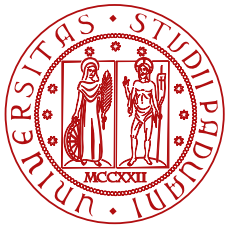


Università
degli Studi
di Padova

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Corso di Laurea Triennale in Ingegneria Informatica

Reti neurali e computazione quantistica

Relatore

Luca Schenato

Università di Padova

Laureando

Arjun Jassal

1219611

Anno Accademico 2021/2022

Data di laurea: 14/11/2022

A coloro che ci sono stati,
ci sono
e ci saranno

Abstract

Le reti neurali da decenni cercano di imitare il funzionamento del cervello umano tramite modelli matematici ma recentemente si è giunti ad un limite tecnologico ed energetico per cui non è possibile aumentare la complessità senza notevoli costi energetici e computazionali. Una risposta potrebbe trovarsi nel campo della meccanica quantistica che grazie ai suoi concetti potrebbe replicare più fedelmente il funzionamento cerebrale, il quale si è scoperto che funziona in maniera parallela e possiede meccanismi a cui si potrebbe trovare una spiegazione con la fisica quantistica. In questa tesi andrò ad evidenziare come una soluzione per migliorare le prestazioni delle reti neurali sia l'introduzione della computazione quantistica in seguito al progresso nella ricerca dei bit quantistici.

Indice

1

Reti neurali

Il cervello è uno degli organi più importanti per il mondo animale ed è una macchina complessa formata da neuroni che interagiscono tra di loro grazie alle sinapsi. Nel 1943 Warren McCulloch e Walter Pitts[32] diedero inizio agli studi del cervello umano, volti ad astrarre il cervello come una macchina elettrica modellizzabile come un circuito di cui il neurone fosse l'unità logica.
Il cervello umano è in grado di risolvere problemi visivi e linguistici in pochi millisecondi, in molto meno tempo di un computer. Dunque un elaboratore che può imitare il cervello umano ha un enorme potenziale in certe applicazioni: così sono state ideate le reti neurali artificiali(ANN), un modello matematico per rappresentare l'interconnessione di neuroni.
Le ANN sono ispirate alla biologia e utilizzano un'elaborazione dei dati analoghe ai neuroni: imparano dalle precedenti esperienze, generalizzano per nuovi input e sono in grado di prendere una decisione. Così come i neuroni si attivano ed inviano impulsi superata una determinata soglia, anche le reti neurali artificiali utilizzano lo stesso funzionamento, dove questa soglia è determinata da una funzione matematica applicata sugli input del neurone.
Al giorno d'oggi esistono reti neurali con molti strati e vengono usate in machine learning[14] e in svariati campi di ricerca e settori industriali e grazie alla loro capacità di lavorare in parallelo un comune esempio di uso è il riconoscimento di immagini.

1.1 Neuroni biologici

Il neurone è l'unità cellulare(figura 1.1) che costituisce il tessuto nervoso ed è in grado di ricevere, elaborare ed inviare impulsi nervosi sotto forma di segnali elettrici.
Ci si può trovare in due stati: a riposo o attivo, in quest'ultimo stato esso produce un impulso elettrico trasportato dall'assone fino alle sinapsi dove viene provocato un rilascio di sostanze chimiche che possono essere eccitanti o inibitorie così da gestire l'attivazione del neurone successivo. Dunque l'eccitamento di un neurone dipende dalla somma pesata degli ingressi in input, la quale superata una certa soglia provvede all'attivazione.
Ogni neurone di per se non è veloce ma il complessivo sistema nervoso è formato da circa 10^{10} neuroni che lavorano in modo parallelo permettendo un'elevata capacità di elaborazione.
Queste cellule sono costituite dai seguenti componenti:

- **Dendriti:** ramificazioni di fibre dal soma che costituiscono le linee di input e permettono al neurone di ricevere i segnali dai neuroni connessi. Inoltre si occupa di rilasciare le sostanze chimiche per eccitare o inibire il neurone in base a ciò che riceve.
- **Soma:** corpo principale del neurone dove avviene l'elaborazione dei segnali ricevuti.
- **Assone:** ramificazione principale collegata al soma che trasporta il segnale di output alle sinapsi.

