bad decisions and provide a reward to the good ones, based on the targeted output. Reinforcement learning is usually used in tasks where information about the target already exists. It is the most intriguing yet tricky among the three basic learning paradigms. In QML the majority of proposals focus on a quantum-assisted approach, where a quantum agent interacts with a classical environment that provides the feedback for his actions. This way the agent is assisted to adapt its behaviour towards the goal result. The main potential in the quantum reinforcement learning comes either from the processing capabilities the agent inherits from its quantum nature or from a possible superposition in the environment's data. It is the most recently developed area of QML, although steady progress is made with implementations of quantum reinforcement learning suggested for several quantum protocols[58–60] . The relevant literature met a rise of interest in the last two years, with variant proposals for quantum reinforcement learning implementation schemes[61–63].

Finally, the dominant type of learning is supervised. In supervised learning the algorithm is trained with examples accompanied by information regarding the desired output of the given task. Supervised learning models are basically functions that map inputs to outputs based on known pairs of inputs and outputs, like classification, regression and anomaly detection. The domination of supervised learning holds on the QML and will be extensively explored for the rest of this thesis.

2.1.2 Strategies for QML algorithms

To build a quantum algorithm is a tricky task. To build a QML algorithm is even trickier mainly due to the fact that ML algorithms are fundamentally complex both mathematically as well as conceptually in order to be generic enough and applicable to a variety of data forms and tasks. In QML literature one can distinguish two approaches in algorithmic design; translate classical algorithms to quantum ones or create quantum algorithms based on a quantum machine's abilities.

The first approach dominated the domain for more than 15 years. The goal is to build a quantum translation of a given classical model to reproduce the same results or outsource the computation or parts of the classical algorithm to a quantum computer. It is a speedup-orientated strategy that considers QML as an application of quantum computing rather than a scientific field. The challenge is, therefore, to create quantum routines that "mimic" the classical ones and produce the same results with the lowest possible resources. The models for inference presented in Section 3 follow this approach.

The second approach on the other side, named exploratory approach, considers QML a freestanding domain of quantum theory and a field of research. This is the modern approach, which gained interest in the last decade. The concept of this approach is that instead of focusing on the classical algorithms and force a quantum translation, the building of a QML algorithm should start from the quantum device and be designed based on what kind of QML models and ML tasks this device is capable of. The aim, in this case, is not necessarily speedups, but to contribute to the ML community in any possible way, by offering new inspiring techniques or models that can handle classically intractable problems. Thus, it does not rely explicitly on digital, universal quantum computers but considers any system obeying to quantum laws, such as annealers, to train and implement a model suitable for learning. The training techniques and learnable models of Section 3 are examples of the explanatory approach.

2.1.3 Types of QML Definitions

In a first effort to make the definition of Quantum Machine Learning a bit less abstract, and categorize the proposed QML algorithms in fundamental classes, Menneer and Narayanan[9] developed the *C-Q* notation to declare classical and quantum part combinations in a model. This notation was later developed to the "four approaches" typology by Aïmeur, Brassard and Gambs[64] which is by now generally established in the domain.

CC refers to Classical data being Classically processed and includes traditional ML methods which are inspired by quantum information research. Such cases are tensor networks[65] which can be used for training an ANN or optimization of ML routines[66]. In the same manner, QC concerns Quantum data that are Classically processed in a way that ML is deployed to assist quantum computing. It displays several applications like ANN used to learn phase transitions in many-body systems[67], discriminate quantum states emitted by a source[68] or detect entanglement in a quantum system[69] and many more.

QQ, on the other hand, is for Quantum data being Quantumly processed. These quantum data can be the measurements of a quantum system in a physical experiment or derive from simulations in a quantum computer. Such an example is the usage of quantum annealers as sampling devices. Finally, in the CQ case, which will be mainly concerned in this thesis, the dataset is Classical and it can be of any kind such as text, image, audio or simply numeric values and a quantum machine is used to analyse these data. This hybrid case accommodates models from exploratory as well as translation approaches and aims to build algorithms to cope with classical ML issues. The challenge is that it requires a quantum-classical interface and classical information must be encoded in the quantum system. This information encoding ends in being a part of these algorithms and effects heavily their efficiency and complexity. In the next subsection information encoding techniques for the CQ cases are presented.

2.1.4 Information Encoding

In the CQ case, the classical data's features must be converted to a representation that is suitable for quantum processing, and also to be readable by the quantum machine. The CQ case will be revisited many times as the QNNs discussed in Section 3 are designed in this spectrum to be applicable to existing real-life ML-solvable problems. In fact, the majority of QML algorithms proposed so far concern the CQ approach and there are two reasons for that; one the one hand, the vast majority of data produced at this moment are classical and not

quantum, therefore a real assistance to ML should be based on what is available and mainly used at the moment in order to provide practical solutions to real problems. On the other hand, almost any QQ model can also work with classical data, and, in fact, this is a much simpler case as will be evident later on. The CQ case can be seen as a special category of QQ, that requires an extra step to prepare the classical data in a quantum form (data representation) and load it to the quantum device (data transfer to hardware), which in QML field is named "state preparation".

There are four main strategies to encode classical information in a quantum system at the moment and all four of them shall be discussed. It is important to note that state preparation routines are actually the first part of a QML CQ algorithm, and therefore this choice affects both the design and the complexity of the algorithm. Furtherly, the probabilistic nature of a quantum world inherited by most QML models means that the results must be estimated by numerous repetitions of the algorithm. And due to no-cloning theorem, the state preparation of data must be repeated too, for every repetition of the algorithm. Thus, possible speedups provided by a QML model heavily depend on the cost of state preparation, making the choice of strategy for this procedure even more crucial. In ML the efficiency of an algorithm depends on the dimension d of the data and size N of the dataset to be processed. In QML, the efficiency of the algorithm is based on qubits and the goal is to keep it polynomial in their number. Since in a quantum system both the qubit as well as the amplitudes can encode data, QML algorithms can be distinguished in amplitude-efficient and qubit-efficient.

The encoding strategies proposed so far are; Amplitude encoding and Hamiltonian encoding in case of continuous data and Basis encoding or Qsample for binary data. The first three approaches are used to represent the full dataset in the quantum system, while Qsamples represent probability distributions over random variables. In order to discuss all four approaches on the same basis, an n-qubit system with initial state $|0000..00\rangle$ and a d-dimensional dataset D of N samples are assumed. Under these assumptions, the following table provides a quick outlook of the best runtime estimations for each strategy.

Table 1 Comparison of the 4 encoding strategies

Encoding Strategy	Number of qubts	Runtime for state prep	Input features
Basis	d	O(dN)	Binary
Amplitude	logd	O(dN)/O(log(dN))	Continuous
Hamiltonian	d	$O(2^d)$/O(d)	Binary
Qsample	logd	O(dN)/O(log(dN))	Continuous

Basis Encoding

In the basis encoding strategy, as the name implies, the classical data are represented in the quantum state of a system. This can be applied in binary data where a classic sample's features are binary encoded and the resulting n-bit string is associated with an n-qubit string. There are two possible approaches met in the literature for basis encoding; represent each one of the *N* inputs of the dataset in a state and train the algorithm with one input at a time or bring all inputs in a superposition state. This strategy is mainly employed in cases where non-linearities are aimed, like in QNN's activation functions.

The first tactic is rather straight forward and is used in the majority of QNN who aim to be implemented in near-term applications. Given a binary data *x*, the input state that represents this data is $|x\rangle$. The second tactic offers a natural speedup to the algorithm since it allows the parallel process of the full dataset at once by the quantum model, which has no classical equivalent. The initial input state of superposition of the quantum system, which carries the information for the whole dataset *D* can be defined as

$$|D\rangle = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |x_i\rangle, (2.1)$$

where $x_i \in D$ is encoded in state $|x_i\rangle$, as mentioned. The amplitudes in state (2.1) are $\frac{1}{\sqrt{N}}$ for basis states corresponding to an input data and 0 for the remaining states, which in general is expected to give rise to a sparse amplitude vector.

The main problem with the second case is that in order to take advantage of the quantum superposition, a lot of resources must be consumed to prepare, manipulate and store the superpositional state of data in a quantum register. There are some proposals about devices that

load patterns in parallel in a quantum register, called qRAMS but they are still not implementable in experimental conditions mainly due to hardware issues which will be discussed in Section 4. Other strategies about state preparation of inputs in superposition proposed can be found in Appendix A.

These strategies require a complex procedure of *O(dN)* steps to be repeated *N* times in a quantum system with three registers. For an average dataset of 1000 samples and 40 features, the quantum circuit required to realize the procedure is of depth greater than 2000. Both the cost as well as the complexity of those strategies make the encoding of data in a superposition a task very demanding for the expected near-term hardware. They rise significantly the complexity of the QML algorithm, possibly vanishing the speedups gained through superposition. That is why the recent literature aiming in realistic near-term application of QNNs mainly exploits the first tactic.

Amplitude Encoding

Amplitude encoding is also self-described by the name and is an encoding technique used by a large number of QML algorithms, mainly because when combined with amplitude amplification techniques like Grover's algorithm provides possible exponential speedups. It also allows the data to be complex-valued vectors, enabling continuous features' representation, with the constraint that these vectors must be normalized.

A normalized classical vector $x \in C^{2d}$ can be represented by the amplitudes of a quantum state $|\psi\rangle$ in the Hilbert space via:

$$x = (x_1 ... x_{2d})^T \leftrightarrow |\psi_x\rangle = \sum_{i=1}^{2d} x_i |i\rangle, (2.2)$$

Similarly, a classical matrix $A \in C^{2d \times 2n}$ with normalized entries, that is $\sum_{ij} |a_{ij}|^2 = 1$, can be encoded as:

$$|\psi_A\rangle = \sum_{i=1}^{2N} \sum_{j=1}^{2d} a_{ij} |i\rangle |j\rangle, (2.3)$$

Therefore, the whole dataset of *N d*-dimensional samples *D* can be encoded in the amplitudes of the initial quantum state:

$$|\psi_D\rangle = \sum_{i=1}^{N} \sum_{j=1}^{d} x_j^i |j\rangle |i\rangle = \sum_{n=1}^{Nd} |\psi_{x^n}\rangle |n\rangle, (2.4)$$

As it follows, the initial quantum state, considered the "input" state of a quantum algorithm, has a *dN*-dimensional amplitude vector. The output can also be basis encoded in an extra qubit and be entangled to the $|n\rangle$ register in the initial state or also amplitude encoded in a separated quantum register.

This routine can be summarized as the preparation of an initial arbitrary state $|\psi\rangle = \sum_i a_i |i\rangle$, where *a* is the amplitude vector of *dN* dimensions resulted from the concertation of all data inputs. Routines to do so efficiently and robustly vary, depending on whether the runtime or the number of qubits required is of essence. In general, amplitude encoding techniques are very complex and the depth of an arbitrary state preparation circuit is computed to be 2^{n-1}.[70] A linear in time routine can be obtained by reversing a quantum algorithm developed for mapping an arbitrary state to a ground state, but is not qubit-efficient as the number of operations scales exponentially to the number of qubits, thus the dimensions of the data. Qubit-efficient or at least with logarithmical dependency to the data's dimensions state preparation routines have been proposed based on parallelism, oracles or the usage of a qRAM memory, but they also exhibit caveats in terms of imposing extreme constraints for the data's structure or depend on hardware that is unavailable for the moment being.

The main advantage of amplitude encoding is that it allows the data to be continuous. Moreover, the qubits required for the representation of data are logarithmic to the multiple of dataset size and data's dimension, therefore such algorithms are qubit-efficient, as long as so is the state preparation routine. In fact, there are some cases when, if the data is of a certain structure, that can be achieved, but in general case the number of qubits required for the preparation of amplitude encoded data varies. Another thing to consider is the ability to retrieve information from amplitudes. If for example the result of a model is encoded only in the amplitude a_i, the number of measurements needed to retrieve this result scale with the number of amplitudes, thus the number of data's size. This is why algorithms based on amplitude encoding usually add an extra ancilla qubit as the results carrier. Finally, due to the unitarity that must be preserved in quantum systems, this type of encoding prohibits any non-linear map to be implemented in a unitary fashion.

Qsamples

Qsample encoding is used in probabilistic QML models and quantum Boltzmann Machines. The idea is that the process of measuring *n* qubits in the computational basis is equivalent to the process of sampling an *n*-bit string from a discrete probability distribution. Thus the amplitude vector can be used to represent a classical discrete probability distribution $p_1, p_2, \ldots p_d$. This can be seen as a hybrid method combining both basis as well as amplitude encoding, since the information the model is interested in is encoded in amplitudes, using one of the state preparation routines for amplitude encoding, while the *d* features are encoded in qubits. The state whose amplitude vector encodes this distribution is called qsample.

Qsample encoding exploits the probabilistic nature of quantum word to obtain easy manipulations of a classical discrete distribution via the quantum manipulations. Join distributions can be easily prepared, marginalization over qubits occurs in a natural way, rejection sampling can be performed and more. The QNN models that use Qsample encoding are those that are proposed for deep learning in a quantum Boltzmann Machine and this is not the case this thesis focuses on, for various reasons mentioned in Section 1 and Section 3. Thus, there is no need to delve into how qsamples are used to perform the above manipulations and we shall proceed to the 4rth and final strategy; Hamiltonian Encoding.

Hamiltonian Encoding

Hamiltonian encoding, also called Dynamic encoding, is used in the final QNN family presented in Section 3, the Continuous-Variable QNN. This strategy differentiates itself from the previous approaches as it implicitly encodes the feature's information by letting them define the evolution of the quantum system. To be more specific, instead of preparing an initial state, the Hamiltonian that describes the evolution applied to the state is prepared. To do so, the Hamiltonian *H* is associated with the dataset represented in a -possible pre-processed to become Hermitian- data matrix *A*, assuming that data are tidy- each row represents a sample and each column a feature- or a Gram matrix A^TA. Hamiltonian encoding offers to the algorithm the ability to extract eigenvalues of the feature-matrix or multiply these eigenvalues to the amplitude vector as well as several other interesting potentials.

If, for simplicity and efficiency, consider only the case of time-dependent Hamiltonians, any unitary transformation can be described as $e^{-iH_A t}$ and therefore, Hamiltonian encoding depends on the ability to implement in a quantum computer the evolution

$$|\psi'\rangle = e^{-iH_A t}\,|\psi\rangle, (2.5)$$

If the initial state of *n* qubits is $|\psi\rangle$, then $|\psi'\rangle$ is the quantum state that carries the information that *H* indicates.

The process of implementing (2.5) in a quantum hardware is called "Hamiltonian Simulation" and in case of gate-based quantum computers, it relates to the task of efficiently decompose a unitary in elementary and easy to implement gates. Note that, in case *A* is not a Hermitian matrix,

$$\tilde{A} = \begin{bmatrix} 0 & A \\ A^{\dagger} & 0 \end{bmatrix}$$

can be used instead, and only part of the output should be considered. This trick allows the eigenvalues of *A* to be processed by the quantum machine.

Polynomial time Hamiltonian simulations are based on the first-order Suzuki-Trotter formula to cope with the fact that the factorisation rule does not stand for scalar exponentials[71]. That reads, if a Hamiltonian *H* is be written as a sum of elementary Hamiltonians $H = \sum_{i=1}^{T} H_i$, one cannot assume $e^{-i\sum_{i=1}^{T} H_i t} = \prod_{i=1}^{T} e^{-i\sum_{i=1}^{T} H_i t}$. Suzuki-Trotter formula states that if the factorization is broken into small time-steps so that the Hamiltonian is expressed as a sequence of small time-steps, an approximation of the Hamiltonian with negligible error at each step can be achieved. However, such a decomposition of a Hamiltonian in elementary and easy to simulate Hamiltonians is not trivial and, also, the smaller the required approximation error, the smaller the length of the time-step has to be; thus, the whole sequence has to be repeated several times.

Furthermore, since the Hamiltonian is used to encode a data matrix, it is important to keep the runtime logarithmic to the dimension of the dataset and thus the Hamiltonian's. Qubit-

efficient Hamiltonian simulations have also been proposed for the special cases of sparse[72] or low-rank[54] Hamiltonians. In the first case, an oracle-based procedure is combined with quite demanding techniques to ensure an almost linear dependency of simulation's runtime to the number of qubits. Therefore, a logarithmic dependency in the dimension of the data matrix encoded in H is achieved. For dense low-rank Hamiltonians, the data are encoded to the Hamiltonian via a density matrix and the density matrix exponentiation routine is used along with a phase estimation routine to design a simulation technique approximately equivalent to the simulation of a swap operator. The proposed design can guarantee a logarithmic dependency given the ability to qubit- efficiently encode a non-sparse density matrix in the Hamiltonian as well as assuming a high redundancy of data.

2.2 Quantum Models for Inference

This chapter is focused on useful approaches to build a QML algorithm for Inference, that is an algorithm used for the application of input-to-output maps or probability distributions in supervised learning tasks. Specifically, given an input x with a label or target y, to apply $f(x)=y$ for some f or $p(y|x)$ respectively. The QML algorithms, routines or strategies of this subsection can be seen as building blocks of a learnable QML model or they can be combined with a classical ML algorithm to compute its output and precision. They all follow the translating strategy in the sense that they explore ways to apply a classical ML technique for inference in a quantum machine.

2.2.1 Linear Models

A linear model is a parametrised linear function that maps an n-dimensional input x to a d-dimensional output y, given a set of $n+1$ weight parameters. Linear models are used in classification or regression tasks and they can also be seen as a linear layer in an NN with identity activation functions in the neurons. Mathematically speaking, a linear model is described via;

$$f(x;w) = w^T x = w_0 + \sum_{i=1}^{n} w_i x_i, \quad (2.6)$$

Interestingly, in the case of $d=1$, that is a scalar output, (2.6) expresses the inner product of the input vector and the weight vector. So, how could a linear model like (2.6) be realized in a quantum computer?

Linear Models in Amplitude Encoding

Consider the case were amplitude encoding is used to encode the input vector *x* in the amplitudes of state $|\psi_x\rangle$ and the weight vector *w* in state $|\psi_w\rangle$. Then (2.6) could be naturally computed in a quantum way by;

$$\langle \psi_w | \psi_x \rangle = f(x; w) = w^T x, (2.7)$$

Although the quantum representation of a linear model came quite naturally, the output of this model cannot be trivially obtained, because measuring the inner products of quantum states is not straightforward. Small quantum circuits that use interference between different branches of a superposition of the initial states are deployed for this task[73].

One way to achieve so is by deploying interference circuits, small quantum circuits that use an ancilla qubit and prepare the superposition of initial states in one register, then apply a Hadamard gate on the ancilla to entangle it with state $|\psi_{x+w}\rangle$ and by measuring the probability of the ancilla state $|0\rangle$, the inner product is indirectly computed. Note that in case of unit vectors their sum and inner product are related via $(w+x)^T(w+x) = 2 + 2w^T x$.

The most well-known technique to obtain $\langle\psi_w|\psi_x\rangle$ is the swap test, which returns the absolute value of (2.7), assuming the states are separated. This can be done if an ancilla qubit is used to create initial state $|0\rangle|\psi_w\rangle|\psi_x\rangle$, prepare a superposition of the ancilla and then create two branches in the superposition, one “swapped” and one “unswapped”. By computing the sum between the swapped and unswapped states in one register and their difference in another, the measurement of the ancilla qubit p_0 reveals the desideratum

$$p_0 = \frac{1}{2} - \frac{1}{2} |\langle \psi_w | \psi_x \rangle|^2, (2.8)$$

Note that the same routine with slight alternations can be applied for mixed states also.

Linear Models in hybrid Amplitude and Hamiltonian Encoding

For this representation, consider the input vector amplitude encoded in the initial state $|\psi_x\rangle$ while the weight vector is encoded in the Hamiltonian of the system, that is a unitary *W* or in general case a quantum circuit. In the first case, a unitary linear model could be realized by

computing the inner product of the final state $W|\psi_x\rangle$ with a uniform distribution $|\psi_w\rangle = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}|i\rangle$, so that the linear model can finally be described as:

$$f(x;w) = \langle\psi_w|W|\psi_x\rangle, (2.9)$$

In order to obtain a scalar output, one just has to sum up the elements of $W|\psi_x\rangle$. A quantum model of (2.9) form can be trained to find the function that accurately fits the right input to the right output by choosing the optimal unitary or circuit.

If unitarity is not respected, a non-unitary quantum circuit can be interpreted as a graphical representation of a neural network. Non-unitary transformations are functions that map an input of *n*-dimensions to an output of $d<n$ dimensions, thus they reduce the dimensionality and the output can be considered a subsystem of system's initial qubits. In order to explain this representation, let's consider a density matrix ρ_x, who encodes the input vector *x* in its diagonal. This can be achieved by encoding the square roots of input vector's elements in the amplitudes of a pure state of $N=logn$ qubits and compute its density matrix. In this formulation the unitary evolution $|\psi_{out}\rangle = W|\psi_x\rangle$ of (2.9) becomes $\rho_{out} = W\rho_x W^T$.

Via density matrices the dimension can be reduced by simply tracing out the first *N-D* qubits to end up with the $D=logd$ qubits of the output state, encoded in a "hidden layer density matrix" $\rho_{hid} = tr_{1,\ldots,N-D}\{\rho_{out}\}$. Vice-versa the dimension can also be increased by adding qubits to the hidden layer density matrix and extend the evolution to these new qubits by $\rho_{hid} \otimes |00..0\rangle\langle 00..0|$. All these can be easily understood if the graphical representation of a NN in Figure 1is used.

Consider the initial state of the quantum system to be $|in\rangle = |q_1, q_2, q_3\rangle$ and the diagonal of α density matrix $\rho_\alpha = |\alpha\rangle\langle\alpha| = |\alpha_1, \alpha_2, \alpha_3\rangle\langle\alpha_1, \alpha_2, \alpha_3|$ at any given point of the evolution to be a layer of neurons. At first a unitary evolution U_1 (linear transformation) forms the intermediate layer $|h_1><h_1| = U_1|in><in|U_1^\dagger$. Tracing out the first qubit leads as to the hidden layer $|h><h| = tr_1\{|h_1><h_1|\}$ followed by joining a qubit in the ground state that stands as the 2nd intermediate layer $|h_2><h_2| = |h_2><h_2| \otimes |q_4 = 0><q_4 = 0|$. Note that the density matrix has again 8 diagonal elements, but four of them have a probability zero and the decision which

qubit is traced out defines the connectivity between a bigger and a smaller layer. Finally, a unitary U_2 on qubits q_2, q_3, q_4 results to the 'output layer' $|out><out| = U_2|h_2><h_2|U^{\dagger}_2$.

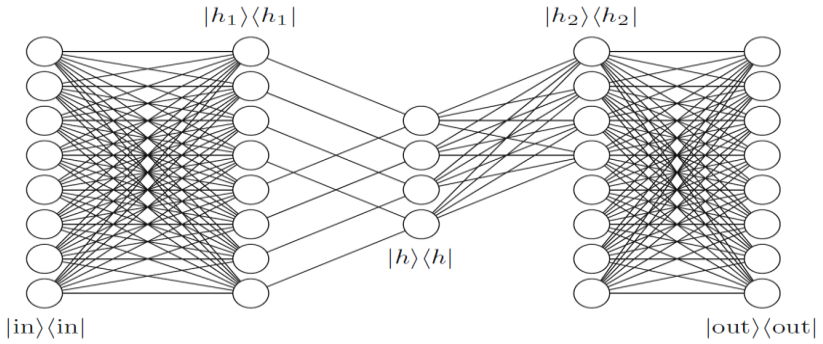


Figure 1The graphical representation of linear quantum circuit as a NN

Linear Models in hybrid Basis and Hamiltonian Encoding

In the final case, both the input vector and the output of the model are encoded in the state of the quantum system, so that a parametrised circuit $U(w)$ applied to an initial state of two registers $|x\rangle|00\ldots000\rangle$ results to the final desired state $|x\rangle|w^T x\rangle$. This can be done through a circuit $U(w)$ such that;

$$U(w) = U_n(w_n)...U_k(w_k)...U_1(w_1)U_0(w_0), (2.10) \quad \text{where}$$

$$U_k(w_k) = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i w_k} \end{bmatrix}, (2.11).$$

So, if the circuit is applied to the initial state, we obtain

$$U(w)|x\rangle|00...0\rangle = 2e^{2\pi i w^T x}|x\rangle|00...0\rangle, (2.12)$$

At this stage, a quantum phase estimation routine can be used to write the expected output of the algorithm in the second register. More details about this routine can be found in Appendix B. To do so, as a preparatory step the second register is prepared as an m-qubit ground state and it is put into a uniform superposition using a Hadamard gate on each qubit. Moreover, an oracle O is also needed in order to apply powers of the parametrised circuit conditioned on the qubits of the second register such as

$$|x\rangle|j\rangle \xrightarrow{O} U(w)^{j-1}|x\rangle|j\rangle, (2.13)$$

and prepare the state

$$\frac{1}{\sqrt{2^m}}\sum_{j=1}^{2m}|x\rangle e^{2\pi i(j-1)w^T x}\,|j\rangle, (2.14).$$

In the next step, via an inverse quantum Fourier transform and for a pre-chosen j that ensures $w^T x \approx (j-1)/2^m$, the result is an approximation of the desired output state $|x\rangle|w^T x\rangle$. Note that if $|w^T x\rangle = |q_1 q_2 \ldots q_m\rangle$, then $w^T x \approx q_1 \frac{1}{2^0} + ... + q_m \frac{1}{2^{m-1}}$, thus the output of the linear model has been computed. This representation has been implemented to simulate a perceptron on a quantum computer in 2014[74].

2.2.2 Kernel-based Models

A kernel is a symmetric positive defined function $\kappa : X \times X \to \mathbb{C}$, for a non-empty set X and is a widely used tactic in a variety of ML algorithms, mainly in order to construct feature maps based on these kernels to project the input data to higher dimensional hyperspaces and potentially increase their separability. Kernel models don't perform computations on the high-dimension hyperspaces but, in fact, keep the complexity as low as possible by limiting the computations in kernel's level which is that of input data.

Kernel methods don't depend on specific architectures of a model, they can be seen as general tools in a wider scope, as indicated by the Represented Theorem. Such is a feature map $\varphi : \mathrm{X} \to \mathbb{C}^x$, $x \to \kappa(.,x)$ which maps an input x to a complex-valued feature vector which is itself a function mapping the input x to a complex number. Feature maps can therefore be used to create a vector space with an inner product that carries the property:

$$\langle \varphi(x) \,|\, \varphi(x') \rangle = \langle \kappa(.,x), \kappa(.,x') \rangle = \kappa(x,x'), (2.15)$$

This can be translated as: A kernel of two inputs computes the inner product of these inputs in a feature space. Also (2.15) implies that a kernel can be constructed from every inner product[75].

This direct relation of kernels and inner products indicates that feature spaces and quantum Hilbert spaces express several similarities in mathematical formalism that can be exploited to build quantum kernels and expand to Hilbert feature spaces. In fact, we have already created quantum feature maps, without realising, in the Information Encoding

subsection. Every process to encode an input x to a quantum system, either via basis or amplitude encoding or any other encoding method can be seen as a feature map φ from the input's set to the Hilbert space of the state $|\varphi(x)\rangle$. So, an inner product of quantum states produced by encoded input data is always a quantum kernel and thus different encoding routines result to different kernels. Some example quantum kernels are:

Basis Encoding: $\kappa(x, x') = \langle x \mid x' \rangle = \delta_{ij}$

Amplitude Encoding: $\kappa(x, x') = \langle \psi_x \mid \psi_{x'} \rangle = x^T x'$

Angle Encoding: $\kappa(x, x') = \prod_{i=1}^{n} \cos(x_i - x_i')$

Several quantum kernel models can be designed based on different kernels and are described in their general form via

$$f(x) = \sum_{m=1}^{M} w_m \kappa(x, x^m), \quad (2.16)$$

where the weights and the inputs can be either classical or quantum parameters. Given the basic properties of the kernel function, a kernel is always a distance function, thus quantum models described by (2.16) called Distance-Based Quantum Classifiers.

Although the investigation of how quantum kernels and feature Hilbert spaces could assist QML is widely unexplored and only very recently draw some attention, several QML algorithms for implementation of Distance-Based Classifiers or kernel models have already been proposed[4,46,76] and one could disguise two approaches. The implicit approach uses a quantum device to evaluate the kernel in a classical algorithm, thus to implement a quantum circuit in charge for the state preparation routine and to compute the inner product of quantum states of the encoded inputs. One the other hand, the explicit approach aims at developing a learnable quantum classifier in feature Hilbert spaces via a Variational Circuit[77]. The Variational Circuits that will be presented in Section 3 constitute a more natural alternative for kernel methods in QNN. As it will be demonstrated, Variational Circuits can in principle explicitly compute in feature space instead of implicitly use kernel functions.

