

UNIVERSITÀ DEGLI STUDI MILANO BICOCCA

Facoltà di Scienze Matematiche, Fisiche e Naturali
CORSO DI LAUREA TRIENNALE FISICA



DIPARTIMENTO DI FISICA "GIUSEPPE OCCHIALINI"

**Classificazione di Dispositivi Quantistici tramite Reti
Neurali Ibride**

Candidato:
Federico Blanco
Matricola: 866552

Relatore:

Prof . Andrea Giachero

Correlatore:

Dr. Roberto Moretti

Sommario

La teoria della meccanica quantistica, sviluppatasi tra gli anni '20 e '30 del secolo scorso, ha rivoluzionato la fisica grazie alla sua capacità di descrivere accuratamente fenomeni su scala atomica e subatomica. Richard Feynman, premio Nobel per la Fisica nel 1965, fu tra i primi a intuire i limiti dei computer classici nella descrizione della natura a livello microscopico, auspicando la realizzazione di un computer quantistico basato sui principi della meccanica quantistica, come la sovrapposizione e l'entanglement. Ad oggi, il calcolo quantistico (o quantum computing) è un settore di studio affermato ed in continua espansione. L'unità fondamentale su cui si basano i computer quantistici è il qubit, un sistema che può esistere in due differenti stati, $|0\rangle$ e $|1\rangle$. Un computer quantistico, dunque, è un sistema costituito da più qubit fisicamente collegati tra loro.

La presente tesi si colloca nell'ambito del Quantum Machine Learning (QML), un settore emergente del Machine Learning (ML), nel quale algoritmi quantistici vengono impiegati e spesso combinati con quelli classici per sviluppare modelli di apprendimento automatico. Negli ultimi anni, il QML ha conosciuto uno sviluppo rapido, aprendo prospettive innovative per il futuro dell'Intelligenza Artificiale.

Tra i modelli più utilizzati nel contesto del machine learning, vi sono le reti neurali. Nella sua forma più semplice una rete neurale è una struttura composta da neuroni, unità elementari che elaborano un insieme di dati in input e forniscono un output scalare. Tali reti si compongono di strati (o layer) costituiti da più neuroni in parallelo, dove ogni neurone è collegato a ciascuno dei neuroni appartenenti al layer successivo tramite connessioni pesate. Questo studio si focalizza sull'implementazione di una rete neurale classica ed una rete neurale ibrida, ossia una rete che fa uso di strategie sia classiche che quantistiche per l'elaborazione del dato in input. La componente quantistica della rete ibrida consiste in una serie di istruzioni che manipolano lo stato quantistico di più qubit. Queste istruzioni possono essere parametrizzate sia dai dati in input ricevuti, ad esempio dall'output del layer precedente, sia da pesi addestrabili.

L'obiettivo principale dello studio è realizzare un confronto tra le prestazioni delle due reti nel risolvere un problema di classificazione binaria supervisionata utilizzando un dataset artificialmente generato, il quale è composto dall'output prodotto da diversi quantum computer, soggetti a diverse quantità di rumore a seconda della loro architettura. In particolare, ogni elemento del dataset consiste nel risultato ottenuto dall'implementazione dello stesso algoritmo sui qubit di diversi

dispositivi quantistici e misurando la probabilità (attraverso sequenze di misure ripetute) di ottenere determinati stati di qubit.

Lo studio effettuato, oltre a fornire un paragone tra reti neurali classiche ed ibride, si concentra sull'ottimizzazione della rete ibrida stessa. In particolare, si vuole valutare in che misura variabili come l'entanglement ed il numero di qubit presenti nello strato quantistico influiscano sulle metriche di valutazione del modello, quali accuratezza di classificazione, definita come la frazione di elementi del dataset correttamente classificati sul totale e la capacità delle reti di eseguire accurate previsioni anche su dati non utilizzati in fase di addestramento.

La rete classica esibisce un'accuratezza di classificazione del 94.00% mentre le reti ibride raggiungono un'accuratezza compresa tra il 95.00% e il 95.75%. Non è stata riscontrata una differenza significativa tra reti ibride con e senza entanglement. Tale comportamento potrebbe differire aumentando il numero di qubit utilizzati nella rete ibrida, oppure in problemi di classificazione più complessi.

Lo studio mostra come l'integrazione di layer quantistici a più qubit all'interno di una rete neurale possa esibire risultati, in termini di accuratezza di classificazione, confrontabili a quelli di una rete neurale classica. Con il progressivo miglioramento nelle prestazioni dei computer quantistici, sarà possibile testare algoritmi di QML più complessi e potenzialmente introdurre modelli ibridi per risolvere con maggiore efficacia i problemi tipici affrontati dal machine learning classico.

Indice

1	Quantum Computing	5
1.1	Introduzione al Quantum Computing	5
1.2	Qubits	5
1.2.1	Sfera di Bloch	6
1.2.2	Informazione di un qubit	7
1.3	Entanglement Quantistico	7
1.4	Sistemi di qubit multipli	8
1.5	Quantum Gate	9
1.5.1	Gate a singolo qubit	9
1.5.2	Gate a due qubit	11
1.6	Quantum Processing Unit (QPU)	12
1.6.1	Qubit superconduttivo	12
1.6.2	Anarmonicità: La Giunzione Josephson	16
1.6.3	Il Qubit Transmon	18
1.7	Errori di lettura del qubit	18
1.7.1	Rumore nei QPU superconduttiivi	18
1.7.2	Modellazione del rumore e della decoerenza	19
1.8	Quantum Provider	21
1.8.1	Classe 'fake_provider' di IBM	21
1.8.2	Caratteristiche di un provider	21
1.8.3	Transpile Circuit	22
1.8.4	Gate non appartenente ai Native Gate	22
1.8.5	CNOT applicata a due qubit non collegati fisicamente	23
2	Machine Learning	24
2.1	Introduzione al Machine Learning	24
2.2	Struttura di una rete neurale	25
2.2.1	Neurone	25
2.2.2	Percettrone e reti multilayer	27
2.3	Addestramento supervisionato	28
2.4	Error Backpropagation Algorithm	29
2.4.1	Binary Cross Entropy (BCE)	31
2.5	Minimo Locale e Minimo Globale	31

2.6	Performance di una rete neurale	32
2.6.1	Processo di validazione	32
2.6.2	Overfitting e Underfitting	33
2.7	Implementazione di una rete neurale ibrida	34
2.7.1	circuito quantistico variazionale	34
3	Classificazione di dispositivi quantistici	37
3.1	Generazione del dataset	37
3.1.1	Provider quantistici	38
3.1.2	Circuito quantistico	39
3.1.3	Normalizzazione	41
3.2	Descrizione delle reti neurali	42
3.2.1	Rete neurale classica	43
3.2.2	Rete neurale ibrida	44
4	Conclusioni	51
A	Evoluzione dello stato del sistema quantistico	53
B	Programmi per analisi dati	55
B.1	Dataset	55
B.2	Addestramento e validazione delle reti neurali	56

Capitolo 1

Quantum Computing

1.1 Introduzione al Quantum Computing

Il quantum computing utilizza tecnologie specializzate, tra cui hardware e algoritmi che sfruttano la meccanica quantistica, per risolvere problemi complessi che i computer o i supercomputer classici non possono risolvere abbastanza rapidamente per finalità pratiche[5].

Il compito esemplare per i computer quantistici che ha fornito la principale motivazione per il loro sviluppo è l'algoritmo quantistico di Shor per fattorizzare grandi numeri[17]. I migliori algoritmi per la fattorizzazione di grandi numeri infatti, hanno una complessità computazionale quasi esponenziale, mentre applicato ad un computer che sfrutta le proprietà di calcolo quantistiche, la complessità computazionale ha un andamento polinomiale. Questo significa che il tempo di esecuzione cresce molto più lentamente rispetto all'approccio classico, per questo motivo, è uno tra diversi algoritmi quantistici che permettono a computer quantistici di superare i più grandi supercomputer classici nel risolvere alcuni problemi specifici importanti per la crittografia dei dati[12].

1.2 Qubits

Un qubit, o bit quantistico, è l'unità di informazione utilizzata per codificare i dati nel quantum computing e può essere inteso come l'equivalente quantistico del bit tradizionale utilizzato dai computer classici per codificare le informazioni in formato binario. Il termine "qubit" è attribuito al fisico teorico americano Benjamin Schumacher[6].

I qubit sono l'unità fondamentale di informazione nella computazione quantistica. Mentre i bit classici possono essere 0 o 1, i qubit possono esistere in una sovrapposizione di stati, che è una combinazione lineare di $|0\rangle$ e $|1\rangle$. Questa proprietà consente ai qubit di rappresentare molte più informazioni rispetto ai bit classici.

Un qubit può essere rappresentato matematicamente come una combinazione lineare di stati base appartenenti ad uno spazio di Hilbert di dimensione 2.

Lo stato generale $|\psi\rangle$ che descrive un qubit può essere dunque scritto come:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

dove α e β sono due numeri complessi tali che:

$$|\alpha|^2 + |\beta|^2 = 1 \quad (1.1)$$

$|\alpha|^2$ e $|\beta|^2$ rappresentano rispettivamente le probabilità di misurare il qubit nello stato $|0\rangle$ o $|1\rangle$.

1.2.1 Sfera di Bloch

Per comprendere meglio il qubit, è utile considerare la seguente rappresentazione geometrica:

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle \quad (1.2)$$

In cui θ e ϕ definiscono un punto sulla sfera unitaria, anche chiamata **sfera di Bloch**, mostrata in Fig.1.1. Da notare che la (1.1) è verificata nella definizione di (1.2).

Nonostante un qubit sia descritto da due numeri complessi α e β , i gradi di libertà non sono quattro, bensì due. Questo è dovuto al vincolo introdotto sui coefficienti stessi nella relazione (1.1) e all'imposizione di una fase globale che rende reale l'ampiezza di probabilità associata a $|0\rangle$.

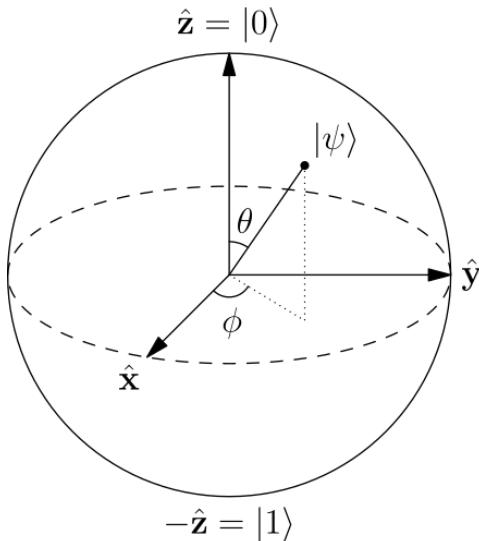


Figura 1.1: stato di un qubit rappresentato sulla sfera di Bloch.

1.2.2 Informazione di un qubit

Definito matematicamente un qubit, è ora importante comprendere quanta informazione può contenere e cosa succede quando viene effettuata una misura su di esso. Contrariamente ai bit classici, che contengono un'informazione minima, esiste un numero infinito di punti sulla sfera unitaria, quindi in linea di principio si potrebbero memorizzare infinite informazioni. Tuttavia, questa conclusione risulta essere fuorviante, a causa del comportamento di un qubit quando viene osservato.

Dopo una misurazione infatti, il qubit avrà come risultato "0" o "1", il suo stato inoltre collasserà sullo stato di base coerente al risultato della misurazione. Per stimare i valori dei coefficienti è infatti necessario ripetere la misura un numero elevato di volte.

A questo punto viene naturale chiedersi dove effettivamente differisce un qubit da un bit classico, dato che abbiamo appena affermato che dopo una misura lo stato di un qubit collassa dando un risultato binario. Per rispondere a questo quesito, dobbiamo considerare che prima di effettuare una misura, durante l'evoluzione di un sistema, nello stato del qubit è nascosta una grande quantità di informazione contenuta nelle ampiezze α e β , dovuta al fatto che il qubit si può trovare in una sovrapposizione di stati. Il bit classico d'altra parte, può avere solo valore "0" o "1" in ogni istante[14].

1.3 Entanglement Quantistico

L'*entanglement* è stato descritto per la prima volta da Einstein, Podolsky, Rosen e Schrödinger come uno strano fenomeno della meccanica quantistica, mettendo in discussione la completezza della teoria. Successivamente, Bell ha riconosciuto che l'*entanglement* porta a deviazioni sperimentalmente testabili della meccanica quantistica dalla fisica classica[3].

Ma che cos'è l'*entanglement*? Che ruolo ha nell'ambito dell'informazione quantistica? L'*entanglement* quantistico è un fenomeno in cui due qubit (o due o più particelle quantistiche) si intrecciano in modo tale che lo stato di una particella non può essere descritto indipendentemente dallo stato dell'altra, qualunque sia la distanza tra di loro. Il risultato della misura di uno dei due qubit condiziona anche il risultato dell'altro[6].

Infine, con l'avvento della teoria dell'informazione quantistica, l'*entanglement* è stato riconosciuto come una risorsa, abilitando compiti come la crittografia quantistica o il teletrasporto quantistico. Insieme al rapido progresso sperimentale nel controllo quantistico, ciò ha portato a un crescente interesse nella teoria dell'*entanglement*[3].

Per capire meglio il ruolo dell'*entanglement* è necessario comprendere sistemi composti da due o più qubit.

1.4 Sistemi di qubit multipli

Quando si considerano più qubit, il sistema quantistico può essere rappresentato come un prodotto tensoriale degli stati dei singoli qubit.

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \dots \otimes |\psi_n\rangle$$

Ad esempio, per due qubit, gli stati possibili del sistema possono essere:
 $|00\rangle, |01\rangle, |10\rangle, |11\rangle$

Possiamo dunque scrivere lo stato del sistema come sovrapposizione di questi stati base:

$$|\Psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle \quad (1.3)$$

I coefficienti della combinazione sono le ampiezze di probabilità degli stati e così come per il singolo qubit, sono numeri complessi il cui modulo quadro rappresenta la probabilità di misurare il sistema nel rispettivo stato base. Vale la relazione:

$$\sum_i |\alpha_i|^2 = 1 \quad (1.4)$$

La misura di un singolo qubit fa collassare lo stato del sistema in una sovrapposizione di stati base in cui lo stato del qubit misurato è coerente al risultato della misura.

Se ad esempio misuriamo il primo qubit nello stato "0" lo stato collasserà da $|\Psi\rangle \rightarrow |\Psi'\rangle$ con:

$$|\Psi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}$$

Uno stato importante a due qubit è lo "*stato di Bell*" o coppia EPR (*Einstein-Podolsky-Rosen*)

$$|\psi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad (1.5)$$

La principale proprietà di questo stato, consiste nella correlazione tra il primo qubit e il secondo, e dunque della presenza di entanglement. Dall'esito della misura del primo qubit infatti, si ottiene anche l'esito del secondo.