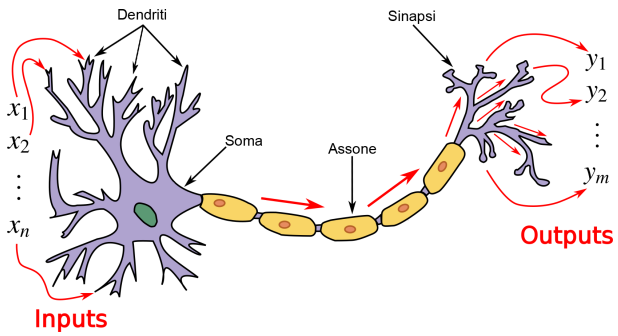


Figura 1.1: Neurone biologico

1.2 Neuroni artificiali

Il neurone artificiale non è altro che una funzione matematica concepita come modello di funzionamento dei neuroni biologici, infatti svolge le attività di ricezione, elaborazione e trasmissione dei segnali ricevuti.

Il primo modello di neurone artificiale(figura 1.2) venne ideato da McCulloch e Pitts[32] e prevede una serie di input $x_i \in \{0, 1\}$ con pesi $w_i \in \{-1, +1\}$. Questi input vengono moltiplicati per i loro pesi e producono una somma pesata che verrà successivamente usata come argomento nella funzione di attivazione.
Dunque avremo la somma pesata $g = \sum_i w_i x_i$ ed avremo un output al neurone dato dalla funzione di attivazione $y = f(g)$ che determina l'attivazione del neurone in base al valore ottenuto, solitamente questa funzione è una funzione a gradino con $y \in \{0, 1\}$.

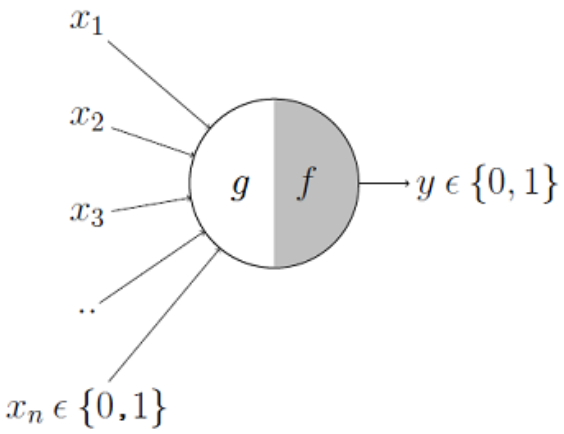


Figura 1.2: Neurone artificiale

1.3 Percettroni

Successivamente al neurone di McCulloch e Pitts, un miglioramento venne effettuato da Frank Rosenblatt nel 1958[8] che propose il percettrone come nuovo modello. Questo nuovo elemento è più flessibile rispetto al neurone artificiale in quanto i valori di input, pesi e output non sono più ristretti al dominio binario.
Lo stato di eccitazione z viene calcolato con la formula $z = \sum_i w_i x_i + b$, dove b è il valore di bias, una costante che non dipende dall'input cosicchè il risultato sia in parte indipendente dall'output dei precedenti neuroni.
Per capire meglio il funzionamento, possiamo immaginare il percettrone che deve decidere tra due scelte, ovvero 0 oppure 1, gli input hanno un peso diverso e dove $w_i x_i < 0$ avremo una scelta più verso lo 0 mentre se $w_i x_i > 0$ più verso 1. La scelta finale ricadrà sulla funzione di attivazione e sul bias b che formano il valore di soglia da superare per l'eccitamento del neurone.