2.2.3 Probabilistic Models

So far, we've considered models for inference based on the application of an input to output function f. These models are called deterministic models. There is a second type of

models for supervised inference, those based on probability distributions of binary random variables, generative or discriminative, called probabilistic models. As mentioned, several times by now, quantum theory is a probabilistic theory and therefore it is a natural corollary to ask how probabilistic models can be realized in a quantum machine.

The answer to this question has actually already be given in Information Encoding. Qsample encoding is used to associate a probabilistic model with a quantum state through mapping the binary random variable to the qubit and consequently the sample of these variables to a quantum state. Therefore the probability *p(x,y)* is interpreted as the corresponding square amplitude and the probabilistic model translated in the quantum world is described as:

$$p(x,y) \Rightarrow |\, p(x,y)\rangle = \sum_{x,y} \sqrt{p(x,y)}\,|\, x,y\rangle, \quad (2.17)$$

In that representation measuring the qubits in the computational basis is equivalent to sampling from the distribution although that demands repeated preparations of the qsample.

Another advantage of qsamples is that they offer an elegant way to obtain the marginalized distribution of the labels *y* by tracing over the qubits associated with the input vector *x*, through

$$p(y) = \sum_{x} p(x,y) \Leftrightarrow tr_x\{|\, p(x,y)\rangle\langle p(x,y)\,|\}, \quad (2.18)$$

In order to use a model based on (2.17) for inference the goal is to be able to prepare a qsample $|p(y|e)\rangle$, for a given $x=e$ and a known qsample $|p(y|x)\rangle$, which unfortunately is not so easily done. Probabilistic models are usually combined with amplitude amplification techniques to achieve some sort of speedups and are exploited to create quantum Bayesian nets[78,79] and quantum Boltzmann machines[60,80,81]. The first case relies on the preparation of qsamples to describe distributions with conditional independence relations, while the distributions required in a Boltzmann machine require a good mean-field approximation.

2.3 Training Techniques for QML

In this subsection, four different approaches for quantum optimization and training of ML algorithms, both classical and quantum, will be presented. Each approach is inspired by a different quantum property and perspective and can respectively be combined with models for inference and learnable models. The first two approaches are mainly based on famous quantum algorithms that guarantee speedups, Harrow-Hassidim-Lloyd (HHL) algorithm for linear algebra calculations in a quantum computer and Grover's routine for search in unstructured datasets. The third one concerns classical gradient descent optimization techniques in hybrid classical- quantum training and finally the fourth deploys adiabatic methods to find the ground state which encodes the optimal result to a given problem. The output of a quantum training routine can be either quantum or classical, based mainly on the information encoding method used and the character of the ML algorithm this routine is applied to.

2.3.1 Based on Linear Algebra Calculus

QML literature developed a wide variety of optimization algorithms based on basic linear algebra routines, called quantum blas[45]. Quantum blas routines exploit the HHL algorithm[82] and its variations in linear algebra tasks like matrix multiplication, matrix inversion and singular value decomposition to create equivalent quantum subroutines such as quantum matrix inversion (HHL), density matrix exponentiation[83](HHL^{DME}) , quantum matrix multiplication (HHL^{-}) , quantum phase estimation or branch selection. These subroutines are combined to create quantum training algorithms. The basic idea relies on Hamiltonian encoding and the usage of quantum systems for linear algebra calculus, where the training input's matrix is encoded in the Hamiltonian of the system and can be therefore manipulated through the above quantum subroutines.

A summary of the HHL algorithm which forms the skeleton of these training schemes is that in order to solve a linear equation $Ax+b$, it encodes b into the amplitudes of a quantum state and matrix A in the Hamiltonian H_A. Then it applies an evolution $e^{iH_A t}$ to the state $|\psi_b\rangle$. The eigenvalues of H_A are extracted in basis encoding through quantum phase estimation (see Appendix B). In a next step, these eigenvalues are quantumly processed to be encoded as amplitudes and the final state is computed by measuring the amplitude of $|\psi_{Ab}\rangle$ (HHL^{-}). With

one extra step to invert the amplitudes the final state becomes $|\psi_{A^{-1}b}\rangle$. HHL with density matrix exponentiation (HHL^{DME}) is an alternative used for kernel matrices deployed in kernel methods, where the input matrix associated with the Hamiltonian is a kernel Gram matrix prepared as a density matrix.

Data matrix inversion based on HHL and HHL^{-} subroutines is the building block of quantum training algorithms proposed for optimization of both sparse[84] as well as low-rank[85] linear regression. The kernel matrix inversion realized by HHL, SWAP circuits and HHL^{DME} is employed to quantumly train Support Vector Machines[4] and Gaussian Processes[86] while adjacency matrix inversion achieved via HHL and HHL^{DME} can be used to train quantum or classical Hopfield Neural Networks[45]. Table 2 offers a summary of the above examples, which are only a few indicative examples of quantum optimization based on linear algebra.

An advantage of training algorithms based on quantum blas is that they inherit runtime speedups due to their logarithmic dependency in the inputs' dimension as well as the size of the input dataset. Their disadvantage is that the quantum subroutines they are based on require a large number of gates to be executed coherently, which due to the limited gate fidelity demand a full-blown fault tolerant quantum machine. This indicates that the above optimization techniques, although carry potentials for exponential speedups, are not indicated for near-term applications.

Classical Method	**Computation**	**Notation**	**Strategy**
Linear regression (sparse)	$(X^{\dagger}X)X^{\dagger}y$	*X:* the input matrix y: the output vector	Data Inversion via HHL+HHL^{-}
Linear Regression (low-rank)	$\sum_{r=1}^{R}\sigma_r^{-1}u_r v_r^T y$	u_r,v_r,σ_r:singular vectors/values of $X^T X$	Data Inversion via HHL^{DME}
Support Vector Machines	$K^{-1}y$	K: kernel matrix κ: kernel function	Kernel matrix inversion via HHL^{DME}
Gaussian Processes	$\kappa^{T}\mathrm{K}^{-1}y, \kappa^{T}\mathrm{K}^{-1}\kappa$		Kernel matrix inversion via HHL+SWAP
Hopfield Neural Network			Adjacency matrix inversion via HHL+ HHL^{DME}

Table 2 Summary of basic examples of linear algebra-based training techniques.

2.3.2 Based on quantum Search

In this second training scheme, the proposed algorithms aim to exploit the quadratic speedup offered by Grover's routine for amplitude amplification[87] in the optimization process. The main idea is that amplitude amplification can be used either to search for the optimal parameters which minimize a cost function in a variational quantum model, or to handle a superposition of the input data, or to search for the best among all possible models for a given task.

In the first case Dürr-Høyer proposed an alternation of Grover's routine[88]. This quantum subroutine has been used for k-Nearest Neighbours[89] methods as a search method to find the closest data points. The main idea is that given a distance function $C(x)$, where data inputs x are basis encoded in $|x\rangle$ and prepared in a superposition with $|x_{input}\rangle$, we want to search this superposition to obtain the binary string that minimizes $C(x)$, thus is the one closest to the input data x_{input}. An oracle is used to mark the $|x\rangle$ states for which $C(x)<C(x_{input})$ and an extra register saves and compares these states to return the optimal one. This can also be extended in a cost function $C(w)$, based on the weights of the model we want to optimize and in the same manner the search problem becomes an optimization problem in the sense of searching for the best parameters. The idea of brute force search for the optimal parameters or the smallest distance offers a quadratic speedup inherited by Grover's search. However, the cost of those techniques in terms of the number of qubits and gates to be implemented is such that near-term quantum devices will not afford.

The second idea revisits the data superposition discussed in basis encoding. It has been proved that such a case can offer exponential memory and computational speedups in case of models based on associative memories, such as quantum Hopfield Neural Networks. The main advantage of a Quantum Associative memory compared to Hopfield's classical associative memory, is that theoretically $N=2^n$ patterns require $n+1$ qubits to be stored, instead of the classic memory capacity of $N/4ln(N)$. In these terms Ventura and Martinez[90] modified Grover's algorithm to handle data in superposition, where a zero amplitude for datapoints that don't appear in the dataset enables a faster quantum recall process. In the quantum computational sense, recalling a state means to measure the system and cause it to collapse in that basis state that corresponds to the pattern we are searching for.

The basic idea is to use an oracle to invert the phase of the sought out basis state and then invert all the basis states about the average amplitude of all the states, repeating $O(\sqrt{2^n})$ (binary encoding of length *n* for each pattern) times. These repetitions will increase the amplitude of our desired basis state, the one closest to the state associated with the input pattern, to near 1, while decreasing the remaining states' amplitudes accordingly. However, these methods also carry the caveats mentioned in the previous paragraph as well as all the difficulties that a data superposition brings to a model.

Finally, the third strategy, proposed to train quantum perceptron models[91] uses amplitude amplification to search for the best model. The idea is to represent the decision boundaries of a perceptron, that is the hyperplanes used to separate the data points, as points in a hypersphere. Training data are respectively represented planes that define the good and bad subspaces in the hypersphere of possible solutions. If K decision boundaries represented by their weight vector are randomly selected and encoded in basis states, these weight states can be prepared in a superposition and given an oracle *O* that marks the states corresponding to decision boundaries that allow the best separability of the data points, Grover's algorithm is used to obtain the best perceptron model. This method of training a perceptron offers a quadratic speedup but it is also followed by all the complexity reasons mentioned above, so the implementation of such an innovative method in a quantum computer will probably require a long wait.

2.3.3 Hybrid Methods for training

Hybrid training methods concern a combination of classical and quantum processes to train a QML model and are inspired mainly from the need for near-term applications. To do so, as will be explained in Section 4, they must constrain themselves to the limited resources offered by the first available quantum hardware as well as demonstrate a certain robustness to noise in order to efficiently work without the requirement of a full error-correction mechanism. Under this spectrum, outsourcing parts of the training process to a classical processor can lead to some optimistic and realistic results. Thus, this training scheme meets an increased interest in the QML literature of the last couple of years and it is used for the training of all QNN models presented in Section 3, where this training scheme will be discussed in greater detail.

The main idea of hybrid training is to use a quantum device to compute the value of an objective function $C(\theta)$ for a given set of classical parameters θ and then a classical algorithm will be combined for the optimisation of these parameters by making queries to the quantum device. This hybrid scheme is beautifully described in Figure 2. The procedure is based on a parametrised circuit $U(\theta)$ implemented in the quantum device to prepare and output state $|\psi(\theta)\rangle$, which also depends in the set of classical parameters θ. In a loose sense, a parametrised circuit forms a family of circuits and its parameter setting defines a circuit in this family. By measuring the output state, the expectation values of a pre-decided objective are obtained, for example the state of a specific qubit or the energy of the quantum system. These expectation values represent the output of the algorithm implemented by the quantum circuit and, via a cost function $C(\theta)$, are deployed to evaluate θ for the given task.

The goal of a variational algorithm is, therefore, to optimize the parameters so that the objective function reaches its minimum. The innovation in this scheme comes from the fact that the parameters are classical and therefore can be saved and processed in a classical device, using classical numerical optimization methods such as gradient descent or simplex. To achieve that the classical device iteratively queries the quantum device to obtain the expectation values that define the values of the cost function, the values of the cost function itself and in some special cases the derivatives of the circuit in order to compute the gradient.

Examples of quantum variational algorithms that can be trained in this scheme include variational eigensolvers[92], a quantum algorithm where the cost function is defined to be the energy function of the quantum system and the optimization is achieved by finding the ground states, thus minimize the value of the cost function. Quantum classifiers based on variational eigensolvers were developed and further details can be found in Appendix C. Another example is the Quantum Approximate Optimization Algorithm[93] which is used to prepare approximate qsamples of Boltzmann distributions and is incorporated in a technique called *quantum approximate thermalisation* which uses the samples from Gibbs state prepared by the quantum algorithm to compute the weight update in a gradient descent training of quantum Boltzmann machines.

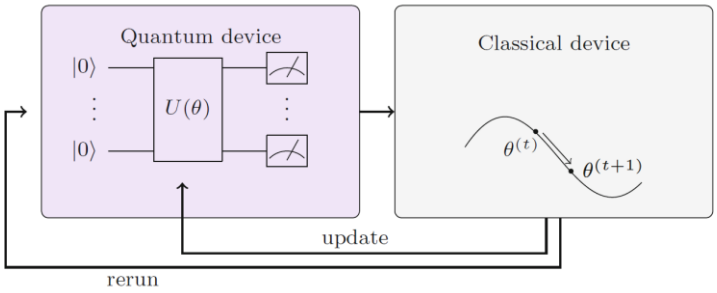


Figure 2 An illustration of the hybrid training scheme

2.3.4 Quantum Adiabatic Methods

Quantum adiabatic methods for training are mainly proposed for training quantum models inspired by adiabatic quantum computing and annealing, therefore for models that fall in QQ type of QML. Adiabatic quantum computing is an alternative to the standard gate model of quantum computing, based on the adiabatic theorem. Several physical implementations have already been demonstrated in quantum annealing devices such as D-Wave with over 100 qubits, although the results are disputed. An advantage of adiabatic computing is that it is expected to be more robust against environmental noise and decoherence than other models of quantum computing[27].

As this thesis is focused entirely in CQ type models and for several reasons refrains from a deep exploration of adiabatic procedures, mainly due to the fact that these methods follow an entirely different interpretation of the quantum mechanics vastly related to the quantum physic, these training scheme will only be superficially mentioned.

Quantum annealers were the first quantum hardware available, thus a big part of QML literature developed adiabatic models for inference and learning which could be experimentally tested in a D-Wave device. However, the experiments so far failed to offer an unambiguous answer to whether the expected speedups are possibly achieved through this framework or not. The initial idea was to encode an optimization problem to the state of a quantum system and use annealing devices in order to prepare the ground state of this system, finding the optimal solution. This routine is called *Quadratic Unconstrained Optimisation* and can be used in the context of memory recall of a quantum Hopfield Neural Network[94]. However, noise and connectivity issues arise in this approach and the recent studies are focused on exploiting a different, proven, advantage of quantum annealers; their ability to generate natural distributions.

The main goal is the preparation of the ground states in a quantum system where the ground state typically represents a qsample for sampling. The input data are encoded in the Hamiltonian of the system and an annealing procedure prepares qsamples sufficiently close to a Gibbs distribution, which can be used to train a Boltzmann machine[95]. Experiments in a D-Wave machine indicate that the "quantum" distributions obtained work well for training and in fact samples from the device have been successfully used to reconstruct and generate handwritten digits[96].

2.4 Learnable Quantum Models

In supervised learning to train a model for inference is to find the optimal set of parameters so given a set of training inputs the model will approximate the function *f* that correctly maps $f(x_i)$ to their labels y_i or approximate the probability distribution to evaluate $p(y|x)$. However, that approximation does not guaranty that the model will be able to correctly generalize in order to provide accurate results for data of the same form which don't belong to the train set. This ability to generalize is called learning an in this subsection we will focus on approaches to build learnable QML algorithms. To achieve so, instead of mimicking classical algorithms and focus on speedups, the researchers followed the exploratory approach in the sense that the algorithms for quantum learning discussed here don't focus on following the design of classical ones but contemplate the quantum mechanic's potentials towards machine learning problems in order to build a new branch of algorithms either without a classical equivalent or as quantum extensions of the existent ones in ML. This approach demonstrates some fruitful results in the terms of near-term applicable QML algorithms mainly due to the fact that it is fixated towards the capabilities of a quantum device to build a quantum algorithm with the minimum required resources instead of trying to achieve the maximum possible speedup.

2.4.1 Quantum Probabilistic Ising-Type models

Ising-type models are not new in the ML, in fact they go way back to some special cases of ANN called Recurrent Neural Networks, like Hopfield Networks[97] for Hebbian learning as well as Boltzmann machines[98,99], which can be seen as the stochastic counterpart of the former. An ising-type model is a physical model of interacting particles of any connectivity and weight, characterized by its energy function:

$$E(s) = -\sum_{i,j} w_{ij} s_i s_j - \sum_i b_i s_i, (3.19)$$

where s_i are the binary-valued units of the Ising-type model and w_{ij},b_i a set of parameters.

In case $w_{ij}=w_{ji}$ and $w_{ii}=0$ as well as $b_i=0$, then (3.19) describes a Hopfield NN which updates its set of parameters with the perceptron rule, aiming to maintain or lower, if possible, the energy of the system. That is:

$$s_i^{t+1} = \text{sgn}(\sum_j w_{ij} s_j^t), (3.20)$$

Boltzmann machines on the other hand are also based on (3.19) by interpreting s_i as binary random variables and keeping the temperature parameter fixed to 1. In such case (3.19) defines the probability distribution over the possible state vector $s=(s_1, s_2,\ldots,s_n)$ via

$$p(s) = \frac{e^{-E(s)}}{\sum_s e^{-E(s)}}, (3.21)$$

.

If constrain $w_{ij}=0$ is added to (3.19), we have the special case of restricted Boltzmann machines.

Starting from (3.19), a quantum version can be obtained if s_i are replaced by qubits and in the special case they are replaced by Pauli single-qubit operators, then an operator which represents the Hamiltonian Energy Function can be defined:

$$H = -\frac{1}{2}\sum_{ij} w_{ij} Z_i Z_j - \sum_i b_i Z_i, (3.21)$$

Consequently, the density matrix of the quantum system's state that can be the quantum analogue of (3.21) is:

$$\rho = \frac{e^{-H}}{tr\{e^{-H}\}}, (3.22)$$

The problem with the above representation is that the diagonal design of H ensures that the density matrix is also diagonal to the computational basis thus no superposition can be achieved and the system is still a classical system translated into quantum formalism. There are two approaches to tackle this issue and add quantum properties to create a quantum system based on (3.21).

The first one involves adding quantum terms to the existing system, like an x-field, to omit the diagonal nature of *H* and enrich it with eigenvalues that correspond to superpositions of states. This technique is used to build and train quantum Boltzmann machines[100] as well as quantum Hopfield NN[101]. A second idea is to start from different equations of the Ising-model to create a quantum system. For example, the replacement of the of Schrodinger's equation with a *Quantum Master Equation*[102] in an open quantum system[103] has been proposed to create quantum Hopfield Neural Networks, or quantum Boltzmann machines based on fermionic instead of stochastic Hamiltonian.

A general framework followed by the researchers exploring the development of a learnable quantum Ising-Type probabilistic model can therefore be outlined in 3 steps; 1) Translate a classical model into quantum language, 2) Add quantum effects to the resulting translation and 3) Analyse the learnability and other potentials for ML in the resulting quantum model. Although the first two steps meet a variety of proposals in the literature, the learnability of the proposed models remains mainly an open question. The probabilistic quantum models of this type are trained with adiabatic methods which naturally makes them ideal applicants for quantum annealing devices, although there exist other proposals for Hopfield QNN based on gate quantum computing[90,104]

2.4.2 Variational Classifiers and Quantum Neural Networks

A very promising approach of QML models, developed under the spectrum of near-term applications is based on Hybrid training. In hybrid training the idea of a variational quantum circuit was introduced as a quantum subroutine of an ML algorithm to achieve inference by computing desired values of the algorithm. As a reminder a variational circuit $U(\theta)$ is nothing more than a quantum circuit with classical parameters inserted in key positions of the circuit. The optimization of these parameters with respect to an objective function results to training and possibly learning. The idea that arises from the specific scheme is that if some circuit parameters are used to insert the input data to the circuit and the output of the circuit is interpreted as a result, variational quantum circuits can form the mathematical framework for supervised learning models. As it will be illustrated in the process, this construction fits elegantly in the QNN context and in Section 3 several QNN proposed based on Variational Quantum Circuits will be analyzed.

A nice property carried from Variational Quantum Circuits is that the single-qubit gates of the circuit can be interpreted as layers of a Neural Network. If the decomposition of $U(\theta)$ into parametrised single-qubit or control single-qubit gates is considered, then the variational circuit is a product of elementary unitary blocks of the form

$$U_\theta = U_L....U_l....U_1 \text{ , where}$$

$$U_k = I_o \otimes I_1 \otimes .... \otimes G_k \otimes .... \otimes I_{n-1} \quad (3.23)$$

and G_k representing a gate applied to the *k-th* qubit of an *n*-qubit quantum system is a parametrised single qubit gate of the form

$$G_k = e^{i\varphi}\begin{pmatrix} e^{i\beta}\cos a & e^{i\gamma}\sin a \\ -e^{-i\gamma}\sin a & e^{i\beta}\cos a \end{pmatrix} = \begin{bmatrix} z & u \\ -u^* & z^* \end{bmatrix} \quad (3.24)$$

The phase parameter can be omitted assuming that $|u|^2+|z|^2=1$, so each single-qubit gate is parametrized by 3 learnable parameters $\theta_i=\{\alpha, \beta, \gamma\}$. U_k therefore is a sparse square matrix of 2^n dimensions. Besides single-qubit gates, control qubit gates are allowed to the circuit, denoted by c_jG_k (G_k is applied to the *k-th* qubit conditioned to the state of *j-th* qubit) to offer universality. Interestingly enough, the elementary building block U_i can be interpreted as linear layer in a neural network, where the architecture is based on the position of qubits to apply the gate on and the occasional control qubits. In an *n*-qubit circuit with information encoded in the 2^n elements of the amplitude vector, a single-qubit gate connects 2^{n-1} sets of variables with the same weights, tying the parameter while the control gate is used to break the symmetry by removing half of these ties and replace them with identities which carry the values of the previous layer.

An important thing to consider in variational circuit models is how the number of parameters is affected by the number of inputs, or the size of the input data. As mentioned before, a QML model designed in the hope of near-term application has to ensure the lowest possible resources. The existent literature presents examples of QNN models designed for classification tasks based on variational circuits with poly-logarithmic[43] as well as linear number of parameters with respect to the number of training inputs and their dimension. Several examples of QNN models of this form will be presented in Section 3, where this approach will be extensively explored.

2.4.3 Other approaches

Quantum Ising-type models and Variational Quantum Circuits represent the two most famous approaches in recent literature and undoubtfully demonstrate a dynamic in terms of the number of proposals and research interest. However, the exploratory approach gained growth in the last half of this decade and demonstrates an enthusiastic variety of ideas and suggestions on other aspects of quantum mechanics that can be used to build a QML model. Although these approaches may have neither reach the mature state nor the research depth of the previous ones, the author believes that some of them worth to be mentioned as indications of the expanded horizons QML acquires under the spectrum of the exploratory approach.

Probabilistic quantum models beyond Ising-type models have been proposed such as quantum random walks[105] or quantum Markov chains[106] based on quantum graphs and projective simulation models. The main idea of a quantum walk is that given a quantum state $|i\rangle|j\rangle$ which represents the edge between *i-th* and *j-th* node in a quantum graph of *N* nodes, an operator

$$\Pi = \sum_{j=1}^{N} |\psi_j\rangle\langle\psi_j|, (3.25)$$

, where,

$$|\psi_j\rangle = \sum_{i=1}^{N} \sqrt{M_{i.j}}\,|\,j\rangle\,|\,i\rangle, (3.26)$$

along with the SWAP operator

$$S = \sum_{i,j=1}^{N} |\,j,i\rangle\langle i,j\,|, (3.27)$$

can be used to create an operator $U=S(2\Pi-I)$ whose application to the quantum system corresponds to one step of quantum random walk. This idea has been applied to create a quantum equivalent of PageRank algorithm and it was proved via simulations to outperform the classical PageRank in terms of sensitivity as well as to "resolve ranking degeneracy" when it comes to similarly weighted pages[107,108].

Another approach contemplates the creation of quantum "committees" inspired by classical ensemble methods where a classifier is constructed by the combination of several models, forming "committees" and the decision is taken based on the majority decision of the committees. The quantum version of this idea[109] is to bring those ensembles in superposition so that parallel implementation will naturally occur. Briefly, given a quantum classifier $A(\theta)$

with input encoded in any way and the output is basis encoded, a model register is used to prepare $A(\theta)$ as a superposition of the possible states, that is all possible parametrizations of this classifier resulting to several classifiers of the same architecture. The state preparation routine of this superposition defines a weighting distribution W_A forming a weighted superposition of all possible predictions and the prediction with the highest probability, that is the one obtained by the majority of quantum classifiers in the committee, is retrieved via measurement.

3 Variational Quantum Circuits Hybrid Learning as Quantum Neural Networks

3.1 Quantum Circuits as Quantum Neurons for Inference

As mentioned before, this section will focus on an effort to translate a classical ANN into the quantum computing language, aiming to the creation of a model that combines quantum routines in such a way that will reproduce the results of it in an efficient way. A model for inference aims to implement the well-known input-to-output map $y=f(x)$ so that it can form a "building-block" for a more complex QML algorithm, in our case a QNN. Under such a spectrum, we may consider this building-block as a quantum neuron analogous to the classical neuron of an ANN.

In section 2 it was presented how the general quantum evolution of a quantum circuit can be interpreted as a certain type of the general linear model. This idea was also expanded to the representation of linear layers of a neural network from such quantum circuits. However, when it comes to Neural Networks, the main problem to be overcome is the problem of non-linearity. The need for linearity in the quantum world comes as an opposition to the non-linear activation function that acts in the weighted input sum of a classical neuron and empowers the model with the ability to deal with non- linearly separable data by capturing non-trivial patterns[110]. This underlines the need to add non-linearities in a quantum model, while respecting the basic axioms of quantum mechanics.

The most intuitive idea to overcome such a problem and thus the oldest one is to exploit the non-linear nature of measuring a quantum state as a form of threshold decision function for

a quantum neuron. The interpretation of the activation function as a measurement is the key idea of several proposed models[9,14], from Kak's very first vision of a QNN[10] till some of the most recent ones proposed. Under this spectrum a quantum circuit that acts as a Quantum Perceptron based on measurement as an activation function will be presented. However, measurement restricts the quantum model to a very specific type of step-function-like decision functions. In the effort to build a quantum neuron that mimics the inference abilities of a classical one, more complex activation functions such as sigmoid, or Relu should be implemented. And in order to do so, angle encoding and Repeat-Until-Success (RUS) circuits shall be employed.

3.1.1 A Quantum Circuit as a Classical Perceptron

A McCulloh-Pitts neuron processes binary-valued input and weight vectors and delivers a binary result (Figure 3). This setup can create an easy to understand quantum equivalent in the case of basis encoded inputs in a quantum circuit, whose output state can either be $|0\rangle$ or $|1\rangle$. The construction of a quantum perceptron based on such a quantum circuit is therefore straightforward and, thus, several proposals have been made in this direction[25,28,30,31,111]. However, most of these ideas have been outdated by now, or simply paused due to the complexity and inability to be implemented in a near-term quantum processor. The idea of a quantum perceptron used for inference was revisited nearly in this year, mostly thanks to IBM's Cloud Quantum Experience that enabled researchers to test their ideas in the small scale of 1-5 qubit quantum circuits. This has been the case of Tacchino, Macchiavello et al[50]. who proposed a quantum circuit that acts as a perceptron neuron and was successfully implemented for pattern recognition. Basis encoding of the inputs is utilized and the measurement adds the necessary non-linearity of the threshold function to the quantum perceptron.

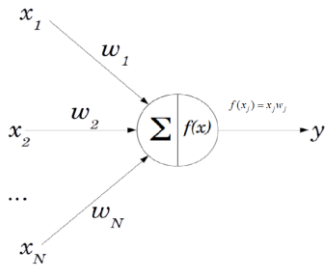


Figure 3 A simple perceptron

Basis Encoding of the Inputs

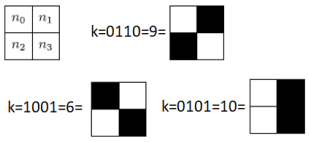
Let $x = \vec{x} = (x_o, x_1, ..., x_{n-1})^T$ be an arbitrary input vector and $w = \vec{w} = (w_o, w_1, ..w_{n-1})^T$ the corresponding weight vector , both of them binary so $x_j, w_j \in \{0,1\}$, define two quantum states, respectively;

$$|\psi_x\rangle = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j \,|\, j\rangle (3.1)$$

$$|\psi_w\rangle = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} w_j \,|\, j\rangle (3.2)$$

The *N*-qubit states $|j\rangle$ from the *N*-dimensional Hilbert space correspond to all possible states of a single qubit being $|0\rangle$ or $|1\rangle$. Thus, for *N* register qubits, there are $n=2^N$ basis states $|j\rangle$ able to encode the *n*-dimensional classical vectors into a uniform weighted superposition of the full computational basis. Apparently in this encoding scheme $log_2 n$ qubits can store and process these *n*-dimensional vectors, analysing $2^{2^N} = 2^n$ different input patterns and all corresponding weights that could be defined.