Con una coppia di qubit si possono costruire quattro stati di Bell:

$$|\psi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}} \quad (1.6)$$

$$|\phi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \quad (1.7)$$

$$|\phi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}} \quad (1.8)$$

Gli stati di Bell risultano fondamentali per diverse ragioni nell’ambito della computazione quantistica, ad esempio, sono utilizzati nel protocollo di teletrasporto quantistico, che permette di trasferire l’informazione quantistica da un qubit a un altro senza il trasferimento fisico del qubit stesso. Questo è realizzato creando uno stato di Bell tra due qubit e utilizzando misurazioni e comunicazioni classiche per trasferire lo stato[14].

1.5 Quantum Gate

I *Quantum Gate* o porte quantistiche, sono l’analogo delle porte logiche usate nella computazione classica. Il loro scopo è quello di manipolare l’informazione quantistica di uno o più qubit.

Un *Quantum Gate* agisce come un operatore sullo stato di un qubit o un sistema di essi appartenenti ad uno spazio di Hilbert di dimensione finita. Sono rispettate le proprietà di unitarietà e linearità.

Una porta quantistica viene solitamente rappresentata come una matrice quadrata $n \times n$ dove n è la dimensione dello spazio di Hilbert contenente lo stato del sistema.

Inoltre è possibile rappresentare gli stati $|0\rangle$ e $|1\rangle$ nel seguente modo:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

1.5.1 Gate a singolo qubit

Un gate che agisce su un singolo qubit viene rappresentato come una matrice quadrata di dimensione 2. Per la proprietà di unitarietà si verifica che per i seguenti gate è valida la relazione

$$U^\dagger U = I$$

Di seguito verranno brevemente esposti i gate più comuni.

X-Pauli Gate o Quantum Not Gate

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Questo gate è l'analogo quantistico del *not-gate* classico. Trasforma lo stato $|0\rangle$ nello stato $|1\rangle$ e viceversa.

$$X|\psi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$$

Y-Pauli Gate

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

La porta Y introduce una rotazione complessa con fase immaginaria nei coefficienti del qubit. Agisce su un generico stato nella seguente modalità:

$$Y|\psi\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} -i\beta \\ i\alpha \end{pmatrix}$$

Z-Pauli Gate

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

La porta Z introduce un'inversione della fase del coefficiente dello stato $|1\rangle$. Agisce su un generico stato nella seguente modalità:

$$Z|\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ -\beta \end{pmatrix}$$

Hadamard Gate (H)

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

La porta H risulta essere un gate molto importante, in quanto introduce una sovrapposizione tra gli stati di base $|0\rangle$ e $|1\rangle$. Nello specifico, l'azione su generico stato risulta essere:

$$H|\psi\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix}$$

La porta H ha l'ulteriore proprietà di essere una matrice autoaggiunta, ossia verifica la condizione

$$H = H^\dagger$$

dunque applicare di seguito due porte H su uno stato non modificherà quest'ultimo, in quanto

$$H^\dagger H = H^2 = I$$

Gate di Rotazione

In verità, i *Gate di Pauli* sono casi particolari di altri tre gate più generici, ossia:

$$R_x(\theta) = \begin{pmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix}$$

$$R_y(\theta) = \begin{pmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \exp(-i \frac{\theta}{2}) & 0 \\ 0 & \exp(i \frac{\theta}{2}) \end{pmatrix}$$

Questi gate rappresentano una rotazione rispetto agli assi x,y,z di un angolo θ . Si può verificare che valgono le seguenti relazioni:

$$X = -iR_x(\pi)$$

$$Y = -iR_y(\pi)$$

$$Z = -iR_z(\pi)$$

Una fase globale tuttavia, non influenza in alcun modo i risultati di una misurazione su un qubit, dunque non è errato affermare che l'azione di ognuno dei *Gate di Pauli* corrisponda ad una rotazione di 180 attorno al rispettivo asse. Più in generale, qualsiasi matrice $2x2$ unitaria si può scomporre come prodotto di tre rotazioni moltiplicata per una fase

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta) \quad (1.9)$$

1.5.2 Gate a due qubit

I gate a due qubit sono rappresentabili come matrici $4x4$ e mantengono le stesse proprietà dei gate a un qubit.

C-NOT Gate

la *controlled-not gate* è la principale porta quantistica a due qubit, corrisponde all'analogo della porta XOR classica.

La matrice associata è la seguente:

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

La sua azione consiste nell'invertire lo stato del secondo qubit (target) nel caso in cui il primo (controllo) sia nello stato $|1\rangle$

$$|00\rangle \rightarrow |00\rangle \quad |01\rangle \rightarrow |01\rangle$$

$$|10\rangle \rightarrow |11\rangle \quad |11\rangle \rightarrow |10\rangle$$

1.6 Quantum Processing Unit (QPU)

I processori quantistici o *Quantum Processing Unit (QPU)* sfruttano le proprietà intrinseche dei sistemi meccanici quantistici, come la sovrapposizione di stati ed entanglement, per risolvere certi problemi in modo efficiente.

Negli ultimi due decenni, ci sono stati rapidi sviluppi nell'ingegneria dei sistemi quantistici e del calcolo quantistico. Si è passati dal regno delle esplorazioni scientifiche su singoli sistemi quantistici isolati verso la creazione e manipolazione di processori multi-qubit. In particolare, i requisiti imposti da QPU più grandi hanno spostato l'attenzione all'interno della comunità scientifica dalla sola scoperta allo sviluppo di nuove e fondamentali astrazioni ingegneristiche associate alla progettazione, controllo, e lettura di sistemi quantistici multi-qubit.

Il risultato è la nascita di una nuova disciplina, chiamata ingegneria quantistica, che si pone l'obiettivo di collegare le scienze di base, la matematica, e l'informatica con campi generalmente associati all'ingegneria tradizionale.

1.6.1 Qubit superconduttivo

Una piattaforma prominente per la costruzione di un processore quantistico multi-qubit coinvolge i *qubit superconduttori*.

In questa tipologia di qubit l'informazione è immagazzinata nei gradi di libertà quantistici di oscillatori anarmonici nanofabbricati (circuiti superconduttori). A differenza di altre piattaforme come ad esempio sistemi di ioni intrappolati o atomi ultrafreddi dove l'informazione è codificata in sistemi quantistici microscopici naturali, i qubit superconduttori sono macroscopici e definiti litograficamente, ossia un processo di microfabbbricazione che consente di costruire circuiti elettrici complessi su scala nanometrica o micrometrica. Questo significa che i qubit superconduttori sono progettati in laboratorio, costruiti tramite processi tecnici avanzati e sono controllabili con maggiore precisione rispetto ai sistemi naturali come gli atomi o gli ioni.

Una caratteristica notevole dei qubit superconduttori è che i loro spettri di livello energetico dipendono dai parametri degli elementi del circuito e quindi sono configurabili. Vengono comunemente definiti *atomi artificiali* proprio per la caratteristica di poter essere progettati in modo tale da esibire spettri energetici simili a quelli degli atomi.

Esistono diversi qubit semiconduttori, uno dei più comunemente utilizzati è il *qubit transmon*, di cui tratteremo brevemente gli aspetti principali nelle sezioni seguenti.

Circuito LC Risonante come Oscillatore Armonico Quantistico

Per comprendere la dinamica del circuito di un qubit superconduttivo, è utile affrontare dapprima una trattazione classica di un circuito LC risonante lineare, come quello schematizzato in Fig1.2

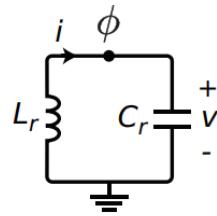


Figura 1.2: Circuito LC

In questo sistema l'energia oscilla tra l'energia elettrica del condensatore (C) e l'energia magnetica dell'induttore (L). L'energia del sistema dipendente dal tempo è data dalla somma infinitesima dei contributi della tensione ai capi del condensatore ($V(t)$) e della corrente nell'induttore ($I(t)$).

$$E(t) = \int_{-\infty}^t V(\tau)I(\tau)d\tau \quad (1.10)$$

Il circuito viene rappresentato da una delle sue variabili generalizzate, carica o flusso.

Di seguito verrà scelto il flusso, associando l'energia elettrica all'energia cinetica (\mathcal{T}) del sistema e l'energia magnetica a quella potenziale (\mathcal{U}). Il flusso è definito come l'integrale nel tempo della tensione

$$\Phi(t) = \int_{-\infty}^t V(\tau)d\tau \quad (1.11)$$

Combinando 1.10 con 1.11, utilizzando le seguenti relazioni

$$V = L \frac{dI}{dt}$$

$$I = C \frac{dV}{dt}$$

Ed integrando per parti, è possibile scrivere \mathcal{T} e \mathcal{U} in funzione del flusso Φ .

$$\mathcal{T}_C = \frac{1}{2} C \dot{\Phi}^2 \quad (1.12)$$

$$\mathcal{U}_L = \frac{1}{2L} \Phi^2 \quad (1.13)$$

L'hamiltoniana classica del sistema si ottiene dalla trasformata di Legendre della Lagrangiana, a sua volta definita come la differenza tra 1.6.1 e 1.13 . Con le opportune manipolazioni algebriche, si ha che l'hamiltoniana rispetta la seguente formulazione:

$$H = \frac{Q^2}{2C} + C \frac{\Phi^2}{2LC} \quad (1.14)$$

Da cui è evidente l'analogia con l'hamiltoniana dell'oscillatore armonico classico

$$H = \frac{p^2}{2m} + \frac{m\omega^2 x^2}{2} \quad (1.15)$$

In cui le variabili generalizzate Q e Φ sono rispettivamente espresse come p e x con $m = C$ e $\omega = \frac{1}{\sqrt{LC}}$.

Per una trattazione quantistica, vengono promosse

$$Q \rightarrow \hat{Q}$$

$$\Phi \rightarrow \hat{\Phi}$$

Tuttavia per semplicità i “cappelli” verranno omessi, da questo punto verranno indicati Q e Φ riferendosi ai rispettivi operatori quantistici.

In un circuito risonante come quello in 1.2 sia L che C sono elementi di un circuito lineare.

Vengono definiti gli operatori *n numero di coppie di Cooper e ϕ , flusso ridotto o fase gauge-invariante*

$$n = \frac{e}{2Q} \quad \phi = 2\pi \frac{\Phi}{\Phi_0}$$

In cui $\Phi_0 = \frac{h}{2e}$ è il *quanto di flusso superconduttore* . Questi due operatori formano una coppia coniugata canonica.

L' hamiltoniana quantistica è dunque:

$$H = 4E_C n^2 + \frac{1}{2} E_L \phi^2 \quad (1.16)$$

E_c è chiamata **energia di carica** e rappresenta la quantità di energia necessaria ad aggiungere una coppia di Cooper all'isola superconduttrice di capacità C . E_L è l' **energia induttiva**.

$$E_C = \frac{e^2}{2C} \quad E_L = \left(\frac{\Phi_0}{2\pi} \right)^2 \frac{1}{L}$$

L'Hamiltoniana nella 1.16 è identica a quella di un oscillatore armonico quantistico (QHO). ϕ viene trattata come la coordinata di posizione generalizzata, in modo che il primo termine sia l'energia cinetica e il secondo termine l'energia potenziale.

La dipendenza quadratica da ϕ dell'energia potenziale influenza le soluzioni degli autovalori come mostrato in Fig.1.3. La soluzione di questo problema agli autovectori fornisce una serie infinita di stati propri $|k\rangle$ le cui energie proprie corrispondenti E_k sono tutte equidistanti

$$E_{k+1} - E_k = \omega_r \hbar \quad (1.17)$$

$$\text{Con } \omega_r = \sqrt{\frac{8E_L E_C}{\hbar}} = \frac{1}{\sqrt{LC}}$$

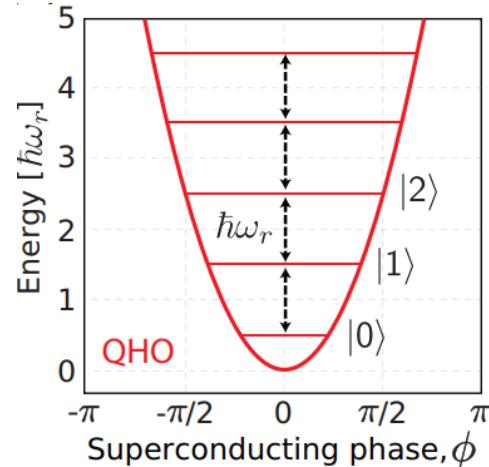


Figura 1.3: Energia potenziale in funzione di ϕ

E' possibile riscrivere l'hamiltoniana dell'oscillatore armonico quantistico in una forma più compatta, introducendo gli operatori di creazione e annichilamento a e a^\dagger tramite la *seconda quantizzazione*

$$H = \hbar\omega_r \left(a^\dagger a + \frac{1}{2} \right) \quad (1.18)$$

Di conseguenza, gli operatori n e ϕ possono essere espressi come:

$$\phi = \phi_0 (a + a^\dagger) \quad n = n_0 i (a - a^\dagger) \quad (1.19)$$

n_0 e ϕ_0 vengono chiamate *fluttuazioni di punto zero*, ossia sono le *fluttuazioni quantistiche del ground state*.

Dal punto di vista quantistico, gli stati sono rappresentati come funzioni d'onda che sono generalmente distribuite su un intervallo di valori di n e ϕ e, di conseguenza, le funzioni d'onda hanno deviazioni standard non nulle. Tali distribuzioni delle funzioni d'onda sono definite *fluttuazioni quantistiche*.

Prima che un sistema come quello di un QHO possa essere usato come qubit, è fondamentale definire uno spazio di calcolo in cui è possibile distinguere una transazione di stato da un'altra. Vengono solitamente presi in considerazione le transazioni tra lo stato fondamentale ed il primo stato eccitato, senza eccitare anche gli stati $|k\rangle$ successivi.

Dato che molte operazioni di gate dipendono dalla selettività della frequenza, le caratteristiche lineari di un QHO, responsabili della spaziatura equidistante dei livelli energetici mostrati in figura Fig.1.3, rappresentano un limite.

1.6.2 Anarmonicità: La Giunzione Josephson

Per mitigare il problema delle dinamiche indesiderate che coinvolgono stati non computazionali, è necessario rendere il sistema non lineare. In breve, richiediamo che la frequenza di transizione tra il *ground state* e il primo stato eccitato sia sufficientemente diversa dalla frequenza di transizione tra il primo e il secondo stato eccitato, in modo tale da poter essere indirizzate individualmente. In generale, è preferibile una anarmonicità elevata. Nella pratica, la quantità di anarmonicità stabilisce un limite su quanto brevi possono essere gli impulsi utilizzati per pilotare il qubit.

Per introdurre la non linearità necessaria a modificare il potenziale armonico, utilizziamo la **giunzione Josephson**, un elemento di circuito non lineare e privo di dissipazione che costituisce la spina dorsale nei circuiti superconduttori.

Sostituendo tale giunzione all'induttore lineare, come mostrato in Fig.1.4, l'energia potenziale del sistema viene modificata nel seguente modo

$$\tilde{\mathcal{U}} = E_J \cos \phi \quad (1.20)$$

Con

$$E_J = \frac{I_c \phi_0}{2\pi}$$

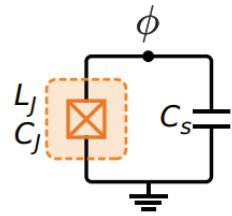


Figura 1.4: Circuito con giunzione Josephson.