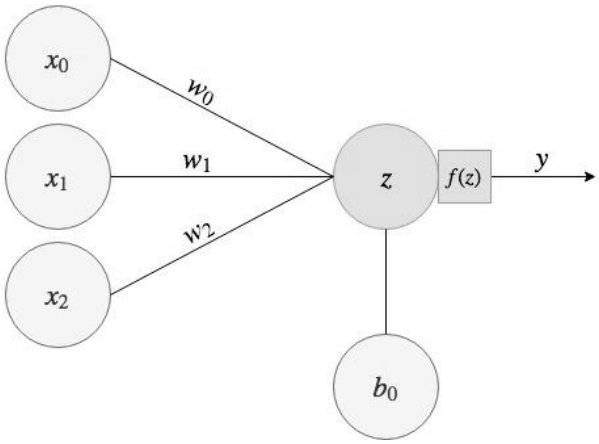
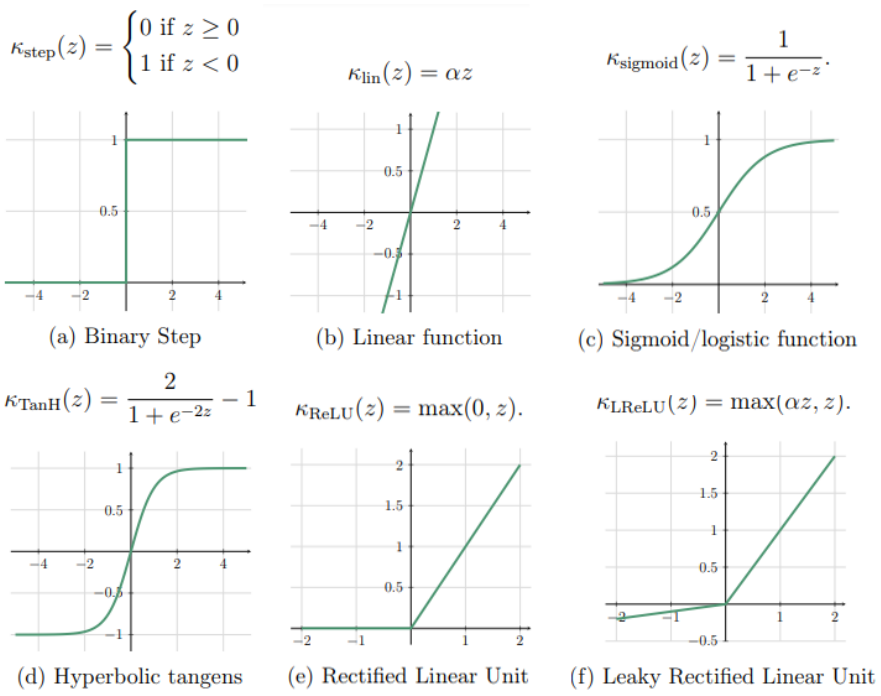


Figura 1.3: Percettrone

L'output dunque assume il valore $y = f(z)$ con f funzione di attivazione che non è più la funzione a gradino introdotta da McCulloch e Pitts perché è molto sensibile ai cambiamenti: infatti in caso di una piccola modifica ai pesi o del bias porterà ad un risultato completamente diverso nella rete finale a causa dei risultati binari. Infatti la scelta di una funzione di attivazione va fatta con attenzione in quanto è molto rilevante nell'addestramento della rete.
Le funzioni di attivazione più comuni sono riassunte in seguito:



(a) Binary Step (b) Linear function (c) Sigmoid/logistic function

(d) Hyperbolic tangens (e) Rectified Linear Unit (f) Leaky Rectified Linear Unit

Figura 1.4: Funzioni di attivazione

1.4 Struttura delle ANN

Una rete neurale artificiale si ispira al sistema nervoso biologico del mondo animale ed è un sistema di neuroni artificiali(chiamati nodi) in più strati(o layers) interconnessi tra di loro. È possibile riscontrare un limite computazionale di una rete con un singolo strato di percettroni in quanto non è in grado di risolvere problemi non lineari come lo XOR, dunque si sono introdotti più layers così da poter risolvere problemi più complessi. Il primo layer è lo strato dove arriva l'input iniziale, infatti prende il nome di input layer, mentre l'ultimo layer prenderà il nome di output layer e rispettivamente sono input e output dell'intera rete neurale. In mezzo a questo si possono trovare uno o più strati e questi prendono il nome di hidden layers. Questa struttura prende il nome di *Multilayer Perceptron*.
Ogni neurone riceve un segnale dai nodi a cui è connesso, che non sono altro che numeri reali, elabora e trasmette a sua volta un segnale ai neuroni a cui è connesso. Il segnale output viene calcolato con una funzione non lineare sulla somma degli input del nodo. Le interconnessioni tra nodi sono chiamati archi e ognuno di essi ha un valore, chiamato peso, che stabilisce la potenza del segnale in quella connessione.
L'architettura più semplice che si possa trovare è la *feed-forward* dove i neuroni di un certo layer prendono come input l'output di quello precedente e non ci sono cicli ma le informazioni si muovono solo in una direzione: dall'input iniziale a quello finale.