An easy visualization of this process can be provided if one considers the binary inputs and weights as black and white patterns. For example, given N=2 qubits, 2^2=4 binary images can be managed resulting to 2^4=16 patterns to be analysed. This is illustrated in the Figure 3:



$$j_i = (-1)^{n_i}, n_i \in \{0,1\}, j \in \{x, w\}$$
$$j_i = n_0 n_1 n_2 n_3$$

Figure 4 Figure 3 The scheme used to label 2X2 patterns and a few examples of these patterns.

Prepare the states and compute the final one

In order to prepare the state (3.1) one can start from an initial ground state $|0\rangle^{\otimes n}$ and use and *nXn* matrix U_x , having *x* in its first column, to act on this state, so that $U_x |0\rangle^{\otimes n} = |\psi_x\rangle$. The next step is to compute the inner product of the initial classical vectors *wx*. To do that, a *nXn* matrix U_w, having w^T in its last row is employed, so that the weighted quantum state is

rotated as $U_w|\psi_w\rangle = |1\rangle^{\otimes n} = |n-1\rangle$. Based on that, the scalar product of the states (3.1), (3.2) can be computed by

$$\langle\psi_w|\psi_x\rangle = \langle\psi_w|U_w^{\dagger}U_w|\psi_x\rangle = \langle n-1|\psi_{x,w}\rangle = c_{n-1} \overset{(3.1),\,(3.2)}{=} \frac{wx}{n}, (3.3)$$

(3.3) makes the role of U_w in this scheme quite eminent; it cancels out some of the transformations U_x applies on the initial state, or even all of them when $w=x$. Therefore, if U_w is applied after U_x in the initial ground state, the required finale state is obtained:

$$U_w U_x |0\rangle^{\otimes N} = U_w|\psi_x\rangle = \sum_{j=0}^{n-1} c_j |j\rangle \equiv |\psi_{x,w}\rangle, (3.4)$$

which contains the desired result in its coefficient c_{n-1}.

Extract and explain the results

The expected output can be obtained using a measurement on an ancilla qubit *(z)*, initially prepared on state $|0\rangle$. This measurement will add the non-linearity required by the threshold decision function of the quantum perceptron. This can be done by applying a multi-controlled NOT gate between the *N* encoding qubits and *(z)*, leading to:

$$|\psi_{x,w}\rangle|0\rangle_z \overset{C-NOT}{\rightarrow} \sum_{j=0}^{n-2} c_j |j\rangle|0\rangle_z + c_{n-1}|n-1\rangle|1\rangle_z, (3.5).$$

This results to measuring *(z)* in the state $|1\rangle_z$ and corresponds to an active neuron with probability $|c_{n-1}|^2$.

Models Complexity, Limitations and expected Quantum Advantages

The quadratic character of the threshold function provides a quantum advantage to the quantum perceptron over the classical one, as both parallel and anti-parallel input and weight vectors result to an activation of the perceptron, while orthogonal vectors always result in the ancilla being measured in the state $|0\rangle_z$, therefore the quantum perceptron overpass the limited linear abilities of classical perceptron. Moreover, this algorithm presents an exponential quantum advantage due to the storage capabilities of representing 2^N bits of information using only N qubits.

Finally, one should consider the strategies for an efficient realization of the unitary transformations that should be followed to implement the preparation of the input state $|\psi_x\rangle$ and U_w in a quantum hardware. Strategies such as the applications of successive flip-flop blocks

have an exponential expense in the number of qubits and gates required to the quantum circuit ($O(2^N)$ gates for N qubits), forcing the quantum perceptron to draw away from possible near-term applications. A more efficient approach suggests an *"hypergraph states generation routine"* that maps the weight and input vector to vertices and hyperedges of generalized graphs that can be easily prepared using multi-controlled Z gates in parallel.

However, not all problems are solved. The depth of such a circuit and the number of unitary gates to be implemented increases exponential with the number of qubits. Furthermore, issues arise concerning the decomposition of several controlled operations to single and two-qubit gates only. The severity of the later issue varies from one quantum processor to the other, and also depends to the desired accuracy, making a universal solution an extremely difficult task.

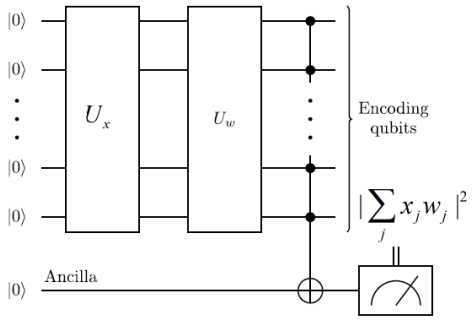


Figure 5 Scheme of the quantum circuit that acts as a quantum perceptron

3.1.2 Non- Linearities Beyond Measurement

RUS are non-deterministic circuits, first introduced by Paetznick and Svore in 2014[112], that form a technique for quantum gate synthesis. The main idea of such circuits is to apply a unitary operation to a quantum state only when a certain expected measurement outcome is observed. This process is repeated until the desired unitary operation is indeed performed, that is the circuit is measured and re-prepared until the desired measurement is observed. RUS circuits are designed so that they have an extremely low resource cost. A runtime analysis performed for a RUS utilized to add non-linearities to the quantum neuron, discussed below, has computed the average depth of such circuit in $O(14^k)$ for k iterations.[38] In the following idea, developed by Cao, Guerreschi and Guzik[38] less than two years ago, a RUS circuit can be employed as the threshold function in a quantum neuron.

The Quantum Neuron

Let's first consider the classical neuron as a non-linear function φ of n variables -the linear combination of n weighted outputs x_i of previous layer's neurons- and maps them to the output value, called the state of the neuron

$$\varphi(\sum_{i=1}^{n} w_i x_i + b) = a \in [-1,1], (3.6)$$

where -1 stands for an inactive neuron and 1 for an active one. For simplicity we will call $\sum_{i=1}^{n} w_i x_i + b = \theta$ the weighted input of the neuron and φ the activation function. Consider also a qubit with the state

$$R_y(a\frac{\pi}{2}+\frac{\pi}{2})|0\rangle = \cos(a\frac{\pi}{2}+\frac{\pi}{2})|0\rangle + \sin(a\frac{\pi}{2}+\frac{\pi}{2})|1\rangle, (3.7)$$

and $R_y(t) = e^{-\frac{itY}{2}}$ the quantum operation of Y Pauli matrix acted on it. As a reminder, the rotation around the y-axis is defined as

$$R_Y(2u) = \begin{bmatrix} \cos u & -\sin u \\ \sin u & \cos u \end{bmatrix}, (3.8)$$

.

The qubit is hence rotated around the y- axis by an angle $a\frac{\pi}{2}+\frac{\pi}{2}$.

Thus, one can easily sense that in case $a=-1$ we have the quantum analogous of an inactive classical neuron and our qubit in state $|0\rangle$, while for $a=1$ the qubit state of $|1\rangle$ can be interpreted as an active quantum neuron. We shall now call this qubit a "quron". However, our quron's capabilities immediately surpass those of a classical neuron when superposition is introduced and that can simply be done for $\alpha \in [-1,1]$.

Under this spectrum, the idea is the following; given a control state $|x\rangle=|x_1x_2\ldots x_n\rangle$ as the representation of the neuron's input or possibly the training input, create a quantum circuit that can act as a quantum neuron of n qubits which returns a state corresponding to the output of the classical neuron, or the output of the considered quron. The goal is to rotate the output qubit by a nonlinear activation ϕ which depends on an angle θ, in other words, we want to prepare the output qubit in state $R_y(2\varphi(\theta))|x\rangle$. By now one can see the importance of angle encoding in this construction. Angle encoding is a special type of information encoding,

developed for special quantum routines depended on RUS. While basis encoding is used for the inputs of the quron, the weighted sum of them is encoded in the angle θ of the rotation.

In order to construct such a quron of *n* qubits, the following quantum operations shall be deployed:

1) $R_y(2w_i)$ is applied to an ancilla qubit conditioned on the *i-th* qubit of $|x\rangle$, followed by $R_y(2b)$ and that is repeated for *n* ancilla qubits. That is analogous to applying

$$R_y(2(\sum_{i=1}^{n} w_i x_i + b)) = R_y(2\theta)$$

to the ancilla qubit conditioned to the state $|x\rangle$.

2) $R_y(2\varphi(\theta))$ is a rotation performed to state $|x\rangle$. Since φ is a non-linear activation function, the only tricky part is to find out how to approximate such rotation.

3)RUS circuits used to approximate $R_y(2\varphi(\theta))$.

Repeat-Until-Success Circuit as an approximator for φ

We will focus, for simplicity, on the RUS circuit developed to approximate a sigmoid like activation function $q(\theta) = \arctan(tan^2\theta)$, thus implement $R_y(2q(\theta))$ to the control state of inputs. RUS needs an ancilla qubit to be measured and upon this measurement's output the circuit's output depends. The idea is really simple: if the ancilla qubit is measured in $|0\rangle$ state, this indicates that $R_y(2q(\theta))$ was successfully applied to the output qubit. If ancilla measures $|1\rangle$, the circuit has implemented a rotation by $R_y(\frac{\pi}{2})$ and in order to correct it $R_y(-\frac{\pi}{2})$ should be applied to the output and the circuit will be repeated. See Figure 1-3 for the realization of the RUS circuit in one or several iterations.

As mentioned before, RUS will act as a threshold function in θ. The circuit can be repeated, say *k* times, to move any point $\theta \in [-1,1]$ closer to one of the ends of the interval [0,π/2]. Therefore, points 0 and 2π have the role of attractors. In the case of $q(\theta)$ this threshold function can be described as;

$$S(\theta)=\begin{cases}\theta>\dfrac{\pi}{4}\Rightarrow R_y(2q(\theta))\rightarrow R_y(\pi)\,|0\rangle=|1\rangle\\ \theta<\dfrac{\pi}{4}\Rightarrow R_y(2q(\theta))\rightarrow R_y(0)\,|0\rangle=|0\rangle\end{cases},(3.9)$$

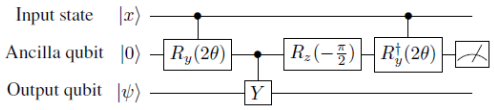


Figure 6 A simple RUS circuit

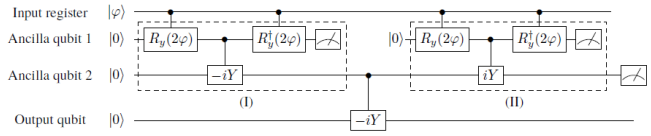


Figure 7 Two iterations of RUS circuit

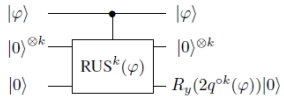


Figure 8 General case of k iterations of RUS circuit

A very interesting property of a RUS circuit, which enriches the constructed quron with an immediate quantum advantage, is that RUS can also be applied in a quantum superposition. Consider a circuit where the input is initialized to a uniform superposition $\frac{1}{\sqrt{n}}\sum_{i=1}^{n}|i\rangle$, controlled by a *n*-dimensional register. Thereafter, the controlled rotation that imitates the activation function can be

$$\frac{1}{\sqrt{n}}\sum_{i=1}^{n}|i\rangle\langle i|\otimes R_y(2\varphi_i),(3.10)$$

and conditioned on the output measurement being $|0\rangle$ the desired final state is a superposition of rotated output states

$$\frac{1}{\sqrt{n}}\sum_{i=1}^{n}F_i\,|i\rangle\otimes|0\rangle\otimes R_y(2q(\varphi_i))\,|\psi\rangle,(3.11)$$

.

Note that F_i is a factor standing for the amplitudes of the superposition, depending on both the angles and the number of successful and failed iterations of RUS, thus F_i is a random variable. A more detailed analysis on such case can be found in Appendix B on the original paper by Cao et al.[38]

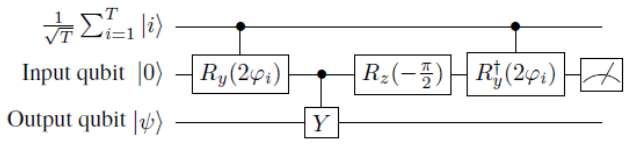


Figure 9 RUS applied in superposition of input rotations

The idea of a Feedforward Neural Network

The immediate idea after constructing such a quron is to use it as a building-block for a QNN to reproduce the behaviour of a classical ANN. Furthermore, due to the ability to handle inputs in superposition, such a quron could handle larger training sets than its classical competitor. Let's try to build a Quantum analogous of a three-layer MLP (Input-Hidden-Output Layer) used as a classifier for binary encoded input data. In the general case

$$\theta_i = \sum_j w_{ij} x_j + b_i, (3.12)$$

represents the input in the hidden layer's *i*-th neuron and

$$\sigma(\theta_i) = \begin{cases} 1, \theta_i > 0 \\ -1, \theta_i < 0 \end{cases}, (3.13)$$

the respective activation function. The *m* neurons in the output layer represent the *m* distinct categories (classes) with neuron's state being 1 representing "belong" and -1 "not belong" to the class. If $z^{(l)}$ is a vector representing the state of *l*-th layer in the MLP, then information's propagation in the network is described by

$$z^{(i)} = \sigma(W^{(i)} z^{(i-1)} + b^i), i = 1, 2, 3, (3.14)$$

and the cost function to be minimized could be the simple Mean Square Error.

In order to transfer such a model in a quantum setting, consider one qubit s^i for each neuron in a layer forming an input state $|s_j^{\,1} s_j^{\,2} \ldots . s_j^{\,n}\rangle$ for each layer. Thus, the state of the qubits corresponding to the previous layer serves as the control register for the next layer. The input state of the input layer could also be a superposition of train data as inputs and the target labels such as

$$\frac{1}{\sqrt{n}}\sum_{j=1}^{n}|x_j\rangle|y_j\rangle, (3.15)$$

The circuit also requires *k* ancilla qubits for the RUS circuits, *k* being the number of iterations of RUS performed to realize the propagation from *i* to *i+1* layer of the QNN -which is nothing but the angle $\varphi_j^{(i+1)}$ that encodes the weighted sum of the inputs (plus the bias) of the *j*-th neuron in *i+1* layer. Note that, the ancilla qubits necessary for the RUS circuit can be reused for all neuron updates, and therefore this construction requires a single qubit for each extra neuron. The general architecture of a circuit that realizes such a QNN can be seen in Figure 5.

However, a QNN cannot bear any true resemblance to a classical NN unless a cost function and training rules for weight updates are set. In a neural network the goal is to minimize the error of the output layer $f(x_j)$ according to the known attributes of the dataset y, so the cost function can be the MSE

$$\min_{W,b}\Sigma_{j=1}^{T}||y_j - f(\varphi_j)||_{2^n}^2, (3.16)$$

A backpropagation with gradient descent is one of the most traditional ways to train an MLP, thus update its weights.

In this QNN, consider that in the first iteration we randomly choose the weights. Using backpropagation in a hybrid training scheme -which is vastly discuss in the following section- the weights (thus the angles) are updated based on the objective function, and at the end of the training the training accuracy can be defined as:

$$\langle\overline{ZZ}\rangle = \frac{1}{m}\sum_{j=1}^{m}\langle z_{a_j}z_{b_j}\rangle \text{ or } \langle\overline{ZZ}\rangle = \prod_{j=1}^{m}\langle z_{a_j}z_{b_j}\rangle$$

where, z_a the states of the qubits in dataset and z_b the states of the output layer, and *m* the length of dataset and output layer. $\langle\overline{ZZ}\rangle = 1$ the best accuracy.

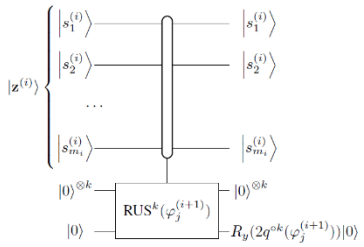


Figure 10 Quantum circuit realization of the quantum neuron propagation

Moving beyond periodic activation functions

The activation function proposed so far is based on the periodic tangent function. The periodicity it introduces creates some serious limitations and problems in the realization of a QNN based on qurons that apply this activation function. First of all, the input θ is restricted to be in the first quadrant, plus neurons' derivatives oscillate quickly making training with gradient descent extremely difficult. Based on the work presented so far, Wei Hu[39] created a non-periodic and non-linear activation function that can be approximated using a RUS and be applied in a quron as described so far.

The proposed activation function is

$$f(\theta) = \arcsin(\sqrt{sigmoid(\theta)}) \text{, where } sigmoid(\theta) = \frac{1}{1+e^{-\theta}}, (3.17)$$

thus, it's derivative $f'(x) = f(x)(1-f(x))$ is easy to compute and can be trained with efficient gradient descent. Moreover, it is not periodic so it can take any real numbers and finally It can generate a ReLU function that looks more like a classical ReLU function, according to simulations of qurons with both *q(θ)* and *f(θ)*, provided by the author. However, that doesn't mean that all our problems are solved. The vanishing gradients problem appears; as max*f'(θ)*=0.25, the backpropagation training squeezes the errors at least by a quarter at each layer and near the two ends of the sigmoid function, its values tend to be flat, implying the gradient is almost zero.

Conclusion

The suggestion of this quantum circuit's architecture utilized to mimic the neuron of a classical NN comes with a performance guarantee[38]; Given an input angle *φ(θ)* encoding the input if *n* qurons, the minimum *k* iterations needed for successfully prepare the desired output state $R_y(2q^{\circ k}(\varphi(\theta))|0\rangle$ for an arbitrary error ε such that

$$|q^{\circ k}(\varphi(\theta)) - g(\varphi(\theta))| \leq \varepsilon, (3.18)$$

$$g(\varphi(\theta)) = \begin{cases} 0, \varphi(\theta) < \frac{\pi}{4} \\ \frac{\pi}{2}, \varphi(\theta) > \frac{\pi}{4} \end{cases}, (3.19)$$

then the expected runtime of the circuit is proved to be $O((\frac{n}{\delta})^{2.075}(\frac{1}{\varepsilon})^{3.15})$, where error ε corresponds to the distance of $g(\varphi(\theta))$ from its attractor 0 or $\pi/2$ and δ its distance from the threshold $\pi/4$.

The dependence on the number of iterations k emphasises the non-deterministic nature of RUS circuits and implies that the runtime is on average slightly longer than the feed-forward pass in a classical neural network. Moreover, on average the circuit depth scales as $O(14^k)$. The authors also provided an estimation for k to be $O(\log(1/\delta\varepsilon))$. The runtime, although polynomial, is still challenging, especially taking into account that state preparation routines and the generation of superposition of states is not taken not account.

The ability to process all input data in superposition is a quantum advantage of such a QNN, that has no analogous to its classical counterpart. However, in such a case in order to simulate an l-layered classical MLP with max layer size d ,an error at max ε and under the constraint that network's weights and bias can be represented in resolution δ ($w=k_w\delta$ and $b=k_b\delta$ for k_i integers), the number of qubits needed is estimated to be $O(nl+\log(n\backslash\delta\varepsilon^{1.52}))$. This yields a linear relationship between the number of qubits and the number of neurons in a classical MLP. Under very certain circumstances, a roughly linear speedup can be achieved in terms of model complexity, but still these computations take into account neither the cost and complexity of state preparation if the inputs to encode them in basis and then prepare them in superposition, nor the complexity added to the model to manipulate and preserve this superposition.

If we consider the main requirements for a QNN proposed in Section 1.3 , the proposed architecture satisfies all three of them; the mapping of its neuron to a qubit ensures the ability to encode any binary N-dimensional input to the initial state, the threshold dynamics represented by RUS circuits correspond to the integrate-and-fire mechanism of the classical neurons and finally quantum effects such as superposition are exploited, providing a quantum advantage while unitary and linearity of the quantum system are preserved, ensuring consistency with quantum theory.

3.2 Hybrid Training

As already mentioned in Section 2, by the term training we mean strategies that aim to the optimization of a quantum model or the utilization of a quantum model to train a classical algorithm with a quantum speedup. The basic 4 approaches are the utilization of linear algebra calculus, quantum search of optimal parameters based on Grover's routine, quantum adiabatic routines designed for entirely quantum models and, what is considered to be the approach closest to a near-term application, hybrid training routines of variational algorithms.

This subsection will focus on the latest case in which inference is made in a quantum device using methods as the ones presented in subsection 3.1 as well as more elaborate ones that will be presented in extend in the next subsection, and gradient descent backpropagating methods -which have no quantum equivalent- are performed in a classical device. As in this whole section, we will focus on models and ideas that can be realized by quantum circuits, in the hybrid training scheme called variational circuits. Variational circuits are parametrized quantum circuits used for inference, in the sense of computing terms of an objective function, or its derivatives. The objective function is used for training, aiming to minimize the inference error of the circuit or to maximize its accuracy. The hybrid nature of the training comes to the fact that classical processing is used to compute better parameters for the circuit, in an iterative process until a threshold value for the objective function is reached. Moreover, classical methods can be utilised to find the optimal initial parametrization of the variational circuit[51].

The hybrid training methods lay their foundations not that long ago, as the theory of variational hybrid training was developed in 2015[40]. These methods, especially the ones concerning quantum circuits, are a part of the exploratory approach, mentioned in Section 1, because the aim of such algorithms is not to mimic a classical ML algorithm, or an ANN in our case, but to accomplish the same task and reproduce the same results, offering runtime speedup and resolve storage issues. In contrast to the "building-block" qurons proposed for Inference, whose resemblance to a classical neuron was clear, the quantum models proposed to form a QNN from now on will focus on reproducing an ANN's results rather than be straightforwardly inspired by the classical model.

The main reason this thesis focusses on the variational scheme is the belief of the author, supported by the recent progress in literature and research, that this scheme is the one most

suitable for near term quantum applications. This belief is strongly based on the acknowledgment that only a part of such an algorithm has to run coherently, allowing much smaller circuits, and such circuits have a universal design that can be easily customized based on the quantum device to be implemented. Finally, an iterative optimization process, which is fairly simple and easy to understand and surprisingly easy to compute in quantum case, comes with robustness against noise in the estimations, and since error correction has a very long way to go in quantum computing technologies so far (more on this on the Section 4), this is a long-awaited feature.

The theory of variational algorithms

Two of the very first variational algorithms that can be implemented for QML, appeared not that long ago, in 2014 and are; Quantum Variational Eigensolvers[92] (VQE), inspired from the physical problem of finding the minimum energy eigenstates- the ground states of a quantum system- and Quantum Approximate Optimization Algorithm[93] (QAOA) designed to deal with combinatorial optimization problems. These are the first two variational algorithms that found application in QML with the VQE altered to build quantum classifiers and QAOA utilized as a preparation routine for Qsamples in Boltzmann machines. Both of these can be realized in a quantum circuit, using Hamiltonian encoding, thus the problem is encoded in a Hermitian matrix A, whose expectation value $\langle A \rangle$ is iteratively optimised by tuning parameters θ of an ansatz state $|\psi(\theta)\rangle$.

As this thesis focuses on Quantum Circuit models used as QNNs, there will be no further discussion concerning QAOA or VQE. In terms of its potential use as a tool in a QNN, detailed information about how a VQE can be used as a classifier is provided in Appendix C. However, our main concern will focus on a third algorithm, Quantum Circuit Learning[42] (QCL), published just earlier this year. This recent job not only proposes a complete scheme for training Quantum Circuits to be used for learning, but it also proves the simplicity of gradient-based systematic optimization in certain scenarios.

But before dive into the construction and of a QNN based on a variational quantum circuit, one should address the ways to optimise circuit's parameters. At this point, the reader should consider that such quantum algorithms return an estimation of the output in the form of

measurements for the expectation of an operator, thus precision can be increased by repeating the algorithm and measurement, concurrently increasing the complexity.

3.2.1 Derivative free Optimization

The simplest way of training a quantum model does not require the computation of gradients. The idea is an iterative algorithm that successively evaluates the objective function. Such an example is the Nelder-Mead method[113], one of the most famous methods for gradient-free optimization of multivariable functions in ML. The method is based on a simplex is a special polytope of $n + 1$ vertices in n dimensions -a line segment on a line, a triangle on a plane.

The main idea is to start with a randomly generated initial simplex and iteratively reshape it, one vertex at a time, towards an optimal region. Every iteration of the algorithm tries a few modifications of the current simplex and ends up with the one shifting the simplex towards the optimal region. In the final iterations of the algorithm the simplex should ideally start to shrink inwards to the best point inside it, resulting to the most optimal objective value.

Consider an n-dimensional space, for an n-variable function's $f(x)$ optimization or an n-parameters variational circuit and a randomly chosen initial simplex that consists of $x_1,x_2,...,x_n$ points. Then each iteration of the algorithm performs the following steps:

Step 1: Order all $x_1,x_2,....,x_n$ points such that $f(x_a)<f(x_b)<...<f(x_z)$

Step 2: Consider all points except the worst x_a and compute their centroid: $C=\frac{1}{n}\sum_{i\neq a} x_i$

Step 3: Transform the simplex. There are several ways to perform such a transformation. The transformations tried in the order of presentation:

a) Reflection: The reflected point $x_r=C+d(C-x_a)$ is computed, for a reflection parameter d. If $f(x_b)<f(x_r)<f(x_z)$, which is translated to x_r being better that x_b, x_a is replaced with x_r and the first iteration is finished. Such a transformation attempts to move the simplex in a direction away from the sub-optimal region around x_a.

b) Expansion: If the reflected point is found to be better that the current best, $f(x_r)>f(x_z)$, the goal is to move from C in the direction of x_r, trying to find a better solution. This can be

done by defining an expanded point $x_e=C+m(x_r-C)$, for an expansion parameter m, usually set to 2. Subsequently x_a is replaced by the better of two points, x_r or x_e.

c)Contraction: In case x_r is worst that x_b, so transformations (a) and (b) have no value, this indicates that the direction defined by x_r may not be optimal. If that is the case the step to take is to contract the simplex. Define a contraction point $x_c=C+p(x_a-C)$, p a parameter close to 0.5. If $f(x_c)>f(x_a)$, we replace x_a with x_c in the simplex.

d)Shrink contraction: In case all the above transformations fail, the entire simplex needs to be redefined. That is, keeping x_z and define all other points with respect to the old ones so that the j-th point will be $x_j=x_z+\gamma(x_j-x_z)$, γ again being a parameter close to 0.5

Step 4: Terminate the algorithm. Termination criteria may vary according to algorithmic needs for high accuracy or lower complexity. Some of the most famous ones are a limit in the number of iterations, or the simplex minimum size.

3.2.2 Numerical Gradient-Based Optimization

This optimization method, called the *finite-differences method*[52], is employed when a model has a black-box access to the cost function. The idea is rather simple; Given a cost function $C(\theta)$ to be optimized, its gradient can be numerically computed by:

$$\frac{C(\theta_1,...,\theta_\mu,....\theta_M)-C(\theta_1,....\theta_\mu+\Delta\theta_\mu,...\theta_M)}{\Delta\theta_\mu}+O(\Delta\theta^2{}_\mu)+O(\frac{\varepsilon}{\Delta\theta_\mu}),(3.20)$$

,

ε being the error of the estimation of $C(\theta)$. It is proven to be extremely important for the success of finite-difference method to ensure that

$$C(\theta_1,...,\theta_\mu,....\theta_M)-C(\theta_1,....\theta_\mu+\Delta\theta_\mu,...\theta_M)<2\varepsilon,(3.21)$$

so that there is no overlapping in evaluation intervals of the two cost functions. This is equivalent to $2\varepsilon\le\Delta\theta_\mu\frac{\partial C(\theta)}{\partial\theta_\mu}$.

This final rule implies that the smaller the gradient, the highest the estimation's precision needed thus more iterations of the algorithm required. This is particularly inconvenient in case the Variational Circuit creates measurements of high variance or a very close approximation to the minimum of the cost function is desired.

3.2.3 Analytical Gradient Based Optimization

This last type of optimization of the objective function, aka for training a model, is the best suited one in case of Variational Quantum Classifiers, vastly used in recent research in the field and the one chosen to train the QNN models this thesis focuses on in Learning subsection. It is also ideal for near-term applied QNN models since, as will be evident soon, it is proven that when dealing with the Pauli-matrix-generator-form the analytical gradients' computation becomes surprisingly simple[42].