I_C rappresenta la corrente critica della giunzione.

Dopo aver introdotto la giunzione Josephson nel circuito, l'energia potenziale non assume più una forma parabolica, ma piuttosto presenta una forma cosinusoïdale come mostrato in Fig.1.5, il che rende lo spettro energetico non degenerato. Pertanto, la giunzione Josephson è l'ingrediente chiave che rende l'oscillatore anarmonico e consente così di identificare un sistema quantistico a due livelli univocamente indirizzabile.

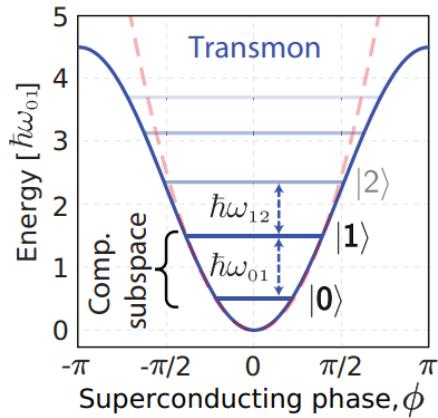


Figura 1.5: Energia potenziale dopo l'introduzione della giunzione Josephson

1.6.3 Il Qubit Transmon

La dinamica del sistema è governata dall'energia dominante tra E_C e E_J . Si tende a progettare qubit superconduttori tali che $E_J \gg E_C$, in modo da mitigare le fluttuazioni dell'operatore n .

Per accedere a questo regime, un approccio utilizzato è quello di aumentare la capacità C in modo da diminuire E_C . Il **qubit transmon** non è altro che un circuito in cui viene messo un condensatore in parallelo alla giunzione. In questo limite, la fase superconduttrice ϕ è un buon numero quantico, cioè la diffusione (o fluttuazione quantistica) dei valori di ϕ rappresentata dalla funzione d'onda quantistica è piccola[11]. Gli autostati a bassa energia sono quindi, in buona approssimazione, stati localizzati nel pozzo potenziale.

Possiamo vedere che il qubit transmon è fondamentalmente un oscillatore debolmente anarmonico (AHO). Se l'eccitazione verso stati non computazionali superiori è soppressa durante qualsiasi operazione di gate, possiamo trattare efficacemente l'AHO come un sistema quantistico a due livelli.

Tuttavia, bisogna sempre ricordare che i livelli superiori esistono fisicamente. La loro influenza sulla dinamica del sistema deve essere presa in considerazione durante la progettazione del sistema e dei suoi processi di controllo.

1.7 Errori di lettura del qubit

Processi fisici casuali e incontrollabili nell'equipaggiamento di controllo e misurazione del qubit o nell'ambiente locale che circonda il processore quantistico sono fonti di rumore che portano alla decoerenza e riducono la fedeltà operativa dei qubit.

1.7.1 Rumore nei QPU superconduttori

In un sistema chiuso, l'evoluzione dinamica dello stato di un qubit è deterministica. Conoscendo lo stato iniziale del qubit e la sua hamiltoniana, è possibile prevederne lo stato in qualsiasi momento futuro. In un sistema aperto la situazione è differente: il qubit infatti interagisce con gradi di libertà incontrollati, definiti proprio come fluttuazioni o rumore, la cui presenza con il passare del tempo rende lo stato del qubit sempre meno simile allo stato previsto inizialmente. Esistono molte diverse fonti di rumore che influenzano i sistemi quantistici, e possono essere categorizzate in due tipi principali: rumore sistematico e rumore stocastico.

Rumore sistematico

Il rumore sistematico deriva da un processo riconducibile a un errore fisso di controllo o di lettura. Una volta identificati, possono essere mitigati attraverso una più accurata calibrazione del dispositivo, o con il miglioramento delle tecniche di design e fabbricazione dei QPU.

Rumore stocastico

il rumore stocastico deriva da fluttuazioni casuali di parametri che sono accoppiati al nostro qubit. Ad esempio, l'oscillatore che fornisce il segnale portante per un impulso di controllo del qubit può avere fluttuazioni di ampiezza o fase. Inoltre, campi elettrici e magnetici che fluttuano casualmente nell'ambiente locale del qubit possono accoppiarsi con esso. Ciò crea fluttuazioni sconosciute e incontrollate di uno o più parametri del qubit, portando alla decoerenza.

1.7.2 Modellazione del rumore e della decoerenza

Cenni al modello di Bloch-Redfield

Nel quadro standard di Bloch-Redfield[11] della dinamica dei sistemi a due livelli, le sorgenti di rumore debolmente accoppiate ai qubit hanno tempi di correlazione brevi rispetto alla dinamica del sistema. In questo caso, i processi di rilassamento sono caratterizzati dal **tasso di rilassamento longitudinale** (Γ_1) e dal **tasso di rilassamento trasversale** (Γ_2), rispettivamente definiti come l'inverso dei tempi di rilassamento longitudinali e trasversali:

$$\Gamma_1 = \frac{1}{T_1} \quad (1.21)$$

$$\Gamma_2 = \frac{1}{T_2} = \frac{\Gamma_1}{2} + \Gamma_\phi \quad (1.22)$$

Per uno stato iniziale $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, è possibile definire la **Matrice densità di Bloch-Redfield** nel seguente modo:

$$\rho_{BR}(t) = \begin{pmatrix} 1 + (\alpha^* \beta e^{i\delta\omega t} - 1) e^{-2\Gamma_2 t} & \alpha \beta^* e^{i\delta\omega t} e^{-\Gamma_1 t} \\ \alpha^* \beta e^{-i\delta\omega t} e^{-\Gamma_1 t} & e^{-\Gamma_1 t} \end{pmatrix} \quad (1.23)$$

La (1.23) descrive l'evoluzione temporale di un qubit a contatto con un ambiente rumoroso. I termini $e^{-2\Gamma_2 t}$ e $e^{-\Gamma_1 t}$ rappresentano le **funzioni di decadimento trasversale e longitudinale**, mentre il termine $e^{i\delta\omega t}$ con $\delta\omega = \omega_q - \omega_d$ introduce un accumulo di fase esplicito per tenere conto dei casi in cui la frequenza del qubit ω_q sia diversa dalla frequenza di controllo ω_d , ossia la frequenza applicata tramite un campo esterno per manipolare lo stato del qubit. La frequenza di controllo ω_d serve ad implementare i gate, nel caso dei qubit superconduttori tali impulsi rientrano nello spettro delle microonde.

Importante notare come per $t \gg (T_1, T_2)$ la matrice ρ_{BR} diventi

$$\rho_{BR}(t \rightarrow \infty) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

mostrando come dopo tempi sufficientemente grandi il qubit decada sul suo stato fondamentale.

Tasso di rilassamento longitudinale Γ_1

Il tasso di rilassamento longitudinale Γ_1 descrive la depolarizzazione lungo l'asse di quantizzazione del qubit, spesso indicata come **decadimento energetico** o **rilassamento energetico**. In questo contesto, un qubit con polarizzazione $p = 1$ si trova nello stato fondamentale $|0\rangle$ (polo nord sulla sfera di Bloch), mentre un qubit con polarizzazione $p = -1$ si trova nel primo stato eccitato $|1\rangle$. $p = 0$ invece indica uno stato misto depolarizzato al centro della sfera di Bloch.

La depolarizzazione si verifica a causa dello scambio di energia con l'ambiente, Γ_1 inoltre può essere visto come somma di un tasso di transizione verso l'alto Γ_\uparrow e di un tasso di transizione verso il basso Γ_\downarrow , i quali indicano rispettivamente il tasso di eccitazione da $|0\rangle \rightarrow |1\rangle$ e di rilassamento da $|1\rangle \rightarrow |0\rangle$.

Γ_\uparrow e Γ_\downarrow sono legati dalla seguente relazione:

$$\Gamma_\uparrow = e^{-\frac{\hbar\omega_q}{k_B T}} \Gamma_\downarrow \quad (1.24)$$

Dove T indica la temperatura dell'ambiente e k_B la costante di Boltzman.

Tipicamente un qubit opera ad una temperatura nell'ordine dei mK con frequenze nell'ordine dei GHz. In questo limite il tasso di eccitazione è esponenzialmente soppresso, in modo da rendere meno probabili le eccitazioni termiche dell'ambiente.

Tasso di decoerenza pura Γ_ϕ

Il tasso di decoerenza pura Γ_ϕ descrive la polarizzazione nel piano $x - y$ della sfera di Bloch. La decoerenza pura è causata dal rumore longitudinale che si accoppia al qubit lungo l'asse z . Questo rumore longitudinale causa fluttuazioni nella frequenza del qubit ω_q in modo tale che non sia più uguale alla frequenza del sistema pilota ω_d , provocando la precessione del vettore di Bloch in avanti o indietro nel sistema di riferimento rotante.

Ci sono alcune importanti distinzioni tra la decoerenza pura e il rilassamento energetico. In primo luogo, a differenza del rilassamento energetico, la decoerenza pura non è un fenomeno risonante; il rumore a qualsiasi frequenza può modificare la frequenza del qubit e causare decoerenza. Pertanto, la decoerenza del qubit è soggetta a rumore a banda larga. In secondo luogo, poiché la decoerenza pura è un fenomeno elastico (non c'è scambio di energia con l'ambiente), è in linea di principio reversibile. Cioè, la decoerenza può essere "annullata" – con la conservazione dell'informazione quantistica – attraverso l'applicazione di operazioni unitarie, ad esempio, impulsi di disaccoppiamento dinamico

Tasso di rilassamento trasversale Γ_2

Il tasso di rilassamento trasversale Γ_2 descrive la perdita di coerenza di uno stato di sovrapposizione. La decoerenza è causata in parte dal rumore longitudinale, che provoca fluttuazioni nella frequenza del qubit e porta alla decoerenza pura Γ_ϕ ed in parte dal rumore trasversale, con un tasso Γ_1 , il cui processo rompe la fase,

dato che una volta verificato, il vettore di Bloch sarà puntato verso $|0\rangle$ perdendo tutta l'informazione su quale direzione stesse puntando il vettore di Bloch lungo l'equatore.

[11]

1.8 Quantum Provider

Un *Quantum Provider* è un servizio in grado di mettere a disposizione l'accesso ad un *hardware* quantistico o ad una simulazione di esso. Un provider rende possibile l'esecuzione di un **circuito quantistico**, ossia una sequenza di gate applicati ad uno o più qubit.

1.8.1 Classe 'fake_provider' di IBM

Nei capitoli successivi, verranno utilizzati dei "*Fake Provider*", ossia delle simulazioni di dispositivi quantistici appartenenti ad IBM, il cui accesso è possibile tramite il modulo `'qiskit_ibm_runtime.fake_provider'`

In seguito, i termini "provider" e "dispositivo quantistico" faranno riferimento ad un oggetto appartenente al modulo di IBM appena citato.

1.8.2 Caratteristiche di un provider

Tra le caratteristiche fondamentali di un provider, troviamo il numero di qubit da cui è composto, la loro disposizione reciproca e i gate in grado di essere eseguiti direttamente da un dispositivo.

coupling map

La *coupling map* descrive la connettività fisica tra i qubit di un dispositivo quantistico. La conoscenza di tale struttura è importante al fine di una progettazione del circuito quantistico atta a rispettare i collegamenti fisici tra i qubit. Oltre a ciò, è possibile visualizzare informazioni importanti come i tempi di rilassamento T_1 e T_2 di ogni qubit, oppure gli errori di lettura dei singoli qubit e gli errori legati ad alcuni gate come la CNOT o la Hadamard.

Viene mostrata la *coupling error map* del provider "FakeLagosV2" appartenente a IBM in Fig.1.6.

native gate

I *native gate* corrispondono ad una lista di gate ad uno o più qubit direttamente eseguibili dall'hardware. Possono essere diversi da un provider all'altro, ma tipicamente sono inclusi tra i *native gate* l'identità e i gate di rotazione, in quanto da quest'ultimi è possibile ottenere tutti gli altri gate ad uno o due qubit, come mostrato

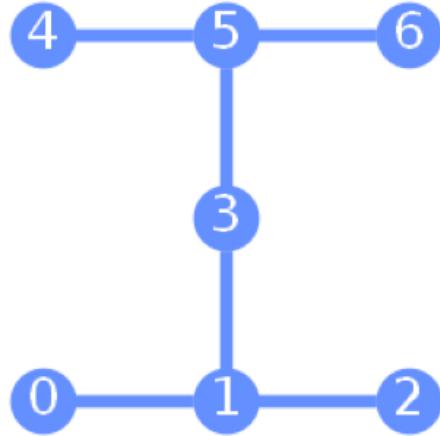


Figura 1.6: Coupling Map di FakeLagosV2, provider composto da 7 qubit.

nell'equazione (1.9). Una combinazione dei native gate permette di ricreare anche tutti gli altri gate.

1.8.3 Transpile Circuit

E' stato precedentemente affermato che un provider quantistico rende possibile l'esecuzione di un circuito quantistico su un hardware fisico. Tuttavia, un circuito quantistico non è altro che un formalismo matematico contenente una serie di istruzioni (i gate) e di misurazioni, atte ad influenzare lo stato dei qubit. In un circuito quantistico è possibile, ad esempio, applicare un gate di controllo tra due qubit che non sono direttamente collegati nel provider su cui viene effettuata la misura, oppure possono essere adoperati gate non appartenenti alla lista dei *native gate*. Per ovviare a questo problema, il circuito viene "tradotto" in un circuito equivalente leggibile dal provider. In seguito vengono illustrati brevemente alcuni esempi.

1.8.4 Gate non appartenente ai Native Gate

Viene preso in considerazione il circuito in Fig.1.7. La porta H non appartiene ai gates nativi di "FakeLagosV2", dunque prima di poterlo eseguire è necessario farne il *transpile*, come mostrato in Fig.1.8.

La porta H è stata convertita in una serie di tre gate equivalenti a meno di un fattore di fase globale. Sono state applicate rispettivamente una rotazione intorno all'asse z di $\pi/2$, seguita da una rotazione di $\pi/2$ intorno all'asse x (rappresentata come \sqrt{X} a meno di un fattore di fase globale) ed un'altra rotazione di $\pi/2$ intorno all'asse z .

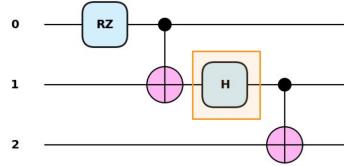


Figura 1.7: Circuito quantistico con gate H.

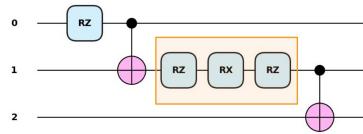


Figura 1.8: Circuito transpile.

1.8.5 CNOT applicata a due qubit non collegati fisicamente

Un ulteriore esempio può essere rappresentato dal circuito in Fig 1.9

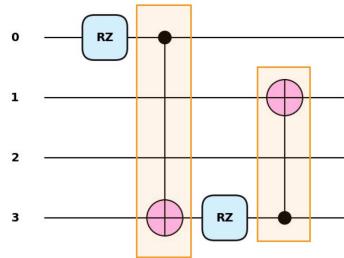


Figura 1.9: Circuito quantistico con gate CNOT applicato a due qubit non collegati fisicamente.