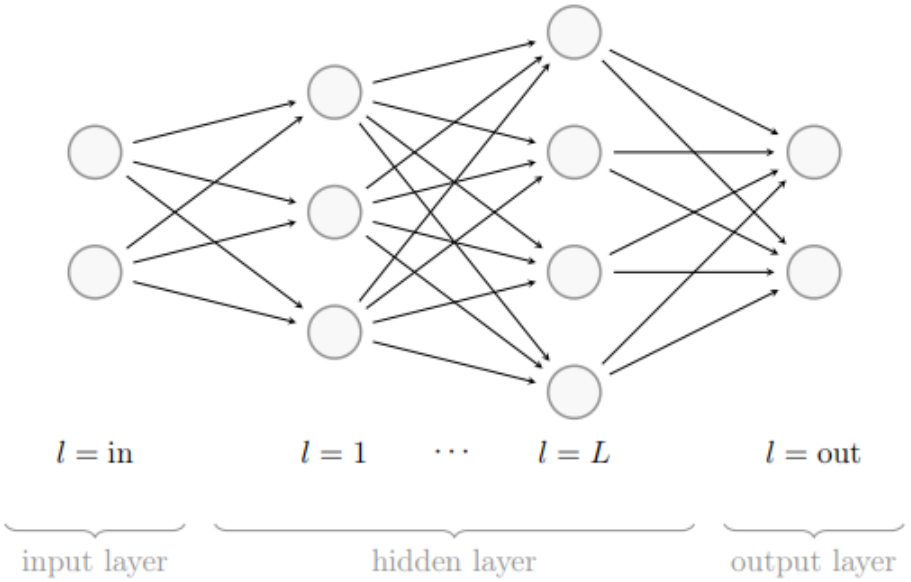


Figura 1.5: Struttura di una rete neurale con L hidden layers

1.5 Addestramento

Il processo di addestramento prevede che la rete neurale impari, nel caso di *supervised learning*, esaminando esempi contenenti un input e il relativo risultato: ovvero mostrando alla rete che ad un certo input corrisponde un determinato output. È possibile quindi calcolare un errore tra ciò che la rete ha prodotto e quello che è il risultato ottimale e questo è fatto dalla loss function, il cui obiettivo è trovare l'errore e successivamente cercare di trovare il valore ottimale dei pesi: infatti la rete viene continuamente aggiornata ad ogni step in base al valore fornito dalla loss function e vengono corretti i pesi in modo da minimizzare l'errore.
Fornendo un traning set di lunghezza N alla rete lo dividiamo nel traning set supervised S formate da coppie (pattern, label), mentre i restanti $N - S$ li utilizziamo per il unsupervised validation per poter controllare se i valori trovati durante l'addestramento siano validi. La funzione più comune utilizzata è la Mean Squared Error(MSE), chiamato anche errore quadratico medio o costo quadratico. Consideriamo S il numero di coppie (pattern, label) fornite alla rete, $\varepsilon_{w,b}(x_i)$ output della rete con valori dei pesi w e di bias b e l'output desiderato y_i, la loss function[11] dell'addestramento supervisionato diventa

$$L_{SV}(w,b) = \frac{1}{S}\sum_{i=1}^{S} ||y_i - \varepsilon_{w,b}(x_i)||^2$$

Se questa funziona diventa sempre più piccola durante il processo allora sappiamo di aver trovato i pesi e i bias corretti per il nostro obiettivo. È altrettanto importante anche la loss function dell'addestramento *unsupervised* in quanto fornisce uno strumento per poter verificare che i valori trovati siano validi

$$L_{USV}(w,b) = \frac{1}{N-S}\sum_{i=S+1}^{N} ||y_i - \varepsilon_{w,b}(x_i)||^2$$

Quest'ultima non può essere trascurata perché l'addestramento in alcuni casi ci fornisce dei valori di pesi e bias che permettono alla rete di performare molto bene nei dati di training set però non è in grado di generalizzare su dati mai visti prima. Questo problema prende il nome di *overfitting* e fa in modo che la rete impari a memoria i dati usati per allenarsi e riesca a riconoscere benissimo solamente quelli. È possibile evitare questo fenomeno fermando l'addestramento laddove L_{USV} inizia ad aumentare, dopo essere sceso, nonostante L_{SV} diventi più piccolo.