Let's consider the case of the Variational Circuit described in the previous section and dive deeper in this idea. Then we will discuss how gradients can be analytically computed for the cost function this circuit estimates. If we have a set of parameters $\theta=\{\theta_1, \theta_2, \ldots, \theta_n\}$, each one assigned to an elementary unitary block $G(\theta_i)$, we can define our quantum circuit as

$$U(\theta) = G(\theta_n)....G(\theta_{n-l})...G(\theta_1), \quad (3.22)$$

and let $C(U(\theta))$ be the cost function related to this circuit that needs to be minimized. In order to compute the analytical gradient of the cost function it is obvious that one has to compute the partial derivatives of the circuit, that is $\partial_m U(\theta)$, for an arbitrary parameter m in θ. At this point a problem arises; there is no guarantee that the derivative of a circuit is a quantum circuit itself, as unitarity is not necessarily preserved by derivation. Thus, a quantum device cannot be straightforwardly used to estimate $\partial_m U(\theta)$. Luckily, there is a special situation where a trick can be exploited; the *Classical Linear Combination of Unitaries*[43], a technique that, under some assumptions, allows the same quantum hardware used to realize $U(\theta)$ to also be used to compute its gradients. For more details on this procedure, see Appendix D.

Let's for simplicity consider the case when m appears only in θ_z. Then the derivative of our quantum circuit becomes

$$\partial_m U(\theta) = G(\theta_n)....\partial_m G(\theta_z)...G(\theta_1), \quad (3.23)$$

The next observation is crucial; Quantum circuits are linear. This linearity of $U(\theta)$ means that the derivative of the circuit is exactly the same as the circuit itself, except for one block. Consider also the special case when the computation of a unitary block's derivative can be expressed as the weighted sum of other unitary blocks;

$$\partial_m G(\theta_z) = \sum_i a_i G_i(\theta_z), (3.24)$$

If that is the case, circuit's derivative can be expressed as a linear combination of circuits;

$$\partial_m U(\theta) = \sum_i a_i G(\theta_n)....G_i(\theta_z)...G(\theta_1) = \sum_i a_i U_i(\theta), (3.25)$$

This means that the gradient can be estimated using slightly different circuits from the original one. We can distinguish two main cases of this scenario. They will be briefly mentioned here and then applied to QNN models described in the next subsection.

Case A: If the variational quantum circuit (3.22) consists of parametrised general single-qubit gates, each gate $G(\theta_i)$ can be described as[114]:

$$\partial_a G(a,b,c) = G(a+\frac{\pi}{2},b,c), (3.26)$$

$$G(a,b,c) = \begin{bmatrix} e^{ib}\cos a & e^{ic}\sin a \\ -e^{-ic}\sin a & e^{-ib}\cos a \end{bmatrix} \rightarrow \partial_b G(a,b,c) = G(a,b+\frac{\pi}{2},0) + \frac{1}{2}G(a,b+\frac{\pi}{2},\pi), (3.27)$$

$$\partial_c G(a,b,c) = \frac{1}{2}G(a,0,c+\frac{\pi}{2}) + \frac{1}{2}G(a,\pi,c+\frac{\pi}{2}), (3.28)$$

This can be seen as a case where $\partial G_i(\theta_z) = G(r_i(\theta_z))$, where a rotation of π/2 applies to some parameters of the circuit and setting others to constant.

Case B: In this case, the variational quantum circuit is not a decomposition of general single-qubit gates, but it rather is parametrised as Pauli matrices. In such a case, each gate $G(\theta_i)$ can be described as:

$$G(m) = e^{imO} \rightarrow \partial_m G(m) = \partial_m e^{imO} = iOe^{imO}, (3.29),$$

where O a tensor product of single-qubit Pauli matrices and the identity operator. Note that Pauli gates can be applied as unitaries, therefore O can be considered and extra gate to $U(\theta)$. This immediately results to an extreme simplicity in the computation of derivatives, by simply adding O as a further gate to the initial circuit. In the unfortunate case that O has to be a non-unitary Hermitian, an effort to decompose it into a linear combination of unitaries should precede.

3.3 Quantum Learning Models

As it has already been clarified in the introductory section, a model that is capable of learning is a model that, trained with a set of parameters $X=\{x_1,x_2,\ldots,x_n\}$ to accomplish a specific task, can generalize the accomplishment of this same task and return predictions with a desired accuracy for new input data $Z=\{z_i\}$, and by new we mean $Z \supset X$. Such a case is, of course, the case of ANNs and in this section we will discuss quantum models that could be seen as quantum analogues of ANN, in the scheme of the exploratory approach. That means that the QNNs based on variational circuits we will discuss are not meant to mimic ANNs, bear the exact architecture or follow the mathematical formalization of them, but they rather are quantum equivalents of an ANN in the sense that they can, given the same -or even smaller- train set X learn to accomplish the same task as an ANN and return predictions for Z.

Of course one can detect some analogies of such a QNN model and an ANN if consider the interpretation of quantum gates in a quantum circuit as linear layers in an ANN, but instead of focussing to find and point out such analogies, our main goal in this subsection is to consider quantum algorithms and models that can form a QNN in near term applications. Most of the existing literature on QML focused on the approach of translating existing ML algorithms into quantum subroutines. Due to the mathematical complexity of these classical algorithms, enhanced by the different natures of quantum computations, these subroutines are non-trivial and resource-intensive algorithms, unable to be implemented on small-scale devices.

This thesis considers “near-term” as something potentially realizable in the next 5 years, at least in terms of 30-50 qubit applications to test the actual performance and complexity of the model. Therefore, the circuits depth -the number of gates that need to be realized- the number of qubits needed for encoding as well as the ancilla qubits required play a major role in the selection of the following models.

Furthermore, the fact that Variational Quantum Circuits can be trained using hybrid methods as discussed previously, enhances this choice. It cannot skip our attention that the research community of QML has recently made a turn towards this approach in the request for a QNN, providing twice as many model proposals in the last 2 years as all proposals for the rest of approaches discussed in Section 2, combined. Furthermore, in the last months, such models started to make their way towards modest realizations of 1 and 2 qubits on IBM Cloud

Quantum Computers. All these being said, the rest of Section 3 will be dedicated to the analysis of 4 QNN models, all of them circuit-centric, all of them coming with a hybrid training framework, and all of them as fresh as 1-2 years old.

3.3.1 Prove of learnability

Before one starts to discuss QNN learnable models based on variational circuits, a very fair demand would be to ensure that there exists the possibility of learning, in the sense that the chosen quantum structure has the mathematical potentials to be adaptable, approximate functions and, thus, detect patterns. This very same reassurance came to the world of the classical ML with the famous *Universal Approximation Theorem*[115,116], a theorem of existence, in the sense that it provides the mathematical justification for the approximation of an arbitrary continuous function by an ANN. The theorem comes in several variations for different ANN architectures, but the first and general formulation, concerning MLP states:

> "*A single hidden layer is sufficient for a multilayer perceptron to compute a uniform approximation to a given training set represented by the set of inputs* $x_1, \ldots, x_m$ *and a desired (target) output* $f(x_1, \ldots, x_m)$*"*

Thankfully very recently a quantum equivalent was proved for the specific case of binary functions[41] stating that *any two-valued label function can be represented by a quantum circuit*, although the price of circuits depth may be forbidding for near term applications. In order to confirm without any doubt, the quantum advantage of a QNN, the research community of QML has to answer the open question of which functions can be compactly represented on a quantum circuit whereas they cannot be on a classical network.

Poof's Highlight

Given the circuit U of the form (3.22), an input state $|x,x_{n+1}\rangle$ where x is the n-dimensional input, encoded in the computational basis so that $x_i=\{-1,1\}$, plus an ancilla qubit ,initialized in $|1\rangle$, to be measured for the inference and a binary label function $f(x)$, we have 2^n n-bit strings and 2^{2^n} possible functions f. Now, consider the case that $U=U_f$, forms a generalized Pauli matrix. That is

$$U_f = e^{iO}, (3.30)$$

for a tensor product of Pauli matrices O.

For simplicity, focus on the case when U_f acts on the input state, it performs a *x*-axis rotation of $\frac{\pi}{4} * f(x)$ on the ancilla. This can be expressed using Dirac notation as;

$$U_f \mid x, x_{n+1}\rangle = e^{(i\frac{\pi}{4} f(x) X_{n+1})} \mid x, x_{n+1}\rangle, (3.31)$$

Now, let *f(X)* an operator diagonal to the computational basis with respect to *f(x)* so that $f(x_i)$ is the element in the diagonal and the *i*-th row of *f(X),* corresponding to the x_i input. Then

$$U^{\dagger}{}_f Y_{n+1} U_f = \cos(\frac{\pi}{2} f(X)) Y_{n+1} + \sin(\frac{\pi}{2} f(X)) Z_{n+1}, (3.32)$$

Since *f(x)* is binary valued, from (3.31) and (3.32) it follows that

$$f(x) = \langle x,1 | U^{\dagger}{}_f Y_{n+1} U_f \mid x,1\rangle, (3.33)$$

That proves that in an abstract level any binary function *f* can be represented by a quantum circuit. Now, to make this work in the framework of Variational Circuits that are in a suitable form to be trained and used as QNN, it is important that U_f can be decomposed as a product of single or two-qubit unitaries. To do so, we will resort to *Reed- Muller* representation of Boolean functions, thus, it is convenient to express x_i to Boolean variables. That can easily be done by defining $x=(x_{1,\dots,}x_n)$ as $x=(b_{1,\dots,}b_n)$ where $b_i=\frac{1}{2}(1-x_i)$ and $f(x)=1-2b$, $b=\{0,1\}$. By such notation, the Reed-Muller representation of the label function will be

$$f(x) = 1 - 2(a_0 \otimes (\bigotimes_{i=1}^{n} a_i b_i) \otimes (\bigotimes_{i=1,j=2}^{n} a_{ij} b_i b_j) \otimes .... \otimes a_{123\dots} b_1 b_2 \dots b_n, (3.34)$$

where addition represents the *mod2* and the coefficients *a* are all 0 or 1. There are in fact 2^n coefficients *a*, resulting in the representation of 2^{2^n} functions *f*.

If we utilise an operator *B*, diagonal to the computational basis corresponding to *b* in the same way that *f(X)* in (3.32) corresponded to *f(x)*, then the operator U_f can be reformed in the following way

$$U_f = e^{(i\frac{\pi}{4} f(x) X_{n+1})} = e^{(i\frac{\pi}{4} X_{n+1})} e^{(-i\frac{\pi}{4} \mathbf{B} X_{n+1})}, (3.35)$$

Each non-vanishing term in the Reed-Muller formula gives rise in U_f to a controlled bit flip on the output qubit. And, as demonstrated in previous literature[38], any controlled one qubit that acts on qubit *n+1* and the control qubit is one of the *n* previous ones, can be written as a product

of n^2 single or two-qubit unitaries. Thus the proof is complete and we can now focus on how to create Variational Circuit models that can exploit the ability to approximate any Boolean function within a reasonable circuit depth and complexity level, so that those models can be used in near term applications or, at least, be tested in an actual quantum system.

3.3.2 Quantum Circuit Learning

The main idea of the QCL algorithm for a supervised learning task can be summarised as; Given a set of train data $\{x_i\}$ with their target vectorized values $\{f(x_i)\}$, the algorithm's output $y_i=y(x_i, \theta)$ is optimized by tuning θ to minimize the distance between the output and target values. The hybrid nature of the algorithm comes from assigning the calculation of y_i to a quantum device and the update of θ to a classical computer. In this sense, the cost function can be some function like the quadratic cost, that is $C = \sum_i \| f(x_i - y_i) \|^2$. In an N qubit circuit, the algorithm can be highlighted as by the following steps:

Step 1: The inputs $\{x_i\}$ are encoded to a quantum state $|\psi_{in}(x_i)\rangle$, using a unitary $U(x_i)$, as described in 3.1.1.

Step 2: A parametrized unitary $U(\theta)$ is applied to the input state prepared in step (1), so that the output state is: $|\psi_{out}(x_i,\theta)\rangle=U(\theta)\ |\psi_{in}(x_i)\rangle$.

Step 3: Use a subset of Pauli operators -as explained above such a choice will make the computation of gradients a much simpler task- $\{B_j\}\subset\{I,X,Y,Z\}^{\otimes N}$ along with an output function F, to measure the expectation values of some chosen observables. Output can be then defined as: $y(x_i,\theta)\equiv F(\{\langle B_j(x_i, \theta)\rangle\})$

Step 4: Define a cost function $C(f(x_i), y(x_i,\theta))$ to be minimised by tuning θ. Optimization can be achieved by either 3 methods mentioned in 3.2.

Step 5: Evaluate model' s performance using a test set.

These steps are in principle the general scheme for variational trainable algorithms designed for hybrid training[40]. QVE and QAOA methods follow the same scheme but design the output in different manners.

The goal of such a quantum circuit is to be able to approximate the function *f* that provides the targets. If that can be done successfully, the model will be able not only to successfully classify the train data given as an input, but also to generalise its predictions correctly to new, unseen data. If that can be achieved, the model is said to be trained to learn!

How QCL can approximate a given function f

Let's for simplicity consider input data is one-dimensional. Then, given an input data *x* corresponding to the input quantum state $|\psi_{in}(x_i)\rangle$, the density operator $\rho_{in}(x)$ can be expanded by the set of Pauli operators B_j with linear coefficient functions $a_k(x)$ such that:

$$\rho_{in}(x) = |\psi_{in}(x)\rangle\langle\psi_{in}(x)| = \sum_k a_k(x) B_k, (3.36)$$

The output state in a such case is obtained by the action of a parametrised unitary transformation $U(\theta)$ in (3.36) and can also be expanded by a subset of B_j, let's name it $B_k=\{b_k(x,\theta)\}$. Thus, the expectation value of a Pauli observable can be written as

$$b_m(x,\theta) = \sum_k u_{mk}(\theta) a_k(x), (3.37)$$

,

with given coefficients $\{u_{ij}\}$, thus the output can be perceived as a linear combination of a_k functions under unitary constraints imposed on the parametrized u_{mk}.

QLC approximates a function *f* considering simple cases where an input state is created by single-qubit rotations. For example, the *N*-qubit state

$$\rho_{in}(x) = \frac{1}{2^N} \bigotimes_{i=1}^{N} [I + xX_i + \sqrt{1-x^2} Z_i], (3.38)$$

can be generated for any binary *x* by implementing single-qubit rotations of the form $\prod_{i=1}^{N} R^i{}_Y(\varphi)$. We have already discussed such rotations and how they can enhance the inference in a quantum circuit in the previous subsection.

The tensor product used to create the input state has an immediate effect; an arbitrary unitary transformation on this state results to an arbitrary *N-th* order polynomial as the expectation values of a selected observable to be measured. However, the highest order term

x^N is hidden in a nonlocal observable $X^{\otimes N}$ and in order to extract it one needs to transfer this observable to a single-qubit observable, using an entangling gate, like a C-NOT for example. Entangling nonlocal operations are the key ingredients of nonlinearity of an output. The nonlinearity created by the tensor product is the key for the approximation of analytical functions.

Compute the Gradients

Now, let's return back to the parametrised unitary $U(\theta)$ of (3.22) and suppose that it follows the usual schema of decomposition to a chain of unitary transformations

$$\prod_{j=1}^{l} U_j(\theta_l)$$

The goal of the circuit that applies $U(\theta)$ to $\rho_{in}(x)$ is evaluated by measuring the expectation value of an observable B, $\langle B(\theta)\rangle$. Then:

$$\langle B(\theta)\rangle = Tr(BU_l...U_1\rho_{in}(x)U_l^{\dagger}....U_1^{\dagger}), (3.39)$$

If it is ensured that $U_j(\theta) = e^{\frac{-i\theta O_j}{2}}$, which is the Case B of 3.2.3 and O_j is a Pauli tensor product

$$\frac{\partial\langle B\rangle}{\partial\theta_j} = -\frac{i}{2}Tr(BU_l...U_j[O_j, U_{j-1}..U_1\rho_{in}(x)U^{\dagger}{}_{j-1}..U^{\dagger}{}_1]U_l^{\dagger}....U_j^{\dagger}), (3.40)$$

Now, exploiting the property of the commutator of an arbitrary operator ρ

$$[O_j, \rho] = i[U_j(\frac{\pi}{2})\rho U^{\dagger}{}_j(\frac{\pi}{2}) - U_j(-\frac{\pi}{2})\rho U^{\dagger}{}_j(-\frac{\pi}{2})], (3.41)$$

and apply it in case of (3.40), where

$$\rho = U_{j-1}..U_1\rho_{in}(x)U^{\dagger}{}_{j-1}..U^{\dagger}{}_1, (3.42),$$

the following evaluation of the gradient is obtained;

$$\frac{\partial\langle B\rangle}{\partial\theta_j} = \frac{i}{2}Tr(BU_l...U_{j+1}U_j(\frac{\pi}{2})\rho_j U^{\dagger}{}_j(\frac{\pi}{2})U_l^{\dagger}....U_{j+1}^{\dagger}) -$$
$$\frac{i}{2}Tr(BU_l...U_{j+1}U_j(-\frac{\pi}{2})\rho_j U^{\dagger}{}_j(-\frac{\pi}{2})U_l^{\dagger}....U_{j+1}^{\dagger}), (3.43)$$

That has a straightforward interpretation; with the insertion *of a* $\pm\pi/2$ rotation generated by a Pauli product O_j the measurement of the respective expectation values $\langle B\rangle_j^+$ and $\langle B\rangle_j^-$ provide

us with an easy evaluation of the gradient $\frac{\partial\langle B\rangle}{\partial\theta_j}=\frac{\langle B\rangle_j^+-\langle B\rangle_j^-}{2}$. Thus, the gradients can be computed by a quantum device.

Conclusion

QCL provides a promising framework for the construction of a QNN that can accomplish both classification as well as regression tasks. The nonlinearities added by the tensor product and the entangling nonlocal operations required for measuring the expectation value of the chosen observable enable such a circuit to approximate any analytical function, which is a promising step towards learning. Moreover, if the parametrised unitaries of the circuit are chosen to be generated by Pauli products, the computation of partial gradients of the circuit is proved extremely easy to compute, allowing the optimization of QCL models to be made by analytical gradient descent where gradients are computed by a quantum device.

Furthermore, another possible quantum advantage of this algorithm comes when considering the equivalent of the input state (3.38) in case *x* is a *d*-dimensional input

$$\rho_{in}(x)=\frac{1}{2^N}\bigotimes_{k=1}^{d}\left(\bigotimes_{i}^{n_k}[I+x_k X_i+\sqrt{1-x_k{}^2}Z_i]\right), \quad (3.44)$$

It is obvious that input states automatically have an exponentially large number of independent functions as coefficient set to the number of qubits, meaning that QCL directly utilizes the exponential number of functions with respect to the number of qubits to model the target function.

3.3.3 Circuit-Centric Classification QNN

The Circuit Model (CM) proposed by Schuld and her colleagues[43] that will be presented here is a result of one of the most consistent multiannual researches in the domain of QML, the research that resulted in the book which inspired and guided this thesis. Summarized in a sentence CV is a strongly entangling parametrised quantum circuit that consists of single and controlled single-qubit gates with classical parameters that, provided a set of inputs encoded in amplitudes of its state, is hybrid trained to perform a classification task. The hybrid-training scheme used corresponds to Case A of 3.2.3 and utilized the strategy of *Classical Linear Combination of Unitaries* to retrieve the gradients and a classical computer to calculate the parameters' updates. The amplitude encoding ensures that the number of parameters needed

grows only poly-logarithmically to the input dimension and is the first variational circuit model to achieve so.

The CM's structure and action for inference with respect to a QNN classifier are resumed in the figure 11. The 4 steps of the algorithm are illustrated (needed for the inference part) and there is a 5th and 6th step needed for training. These 6 steps can be summarised as:

Step 1: State preparation routine S_x, for amplitude encoding of the input data
Step 2: Application of the parametrized unitary circuit
Step 3: Measurement of the first qubit
Step4: Postprocessing of the result and application of a decision step function
Step 5: Quantum computation of the gradients of a Cost function, by realizing a circuit similar this one but slightly changed
Step 6: Classical computation of the updated for the parameters

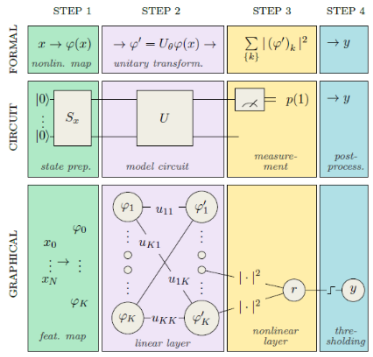


Figure 11 The 4 steps of CV model seen by 3 different perspectives

State Preparation

As mentioned in Section 2, there are several state preparation routines developed for amplitude encoding, with respect to qubit, amplitude or runtime efficiency. However, all of them are of a certain and non-neglectable complexity and the choice of the desired S_x depends heavily on the constraints the data can afford to be imposed to as well as the quantum properties and abilities of the quantum hardware chosen to perform the task. In general terms this can be achieved by applying a kernel function, that is a feature map, to the n-dimensional real-valued input vector x of the normalized input data, possibly pre-processed to be normalized and advanced with some non-zero padding terms to avoid distorting the initial data. Depending on the dimension of the Hilbert space the input data are projected to, the circuit to realize such routines can have linear or polylogarithmic dependency in the dimension n of the input data.

The parametrized unitary circuit

Given the encoded n qubit input state $|\varphi(x)\rangle$, the goal is to use the unitary operation $U(\theta)$ to perform the mapping $|\varphi'(x)\rangle = U(\theta)\,|\varphi(x)\rangle$. By now one should be familiarized with the decomposition of an L-depth circuit $U(\theta)$ of (3.22) into single or two-qubit gates where a single qubit gate G_k is expressed as

$$U_l(I_0 \otimes .... \otimes G_k \otimes ..... \otimes I_{n-1}), (3.45)$$

The 2-qubit control gates will be imprimitive, in the sense that they can map any two-qubit product state into a non-product state. Such a two-qubit gate acting on qubits *a* and *b* in the computational basis can be written as

$$C_a(G_b)\,|\,x\rangle\,|\,y\rangle = |\,x\rangle \otimes G^x\,|\,y\rangle, (3.46)$$

where *G* is a single-qubit gate and the state *x* of a is a pure state. Such a construction of *U* provides it with quantum universality, which means that it can generate the entire unitary group $U(2^n)$.

In order to make the circuit trainable and thus learnable, parameters must be inserted. The unitary nature of single-qubit gates that can be expressed as in the parametrized form of

$$G(a,b,c,\varphi) = e^{i\varphi}\begin{bmatrix} e^{ib}\cos a & e^{ic}\sin a \\ -e^{-ic}\sin a & e^{-ib}\cos a \end{bmatrix}, (3.47)$$

makes this task straightforward. The term $e^{i\varphi}$ can be neglected as one cannot physically measure overall phase factors, so considering only the remaining three parameters as learnable, the familiar form of parametrised single-qubit gates is obtained

$$G(a,b,c) = \begin{bmatrix} e^{ib}\cos a & e^{ic}\sin a \\ -e^{-ic}\sin a & e^{-ib}\cos a \end{bmatrix}, (3.48)$$

However, this is considered a specific case and a general set of elementary parametrised gates can be chosen base on the specific hardware and physical device used.

Circuit's Architecture

Circuit's architecture, i.e. which gate acts in which qubit and which act as the control ones also depends on the physical device that will realise the algorithm, the existent relationships in the input data as well as the specific task to be performed. The only goal is to keep the CM a low-depth circuit. Specifically, it is to keep the depth *L* of the circuit polynomial in the number of qubits *n*. If do so, the elementary qubit operations required will have a polylogarithmic dependency to the dimension *N* of data.

The obstacle that arises in such approach is that by keeping the circuit's depth low, only a limited set of amplitude vectors $|\varphi'(x)\rangle = U(\theta)\,|\varphi(x)\rangle$ can be reached. A solution to this may be offered by utilizing entanglement. If the circuit is strongly entangling, that is able to prepare strongly entangled states, the set of amplitudes it can reach grows impressively. To achieve that, Code blocks B are constructed. Code blocks are a layer of single-qubit gates *G* of the form (3.15) applied to each of the *n* qubits, followed by a layer of *n/gcd(n,r)* control gates, *r* standing for the range of the control. If *r* is relatively prime with *n*, all *n* qubits can be entangled.

The *k*-th controlled qubit gate can be expressed in the form (3.48) and let's denote it as $C_{c_k}\left(G_{t_k}\right)$,where $c_k=kr\ mod n$ the number of the control qubit and $t_k=(kr-r)mod n$ the target's one. Therefore, a code block *B* can be represented as;

$$B=\prod_{k=0}^{n-1} C_{c_k}\left(G_{t_k}\right)\prod_{k=o}^{n-1} G_k, \quad (3.49)$$

Figure 12 offers a handy visualization of an 8 qubit circuit with such architecture, consisting of two building blocks *B1* and *B3* with control range $r_1=1$ and $r_3=3$ respectively and in total *B1* and *B3* are comprised of 17 parametrised single-qubit gates of the form (3.48), and 16 trainable controlled single-qubit gates of the form (3.46).These gates must be decomposed into elementary constant gates suitable for the quantum device that implements the model. In the preferable case when the controlled gates are optimized to be single parametrised, 3(17+16)+1=100 parameters are to be learned. This number seems extremely small compared to the weight matrix of a simple 3layerd MLP of just 10 neurons in each layer. The power of CV is highlighted considering that these 100 parameters are enough to classify inputs of 2^8 = 256 dimensions.

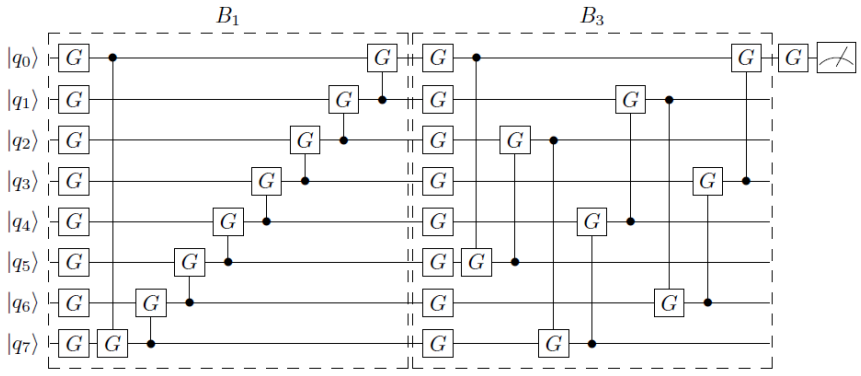


Figure 12 The architecture of an 8 qubit entangling circuit that consists of 2 building blocks

Measurement and postprocessing

The output of the CM is obtained by measuring the first qubit repeatedly, to obtain the probability

$$p(q_o = 1; x; \theta) = \sum_{k=2^{n-1}+1}^{2^n} |(U_\theta \varphi(\theta))_\kappa|^2, (3.50)$$

Or in Dirac notation using a Pauli *Z* operator for the measurement this can be expressed as

$$E(Z) = \langle \varphi(x) | U^\dagger (Z \otimes I \otimes .... \otimes I) U | \varphi(x) \rangle, (3.51)$$

In order to obtain a continuous output, if that is required, we can add a learnable bias *b* via a classical post-processing, so that the final output is now formed as

$$\pi(x; \theta, b) = p(q_o = 1; x; \theta) + b, (3.52)$$

Alike, using Dirac notation $\pi(x; \theta, b) = (\frac{E(Z)}{2} + \frac{1}{2}) + b$. A step function is then applied for a threshold value T, producing the final binary output.

Define the cost function

As a cost function, any cost function can be chosen, depending on the problem. However, the complexity of this choice has an immediate effect on the gradients that we will eventually need to compute. A safe choice is the standard Mean Square Error, which in the present case, given a train set *X* of *m* inputs with their respective labels provided in the set *Y*, with respect in the indexing of *X*, then the cost function to be minimised is:

$$C(\theta, b) = \frac{\pi}{2} \sum_{i=1}^{m} |\pi(x_i; \theta, b) - y_i|^2, (3.53)$$

Training the CM

In order to train the CM, that is optimize the parameters of the circuit, stochastic gradient descent will be used. In fact, single batch gradient descent is the key to keep the CM low-depth, as in case we want to train all inputs in superposition, which is possible thanks to quantum dynamics, the state preparation routine becomes linearly depended in the size of the batch and the dimension of the inputs.