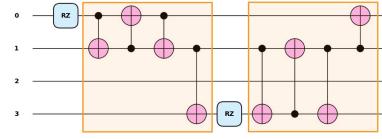


Figura 1.10: Circuito transpile.

In questo caso, per far sì che il circuito sia eseguibile sul provider, è necessario invertire i qubit 0 e 1, in modo da collegare i qubit 0 e 3. Per far sì, vengono applicate tre porte CNOT tra i qubit che si vogliono scambiare, andando a creare l'equivalente di una porta SWAP (scambio). successivamente i qubit vengono nuovamente invertiti per rendere possibile l'applicazione di una CNOT tra il qubit 3 e 1.

E' importante considerare che un circuito transpile avrà sì un azione equivalente al circuito originale, ma a costo di dover applicare un numero maggiore di gate rispetto a quelli originali, aumentando di conseguenza i possibili errori e probabilità di decoerenza.

Capitolo 2

Machine Learning

In questo capitolo viene fornita un'introduzione al machine learning classico, introducendo i concetti fondamentali per comprendere il funzionamento di una rete neurale, ossia uno degli algoritmi utilizzati per l'apprendimento automatico. Viene descritta la struttura tipica di una rete, gli elementi da cui è composta e i passi principali dell'algoritmo di apprendimento di una rete. Vengono inoltre poste le basi sull'implementazione di un circuito quantistico variazionale all'interno di una rete neurale classica, entrando così nel contesto del quantum machine learning e delle reti neurali ibride.

2.1 Introduzione al Machine Learning

Il machine learning è un ramo dell'intelligenza artificiale (IA) che prevede lo sviluppo di algoritmi ispirati ai processi decisionali del cervello umano, che possono apprendere dai dati disponibili per fare classificazioni, regressioni ma anche prendere decisioni sulla base di fattori esterni o generare dati e immagini a partire da una richiesta in input[7].

Le IA prevedono una fase di addestramento, tramite la quale riescono ad apprendere e ad adattarsi ai dati forniti in input e di una fase di validazione, in cui viene testata la capacità del modello di generalizzare anche su dei nuovi dati. I modelli di machine learning vengono suddivisi in tre categorie sulla base del processo di apprendimento.

apprendimento supervisionato L'apprendimento supervisionato è definito dall'uso di dataset etichettati. Questa caratteristica indica che il dataset è provvisto del target di output, il quale viene presentato al modello durante la fase di addestramento. Successivamente durante la fase di validazione vengono dati in input campioni sprovvisti di target al modello, che li confronta con gli esempi forniti durante l'addestramento in modo tale da prevedere l'etichetta corretta da assegnare[15].

apprendimento non supervisionato L’Apprendimento non supervisionato, utilizza algoritmi di apprendimento automatico per analizzare e raggruppare dataset non etichettati (cluster). Questi algoritmi scoprono modelli nascosti o raggruppamenti di dati senza la necessità dell’intervento umano.

apprendimento semi-supervisionato L’apprendimento semi-supervisionato offre una via di mezzo tra l’apprendimento supervisionato e quello non supervisionato. Durante l’addestramento, utilizza un piccolo set di dati etichettati per guidare l’apprendimento da un set di dati più grande e non etichettati [8].

Le reti neurali fanno parte degli algoritmi utilizzati nell’ambito del machine learning. Prende decisioni in modo simile al cervello umano, utilizzando processi che imitano il modo in cui i neuroni biologici lavorano insieme per identificare fenomeni, pesare le opzioni e arrivare alle conclusioni [10].

2.2 Struttura di una rete neurale

Come affermato in precedenza, una rete neurale è un modello che vuole imitare il funzionamento del cervello umano. Nelle reti neurali biologiche, i neuroni, quando sono ”eccitati”, inviano neurotrasmettitori ai neuroni interconnessi per modificare i loro potenziali elettrici. Quando il potenziale elettrico supera una soglia, il neurone viene ”eccitato” e invierà neurotrasmettitori ad altri neuroni .

Allo stesso modo, una rete neurale artificiale è composta da neuroni artificiali che elaborano i dati in input in un dato di output, il quale a sua volta viene elaborato come dato di input dai neuroni connessi. Alle connessioni tra i neuroni sono associati dei pesi variabili e una soglia, al di sopra della quale un neurone si attiva.

Dal punto di vista dell’informatica, possiamo semplicemente considerare una rete neurale come un modello matematico con molti parametri[18].

2.2.1 Neurone

Un **neurone artificiale** è la componente elementare da cui è composta una rete neurale. Nel 1943, McCulloch e Pitts hanno astratto il processo sopra descritto in un modello semplice chiamato modello di McCulloch-Pitts (*M-P neuron model*) o unità logica a soglia (TLU), che è ancora in uso oggi. Questo modello rappresenta uno dei primi tentativi di formalizzare il funzionamento di un neurone artificiale. Il modello M-P descrive il neurone come un’unità logica che riceve vari input $\{x_i\}$ ponderati da pesi $\{\omega_i\}$. l’output del neurone è definito nel seguente modo

$$y = \sum_{i=0}^n \omega_i x_i - \theta$$

Se la somma supera la soglia (θ), il neurone si attiva e genera un output; altrimenti, non si attiva e produce un output nullo [18]. Il suo funzionamento è schematizzato in Fig.2.1 .

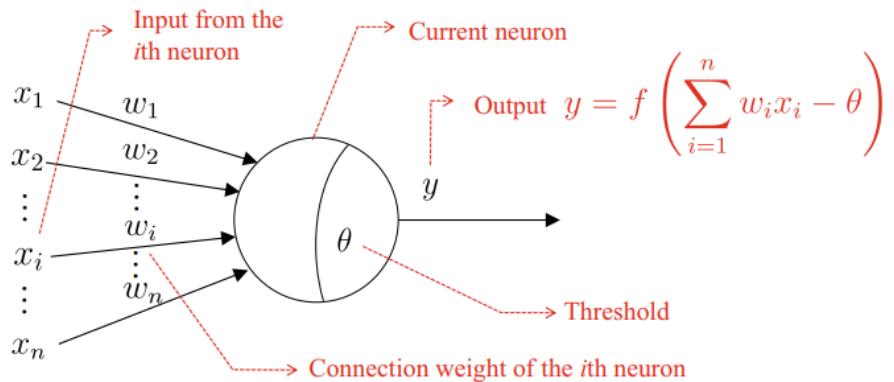


Figura 2.1: Schema di un neurone

All'output così definito, viene inoltre applicata una **funzione di attivazione**, il cui fine è quello di introdurre non linearità nel modello. Di seguito sono descritte le funzioni di attivazione utilizzate ai fini del lavoro di tesi.

ReLU

La funzione ReLU (*Rectified Linear Unit*), la cui forma funzionale è mostrata in Fig.2.2, è definita come $\max(0, x)$. Oppure in modo analogo può essere scritta come segue

$$Relu(x) = \begin{cases} x & \text{se } x \geq 0 \\ 0 & \text{se } x < 0 \end{cases}$$

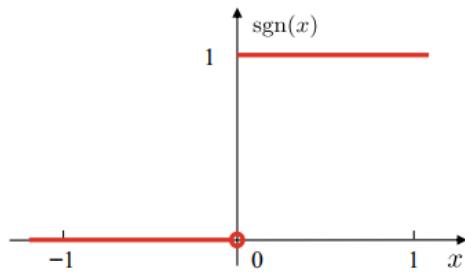


Figura 2.2: Grafico della funzione segno, condivide la stessa forma funzionale della ReLU.

Sigmoide

La funzione sigmoide mappa il valore di input in un continuo di valori compresi tra 0 e 1.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

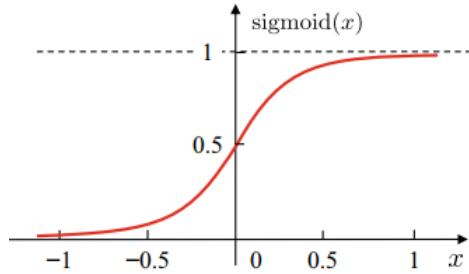


Figura 2.3: Grafico funzione sigmoide

Tanh

La tangente iperbolica è una funzione dalla forma simile alla sigmoide ma che mappa i valori di input nell'intervallo $[-1, 1]$

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

2.2.2 Percettrone e reti multilayer

a struttura più semplice con cui costruire una rete neurale è costituita da un layer di neuroni che riceve i segnali di input e un layer di output con un singolo neurone, noto anche come **percettrone**. Una rete così descritta può essere utilizzata per la classificazione binaria di un dataset linearmente separabile, ossia un insieme di dati generalmente descritti da vettori n-dimensionali le cui classi possono essere separate da un singolo iperpiano lineare.

Per risolvere problemi più complessi, è necessario introdurre dei layer intermedi nascosti (o *hidden layer*). Una rete strutturata come una sequenza di layer è chiamata **multi-layer feedforward neural network**. Ogni neurone di un layer è completamente connesso ai neuroni del layer successivo, ma non a quelli appartenenti allo stesso layer o a layer non adiacenti.

In una rete neurale feedforward, i dati vengono letti dal layer di input e propagati in avanti attraverso i layer nascosti fino a raggiungere il layer di output, come mostrato in Fig.2.4[18].

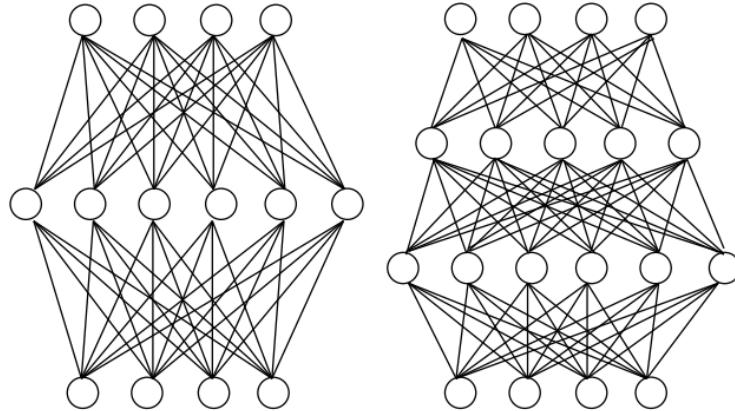


Figura 2.4: esempio di rete feedforward con uno e due hidden layer

2.3 Addestramento supervisionato

L'apprendimento supervisionato si basa su algoritmi che utilizzano un set di addestramento etichettato per insegnare ai modelli a produrre l'output desiderato. Questo set di dati di addestramento include sia gli input sia gli output corretti, consentendo al modello di apprendere e dunque tentare di replicare le risposte corrette. L'algoritmo misura la sua precisione attraverso una funzione di costo, e si regola fino a quando l'errore non è stato sufficientemente ridotto[9].

Nel contesto della classificazione binaria, ossia quando si classifica tra due sole classi, viene utilizzato un dataset di addestramento in cui ogni elemento è composto da diverse **feature**, ossia le caratteristiche che fungono da input per la rete. Il dataset è inoltre etichettato, il che significa che ad ogni elemento viene assegnata una **label**, che rappresenta a quale classe l'elemento appartiene (ad esempio, 0 o 1 nel caso binario).

La rete neurale sarà quindi composta da un layer di input, contenente un numero di neuroni pari al numero di feature, una serie di layer nascosti a cui viene applicata una funzione di attivazione (come ReLU), e un layer di output costituito da un singolo neurone, su cui viene applicata la funzione di attivazione sigmoide. Quest'ultima produce un output compreso tra 0 e 1. L'output della rete viene interpretato come la probabilità di appartenenza a una delle due classi: valori inferiori a 0.5 associeranno l'elemento alla classe 0, mentre valori superiori lo associeranno alla classe 1[2].

Una rete neurale apprende tramite algoritmi di apprendimento, tra cui l'algoritmo di retropropagazione dell'errore (o **Error Backpropagation Algorithm**) è uno dei più utilizzati.

2.4 Error Backpropagation Algorithm

L'algoritmo di backpropagation (BP) utilizza un dataset di addestramento

$$\mathcal{D}_T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$$

in cui ogni $x_i \in \mathbb{R}^d$ sono i dati di input con d il numero di feature, mentre $y_i \in \mathbb{R}^l$ sono i valori di output target di dimensione l . Viene considerata per semplicità una rete composta da un solo hidden layer.

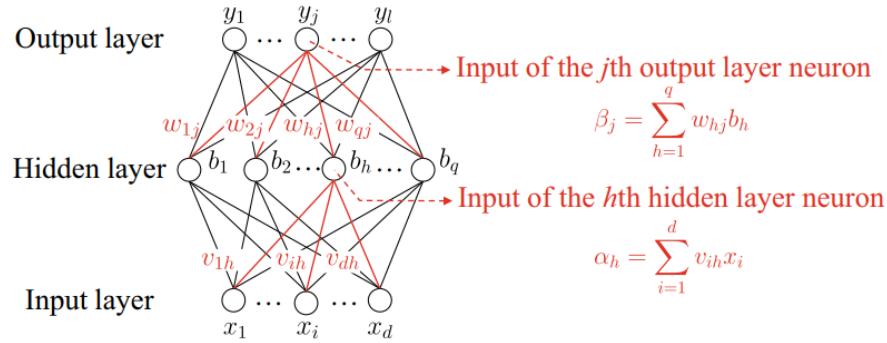


Figura 2.5: rete neurale feedforward in cui sono rappresentati i parametri utilizzati nell'algoritmo di backpropagation

La rete mostrata in Fig.2.5 è composta da d neuroni nel layer di input, q neuroni nel layer nascosto e l neuroni nel layer di output. I pedici h, i, j fanno riferimento rispettivamente ai neuroni nel layer nascosto, di input e di output; i pedici doppi invece si riferiscono alle connessioni tra neuroni di layer diversi. I parametri sono descritti nel seguente modo.

- v_{ih} : peso della connessione tra neurone i -esimo e neurone h -esimo
- γ_h : soglia del neurone h -esimo
- w_{hj} : peso della connessione tra neurone h -esimo e neurone j -esimo
- θ_j : soglia del neurone j -esimo

Da cui vengono definiti α_h , input del neurone h -esimo nel layer nascosto

$$\alpha_h = \sum_{i=1}^d v_{ih} x_i - \gamma_h$$

e β_j , input del neurone j -esimo nel layer di output

$$\beta_j = \sum_{h=1}^q w_{hj} b_h - \theta_j$$

dove i vari b_h rappresentano i neuroni nel layer nascosto.

Calcolo del gradiente ed aggiornamento dei pesi

L'algoritmo viene descritto come una serie di passi

1. Viene preso in considerazione un singolo elemento del dataset (x_k, y_k) , il quale viene fatto passare tramite la rete e propagato fino al layer di output, dove verrà prodotto un valore di output $\hat{y}_k = \{\hat{y}_{1k}, \hat{y}_{2k}, \dots, \hat{y}_{lk}\}$

$$\hat{y}_{jk} = f(\beta_j)$$

dove f è la funzione d'attivazione.