L'aggiornamento dei pesi avviene in due fasi:

- **Forward-propagation:** la propagazione delle informazioni nella rete avviene in avanti, dall'input all'output.
  Per esempio in una MLP a 3 strati (x, y, z), l'informazione si propaga al nodo z_k nel seguente modo:

$$z_k = f(\sum_{j=1} w_{jk} y_j + w_{0k}) = f(\sum_{j=1} w_{jk} f(\sum_{i=1} w_{ij} x_i + w_{0j}) + w_{0k})$$

- **Backward-propagation:** la propagazione delle informazioni in questo caso avviene dall'output all'input. Si procede all'indietro sistemando i valori di pesi e bias grazie alla loss function.

In questa maniera si ottengono i valori ottimali di pesi e bias dopo averli fissati casualmente all'inizio.
Un altro metodo per trovare i pesi ottimi è la discesa del gradiente che consiste nel trovare il valore minimo locale cambiando i valori dei parametri w_t e b_t nella direzione del gradiente della loss function così da ottenere

$$w_{t+1} = w_t - \eta \nabla_{w_t} L_{SV}(w_t, b_t)$$

$$b_{t+1} = b_t - \eta_{b_t} L_{SV}(w_t, b_t)$$

dove t è lo step di addestramento e η è il *learning rate*, ovvero un valore che decide quanto è grande lo step preso dal gradiente. Un learning rate troppo grande permette alla rete di imparare prima ma fa in modo che i valori di pesi e bias saltino troppo da un valore all'altro creando un'oscillazione oppure una convergenza ad un valore non ottimale, infatti è preferibile un learning rate basso così la rete può imparare con calma e raggiungere i valori ottimali in un tempo più lungo però affidabili piuttosto che in breve tempo e sbagliati.

2

Quantum bit

In questo capitolo introduciamo il concetto di informazione quantistica confrontando il qubit, l'unità di informazione quantistica, alla sua controparte nella computazione classica, il bit.
Il bit è l'unità base dell'informazione nella computazione classica e può rappresentare due stati $(0, 1)$ con i quali i dispositivi attuali possono processare dati. Invece il qubit, è l'unità di informazione quantistica e consiste nella sovrapposizione dei due stati ed è usata dalle recenti tecnologie di sistemi non classici.

2.1 Concetti della meccanica quantistica

I sistemi quantistici sono caratterizzati dalla funzione onda ψ nello spazio di Hilbert[29]. Questo spazio è formato da un insieme di stati $|\phi_i\rangle$ che formano una base e il sistema è descritto dallo stato quantistico

$$|\psi\rangle = \sum_i c_i |\phi_i\rangle$$

dove $|\psi\rangle$ è la superposizione lineare della base di stati $|\phi_i\rangle$.
Mentre a livello macroscopico i sistemi possono trovarsi solo in un determinato stato, è abbastanza controintuitivo pensare al fatto che nel mondo microscopico i sistemi siano descritti da $|\psi\rangle$, che non è altro che una superposizione che racchiude tutti gli stati possibili, in altre parole il sistema non è altro che una sovrapposizione di tutti gli stati in cui può essere.

Fondamentali anche i concetti di coerenza e decoerenza: un sistema è detto coerente se si trova in una superposizione dei suoi stati base, se un sistema in superposizione interagisce con l'ambiente esterno, la sua superposizione è distrutta e si ha un sistema decoerente descritto da dalla sua funzione onda ψ.
I coefficienti c_i sono le ampiezze degli stati e $|c_i|^2$ è la probabilità che $|\psi\rangle$ collassi nello stato $|\phi_i\rangle$. La somma delle ampiezze deve essere uguale a 1, infatti:

$$\sum_i |c_i|^2 = 1$$

Per poter visualizzare meglio la superposizione, si può pensare al lancio di una moneta: finché non atterra non si può sapere in che stato sia, se è testa, croce o una combinazione di entrambe. Per poter effettivamente ottenere la misura dello stato della moneta bisogna interrompere la sua rotazione, dunque interagendoci rompendo la sua superposizione.
Prendiamo ad esempio[5] il lancio di una moneta truccata con probabilità che esca testa($|1\rangle$) il doppio della probabilità che esca croce($|0\rangle$), dunque i due stati possibili sono rappresentati da testa e croce.