For each parameter in the set θ, and similarly for b, the update rule given a learning rate h, is set to be

$$\theta_i^{(t)} = \theta_i^{(t+1)} - h\frac{\partial C(\theta, b)}{\partial \theta_i}, (3.54)$$

The derivative of the cost function is

$$\frac{\partial C(\theta)}{\partial \theta} = (\pi(x^m;\theta) - y^m)\partial_\theta \pi(x^m;\theta), (3.55)$$

and is a real-valued function. In order to obtain this value, the only obstacle is the computation of $\partial_\theta \pi(x^m;\theta)$. In case of the bias, this is easy because $\partial_b \pi(x^m;b) = 1$

For the rest of parameters, the derivation is not so trivial. In vector notation, the respective derivatives of the output are given by

$$\partial_\theta \pi(x^m;\theta) = \partial_\theta p(q_0 = 1; x^m;\theta) =$$
$$\partial_\theta \sum_{k=2^{n-1}+1}^{2^n} (U_\theta \varphi(x))^\dagger{}_\kappa (U_\theta \varphi(x))_k =$$
$$2\,\mathrm{Re}\{\sum_{k=2^{n-1}+1}^{2^n} (\partial_\theta U_\theta \varphi(x))^\dagger{}_\kappa (U_\theta \varphi(x))_k\}, (3.56)$$

Or in Dirac notation

$$\partial_\theta \pi(x^m;\theta) = 2\,\mathrm{Re}\{\langle(\partial_\theta U_\theta)\varphi(x^m) \,|\, Z \,|\, U_\theta \varphi(x^m)\rangle\}, (3.57)$$

Now, the only problem to solve is the fact that the gradient of a circuit is not necessarily the gradient. Thankfully, using the Classical Linear Combination of Unitaries routine, which is analysed in Appendix D, the gradients can be quantumly computed for the rest of the parameters as easily, by realizing slight variations of the initial circuit

Conclusion

The CM seems to be an excellent competitor as a near-term applicable QNN. Given N-dimensional data encoded in the amplitudes, the upper limit of $log_2 N$ qubits applies and this is the only case were no ancilla qubits are needed from the model. The power of this is illustrated from the fact that for the process of a 1000-dimensional dataset, only 10 qubits are enough. This limit is upper because CM model is designed in the simplest version possible to be easier to apply, but it is easily upgradable with more quantum advantages -given a complexity cost- such as the superposition of train data or the use of a tensorial feature map (kernel methods).

Simulations also shown that this model demonstrates robustness to noise, which is a much favourable skill for near-term applications, sine error correction in quantum hardware is in a very primary stage. Another advantage, inherited from the variational nature of the circuit, is the fact that learning to counterbalance systematic errors in the device architecture is possible.

The number of repetitions of the circuit needed from the measurement can be seen as sampling from a Bernoulli distribution *s* times. Using amplitude estimation, given an error ε and requiring an estimation probability >0.75, *s* is $O(1/\varepsilon)$ by the circuit depth $O(1/\varepsilon)$. The most "expensive" part of CM is the complexity added by the state preparation routine. As an example, for the state preparation of the 1000-dimensional dataset above 2048 gates are needed. Therefore, developing more effective state preparation routines is a very much desired next step for enhancing not only CM, but every possible QNN model. This problem can be abolished in case the CM is used to perform a task in quantum input data.

Finally, several simulations in classical datasets, provided by the authors, revealed that in case of several benchmark classical models like Perceptron, MLP or SVM, compared to which CM is much slimmer and compact, the model tends to overfit a lot. Even when regularization techniques like *dropout qubit* used, it still overfits badly. This sadly indicates that although trained properly, the QM model is not exactly a learnable model that can generalise. Moreover, how this general framework can be customized for every specific problem is a largely uninvestigated domain. Customization refers to basic decisions in the designment of the CM including the quantum circuit's architectures, which gates are applied in which qubits, initial parametrization as well as the number of code blocks used to create entanglement to the circuit. As a matter of fact, these practical issues arise in all variational quantum circuit QNNs met in the literature so far, forming a new domain of research necessary to make this framework realistically applicable. It should be noted that in the last months, some techniques for heuristics initialization and optimal architecture have been published[51,52,117], although considering very specific cases of variational circuits.

3.3.4 Another circuit-centric QNN

In this subsection another model that bears great resemblance to the previous one, published in the same year, is presented[41]. Although this model follows the same framework

created from variational circuits used as QNN models to be trained by hybrid methods, the main differences is that this one is based in basis encoding of input states -although it can also receive quantum data- possibly prepared in superposition, and analytically computes the gradients of the circuit as in Case B of 3.2.3 section. It can be used for inference in binary classification tasks, provided a labelling function *f(x)* along with train data *x* and in case the *f* is not a random function, which is translated to the existence of a pattern in training data, it can learn to generalise its predictions in new input data.

The main idea is that given a set of parametrized unitaries, with real value parameters, the unitary $U(\theta)$ of length L can be decomposed in the form (3.22). An input $x=x_1x_2....x_n$, $x_{i=}\pm 1$, can be encoded in the basis state $|x,1\rangle$ when an ancilla,(called readout by the authors) qubit is added in the end for the measurement. It is obvious that $n+1$ qubits are required for the encoding of the n-dimensional input. After the application of $U(\theta)$ on the input state the $n+1$-th ancilla qubit is measured using a Pauli operator, Y. Of course, due to the probabilistic nature of quantum systems, the final outcome is obtained after the procedure is repeated in several copies of the output. The classical parameters of $U(\theta)$ are optimized so that the outcome of the measurement meets, or comes as close as possible, to the provided label for x, $f(x)$. In order to do so, a cost function is defined such as;

$$C(\vec{\theta}, x) = 1 - f(x)\langle x,1|U^{\dagger}(\vec{\theta})YU(\vec{\theta})|x,1\rangle, (3.58)$$

At this point, it should be underlined that there are two ways to go. The input state can either encode one train input at a time, or all of them together in a superposition. Because of the complexity that comes with the superposition and especially the cost concerning circuits depth, we will focus our analysis in the first case and briefly discuss the potentials of the later in the end.

As in the scheme described each unitary U_i acts on the outcome of the previous unitary in the circuit, there are non-explicit non-linearities introduced. Thus, this is one of the cases that the is used to measurement adds the non-linearity needed for the QNN, standing for the activation function of the model.

Parametrised gates of the circuit

Until now, no new information was provided. The first distinction of this model is that the individual unitaries U_i of the variational circuit are not a decomposition of single-qubit gates, but as generalised Pauli matrices, therefore given a tensor product of Pauli matrices O, they can be expressed as $U_a(\theta) = e^{i\theta O}$ .This has the immediate effect

$$\| \frac{\partial U_a(\theta)}{\partial \theta} \| \leq 1 \Leftrightarrow \| \frac{\partial C(\theta)}{\partial \theta} \| \leq L, \quad (3.59)$$

L being the number of parameters. This is a very handy property for optimization methods based on derivatives, as it ensures that gradients will not blow up, a problem usually faced in classical ML.

Moreover, the quantum analogous of the Universal Approximation Theorem presented in 3.3.1 ensures that given a two-valued label function f we can construct a quantum circuit U that can be decomposed in one or two-qubit elementary gates $\underline{U_i}$, to approximate this function. By adding learnable classical parameters to these gates, the quantum circuit can be trained to achieve this approximation and this can be used to perform a binary classification task. However, the construction of these quantum circuit depends heavily on the choice of the tensor products of Pauli matrices O_i and, so far, there is very little known, if any, about a strategy to effectively chose these operators.

Compute the gradients and update the parameters

For the sake of argument, consider the problem of effectively chose the architecture of the quantum circuit as resolved; after all is far beyond this thesis scope as well as the author's knowledge to try to tackle such an issue. Then we may proceed to training.

Stochastic gradient descent will be used for a batch size of one train input at a time. Each input will be separately inserted in the circuit and then the parameters will be updated to move the output closer to the desired target value provided by f. A batch of bigger size can be achieved in case of superposition but, as already discussed, this will dramatically increase circuit's depth, thus this case doesn't tie with our near-term applicable perspective.

Repeated measurement of a Y Pauli operator acting on the ancilla qubit result to the output and the derivation of the cost functioned defined from this output in (3.30) is the next

step to go. The computation of the gradients falls with Case B of section 3.2.3 as the construction of this model ensured that each individual unitary of the circuit is of the form $U_k(\theta_k) = e^{i\theta_k O_k}$. In this special case it was recently proven[118] that the derivative of the cost function can very easily be computed quantumly as

$$\frac{\partial C(\theta, x)}{\partial \theta_k} = 2\,\mathrm{Im}(\langle x,1|U_1^\dagger U_2^\dagger ... U_L^\dagger Y_{n+1} U_L ... U_{k+1} O_k U_k .. U_1 | x,1\rangle), (3.60)$$

And since a linear product of unitary operators in a unitary operator, let's call it *D*, the above expression of a *L+2* depth's *D* can be written as

$$\frac{\partial C(\theta, x)}{\partial \theta_k} = 2\,\mathrm{Im}(\langle x,1| D | x,1\rangle), (3.61)$$

It is obvious that this can be quantumly computed by letting the *L+2* depth circuit to act on the input state and perform a measurement to the ancilla. By making repeated measurements of this ancilla, we can obtain a good estimation of the gradients in the price of an *L+2* circuit and 1 extra qubit. After done so, the derivative is really easy to be classically computed via the chain rule and also traditional ML rules can be used to classically update the parameters.

Superposition of input data

As promised in the beginning, we will now briefly discuss the potential of this QNN to work with data in superposition. If the problems of computational complexity and the cost of realizing quantum circuits of extreme depth will once be resolved, this potential offers a tremendous quantum advantage to a QNN in comparison with an ANN.

Such a superposition and how it can be achieved in basis encoded inputs have already been discussed in 3.1 and an efficient state preparation routine is analysed in Appendix A. In this case, consider two batches A and B of train samples, each one corresponding to the two possible values of our label function f. These states can be described in the framework of this QNN as:

$$f(A) = |1\rangle = N_+ \sum_{x_A \in A} e^{i\varphi_A} | x_A, 1\rangle$$
$$f(B) = |0\rangle = N_- \sum_{x_A \in A} e^{i\varphi_B} | x_B, 1\rangle$$

for N_+, N_- be normalizing factors and initial phases are set to 0. This, of course, means that data must be pre-processed from a quantum devise that prepares the superposition, and that one is a complex task.

Now, equation (3.35) describes the association of the circuit's unitary U with any label function. The fact that U is constructed to be diagonal in the computational basis of data qubits has the immediate effect of vanishing cross terms and make phases irrelevant. This means that state $|1\rangle$ is associated with the expected value 1 and state $|0\rangle$ is associated with the expected value 0. Note that in case the diagonality of U is not assured, no such assumptions can be made.

In case of a parametrised diagonal unitary $U(\theta)$, we can compute the expected value of Y_{n+1} acting on the state $U(\theta)|i\rangle$ for $i=1$ or $i=0$, corresponds to the average over all samples with the label i of QNN's predicted label values for the inputs. This leads to the empirical risk of the QNN to form a cost function;

$$C(\theta) = 1 - \frac{1}{2}(\langle 1|U^{\dagger}(\theta)Y_{n+1}U(\theta)|1\rangle - \langle 0|U^{\dagger}(\theta)Y_{n+1}U(\theta)|0\rangle), (3.62)$$

The minimization of this cost function is the training goal of a QNN in superposition. The derivation of this function creates several difficulties when trying to compute it analytically, so the development of efficient strategies to tackle this issue is still under investigation.

Conclusion

The work presented in this framework is an effort to design a very general framework for a QNN or in general an ML model to perform supervised tasks in a quantum computer. This framework is general enough to be adjustable on selected hardware's unique properties and offers the potential exploitation of several quantum properties such as entanglement and superposition to achieve quantum advance. It also tries to respect the restrictions imposed by the limited quantum resources of near-term much-expected quantum devices as well as the lack of error correction, by keeping circuits depth to as low as possible. Proof about the learnability of such a model is provided and numerical simulations have demonstrated its ability to correctly perform classification in Parity and Majority problems. Moreover, the hybrid framework that this model can be trained in is easy to understand and realizable.

However, the generality of this framework arises as many questions as it answers, since the most important key idea, the architecture of the circuit and the strategy to select the unitaries used to approximate a label function is not investigated at all. Another limitation comes from the discrete nature of the data that basis encoding dictates and the very specific nature of the Boolean label function. Basis encoding creates another problem that can be detected in comparison to the previous model presented that utilises amplitude encoding techniques. That is the exponential number of qubits required. In addition, complexity added by the state preparation routines as well as the necessity of repeated measurements of the output to obtain an acceptable estimation of observables expected value also holds in this case. Finally, accuracy issues related to the lack of proper error correction routines in quantum hardware also apply in this case.

3.3.5 A Continuous-Variable QNN

Quantum computation methods are traditionally designed with respect to discrete systems based on manipulating qubits as a computational unit. However, a quantum system consists of several other variables, rarely discussed, such as position, momentum or the, less rarely discussed, amplitude of their states, which are continuous. Given the continuous nature of computations in an ANN, it arises as an interesting approach to turn into *continuous-variable quantum computing*[119]. A brief introduction in CV computing based on phase space formulation can be found in Appendix E.

The first idea for QML algorithms in a CV model came less than a couple of years ago based on QML subroutines designed for an all-photonic continuous-variable quantum computer[120]. A quantum classifier[46] and a Gaussian regression algorithm[121] as well as the framework for a QNN have been proposed in this context, however, the CV model is yet a rather unexplored corner in the field of QML. The main reason this approach is reluctantly seen from the research community is the fact that noise and precision difficulties particularly arise when dealing with continuous variables, especially in comparison with their discrete counterparts. Unfortunately, previous work in quantum error correction[122,123] in case of continuous variables did not go too far, although interesting proposals made their way to publication in recent years[124], giving birth to an interest in developing a QNN based on quantum CV computing. Also, the possible realization of a CV computational model can, among others, be achieved through optical systems[125] like the photonic quantum computer which, due to its lower cost compared to others, has drawn heavy attention from the industry

in recent years. In fact, several start-ups worldwide are working on creating their own photonic quantum computing device (see Xanadu for example).

In this final part of Section 3, the general scheme for a QNN as a quantum circuit in a CV model will be presented and discussed, following the work of Killoran[47] et al. Table 3 presents a tidy outline of the analogues between a ANN and a CV-QNN. Briefly, following the phase space formulation, a CV model encodes information in two conjugate real valued variables x, the position, and p, the momentum, assigned to the quantum state of a bosonic mode, called qumode. Qumodes are the continuous analogue of qubits. X and P operators are defined based on the two variables, so that real=valued functions $F(X,P)$, the quasiprobability distributions, represent the qumode states. A universal gate set consists of 3 basic single-mode Gaussian gates; the Rotation gate $R(\varphi)$, the Displacement gate $D(a)$, the Squeezing gate $S(r)$, a two-mode Gaussian gate; the phaseless Beamsplitter $SB(\theta)$ - a rotation between two qumodes- and a non-Gaussian gate $\Phi(\lambda)$ like Cubic phase or Kerr gate, or even a measurements of specific qumodes. A detailed description for the above framework can be found in Appendix E.

Table 3 Analogues of Continuous Variable-QNN and classical ANN

Classical ANN	**CV -QNN**
feedforward NN	CV variational circuit
weight matrix W	symplectic matrix M
bias vector b	displacement vector a
affine transformations	Gaussian operators
nonlinear activation function	non-Gaussian gates
weight/bias parameters	gate parameters
variable x	position operator X
derivative dx	conjugate momentum operator P
no classical analogue	entanglement
no classical analogue	superposition

The main idea is that a quantum variational circuit can represent one layer of the QNN and the combination of several circuits will result in a QNN which, depending on the circuit's architecture can be the quantum equivalent of classical feedforward Multilayer Perceptron, or

possibly Recurrent, Convolutional or Residual NN. A layer consists of every gate in the gate set and can be described as:

$$L := \Phi \circ \otimes_{i=1}^{N} D(a_i) \circ U_2 \circ \otimes_{i=1}^{N} S(r_i) \circ U_1, (3.63)$$

where $U_i = U_i(\theta,\varphi)$ are n-mode unitary operators that can be decomposed in basic beamsplitters and rotation gates, called linear optical interferometers. The classical parameter set to be optimized is $(\theta,\varphi,r,\alpha,\lambda)$, therefore a hybrid training scheme from those presented in 3.2 can be utilized. The architecture described in (3.63) is illustrated in Figure 12. The first four blocks of unitaries in the circuit carry out a parametrized affine transformation on the n input qumodes which, given a symplectic matrix M can be described as:

$$\begin{bmatrix} x \\ p \end{bmatrix} \rightarrow M \begin{bmatrix} x \\ p \end{bmatrix} + \begin{bmatrix} \mathrm{Re}(a) \\ \mathrm{Im}(a) \end{bmatrix}, (3.64)$$

Therefore, (3.64) can be interpreted as an equivalent to the weighted sum of a neuron's inputs in an ANN and $\otimes_{i=1}^{N} D(a_i) \circ U_2 \circ \otimes_{i=1}^{N} S(r_i) \circ U_1$ as the weight matrix of a fully connected layer. The non-linearity of an activation function is introduced to the model quite naturally and without disturbing the unitary nature of the system by the non-Gaussian function Φ, so that:

$$L(x, p) = \Phi(M(x, p) + a), (3.65)$$

A QNN can be constructed by combining several layers of the above form, where the output of each layer is the input of the next one. Moreover, layer's width can vary based on the number of qumodes available in each level, for example tracing out extra qumodes, likewise tracing out qubits discussed in 2.1.2. A QNN of this form can work with both quantum and classical data. The state preparation required for the encoding of classical train data in the qumodes can be achieved via displacements which carry the input data information on their parameters and are applied to the ground state. Finally, the output of the QNN can is obtained via a measurement to the last circuit's output state, or repeated measurements to estimate the expectation value of a pre-chosen observable. In a CV model a measurement can be achieved via several operators, like photon-counting.

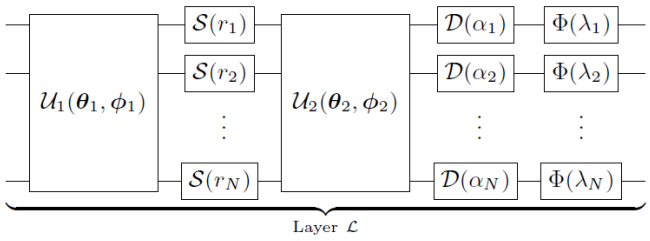


Figure 13 The circuit structure for a layer of a Continuous Variable QNN

Now, we will discuss the simplest case where superposition and entanglement are omitted and a quantum equivalent of a classical feedforward ANN can be built in this framework. Consider the input data to be real-valued vectors $x=(x_1,..,x_n)$ of n-dimensions, encoded in the eigenstates $|x_i\rangle$ of X operators assigned in n qumodes, so that:

$$\vec{x} \rightarrow |x\rangle = \otimes_{i=1}^{n} |x_i\rangle, (3.66)$$

As a reminder, X_i and P_i operators for one qumode are defined to be:

$$X_i = \int_{-\infty}^{\infty} x_i \,|\, x_i\rangle\langle x_i \,|\, dx_i, (3.67)$$
$$P_i = \int_{-\infty}^{\infty} p_i \,|\, p_i\rangle\langle p_i \,|\, dp_i, (3.68)$$
$$, where \langle x \,|\, x'\rangle = \delta(x-x') \quad \langle p \,|\, p'\rangle = \delta(p-p') \quad \langle p \,|\, x\rangle \sim e^{ipx}$$

(3.66), given a bosonic annihilator operator $A = \frac{1}{\sqrt{2}}(X + iP)$ can be described as:

$$|x\rangle = \otimes_{i=1}^{n} |x_i\rangle = \pi^{-\frac{n}{4}} e^{(-\frac{1}{2}x^T x + \sqrt{2}x^T A^\dagger - \frac{1}{2}(A^\dagger)^T A^\dagger)} |0..0\rangle (3.69)$$

As a first step, we furtherly require that the QNN will not mix among X and P operators, thus momentum variable will be discarded. The goal is to create a quantum circuit that given an input state $|x\rangle$ will return an output state $|\sigma(Wx+b)\rangle$,σ the selected activation function. The weight matrix W, which for simplicity is assumed full ranked, is a symplectic matrix assigned to the act of Gaussian operators in the system, so it can be decomposed via Euler decomposition to $W=O_2BO_1$. O_1 is an orthogonal matrix related to an interferometer U_1, constructed solely from phaseless beamsplitters, so that the matrix is kept block-diagonal to prohibit the mix of X and P. Therefore

$$U_1 |x\rangle = U_1[\otimes_{i=1}^{n} |x_i\rangle] = \otimes_{i=1}^{n}[|\sum_{j=1}^{n} o^1_{ij} x_j\rangle] = |O_1 x\rangle, (3.70)$$

In the same manner the interferometer U_2 is correlated with the orthogonal matrix O_2 so that

$$U_2 \mid x\rangle = U_2[\otimes_{i=1}^{n} \mid x_i\rangle] = \otimes_{i=1}^{n}[\mid \sum_{j=1}^{n} o^2{}_{ij} x_j\rangle] = \mid O_2 x\rangle, (3.71)$$

In the next step, the squeezing gate S is constructed with respect to matrix the positive diagonal matrix *B*. If *B* is described as $B:=diag(\{c_i\})>0$, then for $r_i=log(c_i)$ the squeezing gate is defined via

$$S(r) \mid x\rangle = e^{-\frac{1}{2}\sum_i r_i} \mid Bx\rangle, (3.72)$$

The act of a squeezing gate on the eigenstates of X_1 for the operator of a single qumode is therefore

$$S(r_i) \mid x_i\rangle = \sqrt{c_i} \mid c_i x\rangle, (3.73)$$

The bias vector b is added through the set of single-mode displacement operator given the selected parameter vector b, so that

$$D(b) \mid x\rangle = \mid x + b\rangle, (3.74)$$

Therefore, the affine transformation corresponding to the weighted sum of inputs in one Layer of the QNN is prepared in the quantum circuit by

$$D \circ U_2 \circ S \circ U_1 \mid x\rangle \rightarrow \mid O_2 B O_1 + b\rangle = \mid Wx + b\rangle, (3.75)$$

The final step is the quantum equivalent of a nonlinear real-valued activation function σ, which is inserted to the quantum circuit via a non-Gaussian operator Σ associated with σ, so that

$$\Sigma \mid x\rangle = \mid \sigma(x)\rangle, (3.76)$$

Note that if Σ is chosen to be a polylogarithmic function of a fixed degree which locally acts on each mode, as the activation function in an ANN locally acts on the input of each neuron, any function with a convergent Taylor series can be efficiently approximated by the QNN. Thus, the final quantum circuit representing one layer is

$$L := \Sigma \circ D \circ U_2 \circ S \circ U_1 \mid x\rangle = \mid \sigma(Wx + b)\rangle, (3.77)$$

The output state *L* is used as the input for the next quantum circuit layer and in the final *L* the QNN's output is obtained via a measurement. The circuit corresponding to (3.77) is graphically represented in Figure 14. This measurement in the CV model can be a homodyne detection in

each qumode, which projects onto $|x_i\rangle$ states. Finally, a QNN designed as above can be hybrid trained with any of the methods discussed in 3.2, given a cost function C to be minimized.

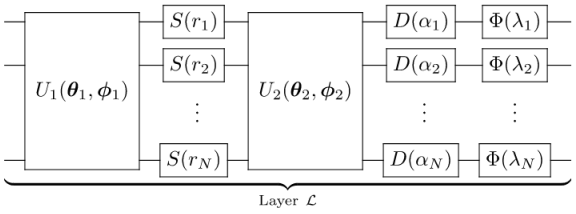


Figure 14 The circuit of a layer in the CV-QNN

Conclusion

Given the fact that each quantum circuit forming a layer in the proposed QNN incorporates every gate in a set of universal gates, it automatically inherits all the capabilities of a CV gate quantum computer. Moreover, nonlinearities respecting to activation functions are elegantly introduced via non-Gaussian operators without disrupting the quantum nature of the model. This framework is so generic and flexible that theoretically can be modified to create several NN architectures, classical and fully quantumized.

Realistically, although it constitutes a very promising approach for building a QNN, it requires further research to construct a guideline of circuit architectures and gate combinations corresponding to these architectures. Moreover, the model complexity of such a circuit, its dependency on qumodes and elementary gates is uninvestigated, so although the hybrid training scheme subsumes this idea in the family of possibly near-term applicable QNN models, no absolute conclusions can be drawn. Finally, the possible exploitation of quantum properties such as superposition, symmetry, interference and entanglement remains an open question, promising further quantum advantages added to this scheme.

4 Quantum Supremacy and Challenges

4.1 Expected Quantum Advantages

There have been several proposals on how to measure the quantum advantage of a model in comparison to a classical one as well as in their general assistance in the domain of ML. The literature till 2010 summarized the expected quantum advantages in the following list[7,9,13,31,126]:

1) Exponential Memory capacity

2) Higher performance with lower complexity (a smaller number of neurons, less required training data)

3) Faster learning (due to superposition)

4) Elimination of spurious or erased memory duo the absence of pattern interference

5) Solution to linear inseparable problems with single layer QNN

6) Absence of wiring (thanks to entanglement property)

7) Higher stability and reliability

8) Processing speed (10^{10}bits/s)

9) Small scale (10^{11} neurons/mm^3)

Almost ten years later, some of these advantages have seen some kind of proof, mathematical or numerical, and others although lay their foundations in the very nature of quantum mechanics, such as (3), (6) or (8), come with the price of qubit and gate complexity in a model, a price that the hardware development so far cannot afford. Of course, a QML model does not need to offer all shorts of the quantum advantages expected, even one from the list is enough to declare supremacy over traditional models for the same task. In recent years,

the conversation of the expected quantum advantages has taken a more realistic and pragmatistic turn and recent literature[27,33,35] discusses three dimensions of quantum advantage; computational complexity, sample complexity and model complexity. This thesis will focus on the latest approach and in this subsection discuss these three dimensions will be discussed in an effort to see which quantum advantages from the list of expectancies of the past survived the progress in Quantum Computing and in which dimension it is allocated.

4.1.1 Computational Complexity

The runtime speedups in QML algorithms lay their foundations in quantum computing itself and naturally they lead the discussion of expected quantum advantages in the domain of ML. In order to look at computational complexity, the terms of runtime of an algorithm and the meaning of asymptotic computational complexity should be clearly defined. The runtime for the execution of an algorithm is simply the multiple of the number of algorithm's elementary operations times their respective execution times. Since this is a theoretical measure and depends on the device which implements the algorithm, a more appropriate and universal measurement is required. Thus, computational complexity theory used the asymptotic complexity of an algorithm instead. By the term asymptotic one means the rate of growth of the runtime with respect to the input's size N. In general terms for a conventional computer, an algorithm is tractable if it has a polynomial dependence on N, that is no more than k^N, for a reasonably large k. In another case, if the dependency is exponential, the algorithm is intractable.

As far as a quantum algorithm is concerned, quantum complexity theory tries to answer the question of whether quantum computers can offer runtime speedups, also called advantages or enhancements, in computational problems compared to classical ones. In case they do so, they can be distinguished in qubit-efficient and amplitude-efficient, concerning whether these algorithms are polynomial to the number of qubits or amplitude respectively. If a quantum algorithm demonstrates an exponential speedup, it is often called quantum supremacy. For example, Grover's algorithm[87] is an amplitude-efficient algorithm with quantum supremacy.

In general terms a wider definition of quantum speedups came as a need in order for QML to move towards explanatory approaches instead of hunting the holy grail of quantum supremacy and results to some early applications to demonstrate the theories. Thus, the recent

QML research community has adopted the following categorization of possible quantum runtime speedups[127]:

1)Provable quantum speedup: There is proof that the algorithm performs better than any classical algorithm could ever perform. Such an example is, of course, the Grover's algorithm. A very recent example, that also comes from QC and not QML domain in particular, is this of Google's 53-qubit superconducting processor solving the classically intractable task of sampling the output of a pseudo-random quantum circuit[128]

2)Strong quantum speedup: The algorithm performs better than the best possible classical algorithm. This is the speedup that Shor's algorithm provides[129]

3)Common quantum speedup: The algorithm performs better than the best available classical algorithm. Such an example is the quadratic speedup achieved in an algorithm proposed for training classical perceptron by utilizing amplitude amplifications[91].