2. Viene calcolato l'errore commesso dal modello per il dato x_k

$$E_k = \mathcal{L}(y_k, \hat{y}_k)$$

l'errore è calcolato tramite una **funzione di costo** (*loss function*) che riceve in ingresso l'output fornito dalla rete \hat{y}_k e l'output target y_k . Possono essere utilizzate diverse funzioni di perdita, come la *Mean Squared Error* o la *Binary Cross Entropy*, quest'ultima verrà opportunamente descritta in seguito.

3. I pesi vengono aggiornati tramite la **discesa del gradiente**:

$$w'_{hj} = w_{hj} - \eta \frac{\partial E_k}{\partial w_{hj}}$$

in cui η è il tasso d'apprendimento (**learning rate**). È possibile riscrivere la derivata parziale dell'errore rispetto al peso hj -esimo utilizzando la *chain rule*

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_{jk}} \frac{\partial \hat{y}_{jk}}{\partial \beta_j} \frac{\partial \beta_j}{\partial w_{hj}}$$

In particolare, il primo fattore dipende dalla funzione di perdita utilizzata, il secondo fattore varia a seconda della funzione di attivazione utilizzata (ad esempio se la funzione di attivazione è la sigmoide si ha che $f' = f(1 - f)$), mentre il terzo fattore è semplicemente b_h .

Allo stesso modo vengono aggiornati anche i parametri tra il layer nascosto e il layer di input. Nel caso in cui parametri vengano aggiornati dopo ogni elemento del dataset si parla di **Stochastic Gradient Descent (SGD)**, se altrimenti vengono aggiornati dopo un insieme di elementi (*batch*) si parla di **Batch Gradient Descent**.

4. Finita un'intera iterazione sul dataset di addestramento è trascorsa un'**epoca**. L'algoritmo viene ripetuto fino all'ottenimento di un errore sufficientemente basso oppure dopo un determinato numero di epoche[18].

2.4.1 Binary Cross Entropy (BCE)

La BCE è una delle funzioni di costo maggiormente utilizzate ai fini di una classificazione binaria. Sia y il target (o *label*) reale (0 o 1) e \hat{y} il valore calcolato dalla rete compreso nell'intervallo [0,1]; La BCE si esprime come:

$$BCE(y, \hat{y}) = -[y \ln(\hat{y}) + (1 - y) \ln(1 - \hat{y})]$$

Vengono analizzate le varie casistiche:

valori output-target concordi: l'output della rete tende al valore target, la funzione di perdita tende correttamente a 0

valori output-target discordi: in questo caso l'output della rete è sbilanciato verso il target sbagliato, la funzione tende a $+\infty$ [1].

2.5 Minimo Locale e Minimo Globale

La funzione di costo di una rete neurale è una funzione dei pesi e dei bias. Da questo punto di vista, il processo di addestramento è un processo di ottimizzazione

dei parametri. L’obiettivo dell’ottimizzazione è trovare la configurazione dei parametri che corrisponda al minimo globale della funzione. Tuttavia, può accadere che il processo di ottimizzazione raggiunga dei minimi locali, come mostrato in Fig.2.6. Nella fase iniziale dell’addestramento i parametri vengono inizializzati casualmente, solitamente secondo una distribuzione normale. Ciò si traduce in un punto preciso nello spazio dei parametri. Algoritmi quali la **discesa del gradiente** effettua iterazioni spostandosi verso la direzione del gradiente negativo. Il **learning rate** (η) corrisponde alla lunghezza del ”passo” percorso verso tale direzione.

Per questo motivo, la scelta di η risulta fondamentale nell’ottimizzazione di una rete neurale. Valori troppo alti, ad esempio, potrebbero non far mai convergere il modello verso il minimo, al contrario un valore troppo basso, potrebbe far sì che il modello rimanga intrappolato in un minimo locale, senza possibilità di raggiungere il minimo globale[18].

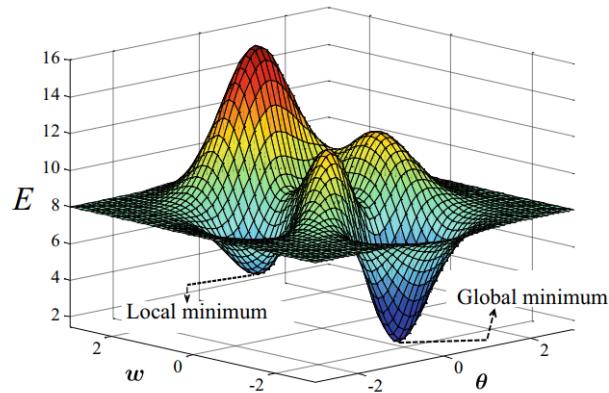


Figura 2.6: Rappresentazione grafica dello spazio dei parametri nel caso di un solo peso e un solo bias.

2.6 Performance di una rete neurale

Nelle sezioni precedenti è stato descritto il processo di addestramento di una rete neurale feedforward. In questa sezione si vuole vedere come valutare le prestazioni di un modello e monitorare problematiche come l’overfitting e l’underfitting, descritte in seguito.

2.6.1 Processo di validazione

Al termine di ogni epoca di addestramento, per valutare le prestazioni di un modello vengono inviati alla rete nuovi elementi appartenenti ad un **dataset di validazione**. A differenza di quanto accade con il dataset di addestramento, i parametri del modello non vengono aggiornati durante la lettura degli elementi appartenenti al

dataset di validazione. Viene comunque calcolata la funzione di costo ed introdotta una metrica per la valutazione del modello, come l'**accuratezza**.

In una classificazione binaria, nel caso in cui l'output della rete sia superiore o inferiore ad una certa soglia (tipicamente 0.5), il dato viene associato alla classe 1 o 0. L'accuratezza è definita come la frazione dei dati correttamente classificati sul totale.

Confrontando l'andamento dell'errore in fase di addestramento e valutazione in funzione delle epoches, è possibile valutare in che modo il modello sta apprendendo, monitorando fenomeni come overfitting e underfitting [2].

2.6.2 Overfitting e Underfitting

Quando il modello apprende troppo bene gli elementi del dataset di addestramento, è probabile che alcune peculiarità di tale dataset vengano considerate come proprietà generali che tutti i campioni potenziali avranno, portando così ad una riduzione dell'accuratezza. Pertanto, la rete neurale perde di generalità, in quanto le previsioni saranno meno affidabili su dataset differenti da quello utilizzato per l'addestramento. Questo fenomeno è noto come **overfitting**, in figura Fig.2.7 è mostrato un tipico plot della funzione di costo in fase di addestramento validazione. Si può notare come, a partire da una determinata epoca, la funzione di costo sui dati di validazione inizi ad aumentare, a differenza di quanto accade sul dataset usato in addestramento. Questa è la conseguenza del fatto che il modello ha imparato troppo bene i dati di addestramento senza riuscire a generalizzare sui nuovi dati usati in fase di validazione. Il fenomeno opposto è noto come **underfitting**, ossia il modello non riesce ad apprendere le proprietà generali degli esempi di addestramento. In figura [Fig.2.8] è mostrato un esempio intuitivo dell'overfitting e dell'underfitting.

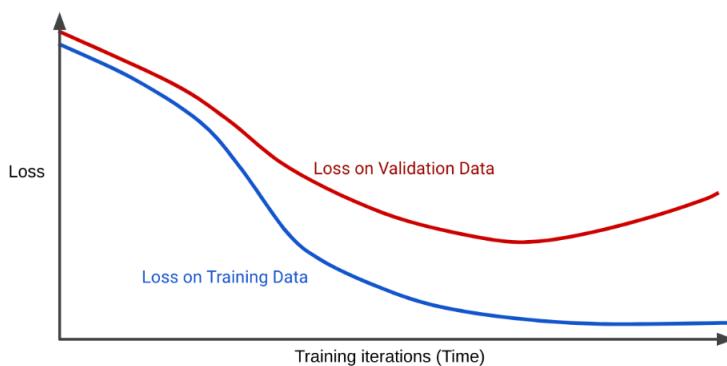


Figura 2.7: comportamento delle curve di perdita in fase di addestramento e validazione in funzione delle epoches

Tra le possibili ragioni, una capacità di apprendimento eccessivamente alta o eccessivamente bassa è una causa comune di overfitting o underfitting. Da un punto di vista pratico, l'underfitting è relativamente facile da superare. Ad esempio, è

possibile aggiungere più epoche di addestramento oppure aumentare il numero di neuroni. Tuttavia, l'overfitting è una difficoltà fondamentale nell'apprendimento automatico, e quasi tutti gli algoritmi di apprendimento implementano alcuni meccanismi per limitarne la comparsa. Un metodo comunemente utilizzato, consiste nell'introduzione del **dropout**, ossia la disattivazione casuale di una frazione di neuroni in ogni layer, al fine di limitare il fenomeno dell'overfitting [18].

Dropout

Per comprendere meglio il ruolo del dropout, immaginiamo un layer nascosto di 10 neuroni. Senza dropout, tutti i neuroni possono attivarsi in risposta ai dati in input. Tuttavia, potrebbe accadere che un piccolo gruppo di neuroni si attivi più frequentemente rispetto agli altri. Questo fenomeno può portare questi neuroni a specializzarsi eccessivamente su specifiche caratteristiche dei dati di addestramento e di conseguenza, tali neuroni saranno meno efficaci nel generalizzare su dati con caratteristiche diverse.

Allo stesso tempo, i neuroni che si attivano meno frequentemente non hanno la possibilità di apprendere in modo significativo, poiché non ricevono abbastanza stimoli dai dati. Questo rende il modello complessivamente meno capace di generalizzare su nuovi dati, andando incontro all'overfitting.

Il dropout è una tecnica che aiuta a contrastare questo problema. Funziona disattivando casualmente una percentuale di neuroni in ogni passaggio dell'addestramento. Per esempio, viene preso in considerazione un tasso di dropout del 30% in un layer con 10 neuroni: questo significa che, a ogni iterazione, 3 neuroni verranno disattivati in modo casuale.

Disattivando neuroni diversi a ogni iterazione, il modello è costretto a non dipendere sempre dagli stessi neuroni più "attivi". Questo riduce la possibilità che un piccolo gruppo di neuroni domini l'apprendimento. Inoltre, i neuroni che normalmente si attivano meno frequentemente vengono forzati a contribuire all'apprendimento, migliorando la capacità del modello di generalizzare su nuovi dati[13].

2.7 Implementazione di una rete neurale ibrida

Una rete neurale ibrida (o *hybrid neural network*) consiste in una rete neurale classica in cui uno o più layer classici vengono sostituiti da uno o più layer quantistici, ognuno dei quali è costituito da un **circuito quantistico varazionale**.

2.7.1 circuito quantistico varazionale

Nel primo capitolo sono stati introdotti i concetti di qubit, gate e circuito quantistico. Un circuito quantistico varazionale è un circuito quantistico dipendente da un vettore di parametri θ in grado di essere aggiornati nel processo di addestramento della rete neurale. Si prenda in considerazione un circuito quantistico composto

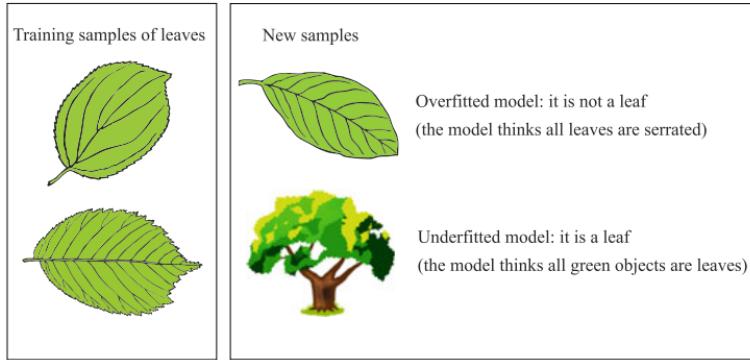


Figura 2.8: Esempio intuitivo dell’overfitting e dell’underfitting

da due qubit e di volerlo utilizzare come layer quantistico in una rete neurale. È possibile scrivere lo stato quantistico $|\psi\rangle$ come combinazione lineari degli stati di base (Eq. 1.3).

Il circuito, è caratterizzato da dei valori di input, ossia i valori di output dei neuroni del layer classico precedente e da dei parametri addestrabili. Nei circuiti utilizzati in questo lavoro di tesi, i valori di output vengono esclusivamente utilizzati per inizializzare le ampiezze del circuito. Le feature classiche vengono normalizzate in modo da rispettare la [1.1].

Sia $X = \{x_1, x_2, x_3, x_4\}$ il dato in input del circuito, si vuole mappare:

$$x_i \in \mathbb{R} \rightarrow \alpha_i \in [0, 1]$$

in modo che:

$$\sum_i^4 \alpha_i = 1$$

la normalizzazione è data da:

$$\alpha_i = \frac{x_i}{\|x\|}$$

con

$$\|x\|^2 = \sum_i^4 |x_i|^2$$

Il parametri θ invece, svolgono un ruolo analogo a quello dei pesi delle connessioni tra neuroni. Vengono utilizzati come angoli di rotazione nei gate di rotazione. Il circuito può essere reso più complesso applicando ulteriori gate al fine di modificare lo stato quantistico. Tra questi, la CNOT-gate è risultata essere fondamentale nel lavoro di tesi, grazie alla sua capacità di poter introdurre entanglement tra i qubit.

L’output del circuito corrisponde alla misura di uno o più qubit del circuito, il quale verrà propagato al layer classico successivo.

La trattazione è valida anche nel caso di circuiti con un numero superiore di qubit.

In sintesi, gli elementi di un dataset vengono elaborati dai vari layer classici e propagati fino a raggiungere il circuito variazionale utilizzato come layer quantistico. I valori ricevuti in input definiscono lo stato iniziale dello stato quantistico, tramite il processo che mappa i valori classici in ampiezze del circuito. Successivamente vengono applicati dei gate ai qubit, tra questi, i gate di rotazione utilizzano i parametri θ come angoli di rotazione, l'output del circuito, corrispondente alla misura di uno o più qubit viene a sua volta propagato al layer classico successivo, fino ad arrivare al layer di output della rete. In seguito, l'algoritmo di backpropagation oltre ad aggiornare i pesi della rete classica, ottimizza anche i parametri θ del layer quantistico.

Capitolo 3

Classificazione di dispositivi quantistici

In questo capitolo, vengono descritte le implementazioni delle reti neurali classica ed ibrida, con l'utilizzo di circuiti quantistici variazionali differenti al fine di poter effettuare un confronto delle prestazioni tra i diversi modelli. Inoltre viene descritta la struttura e la composizione del dataset utilizzato per la classificazione binaria. Per lo svolgimento dell'analisi dati sono stati utilizzati programmi scritti in linguaggio di programmazione Python. Per l'esecuzione dei circuiti quantistici e per la simulazione di dispositivi quantistici (provider) sono state utilizzate le librerie Qiskit e qiskit_ibm_runtime sviluppate da IBM; mentre per lo sviluppo della rete neurale classica è stata utilizzata la libreria Pytorch. L'implementazione dei circuiti quantistici come layer all'interno di una rete neurale è stata eseguita principalmente con le librerie Pennylane, nel presente capitolo è presente un confronto in termini di velocità di esecuzione dell'addestramento di una stessa rete ibrida implementata con Qiskit o Pennylane. Inoltre si è fatto uso della libreria Pandas per la gestione del dataset, della libreria sklearn.processing per la normalizzazione del dataset e della libreria Seaborn per la visualizzazione del plot tra feature del dataset.