$$\begin{cases} P_{testa} = \frac{2}{3} \\ P_{croce} = \frac{1}{3} \end{cases}$$

$$P_{testa} + P_{croce} = 1$$

$$\rightarrow P_{croce} = \frac{1}{3} = \alpha^2, P_{testa} = \frac{2}{3} = \beta^2$$

$$\rightarrow \alpha = \sqrt{\frac{1}{3}}, \beta = \sqrt{\frac{2}{3}}$$

Lo stato della moneta sarà rappresentato come

$$|moneta\rangle = \sqrt{\frac{1}{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$$

Dunque gli stati quantistici possono essere rappresentati come vettori nello spazio di Hilbert, questo forma viene chiamata stato puro. Generalmente i sistemi vengono descritti tramite una combinazione di questi stati, forma chiamata stato misto. Lo stato misto ρ di m stati puri $|\psi_i\rangle$ viene tipicamente descritto come

$$\rho = \sum_i^m p_i |\psi_i\rangle\langle\psi_i|$$

con $\sum_i p_i = 1$.

Assieme alla superposizione, l'entanglement è un altro concetto fondamentale della meccanica quantistica. Questa proprietà necessita di due o più particelle, infatti presi due qubit, questi sono detti legati(o "entangled") quando dopo aver interagito tra di loro non possono più essere descritti singolarmente.
Vediamo un semplice esempio presi due bit in stato puro:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Questo stato descrive la superposizione e abbiamo due casi: dopo la misurazione abbiamo con probabilità 50% che entrambi i qubit si trovino nello stato $|0\rangle$ oppure nello stato $|1\rangle$, dunque misurando un solo qubit, possiamo conoscere lo stato dell'altro senza effettivamente misurarlo.

2.2 Bit e qubit

"Mentre il bit classico è immaginabile come una moneta che, una volta lanciata, cadrà a terra mostrando inesorabilmente una delle due facce, il qubit è immaginabile come una moneta che, una volta lanciata, cadrà a terra continuando a ruotare su sé stessa senza arrestarsi finché qualcuno non ne blocchi la rotazione, obbligandola a mostrare una delle sue facce."[22]
Il classico bit può rappresentare due stati $(0, 1)$ ed è usato per rappresentare informazioni nella computazione classica, per esempio il numero 1 in un sistema binario a 8 bit viene rappresentato come 00000001, il numero 2 come 00000010 e così via; per gli umani questo linguaggio è abbastanza difficile da comprendere, dunque l'elaboratore si occupa di tradurre queste stringhe di 0 e 1 in un linguaggio più comprensibile. L'hardware del computer è in grado di interpretare questi valori in base al cambiamento di corrente: si può assegnare il valore 0 allo stato di assenza("off") e il valore 1 allo stato di presenza di corrente("on"). Ognuno di questi valori trasporta delle informazioni che successivamente vengono decodificate dal computer in parole, immagini, etc.
Invece il qubit, termine coniato da Benjamin Schumacher[1] nel 1955, rappresenta una sovrapposizione dei due stati del classico bit e viene descritto come una combinazione lineare dei valori 0 e 1 con la seguente formula:

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

dove $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$ e $|1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$ non sono altro che i vettori che rappresentano la base su cui si proietta la misura, α e β sono le ampiezze degli stati con modulo tale che la probabilità di misurare gli stati $|0\rangle$ e $|1\rangle$ sia rispettivamente $|\alpha|^2$ e $|\beta|^2$. Dal momento le probabilità di tutti gli stati deve essere in somma uguale a 100%, α e β devono essere scelti tenendo conto della regola di normalizzazione:

$$|\alpha|^2 + |\beta|^2 = 1$$

Fisicamente non è possibile avere una corrente che scorre in entrambe le direzioni sullo stesso cavo, dunque i qubit[15] possono essere realizzati con le due differenti polarizzazioni dei fotoni(verticale ed orizzontale) oppure dai due spin($\uparrow$ e $\downarrow$) o stati di eccitamento un elettrone dal momento che essi rappresentano un sistema quantistico a due livelli. Questi sistemi devono essere ben isolati per permettere a loro di mantenere le loro proprietà, infatti durante l'interazione, per esempio per una misura, con l'ambiente esterno si riscontra il fenomeno di decoerenza che non è altro che una interferenza che potrebbe cambiare lo stato. Per esempio se ci troviamo in una stanza buia e sappiamo che c'è un elettrone in un certo esatto punto, per poterlo vedere dobbiamo cercare con una torcia ma nel momento in cui illuminiamo la particella ricercata la colpiamo con numerosi fotoni e questi andranno ad interagire con l'elettrone cambiando lo stato precedente.
Le porte logiche quantistiche, illustrate in 2.4, creano interferenze nei circuiti in quanto le particelle vicine interagiscono tra di loro e quindi c'è una limitazione sul numero di qubit utilizzabili per circuito[10]. Per questo motivo i sistemi a livello quantistico hanno una certa probabilità di errore per via di queste interferenze, infatti sono stati sviluppati dei dispositivi con lo scopo di ridurre eventuali errori chiamati NISQ(Noisy Intermediate-Scale Quantum) e sono molto sensibili all'ambiente e hanno un numero limitati di qubit[19] e porte. Dunque non è ancora possibile avere una computazione quantistica perfetta ma con lo sviluppo di questi dispositivi sarà possibile sicuramente eccedere le capacità della computazione classica.
I qubit sono esponenzialmente più potenti dei bit, infatti $Bit = 2^{Qubit}$, e possono fare tutto ciò che può fare un bit però possono computare ogni versione degli input in un solo ciclo. Quindi se n bit possono rappresentare 2^n combinazioni, n qubit possono cercare 2^n versioni di un algoritmo in un singolo ciclo. Però la formula $Bit = 2^{Qubit}$ non è propriamente corretta in quanto i qubit sono molto più sensibili ad errori dei qubit data la loro natura.

2.3 Rappresentazione di un qubit

Una rappresentazione grafica di un qubit è data dalla sfera di Bloch, una rappresentazione con proprietà simili al cerchio unitario della trigonometria. Ogni punto all'interno della sfera rappresenta una determinata superposizione che il qubit può assumere.

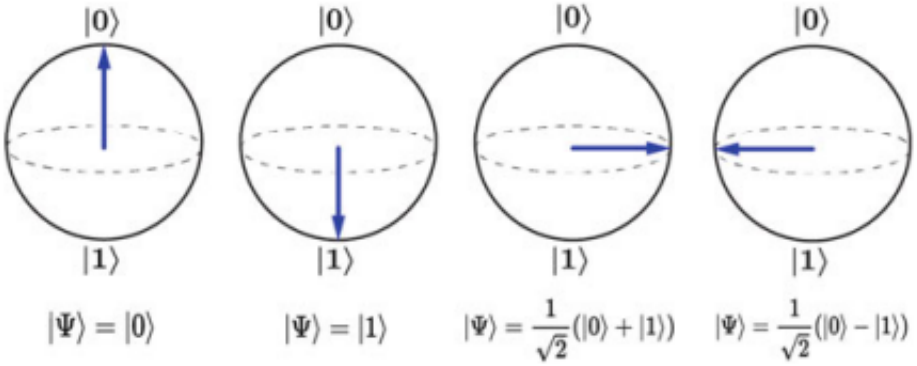


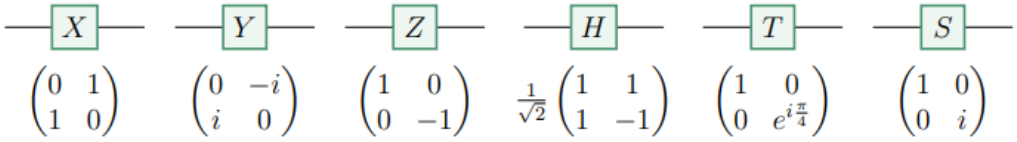
Figura 2.1: Lo stato del qubit è rappresentato dalla freccia nella sfera di Bloch

Nella figura 2.1 è possibile vedere quattro esempi di come usare la sfera di Bloch per visualizzare diversi stati di qubit: i poli corrispondono ai due stati $|0\rangle$ e $|1\rangle$ possibili misurabili, mentre se la freccia punta all'equatore, c'è una probabilità del 50% che il qubit collassi in uno dei due stati.