4) Potential quantum speedup: The algorithm performs better compared to a specific classical algorithm. Such an algorithm to consider is a simple Quantum Perceptron proposed to solve XOR problem, which is intractable by a classical perceptron but not intractable in general ML.[31]

5)Limited quantum speedup: The algorithm performs better compared only to specific corresponding algorithms. Such an example is quantum annealers.[130]

As QNN inherits its enhancements from quantum information processing, the expected speedups in the domain derive directly from the speedups in quantum computing. One of these is quadratic speedup, when Grover's algorithm is utilized in the process of learning such as quantum associative memory models and QNN build as equivalents of Hopfield ANN[13,14,131]. The exponential speedups are mostly met as common speedups when it comes to amplitude encoding and linear-algebra-based algorithms, and are not expected in terms of practical applications due to the severe constraints that need to be imposed in the sparsity of data[54,89,132]. However, amplitude amplification techniques utilized in QML algorithms in circuit architectures demonstrate potentials of some peculiarity that should not be disregarded[133].

Polylogarithmic speedups in case of strong and common speedups is the type of speedups met in all the QNN models presented in Section 3. Variational quantum circuit models demonstrate this speedup compared to classical NN regarding a binary classification task quite naturally, thanks to quantum parallelism. Since a state can be modelled as it is holding both possibilities, any operation could potentially naturally compute both probabilities at once.

4.1.2 Sample Complexity

The second domain that classical ML could benefit from concerns the size of the train set an algorithm needs as inputs in order to be sufficiently trained. That is the objective function to be within an arbitrarily small error of the best possible function, with probability arbitrarily close to 1, or simply said to achieve generalization power. That, in statistical learning theory, is called sample complexity[134]. There are two kinds of elements that can serve as train inputs; these obtained by sampling from a certain distribution and those that are computed as outputs to specific inputs, called queries. Based on this distinction, sample complexity separately for each case will be discussed.

Learning from examples

In classical ML, the question that reflects the sample complexity in the case the train set consists of examples drawn from an original concept that represents the problem via an example oracle, is whether an upper limit can be found for the size of a train set, given a desired error of probability ε, to be less than $1-\delta$, for a small enough δ. In classification tasks that is equivalent to the probability ε of an input to be incorrectly classified being equal to $1-\delta$. The answer is provided by the work of Valiant in Probably Approximately Correct (PAC) learning[135] , and not surprisingly is tangled with the Vapnik-Chervonenkis (VC)-dimension of the model. In summary, it states that a d VC-dimensional algorithm for a task of class C has a lower limit of $\Omega(\frac{1}{\varepsilon}\log\frac{1}{\delta}+\frac{d}{\varepsilon})$ train examples, but no more that $O(\frac{1}{\varepsilon}\log\frac{1}{\delta}+\frac{d}{\varepsilon}\log\frac{1}{\varepsilon})$ are required.

The quantum equivalent of PAC learning, concerning disjunctive normal forms, started to form more than two decades back[136], with the definition of a quantum example oracle as a qsample. In Section 2 the notion of qsamples by $\sum_i \sqrt{p(x)}\,|\,x,\,f(x)\rangle$ was introduced. In this

context, the amplitudes of the basis states represent the distribution *p(x)* from which the examples are drawn. Under very certain circumstances a speedup has been proven by interfering with these amplitudes via a quantum Fourier Transformation, but the restrictions imposed to the data and the computational complexity cost to achieve this interference are forbidding the declaration of a general speedup rule.

In contrary, in 2004 it has been proved that the PAC framework extends to QML learnability[137] and a polynomial equivalence stands for classical and quantum sample complexity:

"[F]or any learning problem, if there is a quantum learning algorithm which uses polynomially many [samples] then there must also exist a classical learning algorithm which uses polynomially many [samples]."

That practically means that for a class of *C* Boolean functions, if a quantum algorithm requires *M* evaluations of a quantum example oracle to achieve learning, then there exists a classical algorithm that can learn the same class of functions if provided with $O(nM)$ classical train inputs. This theory was improved several times[138,139] to finally come to its final form stating that in terms of PAC as well as agnostic learning, classical and quantum sample complexity are equal, up to a constant factor. This means that the very same upper and lower bounds apply in case of quantum learnability and therefore, in terms of sample complexity with train samples provided as examples drawn from a distribution, no exponential speedups are expected and we have to limit ourselves to linear improvements at best.

Learning from queries

Let's now consider the latter case where train inputs can be obtained as outputs to very specific inputs. In case of a classification task, this is called learning from membership queries and refers to the process of approximating (thus learning) a boolean function *f* by querying a membership oracle with specific inputs *z* and obtain the value *f(z)* as the output. In the very same sense, given an input *z* encoded in the computational basis and resulting in a state $|z,0\rangle$, a quantum oracle is a unitary operator U such that $U|z,0\rangle = |z,0 \otimes f(z)\rangle$. Note that the last qubit is once again used as an ancilla to be measured and measurement of this ancilla represents the output of the algorithm.

This seems to be very promising in terms of quantum computing speedups as there are several quantum algorithms that demonstrate the efficiency of only a single quantum query in specific types of problems[140,141], by applying the quantum oracle in a uniform superposition of qubits prepared to a register, achieving parallel processing of all possible inputs at once. These algorithms are extremely useful for a lot sort of computational problems, but they offer no learnability about the function and therefore, no speedup in case of QML is straightforwardly inherited.

The specification of learning boundaries in QML algorithms' sample complexity has been the object of extensive and consecutive research[138,142–144], which in an expanded framework called *"impatience learning*", resulted to the following conclusions:

1)Given a membership oracle query complexity $O(\sqrt{|C|})$, there exists quantum learning algorithm with to resolve a classification task in the class $C=\{C_i\}$, where $|C|$ approaches infinity.

2) Given learning algorithm with membership oracle queries and a parameter $\gamma \leq \frac{1}{3}$ indicating how easy it is to find distinguishing patterns within the classes in C, the sample complexity in the classical ML has an upper bound of $O((\log|C|)^{-\gamma})$ while in QML it is formed to $O((\log|C|)^{-\sqrt{\gamma}})$.

The latter conclusion when furtherly investigated[137] led to a result equivalent to the one in the case train examples are obtained by sampling. That is for a class of *C* Boolean functions, if a quantum algorithm requires *M* quantum membership queries to achieve learning, then there exists a classical algorithm that can learn the same class of functions by at most $O(nM^3)$ classical membership queries. In conclusion, no exponential speedup can be expected in neither this case, as the quantum algorithm provides at most polynomial overhead. Fortunately, the very same literature that proved these limitations of quantum learning provides us with examples of problems that, although intractable in classical ML, can be solvable via QML thanks to computational complexity speedups. Moreover, there has been evidence that in case of oracle-based problems, such as learning parity, a quantum advantage emerges in existing noisy systems[42,145], and this robustness to noise not only arises hopes for near-term applications, as will be discuss afterwards, but it also provides quantum algorithms with

another, widely uninvestigated yet, class of quantum advantages; their possible superior robustness to noisy data. This indicates that one has to look to all three dimensions of quantum enhancements to answer about a QML model's contribution to ML.

4.1.3 Model Complexity

In terms of model complexity, the flexibility, capacity and power of a model are concerned. It is well known from classical ML theory[116,146] that more flexible models, that is easiest to fit train data come from complex model families, unfortunately paying the cost of overfitting[147,148]. In classical ML, a model's complexity is associated with the VC-dimension of the model in terms of expressing an upper bound for the expected generalization error which demands the lowest flexibility of a model with respect to the lowest training error possible. Model complexity can form a quantum advantage in models deriving from the explanatory approach, where the goal is not to mimic a classical ML algorithm but to create a quantum one based on the device available to implement it, designing the algorithm with respect to the strengths and limitations it provides.

A quite straightforward theoretical way to address possible quantum advantages to model complexity would, therefore, be to analyze the VC-dimensions of both a quantum algorithm as well as its classical counterpart's one. However, this task is an extremely challenging task even in case of classical algorithms and demands very complex mathematical and numerical studies. In literature very few ML algorithms, both quantum and classical, come with an analysis of model's VC-dimension, so no assumptions about possible quantum advantages can be confirmed this way. Moreover, this investigation relies heavily on the quantum, or classical, hardware used to implement an algorithm as well as the particularities and subroutines of each algorithm in specific, so no general conclusion can be made and each algorithm, or family of related algorithms, have to be analyzed and compared separately.

Another open question is whether there exist quantum models that, for specific kinds of pattern, demonstrate an increased ability in capturing these patterns or correlations among data. This is, again, unexplored waters although an obvious conclusion can be made regarding the QQ case. When it comes to QML with quantum data, data produced from a quantum system the quantum advantage offered is immediate[33].

In terms of model capacity, there is a specific kind of QML routines that seem to offer a quantum enhancement, in fact an exponential enhancement, in models based on associative memory[90,131,149], paying of course a fair price in terms of computational complexity. Hopfield quantum neural networks utilize the fact that quantum systems are specified by exponentially many complex-valued amplitudes to store input data in those amplitudes and via quantum recall processes output superpositions of data to the QNN, suggesting a more efficient utilization of their associative memory. Due to the severe cost of amplitude encoding, the complexity of preparing data's in superposition to a register and the unavailability of a QRAM to support it this idea unfortunately have failed so far for most applications.

Finally, if loosen from a strictly mathematical prove for quantum enhancements, one can assume some improvements that arise naturally in case of variational quantum circuits as QNNs. One of these is the natural ability of quantum gates to represent a rotation operator as there exist unitary quantum gates which represent rotations around the Bloch sphere. This provides the quantum model with a very simple and efficient way to transform data and easily explore hyperplanes for classification or detect several patterns. Another one is, of course, the possible superposition between '0' and '1'which enables the encoding of N bits information in $log_2 N$ qubits and offers a great improvement in storage capacity of the model. Finally, in case the gates are considered as layers on a QNN, with respect to a classical ANN, then another advantage can be made considering the potential of reduction in features space offered by the quantum variational circuit, leading to reduced number of layers and nodes, thus a simpler model (still though this comes with no proof in principle for the learnability skills of such a model).

4.2 Quantum Challenges

The challenges that QML faces are in a very large part similar to the ones faced by the quantum computation domain. And they all stem from the fact that the available hardware is still in the development process as well as the difficulty to fully explore and prove special quantum properties with no classical analogue to guide this exploration. The macroscopic scale of atoms and subatomic particles that quantum theory applies on as well as the probabilistic nature of quantum mechanics create further obstacles. One the one hand, the technology available to observe and manipulate particles of such scale is in development process during the last two decades, and it also requires a tremendous effort in order to be manipulated without

disturbing the natural need of single quantum systems to be isolated from the rest of the universe in order to remain coherent.

Therefore, scalability is a big issue. The larger-scaled a computer, the harder it is to keep it quantum. The phenomena that promise to make quantum computers really powerful, like entanglement, require controlled qubit interactions. Architectures that allow this control are hard to engineer and hard to scale. Moreover, the thermodynamic conditions of the quantum regime are very challenging to access. When a quantum system is finally achieved, it's difficult to keep it isolated from the environment which seeks to decohere it and make it classical again. Even though the notion of the quantum parallelism is, theoretically, the "quantum advantage" by which a quantum algorithm outperforms a classical one, it has not yet been experimentally demonstrated. This is because it is extremely difficult to maintain quantum phenomena in real physical systems[150]. Thus, all these require extremely low temperatures as well as extremely expensive materials.

On the other hand, a probability-based system is going to have small errors at each piece/circuit/qubit, and when you have systems of 100s, it adds up. The probabilistic nature also requires repeated measurements to obtain a good estimation of the result's expected value, and the no-cloning theorem offers no facilitations. The no-cloning theorem comes with an extra obstacle, regarding the error-encoding: You cannot trivially employ more than one particle to encode a qubit. Instead, all gates must operate so well that they are not just accurate to the single-particle level but even to a tiny fraction of how much they act on a single-particle (to the so-called quantum-error correction threshold)[151]. This is much more challenging than to get gates accurate merely within hundreds of millions of electrons, which is the case met in the classical computer. As the fountains of challenges in QML have been mentioned, let's go deeper and introduce the challenges themselves, grouped in 4 main categories [33] :

(1) The input problem, addressing to the great cost of state preparation routines when dealing with classical data, as well as the struggle to find the most efficient quantum representation for the classical data in the quantum system

(2) The output problem, reflecting the fact that an efficient way to obtain the solution has also to be figured. In case of encoding the expected output in the expected value of a quantum state requires repetitions of the quantum system to serve repeated measurements to

estimate this value. On the other hand, obtaining the solution as a string of bits requires learning an exponential number of bits.

(3) The costing problem, which refers to the computational complexity of an algorithm requiring thousands of gates to be implemented in hundreds of qubits. Such an immense quantum system is very hard to manipulate and control, and the materials required to construct a quantum device of this scale, even if all fidelity, noise and decoherence problems were solved, remains elusive if not forbidden.

(4) The benchmarking problem, closely related to the expected quantum advancements discussed in 4.1. No quantum enhancement can be declared or established until a valid theory for comparing classical and quantum ML algorithms shall be formulated. A clear and quite recent example that demonstrates the confusion this topic creates can be given by the case of quantum-inspired classical algorithms proposed by Tang at the end of 2018. These are classical algorithms inspired by the quantum solutions, based on the HHL algorithm[152,153]. Tang proved that the classical "dequantumized" algorithms solve the respected problems in a similar time than the quantum algorithms, under similar assumptions. Thus, this demonstrated that the much-acclaimed quantum speedup was in these cases not real.

4.2.1 The hardware

The very best scenario for the immediate implementation of QNN models as the ones discussed in this thesis, and in general case all the ones that have been proposed so far in QML following any of the approaches discussed in Section 2, requires the existence of a quantum computer that offers;

a) Universality. It traditionally means that the quantum machine can perform any physically possible computation. This attribute is equivalent to the ability of a quantum computer, based on any quantum system, to implement all possible unitaries

b) Error-correction. This refers to the fact that the outcome of any algorithm realized by the quantum computer is the one expected by the mathematical and physical equations that theoretically describe the algorithm. An error correction mechanism is needed to detect and correct errors as gates have limited fidelity, due to the decoherence of qubits leading to disturbed calculations.

c)Large- scale. The number of qubits available to support high circuit's width as well as the number of gates, responding to circuit's depth, that can be implemented by the quantum computer is large enough to meet the algorithm's demands.

However, the first quantum computing devices meet none of these criteria. In fact, they have been characterized as noisy intermediate-term as well as small-scale devices[154] . In case of universality the challenges vary depending on the technology used to realize a quantum system. Concerning the large-scale just consider the fact that the very best algorithms presented here desire more qubits for the manipulation of 6-dimensional input data than those most recently available by Google's 53-qubit quantum computer, acclaimed just a week before these lines are written[128]. Moreover, near-term applicable QML algorithms are expected to require a circuit-depth of no more than 1000 gates, and we've seen so far that in some cases only for the state preparation of data more than 2000 gates are required.

4.2.2 Decoherence

Decoherence stands for the loss of information occurring in a quantum system by its interaction with the environment, in case of measurement, vibrations, electromagnetic waves or temperature fluctuations for example. Unfortunately, it also occurs spontaneously, even if we do not interfere with the system. These interactions result in decouples of state vector's components and entanglements with their surroundings. At the same time, the global state of the quantum system and the environment remains coherent, it's only the observing system that becomes decoherent. Consequently, this decoherence explains only why the observer can no longer witness the superposition, not the discontinuity of the measurement.

One can detect the conflicts created by the reality decoherence dictates in QML algorithms. From the very begging, we agreed that a QML algorithm should efficiently deploy the unique quantum properties and evolve with respect to quantum effects such as superposition, entanglement and interference while at the same time it should be consistent with the quantum principles[7]. In order for QML algorithms to reach a quantum enhancement or simply be realized, they rely on the undisturbed evolution of quantum superpositions along with extensive control of the quantum system so that non-trivial operations can be implemented. Quantum coherence is fragile and can be preserved for particularly narrow time windows of one second in best and rare cases[155], so all algorithm's operations must also obey to time limitations.

Although some great steps towards controlling errors created by decoherence have been recently made by IBM[156], the techniques proposed can only be applied in quantum circuits of very sort depth and stop demonstrate efficiency for more than 4-5 qubits. Thereby, the needs

of even very simple QML models are refrained significantly from the very best schemes for decoherence control we have so far available.

4.2.3 The need for a qRAM to exploit superposition

A quantum Random Access Memory (qRAM) is, as the name declares, a quantum equivalent of the classical Random-Access Memory with uses $\underline{n}$ bits to randomly address $N=2^n$ distinct memory cells. Likewise, qRAM uses n qubits to address any quantum superposition of N memory cells. Practically, that is to encode in superposition N n-dimensional input vectors into $log(Nn)$ qubits in $O(log(Nn))$ time[157]. Quantum superposition is a crucial notion for QML as this quantum property naturally endows the quantum algorithms with speedups. The importance of superposition to QML derives from the fact that ML is practically used to analyze vast amounts of data. However, the near-term applicable QNN models we so far discussed, although recognized the immediate advancements occurred when encoding input data in a superpositional state, have to adapt to the fact that the technology and resources required to prepare and manipulate such superpositions are not expected to be available quite soon.

The issue of encoding data in a superposition meets a significant research interest, as it constitutes a basic tool for quantum computations. Relevant literature focuses on the development of an efficient qRAM, although some other interesting yet mostly not investigated ideas can be found[158,159]. As expected, this issue is much more trivial in QQ case, where the algorithm's input is quantum and not classical and therefore no computational resources needed for data encoding. The CQ case is more elaborate as it requires a procedure that encodes the classical information into a quantum state. As demonstrated in Section 2, the computational cost of state preparation can be the key element in separating a QML algorithm with quantum speedup from one performing as efficiently as a classical one.

Although the concept of a qRAM dates back to 2008[160,161] and counts numerous references in QML literature[4,7,22,29,38,45], there is yet no practical implementation of a qRAM on a physical device in experimental settings. This is mainly due to the fact that the number of resources required to build a qRAM scales exponentially to the number of qubits needed, If fact, for the N n-dimensional input vectors we mentioned, qRAM requires $log(Nn)$ qubits. To make things more difficult, in case the QML algorithm is not fault-tolerant and requires error correction, which is the case especially with QML models that deploy Grover's algorithm, if not all then at least the majority of qRAMs components should incorporate error-correction

techniques. Moreover, in order for in *O(log(Nn))* memory recall time to be achieved, the qRAM requires a relatively uniform distribution of the data in the quantum register[157], which is not trivially obtained, otherwise the time escalates to $O(\sqrt{N})$.

Finally, if we take into account all the required resources, the dependence of a QML algorithm in qRAM may cost it its quantum speedup both in terms of computational as well as model complexity. Whether the quantum enhancements provided by superposition are diminished by the cost of preparing and manipulating this superposition is still an open question for the QML research community.

4.3 Prospects for Near-Term QML

In the search for the best approach to have near-term applicable QNN, or a QML model in general, the QML community currently speaks of an algorithm that can be realized by approximately 100 qubits, 1000 gates and has some robustness to noise[35]. These constrains are basically formed by the hardware difficulties QML is facing, discussed in the previous subsection. The quantum variational circuit-based models demonstrate a momentum towards this direction, mainly due to their ability to be hybrid trained, thus reduce the needed resources. In fact, the hybrid training offers the advantage of using the quantum device for short routines by outsourcing to a classical device and this also comes with the asset of easier storage and manipulation of the classical parameters used. This storage allows the reuse of the same, optimal parameters to predict several inputs. Taking into account that the development of a quantum memory like much desired QRAMs is yet to come and the no-cloning theory in quantum physics prohibits the replication of a state, this reusage won't be an option in case of a quantum-blas based algorithm which encodes the training parameters in a quantum state.

Another aspect QML strongly focuses on is the quantum advantage it can offer to ML. The majority of early literature follows the "translation" approach, trying to create quantum equivalents of classical ML algorithms, and offers a variety of mathematical proves of upper and lower bounds that could ensure exponential speedups in runtime compared to classical methods. These exponential speedups come with the price of strict constraints been imposed to the data and no practical applications to meet these constraints, as well as models of qubit and gate cost that are realistically out of reach.

However, the very nature of ML is practical, based on intractable problems that have been solved to a satisfactory level through approximations. Moreover, several breakthroughs and developments in ML are owed to huge and repeated mathematical experiments. It is not by accident that the main development of ML started as soon as the first packages developed in popular programming languages such as Python and Matlab, enabling faster and easier experiments. Furthermore, one should take into account that an ML model is not only measured based on complexity, but also its ability to generalize -an attribute that often stems from mathematical simplicity- its wide applicability or interpretability.

Under this spectrum, the focus on quantum advantage being interpreted as an exponential speedup offered to a classical ML algorithm needs to be revoked. The exploratory approach, where variational quantum circuit models are accommodated, follows this direction by aiming to create new models and dynamics. Instead of focusing on gaining an exponential speedup, it aims to expand ML strategies and potential offer a polylogarithmic speedup, which seems a much easier to be reached goal. This also enabled the QML research to move from theoretical proves to numerical examples and, in recent cases, quantum simulations and small-scale realizations to prove a model's potentials. It also moved the discussion from possible quantum speedup to consideration of noise manipulation or overfitting and strategies to overcome such problems in the QML.[41,43,58] Therefore, even a quadratic speedup is a speedup and if we consider cases when the simulation of a quantum model cannot be done classically, this can be considered an exponential speedup on its own. A, not less acceptable and useful, QNN model can be settled with quantum speedups that are common, potential or even limited if it seeks an early application.

One last thing we should consider in the search for near-term quantum applications is the famous Big Data problem. Quantum properties such as entanglement have been proposed to be the holy-grail for big data manipulation and storage frugality[4]. However, the manipulation of high dimensional date seems to be a goal for the distant future and as discussed in sample complexity, not much quantum advantage can be expected from that either, as the train data required in the quantum case are, in general, polylogarithmically less at best. State preparation routines are the bottleneck of every QML algorithm and therefore, without unlimited resources of qubits and gates or a Quantum Random Access Memory of respectable capacity, the nearby QML models have to settle with small data. Hybrid-training routines offer partial relief by processing a subset of samples at a time and if approximate data encoding strategies are used or quantum data replace classical ones, the application of such models moves even closer.

Although the difficulties of dealing with big data may be seen tremendous, one should consider that the same stands for traditional ML, and the latter has established its domain few decades earlier, preceding its quantum twin. Also, there exists an immense variety of problems where data collection is expensive or impossible and seeks for a solution build for extremely small datasets. Several QNN models have proved their ability to be efficiently trained with much less data than an ANN[31], and the less the train data the less the cost of a quantum model. Thus, these problems may be the quest for the first efficient applications of a QNN.

Finally, concerning near-term applications of QML, this thesis has only briefly mentioned the QC approaches, quantum routines and algorithms employed to assist and not replace traditional ML. Some very promising results started to occur recently[49,157] and although this form of quantum algorithms doesn't comply with the definition of QML, at least the one adopted for the purposes of this thesis, it offers some potential that deserves better consideration and investigation or at the very least to be mentioned. First of all, algorithms proposed as quantum assistance in ML processes are the only ones that genuinely and undoubtfully can demonstrate quantum advantage. This derives from the fact that they consider cases of ML where classical routines are intractable and try to resolve such issues.

Moreover, this approach is the first one that considers a vise-versa approach concerning data manipulation; instead of quantumly encode classical data, which has been proven very expensive in terms of complexity, or trying to focus on quantum data which occur to very limited problems, they explore the idea of classical data with quantum like correlations to be benefited from a quantum model. There is another great advantage of quantum assisted ML; the quantum resources needed are much limited compared to a quantum model for ML tasks, even hybrid trained. Therefore, as a conclusion, although this idea has only very currently gained a compact and severe researching interest, the author of this thesis embraces the possible contributions to ML field arising.

4.4 Open Questions

In the one-year process of collection, studying, analyzing and evaluating approaches and methods in QML for this thesis I came across dozens of open questions, blurry spots, misconceptions, poorly established proposals as well as extremely theoretical ideas. Of course, this is unsurprising for a domain that, as mentioned in the introductory section, struggles to obtain a holistic and solid definition. It is also justified by the fact that research interest, guided by the overwhelming potentials offered by quantum computing theory, precedes the hardware to test and support any QML model proposed. And although in some cases mathematical and numerical evidence are enough to draw conclusions about a QML models' advantages and disadvantages, in general terms possible speedups or the ability to tackle NP-hard problems unsolvable by the available classical hardware, the efficiency of such methods, the actual cost to obtain the output of an algorithm after repeated executions and measurements or the model

complexity will remain unanswered, at least till the available quantum hardware reach several milestones.

During this study, I kept all the open questions in a separate file. Questions that the authors of a paper declared as open for their own work as well as my own unanswered questions. Starting from the very early contributions in QML and moving towards today one can realize with relief that lots of the primitive misconceptions and blurry spots in the domain can now be considered either answered, outdated, updated or at least less poorly defined. Especially in the last 5 years the literature took a much more scientific than romantically exploratory turn and QML even obtained its third book this year, to assist in organization, evaluation and association of previously unstructured knowledge in the domain. Several questions though remained and will remain for a reasonable period of time. Therefore, I deem citing the remaining of my "question-list" as the most appropriate closure to this section.

1)Given a variational quantum circuit QNN, is there any strategy or even basic rules to follow to decide its architecture? In other terms can any good assumption be made about the ansatz of such a circuit given a specific problem? And if so, can this ansatz obey in the restrictions imposed by hardware's available resources while deploy the fullest of its given potentials?
2)Is there a way to train QML models not given as mathematical equations, but as physical quantum algorithms?
3)Can QML models realistically benefit from quantum routines such as classical linear combination of unitaries to overcome numerical optimisation?
4) Are there any good parametrisation strategies for variational QML models and variational quantum circuits in particular? Given the fact that parametrisation is highly associated with the objective function and therefore the complexity in training methods, this is a crucial question to answer.
5) Given any train method of a parametrised quantum model, is there a strategy for optimal initialization heuristics to allow for more consistent convergence of its local optimizers?
6) Is it necessary, and if so, is it possible, to deploy tricks from classical ML's iterative training processes, like momentum or adaptive learning rates, to improve a quantum model's performance?
7) Regarding model's complexity, can we detect a class of quantum models that exhibit superiority at capturing patterns and correlations in data?

8)Is there a particular class of ML problems that would be better served by a QML model, due to matching more to its quantum nature?

9) Which are the actual speedups and impacts of QML proposed algorithms to practical real-life problems?

10) Is there a precise definition of what a QNN is?

11) Can the power of feature maps be exploited to introduce nonlinearities in a QNN.

12) Can the proof of learnability of variational quantum circuit QNNs be extended in functions that are not Boolean?

13) Is there any way to determine the quantum sample complexity for learning concepts whose range exceeds the binary case?

14) How to introduce efficient, quantum appropriate and without great price of complexity, means of regularisation to avoid the problem of overfitting usually witnessed in the QNN models proposed so far?

15) Can a classical computer which uses random bits, produce outcomes which follow the same probability distribution that a quantum circuit would give?

16) How the gapless formulation of the adiabatic theorem influences time complexity?

17) Is there an affordable way to utilise superposition in QML? Are there any other realistic prospects besides a qRAM?

18) Which classes of relevant states for machine learning with amplitude encoding can be prepared qubit-efficiently

4.5 Conclusions

At this final point, the author, will present its personal opinion as a conclusion, formed by one year's research in the vast literature of this field. To begin with, this review of all the proposed approaches on building a QML algorithm and by extension a QNN confirms the choice of Variational Quantum Circuits in a Hybrid training scheme as the best applicants for an implementation in the near future. This is because this framework affords to be modified enough to meet the restrictions discussed in section 4.4 and still be able to learn at some degree from the train data, without depending on the existence of a qRAM as the models that follow the ising-type approach with adiabatic training methods or Quantum Ensamble methods and Quantum Random Walks do. The Hybrid scheme is ideal because it also abstains from superposition effect which is hard both to create and maintain, contrary to the training methods based on search discussed in 2.3.2. It also keeps the computational complexity and the required

resources to a minimum by outsourcing the update of the parameters to a classical machine, decreasing the overall complexity; which is not the case for training methods based on linear algebra presented in 2.3.1, as the HHL algorithm and the quantum routines required depend on the implementation of highly complex quantum subroutines and circuits of a depth of thousand gates. Finally, the choice of classical parameters offers the extra advantage of easy reproducibility, as they can be saved and manipulated by classical hardware such as RAM and CPU or GPU and therefore this part of the algorithm suffers from neither the challenge of decoherence nor the lack of error-correction, which is not a choice offered by adiabatic methods discussed in 2.3.3. However, another potential direction found in the recent literature, though particularly disregarded the previous years and is still in a very primitive stage, seems to be the direction of kernel methods in 2.2.2. The immediate and intuitive relation between information encoding, quantum kernels and inner products in a feature Hilbert space has the immediate advantage of exploiting the pre-existent cost of state preparation routines to gain the representation potentials of its classical counterpart. Thereafter the author believes that this is an interesting direction which worth to be furtherly investigated.