3.1 Generazione del dataset

Il dataset utilizzato è composto da 2000 elementi, ciascuno dei quali caratterizzato da 4 feature ed una label binaria (0 o 1). Le classi sono bilanciate in quantità uguali, ossia il numero di elementi nel dataset etichettati con 0 ed 1 è lo stesso. Ogni campione è generato attraverso la misura di due qubit di un circuito quantistico su un provider quantistico. La misura viene ripetuta 10^4 volte, e ciascuna feature rappresenta la frequenza di uno dei 4 possibili stati di base $|00\rangle$, $|01\rangle$, $|10\rangle$ o $|11\rangle$. La classe (0 o 1) viene determinata in base al provider su cui è stata eseguita la misura. Il dataset è stato successivamente normalizzato affinché ogni feature abbia media 0 e varianza 1.

3.1.1 Provider quantistici

Sono stati utilizzati due differenti simulatori di dispositivi quantistici, rispettivamente denominati 'FakeLagosV2' e 'FakePerth', a cui verrà fatto riferimento rispettivamente con 'Lagos' e 'Perth' da questo punto della trattazione.

Lagos e Perth sono entrambi dispositivi composti da 7 qubit, inoltre condividono la stessa coupling map. La differenza tra i due dispositivi consiste nella differente quantità di rumore a cui i qubit dei diversi provider sono soggetti. Come mostrato in Fig.3.1 e Fig.3.2 i qubit hanno differenti errori di lettura (*readout error* e anche differenti errori legati a gate quali Hadamard e CNOT.

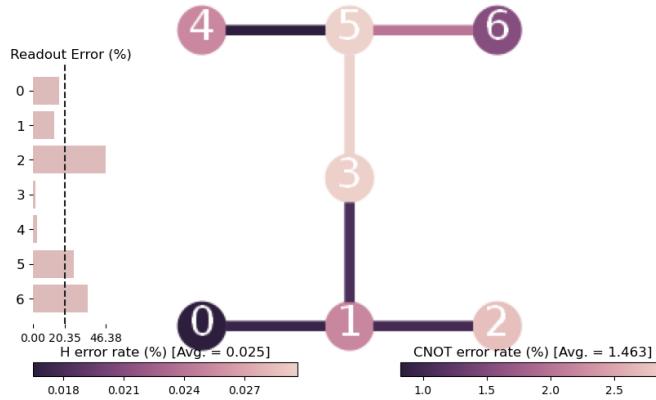


Figura 3.1: error map di Lagos. **a sinistra:** errori percentuali di lettura dei singoli qubit; **in basso a sinistra:** errori percentuali H-gate; **in basso a destra:** errori percentuali CNOT-gate tra qubit adiacenti.

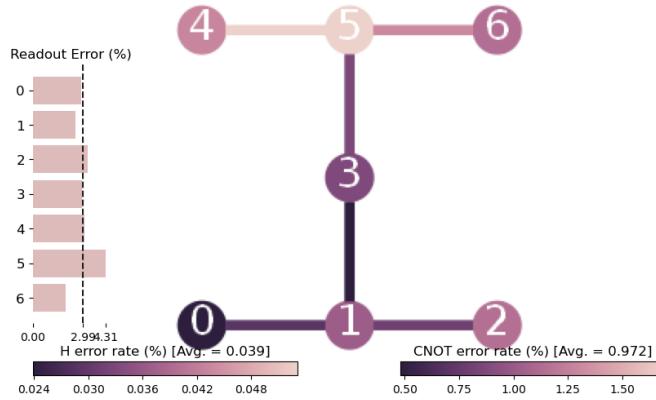


Figura 3.2: error map di Perth. **a sinistra:** errori percentuali di lettura dei singoli qubit; **in basso a sinistra:** errori percentuali H-gate; **in basso a destra:** errori percentuali CNOT-gate tra qubit adiacenti.

3.1.2 Circuito quantistico

Il circuito quantistico è stato progettato tenendo conto della struttura dei provider, sono stati utilizzati gate come Hadamard e CNOT per creare sovrapposizione ed entanglement tra i qubit, ma anche gate di rotazione per introdurre una maggiore variabilità nello stato del circuito. Come mostrato in (Fig. 3.3), si è scelto di misurare il qubit '3' e '4', di conseguenza le operazioni sono state limitate ai qubit dal 3 al 5.

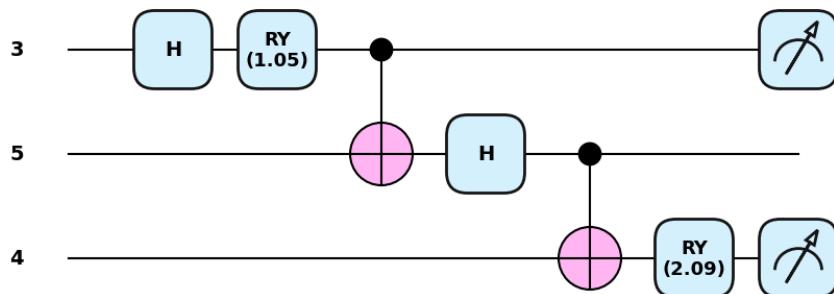


Figura 3.3: Circuito eseguito sui provider Lagos e Perth per la generazione del dataset

Evoluzione dello stato del sistema

Lo stato del sistema prima di essere misurato può essere espresso come

$$|\psi\rangle = |q_3 q_4 q_5\rangle$$

Nello stato iniziale il sistema quantistico si trova nello stato $|\psi\rangle_i = |000\rangle$, mentre dopo l'applicazione dei gate mostrati in (Fig. 3.3) può essere espresso nel seguente modo:

$$\begin{aligned}
 |\psi\rangle_f = & \alpha_- |000\rangle + \beta_- |010\rangle + \\
 & -\beta_- |001\rangle + \alpha_- |011\rangle + \\
 & +\alpha_+ |100\rangle + \beta_+ |110\rangle + \\
 & +\beta_+ |101\rangle - \alpha_+ |111\rangle
 \end{aligned} \tag{3.1}$$

con

$$\alpha_{\pm} = \frac{\sqrt{3} \pm 1}{8} \quad \beta_{\pm} = \sqrt{3} \alpha_{\pm}$$

Dopo la misura dei qubit 3 e 4, lo stato viene compresso in una sovrapposizione di soli due stati del qubit non misurato, ossia il qubit 5, in una forma del tipo:

$$|\psi\rangle_m = |q_3q_4\rangle_m (C_0|0\rangle + C_1|1\rangle)$$

Dove $|q_3q_4\rangle_m$ rappresenta il risultato della misura dei qubit 3 e 4. La probabilità di misurare uno determinato stato è definita come

$$P(|q_3q_4\rangle) = |C_0|^2 + |C_1|^2$$

Per il circuito descritto, le probabilità sono le seguenti:

$$\begin{aligned} P(|00\rangle) = P(|01\rangle) &= \frac{2 - \sqrt{3}}{8} \approx 3.35\% \\ P(|10\rangle) = P(|11\rangle) &= \frac{2 + \sqrt{3}}{8} \approx 46.65\% \end{aligned}$$

Per una trattazione più esaustiva riguardo i passaggi algebrici che hanno portato ai risultati riportati in questa sezione, si invita il lettore a consultare l'appendice A.

Come mostrato in tabella (3.1), le misurazioni di un circuito quantistico eseguite su dispositivi quantistici diversi sono influenzate da quantità di rumore differenti. Il rumore può essere suddiviso in due componenti principali: una componente statistica, dovuta alla natura probabilistica delle misure ed una componente dovuta all'architettura del dispositivo, che riflette le imperfezioni e caratteristiche hardware. Quest'ultima componente è ciò che permette di distinguere e classificare i diversi dispositivi quantistici tra loro.

$ 00\rangle$	$ 01\rangle$	$ 10\rangle$	$ 11\rangle$	label
488	526	4400	4586	0
471	537	4371	4626	0
514	505	4307	4674	0
472	522	4372	4634	0
476	472	4428	4586	0
482	495	4457	4566	1
487	489	4451	4573	1
518	471	4581	4430	1
500	476	4471	4553	1
486	472	4481	4561	1

Tabella 3.1: Esempi di alcuni elementi del dataset prima di essere normalizzati. la label 0 si riferisce alle misure eseguite su Lagos mentre le label 1 alle misure eseguite su Perth.

3.1.3 Normalizzazione

Per ottimizzare il processo di apprendimento delle reti neurali, il dataset è stato normalizzato usando la funzione '*StandardScaler*' appartenente alla libreria '*sklearn.preprocessing*'.

Siano $X = x_1, x_2, \dots, x_n$ i valori di una feature del dataset, lo stesso processo viene poi esteso anche alle altre. La normalizzazione consiste nel trasformare ogni valore x_i in un valore x'_i nel seguente modo:

$$x'_i = \frac{x_i - \mu}{\sigma}$$

in cui μ e σ sono rispettivamente la media e la deviazione standard della feature considerata.

Il dataset così normalizzato ha media e varianza per ogni feature uguali rispettivamente a 0 e 1. La distribuzione delle feature è mostrata in Fig.3.4.

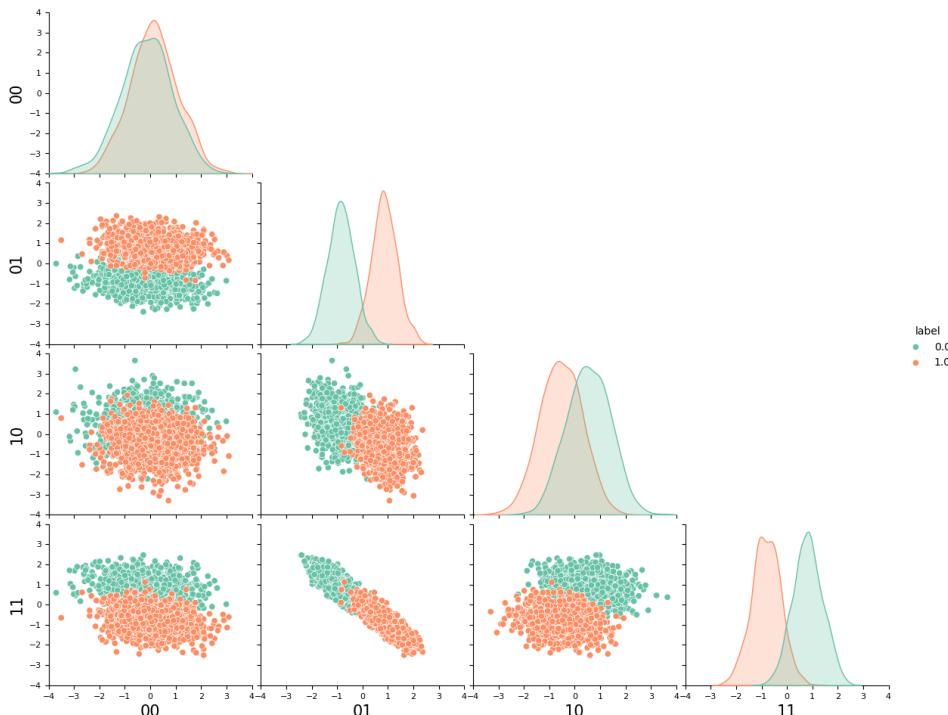


Figura 3.4: Distribuzione delle feature normalizzate. In rosso i dati appartenenti alle misure effettuate su Perth; in blu le misure effettuate su Lagos.

3.2 Descrizione delle reti neurali

Per risolvere il problema di classificazione binaria sono state utilizzate sia una rete neurale classica che quattro varianti di reti neurali ibride. Queste ultime differiscono tra loro esclusivamente per la componente quantistica. In particolare, sono state utilizzate due reti con un layer quantistico composto da 2 qubit e altre due reti con un layer quantistico composto da 3 qubit. Per ciascuna coppia di reti (con 2 o 3 qubit), una rete include l'entanglement tra i qubit, mentre l'altra no.

Prima dell'utilizzo, gli elementi del dataset, già normalizzato, vengono mescolati casualmente. Successivamente, il dataset viene suddiviso in due parti con un rapporto 80/20. La prima parte viene utilizzata per l'addestramento, mentre la seconda viene impiegata per la validazione. Durante ogni epoca di addestramento, i dati vengono ulteriormente suddivisi in batch e rimescolati. Per il dataset di validazione, invece, i dati vengono suddivisi in batch ma senza rimescolamento. Inoltre, durante il processo di addestramento, il learning rate viene ridotto dinamicamente con il trascorrere delle epoche. In particolare il learning rate viene moltiplicato per un fattore $\gamma \in (0, 1)$ dopo un numero specifico di epoche, indicato dal parametro *step*. I parametri utilizzati per l'addestramento sono gli stessi sia per la rete classica che per le reti ibride e sono riportati nella tabella (3.2).

Si noti che l'apprendimento di una rete neurale e di conseguenza le sue prestazioni, possono essere influenzate dall'ordine con cui le vengono presentati i dati, per questo motivo, nei programmi di analisi dati (Appendice B) viene utilizzato lo stesso seme di generazione pseudo-casuale per il rimescolamento degli elementi del dataset.

dimensione batch	40
seme di generazione	25
learning rate	$5 \cdot 10^{-5}$
γ	0.75
step	50
epoche	500

Tabella 3.2: parametri utilizzati per l'addestramento delle reti neurali

Per l'ottimizzazione della rete viene utilizzato l'**Adaptive Moment Estimator (Adam)**[16], uno degli algoritmi di ottimizzazione più utilizzati nel contesto delle reti neurali. La funzione costo utilizzata è invece la **BCEWithLogitsLoss**, ossia una Binary Cross Entropy (sezione 2.4.1) a cui è stata integrata la funzione sigmoide per il valore in input.

3.2.1 Rete neurale classica

La rete neurale classica è composta da un layer di input con quattro neuroni, due layer nascosti con rispettivamente quattro e due neuroni ed un layer di output con un neurone. La struttura è mostrata in Fig.3.5.

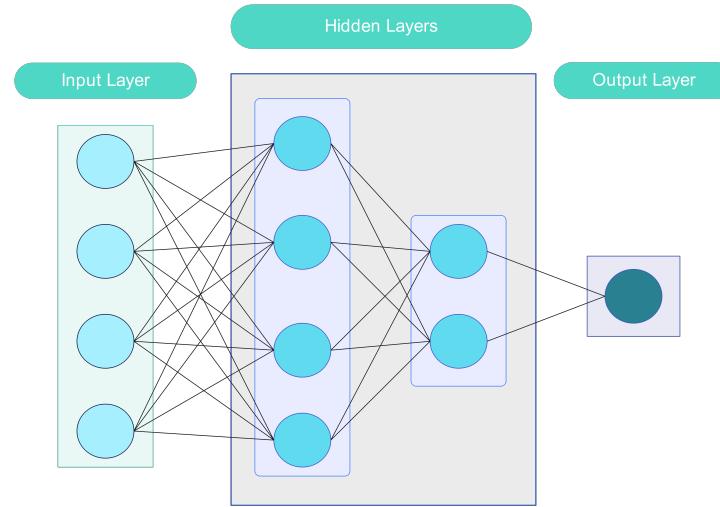


Figura 3.5: Struttura della rete neurale classica utilizzata

I pesi della rete sono stati inizializzati tramite una tecnica proposta da Kaiming He[4]. Questa tecnica consiste nell'inizializzazione dei pesi usando una distribuzione normale con media 0 e varianza $2/n_{in}$, in cui n_{in} corrisponde al numero di neuroni nel layer precedente. I bias invece vengono inizializzati a 0.

La rete ha raggiunto un picco di accuratezza del 94% riducendo la funzione costo da un valore di 0.46 fino ad un valore di 0.33 circa, come mostrato in Fig.3.7 e Fig.3.6.

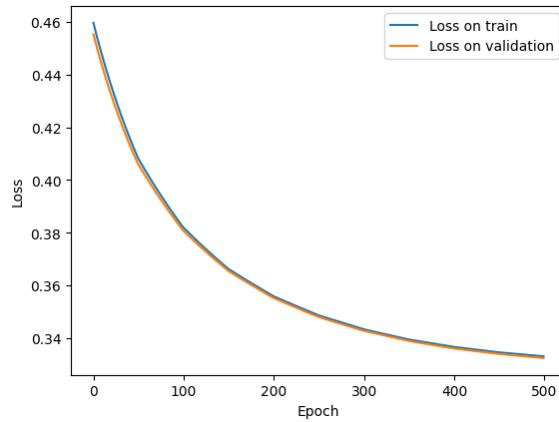


Figura 3.6: Curve di perdita in fase di addestramento (blu) e in fase di validazione (arancione) per la rete classica.

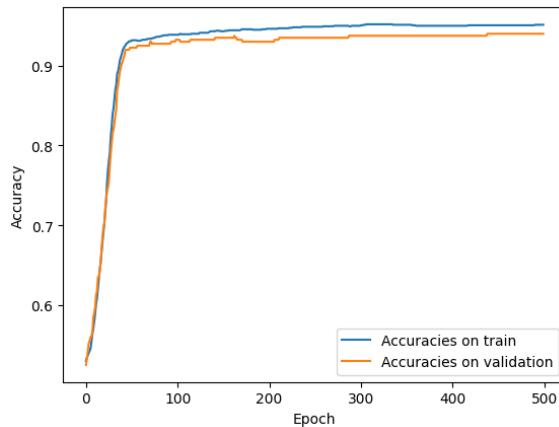


Figura 3.7: Curve dell'accuratezza in fase di addestramento (blu) e in fase di validazione (arancione) per la rete classica.

3.2.2 Rete neurale ibrida

Le quattro reti neurali ibride sono tutte composte da un layer di input classico con 4 neuroni, un layer nascosto classico, un layer quantistico costituito da un circuito quantistico variazionale, un neurone classico utilizzato come layer di output.

I pesi e i bias delle reti neurali ibride vengono inizializzati nello stesso modalità della rete classica. I parametri angolari del layer quantistico invece vengono inizializzati secondo una distribuzione uniforme nell'intervallo $[-\pi, \pi]$.

Circuiti a 2 qubit

Le prime due reti neurali utilizzate sono caratterizzate da un layer quantistico variazionale composto da 2 qubit. I due circuiti sono rispettivamente mostrati in [Fig3.8] e [Fig3.9]

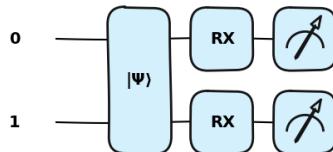


Figura 3.8: Circuito con 2 qubit senza entanglement.

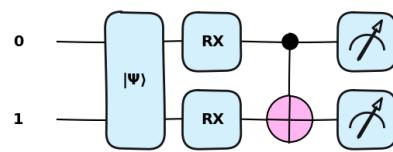


Figura 3.9: Circuito con 2 qubit con entanglement.

Il layer nascosto è composto da 4 neuroni. L'architettura di queste due reti neurali ibride è mostrata in Fig3.10.

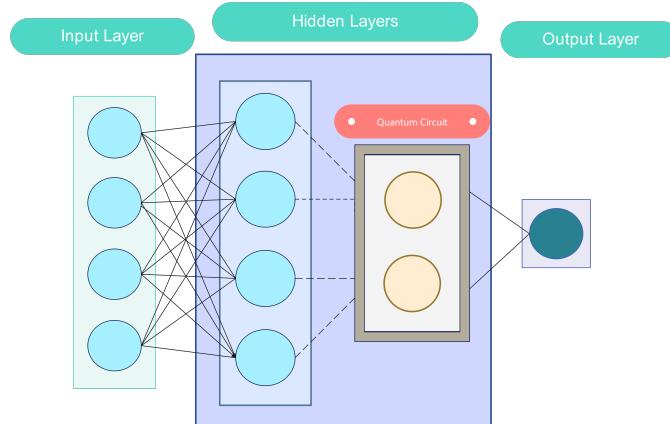


Figura 3.10: Modello della rete ibrida con il layer quantistico composto da 2 qubit

La rete ibrida con il circuito senza entanglement raggiunge un accuratezza sul dataset di validazione del 95.25% e riduce la funzione costo da un valore di 0.69 fino a circa 0.46, come mostrato in Fig.3.11 e Fig3.12.

Anche la rete ibrida con il circuito contenente entanglement tra i qubit raggiunge un'accuratezza sul dataset di validazione pari al 95.25%, riducendo la funzione costo da un valore iniziale di 0.69 ad un valore finale di circa 0.48, come mostrato in Fig3.13] e Fig3.14

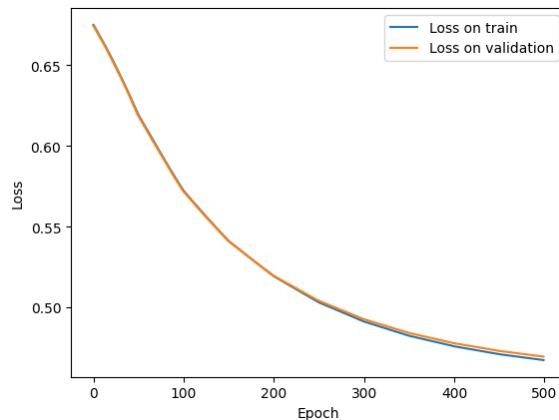


Figura 3.11: curve della funzione costo in fase di addestramento (blu) e in fase di validazione (arancione) per la rete ibrida con layer quantistico con 2 qubit senza entanglement.

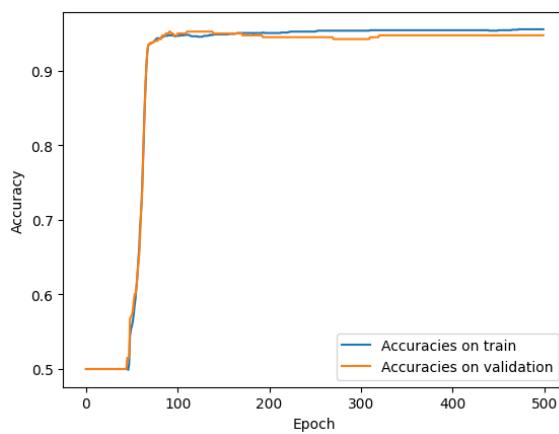


Figura 3.12: curve dell'accuratezza calcolata in fase di addestramento (blu) e in fase di validazione (arancione) per la rete ibrida con layer quantistico con 2 qubit senza entanglement.

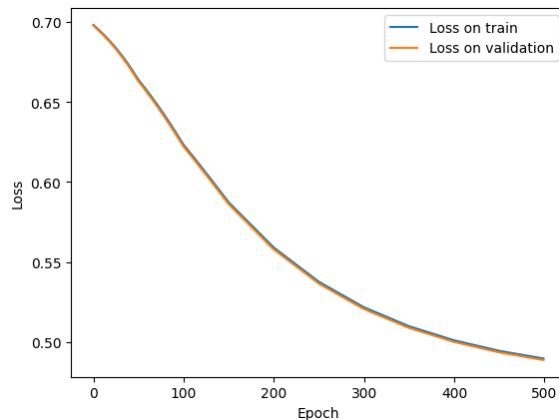


Figura 3.13: curve della funzione costo in fase di addestramento (blu) e in fase di validazione (arancione) per la rete ibrida con layer quantistico con 2 qubit con entanglement.

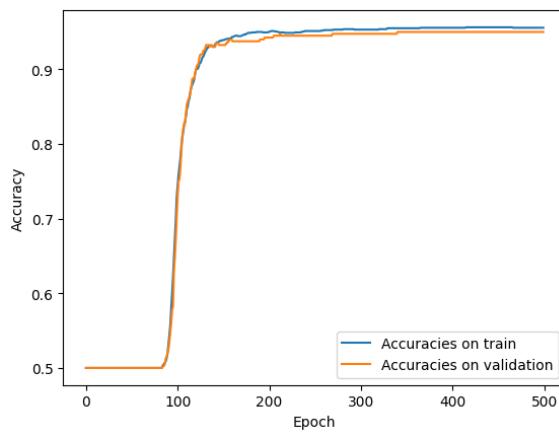


Figura 3.14: curve dell'accuratezza calcolata in fase di addestramento (blu) e in fase di validazione (arancione) per la rete ibrida con layer quantistico con 2 qubit con entanglement.

Circuiti a 3 qubit

Le altre due reti sono composte da circuiti contenenti 3 qubit, definiti come nelle immagini Fig.3.15 e Fig.3.16.

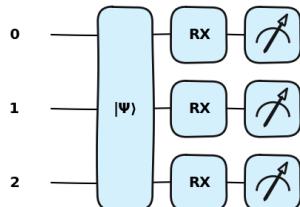


Figura 3.15: Circuito con 3 qubit senza entanglement.

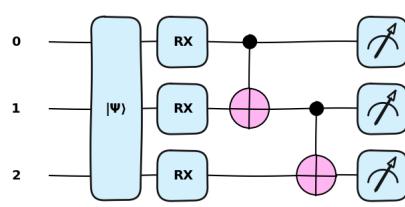


Figura 3.16: Circuito con 3 qubit con entanglement.

Il layer nascosto classico in questo caso contiene 3 neuroni come mostrato in Fig3.17

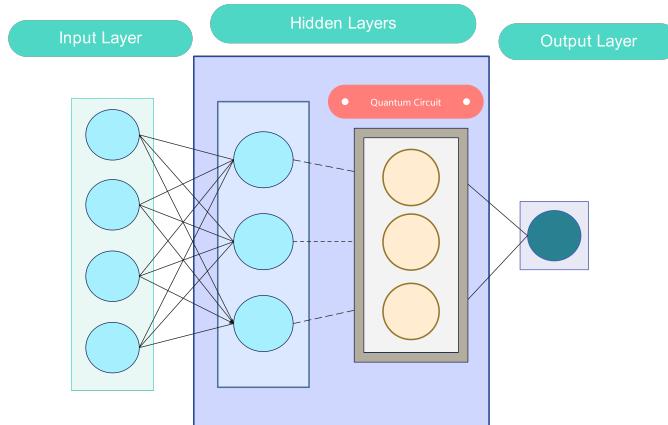


Figura 3.17: Modello della rete ibrida con il layer quantistico composto da 3 qubit.

In questo caso, la rete ibrida con il circuito senza entanglement raggiunge un accuratezza sul dataset di validazione del 95.75%. La funzione costo viene ridotta da un valore iniziale di 0.78 fino ad un valore di circa 0.22, come mostrato in Fig.3.18 e Fig3.19

La rete ibrida con il circuito contenente entanglement mantiene un'accuratezza sul dataset di validazione sempre pari al 95.75%, riducendo la funzione costo da un valore iniziale di 0.78 ad un valore finale di circa 0.28, come mostrato in Fig3.20 e Fig3.21.

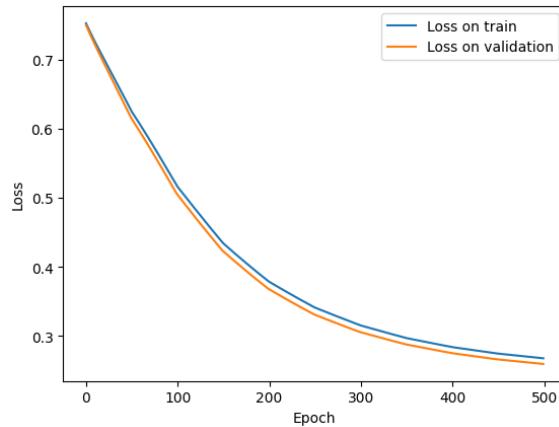


Figura 3.18: curve della funzione costo in fase di addestramento (blu) e in fase di validazione (arancione) per la rete ibrida con layer quantistico con 3 qubit senza entanglement.

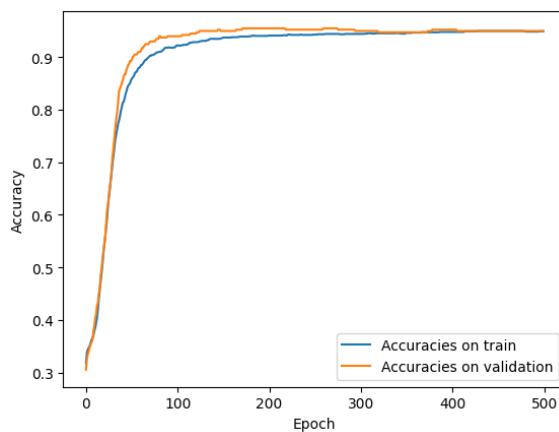


Figura 3.19: curve dell'accuratezza calcolata in fase di addestramento (blu) e in fase di validazione (arancione) per la rete ibrida con layer quantistico con 3 qubit senza entanglement.

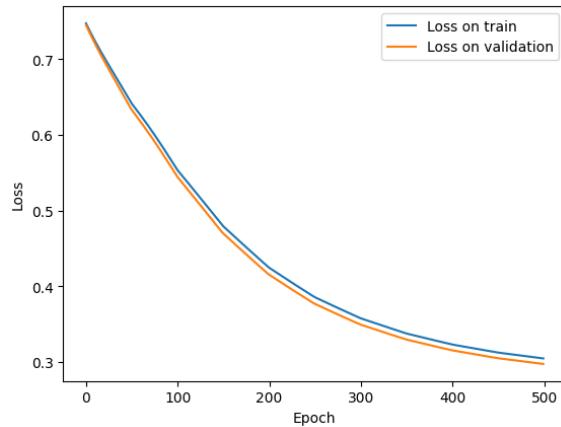


Figura 3.20: curve della funzione costo in fase di addestramento (blu) e in fase di validazione (arancione) per la rete ibrida con layer quantistico con 3 qubit con entanglement.