On the other hand, in 4.2 it has been made clear that there do still exist severe challenges imposed by the lack of hardware to support the implementation of such QNN models and the practical experimentation needed to form some firm and global conclusions about their performance, possible advantages and mostly, their ability to learn. The scatter and very limited simulations provided by the authors for the models presented in section 3 can by no means considered conclusive, due to the fact that the simulated QNNs focus on very specific individual cases which cannot be indicative for their general performance, and even in such cases the results reveal a limited robustness to noise and poor generalization power. This is justifiable, considering that there is still no knowledge concerning the optimal architecture of the quantum circuit or the initial parameters, so a comparison with an ANN' s best results would be unfair, given the extensive knowledge already obtained in the field of ML resulting to assortment of optimization methods developed for the construction of an ANN based on the task's and the dataset's demands. Furthermore, only in the last months the first implementations on actual quantum processors started to occur, as in the case presented in 3.1.1, and these implementations are of the scale of 1-5 qubits and therefore concern the implementation of a just a single quron, which is only one building block of a QNN. Finally, the greatest caveat for these models comes from the complexity imposed by state preparation techniques, as is underlined in 2.1.4, which is an inevitable step when it comes to the manipulation of a classical

dataset. And although the majority of the available data in our days is classical as well as the real-life problems seeking a solution from ML concern classical data, as analysed in 2.1.3, the best chances for the application of a QNN are met when quantum data are concerned instead.

Another conclusion that can be made is that the existing proposals for QNN do rather form a general framework that a specific QNN model. The lack of specific knowledge or even scatter indications about the proper construction of the quantum circuit' s architecture is the "elephant in the room" in case of Variational Quantum Circuits. However, this abstractness is met in general in QNN proposals of all approaches and should not be considered a disadvantage of this specific family of QNNs. There must be extensive research upon the subject and my personal belief is that the research community should focus on the development of more specific strategies and QNN models of one type rather than keep on investing in theoretical frameworks for possible QNNs. There are several issues that need to be investigated besides the architecture of the circuit and these are: the optimal initialization of parameters, development of more non-linear activation functions of the form presented in 3.1.2 based on the low-cost RUS circuits, techniques to avoid overfitting and develop robustness to errors, which types of classical datasets could benefit more from quantum processing, realistically efficient state preparation routines. Until the available knowledge becomes more concrete and the QNN framework less abstract, there will be no way to obtain a benchmark and conclusive comparison of QNN and ANN's potentials.

Machine Learning met his greatest advancements through practical and not theoretical research, which is expected for such a practical domain. In the same manner Quantum Machine Learning has a lot of prospects to develop and growth given the proper hardware to have easy access to repeatable experiments and quantum simulations deep enough that a classical computer cannot efficiently support. Although numerous software started to form in the recent years (Strawberry Fields, PennyLane, Atos, QuantumFlow, Qiskit) to enable quantum Machine Learning on classical computers, based either on CPU or on GPU processing, the creators of these software admit that Quantum Neural Networks cannot be efficiently simulated on a classical computer due to the very large number of elementary quantum gates that must be realized, repeatedly and on hundreds of qubits, leading to an intractable task with increasing network depth and width. In fact, in the site of Xanadu, a company dedicated to the research in QML and the development of software for QML, the creators of Strawberry Fields and PennyLane state *"While the simulator route is useful for quantum machine learning in the*

short term, it is fundamentally not scalable". Ultimately, direct evaluation of hardware will likely be necessary to large scale quantum neural networks, a task that is still in development[162].

It is not by accident that when a proposal about a Quantum Neural Network is provided with a classical simulation of the model, this comes without much, if not any, details from the authors and it always concerns a very specific case of the model. In the specific case of Variational Quantum Circuits discussed in this thesis, given a specific task on a specific dataset, the architecture of the simulated circuit that represents that Quantum Neural Network and parameter's initiation are never discussed. Even when some details about the model's design are mentioned, what led to the choice of these specific gates acting on those specific qubits is not addressed. The first real implementations of Quantum Neural Networks, which tentatively appeared in the last couple of years, are in fact implementations of quantum neurons that can potentially act as building blocks of Quantum Neural Networks and concern 16 qubits at best case. These facts are indicative of the primitive stage Quantum Neural Networks are on.

This imprimitive stage is very similar to the stage of Artificial Neural Networks at 80s and 90s when the classical hardware available was far beyond to support the computational complexity of these models. Therefore, in my opinion, the best of Quantum Neural Networks is just yet to come because history repeats itself. The potential speedups and revolution in the Machine Learning that Quantum Computing can offer are mathematically proved and intuitively straightforward, but the depth of these contributions can only be measured with regards to their cost -which depends on the available hardware- and will increase as the hardware challenges will decrease in time. Technological evolution in the 21st century repeatedly proved that the impossible is possible, thus in not-so-many years from now, with the existence of a qRAM, a quantum computer of some thousands of qubits and quantum error-correction improved, entangling quantum circuits that process in parallel a whole train set in superposition, or a committee of quantum perceptron will form Quantum Neural Networks with no classical equivalent. But until then the QML research community is forced in extremely small-scaled simulations, inefficient to experimentally prove quantum supremacy or to firmly answer open questions regarding optimization, error correction, QNN's architecture, generalization power, robustness to noise and overfitting.

5 References

1. Misra, J. & Saha, I. Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing* **74**, 239–255 (2010).
2. Mead, C. Neuromorphic Electronic Systems. *Proc. IEEE* (1990). doi:10.1109/5.58356
3. Senekane, M. & Taele, B. M. Prediction of Solar Irradiation Using Quantum Support Vector Machine Learning Algorithm. *Smart Grid Renew. Energy* **07**, 293–301 (2016).
4. Rebentrost, P., Mohseni, M. & Lloyd, S. Quantum support vector machine for big data classification. **1**, 1–5 (2013).
5. Dernbach, S., Mohseni-Kabir, A., Pal, S., Towsley, D. & Gepner, M. Quantum Walk Inspired Neural Networks for Graph-Structured Data. (2018).
6. Schuld, M., Sinayskiy, I. & Petruccione, F. Quantum computing for pattern classification. *Lect. Notes Comput. Sci.* **8862**, 208–220 (2014).
7. Schuld, M., Sinayskiy, I. & Petruccione, F. The quest for a Quantum Neural Network. *Quantum Inf. Process.* **13**, 2567–2586 (2014).
8. Gupta, S. & Zia, R. K. P. Quantum Neural Networks. *J. Comput. Syst. Sci.* **63**, 355–383 (2001).
9. Narayanan, A. & Menneer, T. Quantum artificial neural network architectures and components. *Inf. Sci. (Ny).* **128**, 231–255 (2000).
10. Kak, S. C. Quantum Neural Computing. *Adv. Imaging Electron Phys.* **94**, 259–313 (1995).
11. Menneer, T. & Narayanan, A. Quantum-inspired Neural Networks. (1995).
12. Chrisley, R. Quantum learning. *New directions in cognitive science: Proceedings of* (1995).
13. Ezhov, A. A. & Ventura, D. . Quantum Neural Networks. *Futur. Dir. Intell. Syst. Inf. Sci. Futur. Speech Image Technol. Brain Comput. WWW, Bioinforma.* 213 (2000).
14. Perus, M. Neural networks as a basis for quantum associative networks. *Neural Netw. World* **10**, 1001–1013 (2000).
15. Behrman, E. C., Nash, L. R., Steck, J. E., Chandrashekar, V. G. & Skinner, S. R. Simulations of quantum neural networks. *Inf. Sci. (Ny).* **128**, 257–269 (2000).

16. Trugenberger, C. A. Quantum Pattern Recognition. *Quantum Inf. Process.* **1**, 471–493 (2002).
17. Trugenberger, C. A. Probabilistic Quantum Memories. *Phys. Rev. Lett.* (2001). doi:10.1103/PhysRevLett.87.067901
18. Altaisky, M. V. Quantum neural network. 1–4 (2001). doi:10.1006/jcss.2001.1769
19. Cao, H., Cao, F. & Wang, D. Quantum artificial neural networks with applications q. *Inf. Sci. (Ny).* **290**, 1–6 (2015).
20. Ricks, B. & Ventura, D. Training a Quantum Neural Network. *Adv. Neural Inf. Process. Syst.* 1019–1026 (2004).
21. Zhou, R., Qin, L. & Jiang, N. Quantum perceptron network. in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2006). doi:10.1007/11840817_68
22. Da Silva, A. J., De Oliveira, W. R. & Ludermir, T. B. Classical and superposed learning for quantum weightless neural networks. *Neurocomputing* **75**, 52–60 (2012).
23. Sagheer, A. & Metwally, N. Communication via quantum neural networks. *Proc. - 2010 2nd World Congr. Nat. Biol. Inspired Comput. NaBIC 2010* 418–422 (2010). doi:10.1109/NABIC.2010.5716339
24. Panella, M. & Martinelli, G. Neural networks with quantum architecture and quantum learning. *Int. J. Circuit Theory Appl.* (2011). doi:10.1002/cta.619
25. Sagheer, A. & Zidan, M. Autonomous Quantum Perceptron Neural Network. (2013).
26. Liu, C. Y., Chen, C., Chang, C. Ter & Shih, L. M. Single-hidden-layer feed-forward quantum neural network based on Grover learning. *Neural Networks* **45**, 144–150 (2013).
27. Wittek, P. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. *Quantum Machine Learning: What Quantum Computing Means to Data Mining* (2014). doi:10.1016/C2013-0-19170-2
28. Altaisky, M. V., Kaputkina, N. E. & Krylov, V. A. Quantum neural networks: Current status and prospects for development. *Phys. Part. Nucl.* **45**, 1013–1032 (2014).
29. Wiebe, N., Kapoor, A. & Svore, K. M. Quantum Deep Learning. **2**, 1–34 (2014).
30. Zidan, M., Sagheer, A. & Metwally, N. An autonomous competitive learning algorithm using quantum hamming neural networks. *Proc. Int. Jt. Conf. Neural Networks* **2015-Septe**, (2015).
31. Seow, K.-L., Behrman, E. & Steck, J. Efficient learning algorithm for quantum perceptron unitary weights. 1–10 (2015).

32. Adachi, S. & Henderson, M. Application of Quantum Annealing to Training of Deep Neural Networks. (2015).
33. Biamonte, J. *et al.* Quantum machine learning. *Nature* **549**, 195–202 (2017).
34. Dunjko, V. & Briegel, H. J. Machine learning & artificial intelligence in the quantum domain: A review of recent progress. *Reports Prog. Phys.* **81**, (2018).
35. Schuld, M. & Petruccione, F. *Supervised Learning with Quantum Computers*. (2018). doi:10.1007/978-3-319-96424-9
36. Chen, J., Wang, L. & Charbon, E. A quantum-implementable neural network model. *Quantum Inf. Process.* **16**, 1–24 (2017).
37. Wan, K. H., Dahlsten, O., Kristjánsson, H., Gardner, R. & Kim, M. S. Quantum generalisation of feedforward neural networks. *npj Quantum Inf.* 1–7 (2016). doi:10.1038/s41534-017-0032-4
38. Cao, Y., Guerreschi, G. G. & Aspuru-Guzik, A. Quantum Neuron: an elementary building block for machine learning on quantum computers. 1–30 (2017).
39. Hu, W. Towards a Real Quantum Neuron. *Nat. Sci.* **10**, 99–109 (2018).
40. McClean, J. R., Romero, J., Babbush, R. & Aspuru-Guzik, A. The theory of variational hybrid quantum-classical algorithms. *New J. Phys.* (2016). doi:10.1088/1367-2630/18/2/023023
41. Farhi, E. & Neven, H. Classification with Quantum Neural Networks on Near Term Processors. 1–21 (2018).
42. Mitarai, K., Negoro, M., Kitagawa, M. & Fujii, K. Quantum circuit learning. *Phys. Rev. A* **98**, 1–3 (2018).
43. Schuld, M., Bocharov, A., Svore, K. & Wiebe, N. Circuit-centric quantum classifiers. 1–17 (2018).
44. Lloyd, S. & Weedbrook, C. Quantum Generative Adversarial Learning. *Phys. Rev. Lett.* **121**, 1–5 (2018).
45. Rebentrost, P., Bromley, T. R., Weedbrook, C. & Lloyd, S. Quantum Hopfield neural network. *Phys. Rev. A* **98**, 1–13 (2018).
46. Schuld, M. & Killoran, N. Quantum Machine Learning in Feature Hilbert Spaces. *Phys. Rev. Lett.* **122**, (2019).
47. Killoran, N. *et al.* Continuous-variable quantum neural networks. 1–21 (2018).
48. Dendukuri, A. *et al.* Defining Quantum Neural Networks via Quantum Time Evolution. 1–9 (2019).
49. Perdomo-Ortiz, A., Benedetti, M., Realpe-Gómez, J. & Biswas, R. Opportunities and

challenges for quantum-assisted machine learning in near-term quantum computers. *Quantum Sci. Technol.* **3**, 1–13 (2018).

50. Tacchino, F., Macchiavello, C., Gerace, D. & Bajoni, D. An artificial neuron implemented on an actual quantum processor. *npj Quantum Inf.* **5**, 1–8 (2019).
51. Verdon, G. *et al.* Learning to learn with quantum neural networks via classical neural networks. 1–12 (2019).
52. Guerreschi, G. G. & Smelyanskiy, M. Practical optimization for hybrid quantum-classical algorithms. 1–25 (2017).
53. Kerenidis, I., Landman, J., Luongo, A. & Prakash, A. q-means: A quantum algorithm for unsupervised machine learning. (2018).
54. Lloyd, S., Mohseni, M. & Rebentrost, P. Quantum principal component analysis. *Nat. Phys.* (2014). doi:10.1038/NPHYS3029
55. Otterbach, J. S. *et al.* Unsupervised Machine Learning on a Hybrid Quantum Computer. (2017).
56. Liu, N. & Rebentrost, P. Quantum machine learning for quantum anomaly detection. *Phys. Rev. A* **97**, 1–11 (2018).
57. Liang, J. M., Shen, S. Q., Li, M. & Li, L. Quantum anomaly detection with density estimation and multivariate Gaussian distribution. *Phys. Rev. A* **99**, 1–7 (2019).
58. Lamata, L. Basic protocols in quantum reinforcement learning with superconducting circuits. *Sci. Rep.* **7**, (2017).
59. Albarrán-Arriagada, F., Retamal, J. C., Solano, E. & Lamata, L. Measurement-based adaptation protocol with quantum reinforcement learning. *Phys. Rev. A* **98**, (2018).
60. Crawford, D., Levit, A., Ghadermarzy, N., Oberoi, J. S. & Ronagh, P. Reinforcement learning using quantum boltzmann machines. *Quantum Inf. Comput.* (2018).
61. Dong, D., Chen, C., Li, H. & Tarn, T. J. Quantum reinforcement learning. *IEEE Trans. Syst. Man, Cybern. Part B Cybern.* **38**, 1207–1220 (2008).
62. Dunjko, V., Taylor, J. M. & Briegel, H. J. Advances in quantum reinforcement learning. in *2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017* (2017). doi:10.1109/SMC.2017.8122616
63. Cárdenas-López, F. A., Lamata, L., Retamal, J. C. & Solano, E. Multiqubit and multilevel quantum reinforcement learning with quantum technologies. *PLoS One* (2018). doi:10.1371/journal.pone.0200455
64. Aïmeur, E., Brassard, G. & Gambs, S. Machine learning in a quantum world. in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence*

and Lecture Notes in Bioinformatics) (2006). doi:10.1007/11766247_37

65. Stoudenmire, E. M. & Schwab, D. J. Supervised learning with tensor networks. *Adv. Neural Inf. Process. Syst.* 4806–4814 (2016).
66. Huggins, W., Patil, P., Mitchell, B., Whaley, K. B. & Stoudenmire, E. M. Towards quantum machine learning with tensor networks. *Quantum Sci. Technol.* (2019). doi:10.1088/2058-9565/aaea94
67. Carrasquilla, J. & Melko, R. G. Machine learning phases of matter. *Nat. Phys.* (2017). doi:10.1038/nphys4035
68. Sentís, G., Bagan, E., Calsamiglia, J. & Muñoz-Tapia, R. Multicopy programmable discrimination of general qubit states. *Phys. Rev. A - At. Mol. Opt. Phys.* (2010). doi:10.1103/PhysRevA.82.042312
69. Qiu, P.-H., Chen, X.-G. & Shi, Y.-W. Detecting Entanglement With Deep Quantum Neural Networks. *IEEE Access* (2019). doi:10.1109/access.2019.2929084
70. Plesch, M. & Brukner, Č. Quantum-state preparation with universal gate decompositions. *Phys. Rev. A - At. Mol. Opt. Phys.* (2011). doi:10.1103/PhysRevA.83.032302
71. Lloyd, S. Universal Quantum Simulators: Correction. *Science (80-.).* (1998). doi:10.1126/science.279.5354.1113h
72. Berry, D. W., Childs, A. M. & Kothari, R. Hamiltonian Simulation with Nearly Optimal Dependence on all Parameters. in *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS* (2015). doi:10.1109/FOCS.2015.54
73. Cleve, R., Ekert, A., Macchiavello, C. & Mosca, M. Quantum algorithms revisited. *Proc. R. Soc. A Math. Phys. Eng. Sci.* (1998). doi:10.1098/rspa.1998.0164
74. Schuld, M., Sinayskiy, I. & Petruccione, F. Simulating a perceptron on a quantum computer. *Phys. Lett. Sect. A Gen. At. Solid State Phys.* **379**, (2015).
75. Schölkopf, B., Herbrich, R. & Smola, A. J. A generalized representer theorem. in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2001). doi:10.1007/3-540-44581-1_27
76. Havlíček, V. *et al.* Supervised learning with quantum-enhanced feature spaces. *Nature* (2019). doi:10.1038/s41586-019-0980-2
77. Schuld, M. & Killoran, N. Quantum Machine Learning in Feature Hilbert Spaces. *Phys. Rev. Lett.* (2019). doi:10.1103/PhysRevLett.122.040504
78. Low, G. H., Yoder, T. J. & Chuang, I. L. Quantum inference on Bayesian networks.

Phys. Rev. A - At. Mol. Opt. Phys. **89**, (2014).

79. O'Gorman, B., Babbush, R., Perdomo-Ortiz, A., Aspuru-Guzik, A. & Smelyanskiy, V. Bayesian network structure learning using quantum annealing. *European Physical Journal: Special Topics* (2015). doi:10.1140/epjst/e2015-02349-9
80. Amin, M. H., Andriyash, E., Rolfe, J., Kulchytskyy, B. & Melko, R. Quantum Boltzmann Machine. *Phys. Rev. X* **8**, (2018).
81. Wiebe, N., Kapoor, A. & Svore, K. M. Quantum deep learning. *Quantum Inf. Comput.* (2016).
82. Harrow, A. W., Hassidim, A. & Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* (2009). doi:10.1103/PhysRevLett.103.150502
83. Lloyd, S., Mohseni, M. & Rebentrost, P. Quantum principal component analysis. *Nat. Phys.* **10**, 631–633 (2014).
84. Wiebe, N., Braun, D. & Lloyd, S. Quantum algorithm for data fitting. *Phys. Rev. Lett.* (2012). doi:10.1103/PhysRevLett.109.050505
85. Schuld, M., Sinayskiy, I. & Petruccione, F. Prediction by linear regression on a quantum computer. *Phys. Rev. A* (2016). doi:10.1103/PhysRevA.94.022342
86. Zhao, Z., Fitzsimons, J. K. & Fitzsimons, J. F. Quantum-assisted Gaussian process regression. *Phys. Rev. A* (2019). doi:10.1103/PhysRevA.99.052331
87. Grover, L. K. A fast quantum mechanical algorithm for database search. in *Proceedings of the Annual ACM Symposium on Theory of Computing* (1996). doi:10.1145/237814.237866
88. Dürr, C. & Hoyer, P. A Quantum Algorithm for Finding the Minimum. *CoRR* **quant-ph/9**, (1996).
89. Wiebe, N., Kapoor, A. & Svore, K. M. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *Quantum Inf. Comput.* **15**, 318–358 (2015).
90. Ventura, D. & Martinez, T. Quantum associative memory with exponential capacity. in *IEEE International Conference on Neural Networks - Conference Proceedings* (1998). doi:10.1109/ijcnn.1998.682319
91. Wiebe, N., Kapoor, A. & Svore, K. M. Quantum perceptron models. in *Advances in Neural Information Processing Systems* (2016).
92. Peruzzo, A. *et al.* A variational eigenvalue solver on a photonic quantum processor. *Nat. Commun.* (2014). doi:10.1038/ncomms5213
93. Farhi, E., Goldstone, J. & Gutmann, S. A Quantum Approximate Optimization

Algorithm. 1–16 (2014).

94. Seddiqi, H. & Humble, T. S. Adiabatic quantum optimization for associative memory recall. *Front. Phys.* (2014). doi:10.3389/fphy.2014.00079
95. Denil, M. & De Freitas, N. Toward the Implementation of a Quantum RBM. *NIPS Deep Learn. Unsupervised Featur. …* (2011).
96. Benedetti, M., Realpe-Gómez, J., Biswas, R. & Perdomo-Ortiz, A. Quantum-assisted learning of hardware-embedded probabilistic graphical models. *Phys. Rev. X* (2017). doi:10.1103/PhysRevX.7.041052
97. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities (associative memory/parallel processing/categorization/content-addressable memory/fail-soft devices). *Biophysics (Oxf).* (1982).
98. Hinton, G. Boltzmann Machines. in *Encyclopedia of Machine Learning and Data Mining* (2017). doi:10.1007/978-1-4899-7687-1_31
99. Sutskever, I., Hinton, G. & Taylor, G. The recurrent temporal restricted boltzmann machine. in *Advances in Neural Information Processing Systems 21 - Proceedings of the 2008 Conference* (2009).
100. Amin, M. H., Andriyash, E., Rolfe, J., Kulchytskyy, B. & Melko, R. Quantum Boltzmann Machine. *Phys. Rev. X* (2018). doi:10.1103/PhysRevX.8.021050
101. Nonomura, Y. & Nishimori, H. Quantum Hopfield Model. 1–9 (1995).
102. Rotondo, P., Marcuzzi, M., Garrahan, J. P., Lesanovsky, I. & Müller, M. Open quantum generalisation of Hopfield neural networks. *J. Phys. A Math. Theor.* (2018). doi:10.1088/1751-8121/aaabcb
103. Breuer, H. P. & Petruccione, F. *The Theory of Open Quantum Systems*. *The Theory of Open Quantum Systems* (2007). doi:10.1093/acprof:oso/9780199213900.001.0001
104. Rebentrost, P., Bromley, T. R., Weedbrook, C. & Lloyd, S. Quantum Hopfield neural network. *Phys. Rev. A* (2018). doi:10.1103/PhysRevA.98.042308
105. Venegas-Andraca, S. E. Quantum walks: A comprehensive review. *Quantum Information Processing* (2012). doi:10.1007/s11128-012-0432-5
106. Szegedy, M. Quantum speed-up of Markov Chain based algorithms. in *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS* (2004). doi:10.1109/focs.2004.53
107. Loke, T., Tang, J. W., Rodriguez, J., Small, M. & Wang, J. B. Comparing classical and quantum PageRanks. *Quantum Inf. Process.* (2017). doi:10.1007/s11128-016-1456-z
108. Paparo, G. D. & Martin-Delgado, M. A. Google in a quantum network. *Sci. Rep.* (2012).

doi:10.1038/srep00444

109. Schuld, M. & Petruccione, F. Quantum ensembles of quantum classifiers. *Sci. Rep.* (2018). doi:10.1038/s41598-018-20403-3
110. Hinton, G. E. & Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks (Supporting Online Material). *Science (80-.).* (2006). doi:10.1126/science.1127647
111. Zheng, Y., Lu, S. & Wu, R.-B. Quantum Circuit Design for Training Perceptron Models. 1–12 (2018).
112. Paetznick, A. & Svore, K. M. Repeat-until-success: Non-deterministic decomposition of single-qubit unitaries. *Quantum Inf. Comput.* (2014).
113. Nelder, J. A. & Mead, R. A Simplex Method for Function Minimization. *Comput. J.* (1965). doi:10.1093/comjnl/7.4.308
114. Barenco, A. *et al.* Elementary gates for quantum computation. *Phys. Rev. A* (1995). doi:10.1103/PhysRevA.52.3457
115. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks* (1991). doi:10.1016/0893-6080(91)90009-T
116. Haykin, S. S. *Neural networks and learning machines, 3rd Edition*. (2009). doi:10987654321
117. Verdon, G., Broughton, M. & Biamonte, J. A quantum algorithm to train neural networks using low-depth circuits. (2017).
118. Romero, J. *et al.* Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz. *Quantum Sci. Technol.* **4**, 1–18 (2019).
119. Lloyd, S. & Braunstein, S. L. Quantum computation over continuous variables. *Phys. Rev. Lett.* **82**, 1784–1787 (1999).
120. Lau, H. K., Pooser, R., Siopsis, G. & Weedbrook, C. Quantum Machine Learning over Infinite Dimensions. *Phys. Rev. Lett.* (2017). doi:10.1103/PhysRevLett.118.080501
121. Das, S., Siopsis, G. & Weedbrook, C. Continuous-variable quantum Gaussian process regression and quantum singular value decomposition of nonsparse low-rank matrices. *Phys. Rev. A* (2018). doi:10.1103/PhysRevA.97.022315
122. Lloyd, S. & Slotine, J. J. E. Analog quantum error correction. *Phys. Rev. Lett.* (1998). doi:10.1103/PhysRevLett.80.4088
123. Braunstein, S. L. Error correction for continuous quantum variables. *Phys. Rev. Lett.* **80**, 4084–4087 (1998).
124. Dias, J. & Ralph, T. C. Quantum error correction of continuous-variable states with

realistic resources. *Phys. Rev. A* (2018). doi:10.1103/PhysRevA.97.032335

125. Andersen, U. L., Neergaard-Nielsen, J. S., Van Loock, P. & Furusawa, A. Hybrid discrete- and continuous-variable quantum information. *Nat. Phys.* (2015). doi:10.1038/nphys3410
126. Ventura, D. & Martinez, T. Initializing the amplitude distribution of a quantum state. *Found. Phys. Lett.* **12**, 547–559 (1999).
127. Rønnow, T. F. *et al.* Defining and detecting quantum speedup. *Science (80-.).* (2014). doi:10.1126/science.1252319
128. Rieffel, E. G. Quantum Supremacy Using a Programmable Superconducting Processor. *NASA* (2019).
129. Shor, P. W. Algorithms for quantum computation: discrete logarithms and factoring. in (2002). doi:10.1109/sfcs.1994.365700
130. Li, R. Y., Di Felice, R., Rohs, R. & Lidar, D. A. Quantum annealing versus classical machine learning applied to a simplified computational biology problem. *npj Quantum Inf.* (2018). doi:10.1038/s41534-018-0060-8
131. Rebentrost, P., Bromley, T. R., Weedbrook, C. & Lloyd, S. Quantum Hopfield neural network. *Phys. Rev. A* **98**, (2018).
132. Aïmeur, E., Brassard, G. & Gambs, S. Quantum speed-up for unsupervised learning. *Mach. Learn.* (2013). doi:10.1007/s10994-012-5316-5
133. Briegel, H. J. Machine learning & artificial intelligence in the quantum domain.
134. Hastie, T., Tibshirani, R. & Friedman, J. The Elements of Statistical Learning. *Elements* **1**, 337–387 (2009).
135. Valiant, L. G. A theory of the learnable. in *Proceedings of the Annual ACM Symposium on Theory of Computing* (1984). doi:10.1145/800057.808710
136. Bshouty, N. H. & Jackson, J. C. Learning DNF over the uniform distribution using a quantum example oracle. in *Proceedings of the 8th Annual Conference on Computational Learning Theory, COLT 1995* (1995). doi:10.1137/s0097539795293123
137. Servedio, R. A. & Gortler, S. J. Equivalences and separations between quantum and classical learnability. *SIAM J. Comput.* (2004). doi:10.1137/S0097539704412910
138. Atici, A. & Servedio, R. A. Improved bounds on quantum learning algorithms. *Quantum Inf. Process.* (2005). doi:10.1007/s11128-005-0001-2
139. Arunachalam, S. & De Wolf, R. Optimal quantum sample complexity of learning algorithms. *J. Mach. Learn. Res.* (2018). doi:10.4230/LIPIcs.CCC.2017.25
140. Bernstein, E. & Vazirani, U. Quantum complexity theory. *SIAM J. Comput.* (1997).