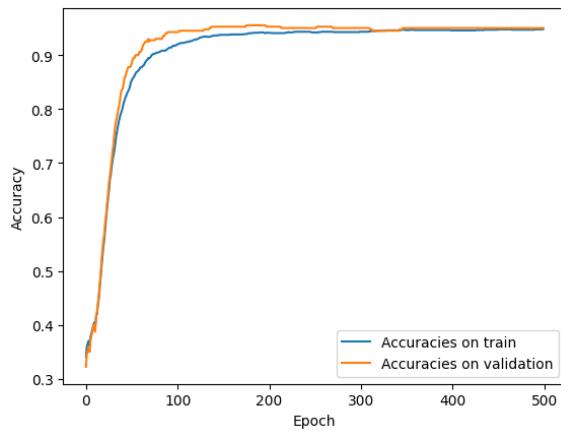


Figura 3.21: curve dell'accuratezza calcolata in fase di addestramento (blu) e in fase di validazione (arancione) per la rete ibrida con layer quantistico con 3 qubit con entanglement.

Capitolo 4

Conclusioni

Il Quantum Machine Learning è un campo in rapido sviluppo che offre potenzialmente maggiori efficienze rispetto al Machine Learning classico in problemi specifici. Questa tesi esplora un caso di studio sulla classificazione binaria del rumore a cui sono soggetti due diversi dispositivi quantistici durante l'esecuzione dello stesso circuito. Nella prima parte, il lavoro si è concentrato sull'ottimizzazione di una rete neurale classica. Successivamente, l'attenzione si è spostata su una rete neurale ibrida, variando il numero di qubit nel layer quantistico e il grado di entanglement tra i qubit.

I risultati delle reti neurali ibride sono confrontabili con i risultati della rete neurale classica. Tutti i modelli utilizzati restituiscono infatti valori di accuratezza tra il 94.00% e il 95.75%. La differenza tra i vari modelli è minima e sulla scala delle dimensioni del dataset non significativa. Per la validazione del modello è stata utilizzato il 20% del dataset originale, ossia un dataset composto da 400 elementi sui 2000 totali. Una differenza massima pari al 1.75% corrispondono ad una differenza di 7 elementi del dataset classificati correttamente.

Una rete neurale ibrida dunque, da questo punto di vista, sembrerebbe offrire solo un lieve miglioramento. Potrebbe essere interessante testare la rete su un dataset di validazione con una dimensione di uno o due ordini di grandezza superiore. In questo caso differenze anche solo del 1% in termini di accuratezza corrisponderebbero a decine o centinaia di dati di differenza.

Un'altra metrica da considerare è la riduzione della funzione di costo. Sebbene il modello classico vanti un valore della funzione costo inferiore rispetto ai modelli ibridi a 2 qubit, bisogna considerare che all'epoca zero il modello classico parte da un valore già inferiore rispetto al valore della funzione costo del modello ibrido. Considerando la differenza tra la funzione costo all'epoca 0 e la funzione costo all'epoca 500 è evidente che i modelli ibridi abbiano una variazione maggiore rispetto al modello classico.

Questa capacità di ridurre in modo più efficiente la funzione perdita, potrebbe avere un significato in termini di confidenza del modello. Un modello ha una confidenza maggiore se le previsioni sono più vicine al valore reale della label. Nel

problema di classificazione trattato la soglia di classificazione utilizzata è quella standard di 0.5. Se ad esempio la rete su un dato appartenente alla classe 1 restituisce un valore di 0.6, verrà classificato correttamente ma avrà un peso diverso sulla funzione costo rispetto ad un dato classificato con un output della rete tendente ad 1. Per verificare questa ipotesi, si potrebbe calcolare l'accuratezza utilizzando una soglia di classificazione inferiore (ad esempio 0.3) e vedere se effettivamente le reti ibride abbiano una confidenza superiore rispetto alla rete classica.

Infine, confrontando modelli ibridi con la presenza o meno di entanglement non si sono riscontrate differenze significative se non in termini di costi computazionali. La complicazione dovuta all'inserimento dell'entanglement tra i qubit infatti ha sistematicamente aumentato i tempi di addestramento delle reti. Il motivo per cui l'entanglement non abbia avuto un ruolo significativo potrebbe trovarsi nella scarsa difficoltà di classificazione del dataset utilizzato. Inoltre l'entanglement potrebbe avere un impatto maggiore in casi in cui il layer quantistico sia composto da un maggior numero di qubit.

Per concludere, lo studio potrebbe essere migliorato ad esempio aumentando la dimensione del dataset di validazione, utilizzando dataset multi-classe o dataset binari con un numero maggiore di feature distribuite in modo tale da aumentare il livello di difficoltà della classificazione. Nonostante ciò lo studio dimostra che l'implementazione di circuiti quantistici come layer in reti neurali feedforward per risolvere problemi di classificazione non solo è possibile, ma risulta competitivo con modelli di reti neurali classici. Questa tesi potrebbe contribuire allo sviluppo di algoritmi ibridi per la risoluzione di problemi di classificazione più complessi.

Appendice A

Evoluzione dello stato del sistema quantistico

In questa sezione viene mostrata l’evoluzione del circuito utilizzato per la generazione del dataset di cui si è discusso nella sottosezione 3.1.2.

Lo stato iniziale di un sistema a 7 qubit può essere scritto come:

$$|\psi\rangle_i = |0000000\rangle$$

Tuttavia, avendo applicato dei gate solo ai qubit 3,4 e 5, è possibile alleggerire la notazione e considerare solo i tre qubit rilevanti ai fini dell’evoluzione dello stato quantistico.

Viene brevemente mostrato come variano gli stati base di singolo qubit $|0\rangle$ e $|1\rangle$ in seguito all’applicazione dei gate H , $CNOT$, e $R_Y(\theta)$, descritti nella sezione 1.5.

H-Gate

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Rotazione Y

$$R_Y(\theta)|0\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)|1\rangle$$

$$R_Y(\theta)|1\rangle = -\sin\left(\frac{\theta}{2}\right)|0\rangle + \cos\left(\frac{\theta}{2}\right)|1\rangle$$

CNOT-Gate

$$CNOT|00\rangle = |00\rangle$$

$$CNOT|10\rangle = |11\rangle$$

Lo stato generale del sistema viene espresso come $|\psi\rangle = |q_3q_4q_5\rangle$.

Stato iniziale:

$$|\psi\rangle_i = |000\rangle$$

H-Gate su 3:

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|000\rangle + |100\rangle)$$

RY(60°) su 3:

$$|\psi\rangle = \frac{\sqrt{3}-1}{2\sqrt{2}}|000\rangle + \frac{\sqrt{3}+1}{2\sqrt{2}}|100\rangle$$

CNOT-Gate da 3 a 5:

$$|\psi\rangle = \frac{\sqrt{3}-1}{2\sqrt{2}}|000\rangle + \frac{\sqrt{3}+1}{2\sqrt{2}}|101\rangle$$

H-Gate su 5:

$$|\psi\rangle = \frac{\sqrt{3}-1}{4}|000\rangle + \frac{\sqrt{3}-1}{4}|001\rangle + \frac{\sqrt{3}+1}{4}|100\rangle - \frac{\sqrt{3}+1}{4}|101\rangle$$

CNOT-Gate da 5 a 4:

$$\begin{aligned} |\psi\rangle_f = & \frac{\sqrt{3}-1}{8}|000\rangle + \frac{(\sqrt{3}-1)\sqrt{3}}{8}|010\rangle - \frac{(\sqrt{3}-1)\sqrt{3}}{8}|001\rangle + \frac{\sqrt{3}-1}{8}|011\rangle + \\ & + \frac{\sqrt{3}+1}{8}|100\rangle + \frac{(\sqrt{3}+1)\sqrt{3}}{8}|110\rangle + \frac{(\sqrt{3}+1)\sqrt{3}}{8}|101\rangle - \frac{\sqrt{3}+1}{8}|111\rangle \end{aligned}$$

Lo stato $|\psi\rangle_f$ corrisponde allo stato mostrato nell'equazione (3.1)

Appendice B

Programmi per analisi dati

B.1 Dataset

```
# Function to create a dataset and a labels

def make_dataset(num_samples, circuit, backend1, backend2, shots, measured_qubits):

    # preallocate a dataset with dimention [2num_samples] x [ 2^(measured_qubit) ]
    dataset = np.zeros((2*num_samples, 2**measured_qubits))

    # generate samples calling the function 'make_counts_array' using backend1 as input
    data1 = make_counts_array( num_samples, circuit, backend1, shots, measured_qubits)

    # generate samples calling the function 'make_counts_array' using backend2 as input
    data2 = make_counts_array( num_samples, circuit, backend2, shots, measured_qubits)

    # Concatenate data1 and data2
    dataset = np.concatenate([data1,data2])

    # create the labels: 0 for samples of backend1, 1 for samples of backend2
    labels = np.concatenate([np.zeros(num_samples), np.ones(num_samples)])

    # return the dataset and the labels
    return dataset, labels
```

Figura B.1: Funzione utilizzata per generare il dataset.

```
# Function for create an array with counts get by 'run_circuit'

def make_counts_array( num_samples , circuit , backend, shots, measured_qubits):

    # preallocate a matrix with dimention [num_samples] x [ 2^(measured_qubit) ]
    data = np.zeros((num_samples , 2**measured_qubits))

    for i in range( num_samples):
        # preallocate the array for the counts
        counts_array = np.zeros(2**measured_qubits)
        counts = run_circuit(circuit, backend, shots, measured_qubits)

        # fill the count_array
        for j, key in enumerate(sorted(counts.keys())):
            counts_array[j] = counts[key]
        data[i] = counts_array

    return data
```

Figura B.2: Funzione per creare un vettore di conteggi.

```

# Function for run our quantum circuit on a backend
def run_circuit(circuit, backend, shots, measured_qubits):
    # transpile the circuit for the backend
    | transpiled_circuit = transpile(circuit, backend)

    # run the transpiled circuit on the backend
    | job = backend.run(transpiled_circuit, shots = shots)

    # get the result counts
    | counts = job.result().get_counts()

    # define all_possible_keys in order to convert natural number in range 2^(number of measured qubit) in binary number
    # example measured_qubit = 2; [ 1,2,3,4 ] --> [ 00 , 01 , 10 , 11 ]
    all_possible_keys = [format(i, '0'+str(measured_qubits)+'b') for i in range(2**measured_qubits)]

    # let's normalize the counts in order to have values in range [0,1]. Additionally, let's add the keys with no counts, assigning them a value of 0
    for key in all_possible_keys:
        if key not in counts:
            counts[key] = 0.0
            counts[key] = counts[key]/shots
    sorted_counts = dict(sorted(counts.items()))
    return sorted_counts

```

Figura B.3: Funzione per eseguire un circuito su un provider (backend) quantistico.

```

#Generate the dataset and the labels. Save them in 2 different file.npy
Backend_list= {
    'Lagos' : FakeLagosV2(),
    'Perth' : FakePerth(),
}

keys = list(Backend_list.keys())
for combination in combinations(keys,2):
    key1, key2 = combination
    file_name = 'Dati/dataset_{}x{}_3q'.format(key1, key2)
    labels_name = 'Dati/labels_{}x{}_3q'.format(key1, key2)
    dataset, labels = make_dataset(num_samples, qc, Backend_list[key1], Backend_list[key2], shots, measured_qubits)

    normalized_dataset = StandardScaler().fit_transform(dataset)
    np.save(file_name, normalized_dataset)
    np.save(labels_name, labels)

```

Figura B.4: Cella contenente l'algoritmo per la creazione di un file contenente i dati e di un file contenente le label

B.2 Addestramento e validazione delle reti neurali

In questa sezione sono contenuti i codici dei programmi utilizzati per la fase di addestramento e validazione delle reti neurali sia classiche che ibride.

```

def evaluate_model( model , loader , loss_fn ):

    all_outputs = []
    all_labels = []

    running_loss = 0
    with torch.no_grad():
        for id, (data, label) in enumerate(loader):
            output = model(data)
            loss = loss_fn(output,label)
            running_loss += loss.item()

            prob = torch.sigmoid(output)

            all_outputs.append(prob)
            all_labels.append(label)

    losses = running_loss/(id+1)
    all_outputs_tensor = torch.cat(all_outputs)
    all_labels_tensor = torch.cat(all_labels)

    predicted = torch.round(all_outputs_tensor)
    accuracy = accuracy_score(all_labels_tensor.numpy(), predicted.numpy())

    return accuracy , losses

```

Figura B.5: Addestramento della rete neurale

```

# This is the actual model training and validation.

train_losses = []
val_losses = []

train_accuracies = []
val_accuracies = []

# training the model
for i in range(epochs):

    train_running_loss = 0

    model.train() # Set the model to training mode
    for id_batch, (data, label) in enumerate(train_loader):
        output = model(data)
        loss = lossFunction(output, label)
        loss.backward()
        opt.step()
        opt.zero_grad(set_to_none=True)

        train_running_loss += loss.item()

    train_losses.append(train_running_loss/(id_batch+1))
    scheduler.step()

    model.eval()
    train_acc , train_loss = evaluate_model( model , train_loader, lossFunction)
    val_acc , val_loss = evaluate_model( model , val_loader, lossFunction)

    train_accuracies.append(train_acc)
    val_accuracies.append(val_acc)
    val_losses.append(val_loss)

```

Figura B.6: Funzione di validazione delle reti neurali

Bibliografia

- [1] PyTorch documentation. BCELoss — PyTorch 2.4 documentation.
- [2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, November 2016.
- [3] Otfried Gühne and Géza Tóth. Entanglement detection. *Physics Reports*, 474(1-6):1–75, 2009.
- [4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [5] IBM. Che cos’è il quantum computing?
<https://www.ibm.com/it-it/topics/quantum-computing>.
- [6] IBM. Cos’è un qubit?
<https://www.ibm.com/it-it/topics/qubit>.
- [7] IBM. Cos’è l’Intelligenza Artificiale (AI)? | IBM, October 2021.
- [8] IBM. Che cos’è il machine learning (ML)? | IBM, May 2024.
- [9] IBM. Cosa è l’apprendimento supervisionato? | IBM, May 2024.
- [10] IBM. Cos’è una rete neurale? | IBM, May 2024.
- [11] Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P. Orlando, Simon Gustavsson, and William D. Oliver. A Quantum Engineer’s Guide to Superconducting Qubits, July 2021.
- [12] Thaddeus D Ladd, Fedor Jelezko, Raymond Laflamme, Yasunobu Nakamura, Christopher Monroe, and Jeremy Lloyd O’Brien. Quantum computers. *nature*, 464(7285):45–53, 2010.
- [13] Ali Masri. An Intuitive Explanation to Dropout, July 2019.
- [14] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

- [15] NVIDIA. NVIDIA Blog: Supervised Vs. Unsupervised Learning, August 2018.
- [16] Pytorch. Adam — PyTorch 2.4 documentation.
- [17] Unathi Skosana and Mark Tame. Demonstration of Shor's factoring algorithm for $N = 21$ on IBM quantum processors. *Scientific Reports*, 11(1), August 2021.
- [18] Zhi-Hua Zhou. *Machine Learning*. Springer Singapore, Singapore, 2021.