doi:10.1137/S0097539796300921

141. Deutsch, D. & Jozsa, R. Rapid Solution of Problems by Quantum Computation. *Proc. R. Soc. A Math. Phys. Eng. Sci.* (1992). doi:10.1098/rspa.1992.0167
142. Kothari, R. An optimal quantum algorithm for the oracle identification problem. in *Leibniz International Proceedings in Informatics, LIPIcs* (2014). doi:10.4230/LIPIcs.STACS.2014.482
143. Ambainis, A. *et al.* Quantum identification of boolean oracles. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)* (2004). doi:10.1007/11683858_1
144. Hunziker, M., Meyer, D. A., Park, J., Pommersheim, J. & Rothstein, M. The geometry of quantum learning. *Quantum Inf. Process.* (2010). doi:10.1007/s11128-009-0129-6
145. Ristè, D. *et al.* Demonstration of quantum advantage in machine learning. *npj Quantum Inf.* **3**, 1–12 (2017).
146. Theodoridis, S. & Koutroumbas, K. *Pattern Recognition*. *Wiley Interdisciplinary Reviews Computational Statistics* **5748**, (2009).
147. Vapnik, V. N. An overview of statistical learning theory. *IEEE Transactions on Neural Networks* (1999). doi:10.1109/72.788640
148. Sain, S. R. & Vapnik, V. N. The Nature of Statistical Learning Theory. *Technometrics* (1996). doi:10.2307/1271324
149. Lu, S. C., Zheng, Y., Wang, X. T. & Wu, R. B. *Book #10 - Quantum Machine Learning by Peter Wittek. Kongzhi Lilun Yu Yingyong/Control Theory and Applications* **34**, (2017).
150. Nielsen, M. a. & Chuang, I. L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. *Cambridge University Press* (2011). doi:10.1017/CBO9780511976667
151. Steane, A. M. Overhead and noise threshold of fault-tolerant quantum error correction. *Phys. Rev. A - At. Mol. Opt. Phys.* (2003). doi:10.1103/PhysRevA.68.042322
152. Tang, E. Quantum-inspired classical algorithms for principal component analysis and supervised clustering. 1–5 (2018).
153. Gilyén, A., Lloyd, S. & Tang, E. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension. 1–10 (2018).
154. Preskill, J. Quantum Computing in the NISQ era and beyond. *Quantum* (2018). doi:10.22331/q-2018-08-06-79
155. Rancic, M., Hedges, M. P., Ahlefeldt, R. L. & Sellars, M. J. Coherence time of over a

second in a telecom-compatible quantum memory storage material. *Nat. Phys.* (2018). doi:10.1038/NPHYS4254

156. Temme, K., Bravyi, S. & Gambetta, J. M. Error Mitigation for Short-Depth Quantum Circuits. *Phys. Rev. Lett.* (2017). doi:10.1103/PhysRevLett.119.180509
157. Ciliberto, C. *et al.* Quantum machine learning: A classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **474**, (2018).
158. Grover, L. & Rudolph, T. Creating superpositions that correspond to efficiently integrable probability distributions. 1–2 (2002).
159. Park, D. K., Petruccione, F. & Rhee, J. K. K. Circuit-Based Quantum Random Access Memory for Classical Data. *Sci. Rep.* (2019). doi:10.1038/s41598-019-40439-3
160. Giovannetti, V., Lloyd, S. & MacCone, L. Architectures for a quantum random access memory. *Phys. Rev. A - At. Mol. Opt. Phys.* (2008). doi:10.1103/PhysRevA.78.052310
161. Giovannetti, V., Lloyd, S. & Maccone, L. Quantum Random Access Memory. *Phys. Rev. Lett.* (2008).
162. Schuld, M., Bergholm, V., Gogolin, C., Izaac, J. & Killoran, N. Evaluating analytic gradients on quantum hardware. *Phys. Rev. A* (2019). doi:10.1103/PhysRevA.99.032331
163. Trugenberger, C. A. Probabilistic Quantum Memories. *Phys. Rev. Lett.* **87**, 67901-1-67901–4 (2001).
164. Shen, C., Zhang, Z. & Duan, L. M. Scalable implementation of boson sampling with trapped ions. *Phys. Rev. Lett.* (2014). doi:10.1103/PhysRevLett.112.050504

6 Appendices

Appendix A

State Preparation Routines

Basis Encoding- Create Superposition of Inputs

Ventura and Martinez[90,126] as well as Trugenberger[163], in separate works, proposed the following quantum routine to prepare a superposition of the input data, basis encoded, as the initial state for a QML algorithm. Consider a dataset of *N*-dimensional data and size *M*. The key operator during this process is S^p, a set of operators that form a set of conditional Hadamard-like transforms. Considering the simple case of 2 qubits required to encode an input data, then:

$$S^p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{\frac{p-1}{p}} & \frac{-1}{\sqrt{p}} \\ 0 & 0 & \frac{-1}{\sqrt{p}} & \sqrt{\frac{p-1}{p}} \end{bmatrix} (A.1)$$

The parameter *p* is different for each one of the training inputs, so for the considered dataset *MN* different parameters will be needed and accordingly *M* different operators of the form (A.1). Another important operator is F^0 which in two-qubit case is also defined as:

$$F^o = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (A.2)$$

F^0 flips the state of the second qubit if the state of the first qubit is $|0\rangle$. In the same manner, an operator F^1 which flips the second qubit if the first qubit is in state $|1\rangle$ is:

$$F^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} (A.3)$$

Obviously (A.2) and (A.3) are two C-NOT gates conditioned to the first qubit. Then, a FLIP operator used to change the basis states corresponding to the different input patterns is constructed as

$$A^{00} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} F & 0 \\ 0 & I_6 \end{bmatrix} (A.4)$$

This FLIP operator flips the state of the third qubit, conditioned to the state of the first two qubits. In the exact similar manner, another three flip operators are constructed for all possible states of the first two qubits, respectively A^{01}, A^{10}, A^{11}. A combination of these operators is used to identify the data inputs in superposition and along with F^1 create the superposition. Now the construction of such operators can be accordingly expanded to the case of N qubits.

The routine requires three quantum registers; a loading register *X* which consists of *N* qubits are always set to $|0..0\rangle$ after each iteration, the *C* register which is an ancilla register of 2 qubits which are the control qubits and indicate the status of each state and finally the *G* register, a storage register of N qubits which is used to hold the superposition. Then an outline of the algorithm is the following:

Consider the system of three registers to be in the state $|X_1..X_N;a_1,a_2;G_1,\ldots,G_N\rangle$

1)Prepare the system in an initial ground state $|0..0\rangle$.

2)Use FLIP gates to flip selective qubits in the *X* register, so that the state in the *X* register corresponds to the state of the first input data in basis encoding.

3)When the state in superposition in *X* register that represents the input data is prepared, it is separated in two branches, a memory one and a processing one and the first one is saved in the G register. That can be considered as:

$$\frac{1}{\sqrt{2}}|0,0...0;0,0;0,...,0\rangle + \frac{1}{\sqrt{2}}|0,0...0;0,1,0,..,0\rangle, (A.5)$$

,

where the term with $a_2=0$ is the memory branch and the left flagged with $a_2=1$ the processing branch. The superpositional state is broken in two unequal pieces and in each iteration the status a_i of the smallest piece is made permanent in the G register.

4)The *X* register with the larger piece is selectively flipped again so that the state corresponding to the second input data is prepared and steps 2-3 repeat for every input data. Note that the C-NOT gates (A.2)-(A-4) conditioned to the second statuses' qubit $a_2=1$ limits the executions to the processing branch.

Iteratively the memory branch stores the first *m* inputs in its storage register, while the storage register of the processing branch is in the ground state. In both branches the loading register is also in the ground state. In the *m-th* step the quantum system's state can thereafter be described by:

$$\frac{1}{\sqrt{M}}\sum_{i=1}^{N}|0,0...0;0,0;x_1^i,...,x_N^i\rangle+\frac{1}{\sqrt{2}}|0,0...0;0,1;0,...,0\rangle,(A.6)$$

Explicitly, in the *(m+1)*-th step of the algorithm the state that corresponds to the *m+1* input data $|x_1^{m+1},...,x_N^{m+1}\rangle$ is prepared in the *X* register and a (A.2)-(A-4) conditioned to the second statuses' qubit $a_2=1$ is applied to copy it to the storage register so that:

$$\frac{1}{\sqrt{M}}\sum_{i=1}^{m+1}|x_1^{m+1},...,x_N^{m+1};0,0;x_1^i,...,x_N^i\rangle+\sqrt{\frac{M-m}{M}}|x_1^{m+1},...,x_N^{m+1};0,1;x_1^{m+1},...,x_N^{m+1}\rangle,(A.7)$$

A CNOT gate is used to flip the first ancilla to $|1\rangle$ if the second ancilla is also set to $|1\rangle$, which stands always only for the processing branch as we have set $a_2=0$ permanently to the *C* register of the memory branch.

The process branch is splitted in the two pieces by applying the key operator S^p, where $p=N+1-(m+1)$ to a_2 controlled by a_1, and this leads to:

$$\begin{aligned}&\frac{1}{\sqrt{M}}\sum_{i=1}^{m}|x_1^{m+1},...,x_N^{m+1};00;x_1^i,...,x_N^i\rangle+\\&\frac{1}{\sqrt{M}}x_1^{m+1},...,x_N^{m+1};10;x_1^{m+1},...,x_N^{m+1}\rangle+\\&\frac{\sqrt{M-(m+1)}}{\sqrt{M}}|x_1^{m+1},...,x_N^{m+1};11;x_1^{m+1},...,x_N^{m+1}\rangle,(A.8)\end{aligned}$$

The branch marked by $|a_1a_2\rangle=|10\rangle$ is the one to be added in the memory branch. This routine will succeed after *O(MN)* steps in total.

Appendix B

Phase Estimation Routine

This quantum routine is used to write information encoded in the phase of an amplitude into a basis state via a reverse quantum Fourier transform, which maps a quantum state with an amplitude vector *x* to a state whose amplitude vector is the Fourier transformation of *x*. Mathematically speaking for an *n*-qubit state a quantum Fourier transform is:

$$\sum_{j=1}^{2n} x_j \mid j\rangle \rightarrow \sum_{j=1}^{2n} \frac{1}{\sqrt{2^n}} \left[\sum_{k=1}^{2n} e^{\frac{2\pi i j k}{2^n}} x_k \right] \mid j\rangle, (B.1)$$

Given an amplitude α which carries the required information in its phase φ, so that $a = \mid a \mid e^{i\varphi}$ or in general terms given a unitary $U=e^{2i\pi\varphi}$ applied to an-qubit state $|\psi\rangle$ so that $U|\psi\rangle = e^{2i\pi\varphi} |\psi\rangle$, then $|\psi\rangle$ is the eigenvector and $e^{2i\pi\varphi}$ are the corresponding eigenvalues, which are all normalized since *U* is unitary.

A quantum circuit of two *n*-qubit registers is used for this routine. In one register, the state $|\psi\rangle$ is prepared and the other register, called the ancilla register, is prepared in the ground state.

Step 1: The initial state of the circuit is prepared to be:

$$|\psi_0\rangle = \mid 0...0\rangle \mid \psi\rangle, (B.2)$$

Step 2: An *n*-bit Hadamard gate operation is applied on the ancilla register to prepare the superposition:

$$|\psi_1\rangle = \frac{1}{2^{\frac{n}{2}}} (\mid 0\rangle + \mid 1\rangle)^{\otimes n} \mid \psi\rangle, (B.3)$$

Step 3: Construct a control unitary operator $C\text{-}U^{2j}$ that applies the unitary operator *U* on the target register only the *j-th* qubit is in state $|1\rangle$.

$$U^{2^j} = U^{2^{j-1}} U = U^{2^{j-1}} e^{2i\pi\varphi} = e^{2i\pi 2^j \varphi}, (B.4)$$

Step 4: Apply all the *n-1* operations derived from (B.4) to state $|\psi_1\rangle$ the resulting state of the circuit is:

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{k=0}^{2^n-1} e^{2i\pi\varphi k} \mid k\rangle \mid \psi\rangle, (B.5)$$

Step 5: Apply an inverse Fourier Transform to the ancilla register to obtain:

$$|\psi_3\rangle = \frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{k=0}^{2^n-1} e^{\frac{2\pi ik}{2^n}(x-2^n\varphi)}\,|x\rangle\,|\psi\rangle, (B.6)$$

Step 6: (B.6) picks for $x=2^n\varphi$ and in case $2^n\varphi$ is an integer the final state in the ancilla becomes

$$|\psi_4\rangle = |2^n\varphi\rangle\,|\psi\rangle (B.7)$$

Appendix C

Variational Eigensolver Classifiers

Variational eigensolvers were the first scheme of variational algorithms proposed. The algorithm is based on the variational principle of quantum mechanics that the ground state $|\psi\rangle$ minimizes the expectation value

$$\frac{\langle\psi|\mathrm{H}|\psi\rangle}{\langle\psi|\psi\rangle}, (C.1)$$

where *H* is the Hamiltonian of the system.

Therefore, given a parametrised quantum system with a parameter set θ, then the cost function for a variational system can be defined as the energy expectation value of the system:

$$C(\theta) = \langle\psi(\theta)|\mathrm{H}|\psi(\theta)\rangle, (C.2)$$

for normalized kets, and the variational eigensolver algorithm aims to find the ground states - that is, lowest energy eigenstates- of quantum system, thus minimize the cost function.

Initially, $C(\theta)$ is quantumly estimated for an initial parameter set by repeated measurements on the state $|\psi(\theta)\rangle$ to obtain the estimation of the expectation value of *H*. Then, if parameters θ are classical real-valued parameters, classical training can be utilized to lower the energy of the system by minimising the cost function in a scheme referred as hybrid.

This routine is used in Variational Classifiers for supervised learning. If the expectation value of an observable *C* is interpreted as the output of a classifier for an input *x*, it results to:

$$f(x;\theta) = \langle\psi(x;\theta)|C|\psi(x;\theta)\rangle, (C.3)$$

(C.3) describes a Variational Classifier with a set of parameters θ.

To associate an expectation value with a binary model output $f(x;\theta)$, a Pauli Z operator can be used on a pre-selected qubit i, which is called the observable, to measure its state. This way, the expectation value can be transformed the probability of the *i-th* qubit to be in state $|1\rangle$ or $|0\rangle$ thus, the binary output of the classifier is obtained. That is simply written as:

$$f(x;\theta) = \langle \psi(x;\theta) \mid Z_i \mid \psi(x;\theta) \rangle, (C.4)$$

In this framework, any cost function can be selected for optimization. For example, given target values y_i for the inputs x_i, $i=1,2,\ldots,N$, the standard squared loss cost function is:

$$C(\theta) = \sum_{i=1}^{N} (\langle \psi(x_i;\theta) \mid Z_i \mid \psi(x_i;\theta) \rangle - y_i)^2, (C.5)$$

After the quantum computation of observable's expectation value, the cost function can be trained by any traditional training scheme.

Appendix D

Classical Linear Combination of Unitaries

Given a variational circuit U_θ of n qubits and L depth, with classical real-values $\theta=\{\theta_1,\theta_2,....,\theta_w\}$ as parameters of the L, where L is $\Omega(4^n)$ single-qubit or two-qubit gates that compose the circuit. Therefore, U_θ can be decomposed to elementary unitaries as;

$$U_\theta = U_L....U_l....U_1$$

For a single qubit gate G_k, we have

$$U_k = I_o \otimes I_1 \otimes .... \otimes G_k \otimes .... \otimes I_{n-1} \text{ and}$$

$$G_k = e^{i\varphi}\begin{pmatrix} e^{i\beta}\cos a & e^{i\gamma}\sin a \\ -e^{-i\gamma}\sin a & e^{i\beta}\cos a \end{pmatrix}$$

So every single qubit gate G_k can be parametrized by a set of 4 parameters $\{\varphi,\alpha,\beta,\gamma\}$.

The two-qubit single control gates are also requested to be imprimitive, meaning they map a two-qubit product state to a one-qubit non product state. Imprimitive gates can be described by the 4 parameters of the single-qubit gate they incorporate for that. Thus, we need at least 4^n parameters for the circuit. However, since the overall phase factor cannot physically be measured, φ is neglected and only the 3 out of 4 parameters are considered learnable.

In case the above assumptions about the circuit are met, the derivative of the circuit $\partial\theta$ can be computed by a procedure named by its inventors[43] as *"Classical Linear Combination of Unitaries"*.

First, consider $\partial_\mu U_i = I \otimes .... \otimes \partial_\mu G(a,\beta,\gamma) \otimes ... \otimes I, \mu = \alpha,\beta,\gamma$.

The respective derivatives of G are easily computed as:

$$\partial_a G = G(a+\frac{\pi}{2},\beta,\gamma)$$

$$\partial_\beta G = \frac{1}{2}G(a,\beta+\frac{\pi}{2},0)+\frac{1}{2}G(a,\beta+\frac{\pi}{2},\pi), (D.1)$$

$$\partial_\gamma G = \frac{1}{2}G(a,0,\gamma+\frac{\pi}{2})+\frac{1}{2}G(a,\pi,\gamma+\frac{\pi}{2})$$

So, in case of differentiation for elementary operators of single-qubit gates with respect to the three learnable parameters we have to;

a) For parameter α: Implement the same gate, alternated by shifting parameter a by $\frac{\pi}{2}$

b) For parameter β: Implement a linear combination of two single-qubit gates where parameter β is shifted by $\frac{\pi}{2}$ and γ is replaced with constants 0 and π

c) For parameter γ: Implement a linear combination of two single-qubit gates where parameter γ is shifted by $\frac{\pi}{2}$ and β is replaced with constants 0 and π

In case of imprimitive two-qubit gates the derivation is not such straightforward. However, if we made an extra agreement and limit to single-control single-qubit gates of the form *C(G)*, via:

$C_a(G_b)|x\rangle|y\rangle = |x\rangle \otimes G^x |y\rangle$, where

G_b: A single-qubit gate acting on qubit *b* which is everything but a global phase factor other than identity

$|x\rangle$: The state of qubit a which is a pure state

Then we obtain:

$$\partial_\mu C(G) = \frac{1}{2}(C(\partial_\mu G) - C(-\partial_\mu G)), \mu = \alpha, \beta, \gamma$$

.

This elegantly leads the computation of the derivative to compute the difference between a controlled derivative gate and the negative version of that same gate, which can be easily computed by the derivative rules of single-qubit gates for each parameter. In case of parameter α, the controlled gates are unitary and in case $\mu=\beta$ or $\mu=\gamma$ is a linear combination of unitaries.

Now, consider a threshold decision function $\pi(x;\mu,b) = p(q_0 = 1, x, \mu) + b$, of the circuit's output state $|\varphi(x)\rangle$, *x* be the input and φ the feature map used for the input encoding to the initial state. The derivative of this function is:

$$\partial_\theta \pi(x^m;\theta) = \partial_\theta p(q_0 = 1; x^m;\theta) =$$

$$\partial_\theta \sum_{k=2^{n-1}+1}^{2^n} (U_\theta \varphi(x))^\dagger{}_\kappa (U_\theta \varphi(x))_k =$$

$$2\,\mathrm{Re}\{\sum_{k=2^{n-1}+1}^{2^n} (\partial_\theta U_\theta \varphi(x))^\dagger{}_\kappa (U_\theta \varphi(x))_k\}$$

And this can be analyzed to:

$$\partial_\theta \pi(x;\mu) = \sum_{j=1}^{J} a_j \,\mathrm{Re}\{\langle U_{\theta[j]}\varphi(x)\,|\,Z\,|\,U_{\theta[j]}\varphi(x)\rangle\}, \quad (D.2)$$

where $j=\alpha,\beta,\gamma,\varphi$ and a_j a corresponding coefficient stemming from the derivation of single qubit gates given by (D.1).

This last derivative can be quantumly estimated by repeated measurements. The ability to quantum estimate (D.2) is proved by the authors and stems from the following observation:

"*Given two unitary quantum circuits A and B that act on an n-qubit register to prepare the two quantum states |A⟩, |B⟩, and which can be applied conditioned on the state of an ancilla qubit, we can use the quantum device to sample from the probability distribution* $p = \frac{1}{2} + \frac{1}{2}\mathrm{Re}\langle A\,|\,B\rangle$ "

Appendix E

Continuous Variable Computing

Continuous-Variable (CV) quantum computing makes use of physical observables[119], like the position and momentum of photons in an electromagnetic field or ions in an ion-trap, whose numerical values belong to countable infinite-dimensional Hilbert spaces, thus continuous intervals. CV quantum protocols can be physically realized either through techniques of quantum optics[125] or via modified ion-trap computers[164]. A basic notion is qumodes, referring to quantum states of bosonic modes which carry the information in a CV quantum system. It is the equivalent of qubits in a discrete quantum model. The terminology of a *mode* of free quantum field $\phi(x)$ refers to its quantum Fourier transform:

$$\varphi(x) = \int \frac{\partial^3 p}{(2\pi)^3} \frac{1}{\sqrt{2\omega_p}} (a(p)e^{ixp} + b(p)^\dagger e^{-ixp}, (E.1)$$

,

where ω stands for the frequency of the mode.

There is a distinction between fermionic and bosonic modes, based on whether the objects a and $a, a^\dagger$ as well as $b, b^\dagger$ commute or anticommute, since they are the creation and annihilation operators for the particle and antiparticle respectively, associated to the field. If they commute, the corresponding particle is a boson - you can pile up arbitrarily many particles into the same state just by applying the creation operator many times. Therefore, (E.1) describes a *bosonic mode*. If they anticommute, the corresponding particle is a fermion - applying the creation operator twice just gives zero, so you can't ever have more than one particle in the same state. In such case (E.1) corresponds to a *fermionic mode*. In a slight generalization, any collection of creation/annihilation operators can be called bosonic or fermionic modes according to their commutation relations, regardless of whether they arose from a quantum field or were just given in some other way.

Thus, qumodes can be seen as the wires of a quantum circuit and based on their dual wave-like and particle-like nature, phase space formulation or Hilbert-space formulation is used for information encoding. If the wave nature is considered, given two conjugate variables *x* and *p*, which can be the position and momentum or the quadrature amplitudes of the qumode, its state can be encoded in $(x, p) \in \mathbb{R}^2$ and be represented by a real-valued function *F(x,p)*.

Such a function *F* is called a *quasiprobability* distribution. Note that, although quasiprobability distributions share some properties of classical probability distributions, they are expanded to negative values and the normalization constraint is loosen so *F(x,p)* must have a unit integral over the space phase. If the equivalent Hilbert space formulation is used instead of the phase space one, qumode states are represented as vectors or density matrices in an infinite Hilbert space spanned by the eigenstates of the photon number operator N, named Fock states $\{|n\rangle\}_{i=0}^{\infty}$. We will focus on the former case.

The operators of a CV model can be divided in two classes; Gaussian and non-Gaussian. The Gaussian transformations map the set of Gaussian distributions in phase space to itself. The basic single-mode and two-mode gates for a real-valued mode φ are;

$$Rotation: R(\varphi): \begin{bmatrix} x \\ p \end{bmatrix} \rightarrow \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} x \\ p \end{bmatrix}, (E.2)$$

$$Displacement: D(a;\varphi): \begin{bmatrix} x \\ p \end{bmatrix} \rightarrow \begin{bmatrix} x + R(a) \\ p + \mathrm{Im}(a) \end{bmatrix} \begin{bmatrix} x \\ p \end{bmatrix}, a \in \mathbb{C}, (E.3)$$

$$Squeezing: S(r;\varphi): \begin{bmatrix} x \\ p \end{bmatrix} \rightarrow \begin{bmatrix} e^{-r} & 0 \\ 0 & e^{r} \end{bmatrix} \begin{bmatrix} x \\ p \end{bmatrix}, r \in \mathbb{R}, (E.4)$$

$$(phaseless)\ Beamsplitter: BS(\theta;\varphi) = \begin{bmatrix} x_1 \\ x_2 \\ p_1 \\ p_2 \end{bmatrix} \rightarrow \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & -\sin\theta \\ 0 & 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ p_1 \\ p_2 \end{bmatrix}, \theta \in [0, 2\pi], (E.5)$$

Gaussian operators are related to the set of affine transformations on phase space. This can be demonstrated by the fact that any Gaussian transformation applied in an *n* modes quantum system can be described by:

$$\begin{bmatrix} x \\ p \end{bmatrix} \rightarrow M \begin{bmatrix} x \\ p \end{bmatrix} + \begin{bmatrix} \mathrm{Re}(a) \\ \mathrm{Im}(a) \end{bmatrix}, (E.6)$$

,

given a symplectic matrix *M* and a complex vector *a*. Note that a symplectic matrix *M* carries the property

$$M^T \begin{bmatrix} 0 & \mathrm{I} \\ -\mathrm{I} & 0 \end{bmatrix}_{2n\times 2n} M = \begin{bmatrix} 0 & \mathrm{I} \\ -\mathrm{I} & 0 \end{bmatrix}_{2n\times 2n}, (\mathrm{E}.7)$$

and can be split into an Euler decomposition (a type of Singular Value Decomposition) of the form

$$M = K_2 \begin{bmatrix} \Sigma & 0 \\ 0 & \Sigma^{-1} \end{bmatrix} \mathrm{K}_1, (\mathrm{E}.8),$$

where Σ a diagonal positive valued matrix and K_1, K_2 are also symplectic and orthogonal real-valued matrices, described via

$$\mathrm{K} = \begin{bmatrix} C & D \\ -D & C \end{bmatrix}, where \begin{matrix} CD^T - DC^T = 0 \\ CC^T - DD^T = \mathrm{I} \end{matrix} (E.9)$$

The importance of Gaussian operators in the general form (E.6) stems from the isomorphism between the intersection of the symplectic and orthogonal groups on *2n* dimensions and the unitary group on *n* dimensions, which enables the transformations *K* to be performed via the unitary action of passive linear optical interferometers. Moreover, any Gaussian operator can be decomposed into a CV circuit containing only the basic gates (E.2)-(E.5), so one could define the Gaussian transformations as those quantum circuits which contain only the gates given above.

Two very special operators in a CV model derive as operator versions of the phase space variables *x* and *p* previously defined as information carriers. Each qumode in a CV model is associated with a pair of operators (X_i, P_i) where,

$$X_i = \int_{-\infty}^{\infty} x_i \,|\, x_i\rangle\langle x_i \,|\, dx_i, (E.10)$$
$$P_i = \int_{-\infty}^{\infty} p_i \,|\, p_i\rangle\langle p_i \,|\, dp_i, (E.11)$$
$$, where \begin{matrix} \langle x \,|\, x'\rangle = \delta(x - x') \\ \langle p \,|\, p'\rangle = \delta(p - p') \\ \langle p \,|\, x\rangle \sim e^{ipx} \end{matrix}$$

and $|x\rangle$ are orthogonal and not normalised. So, a *n*-modes quantum system can be described by *(X,P)* where *X* and *P* are vectors of operators. Note that the Heisenberg uncertainty principle applies for *x* and *p* being simultaneously measured. A special rotation gate $F:=R(\frac{\pi}{2})$ connects (E.10) and (E.11), so that $x=F(p)$. Moreover, *x* operator is the multiplier operator and *p* the derivative operator, so that:

$$X\varphi(x) = x\varphi(x)$$

$$P\varphi(x) = -i\frac{\partial}{\partial x}\varphi(x)$$

Finally, the CV model's set of gates becomes universal -able to approximate any transformation $U_{H(X,P)}=exp(-itH)$ through a quantum circuit of polynomial depth- if, along with the gates (E.2)-(E.5) a single-mode non-Gaussian operator -corresponding to a nonlinear function of the space variables- is added. The most commonly used non-Gaussian gates are:

Cubit Phase Gate: $V(\gamma) = e^{(i\frac{\gamma}{3}X^3)}, (E.12)$

Kerr Gate: $K(\kappa) = e^{(ikN^2)}, (E.13)$, where N is the number operator-the observable that counts the number of particles