2.4 Porte logiche

La computazione quantistica è realizzata grazie alla manipolazione dello stato dei qubit tramite le porte logiche quantistiche. Queste hanno una rappresentazione matriciale e soddisfano la relazione $U^T U = UU^T = I$ in quanto operatori unitari.
Le porte che agiscono su singoli qubit sono le porte di Pauli(X,Y,Z) che ruotano il qubit di π rispettivamente lungo gli assi X, Y, Z nella sfera di Bloch, la porta di Hadamard(H) invece crea una superposizione di stati per una data base(esempio: $|0\rangle \rightarrow \frac{(|0\rangle+|1\rangle)}{\sqrt{2}}$ $|1\rangle \rightarrow \frac{(|0\rangle-|1\rangle)}{\sqrt{2}}$) o in altre parole applica una rotazione di $\frac{\pi}{\sqrt{2}}$ lungo l'asse Y e π lungo l'asse X.
Un'altra porta è la porta Phase Shift rappresentata dalla matrice: $R_p(\phi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$ ed effettua una rotazione di ϕ lungo un asse $p \in \{X, Y, Z\}$ sulla sfera di Bloch e non va a

cambiare le probabilità degli stati ma cambia la fase dello stato quantico. Due esempi comuni sono la porta T dove $\phi = \frac{\pi}{8}$ e la porta S(SWAP) dove $\phi = \frac{\pi}{2}$.



$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \quad \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

Figura 2.2: Porte logiche con singolo qubit

Per esempio applicando la porta quantistica X di Pauli è possibile passare da $|0\rangle$ a $|1\rangle$ e viceversa. Formalmente

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

Le porte quantistiche su singoli bit hanno analogie con i loro corrispettivi classici, ma esistono anche porte multi-qubit come la porta Controlled-Not(CNOT) che richiede due qubit di cui uno è un qubit di controllo il quale se ha valore $|1\rangle$, applica l'operatore NOT sull'altro qubit e la sua rappresentazione matriciale è

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

2.5 Parallelismo

La notevole potenza computazionale offerta dai qubit è data dal principio di sovrapposizione degli stati che permette a un elaboratore quantistico di poter calcolare contemporaneamente un'operazione su tutti gli stati di una funzione d'onda, dunque si ha un'elaborazione parallela di tutte le possibili configurazioni degli stati dei qubit.
Un computer classico esegue solo una funzione alla volta, infatti se consideriamo una funzione che prende due bit come input{(00, 01, 10, 11)}, se volessi l'output di ogni singolo input dovremmo eseguire la funzione quattro volte, mentre grazie alla superposizione, prendendo uno stato quantico di n qubit in input, avremo 2^n operazioni eseguite parallelamente. Grazie al principio di parallelismo è possibile eseguire molti più calcoli di una computazione classica.

Se prendiamo un sistema quantistico a 3 qubit, tutti i suoi stati possibili sono

$$|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle$$

ed essi verranno processati contemporaneamente grazie alle proprietà dei qubit. Nonostante ciò questo vantaggio computazionale lo si ha in casi molto specifici, casi per cui è possibile sfruttare gli effetti della sovrapposizione.
L'algoritmo ideato da Peter Shor[20] è un esempio pratico in cui viene sfruttato il concetto di parallelismo quantistico. Infatti questo algoritmo serve a fattorizzare un numero intero per trovare i fattori primi e viene eseguito in un tempo polinomiale, mentre l'algoritmo di fattorizzazione classico[6] ha un'andamento esponenziale. Purtroppo questo algoritmo, come molti altri algoritmi quantistici, ha un limite dato dal fenomeno di decoerenza[7] in quanto l'interazione tra l'ambiente e le particelle genera interferenze, introducendo errori nella computazione[27].

2.6 Complessità computazionale

I vari problemi computazionali sono classificati grazie alla teoria della complessità computazionale in base alle risorse(tempo di calcolo e memoria) necessarie per risolverli. Le due classi principali sono P(Polynomial) e NP(Nondeterministic Polynomial): il primo comprende problemi che possono essere risolti rapidamente da un computer mentre il secondo comprende problemi che hanno una soluzione che può essere verificata con il computer. Si può notare che P$\subset$NP in quanto saper risolvere un problema implica la possibilità di poter controllare la soluzione.
Per esempio per il problema della fattorizzazione di un intero n non è possibile trovare una soluzione in maniera rapida ma è possibile controllare che sia corretta semplicemente dividendo n per un fattore p trovato, dunque ricade nella classe NP ma non in P. Molti ricercatori credono che esistano problemi in NP che non sono P, infatti esiste la classe NP-Complete di cui un esempio è il problema delle somme parziali[21].
Non si sa ancora se i computer quantistici possano risolvere i problemi NP però un modo per poter risolverli è sfruttare il parallelismo calcolando tutte le soluzioni del problema parallelamente.
Un'altra classe è PSPACE che comprende tutti quei problemi risolvibili non considerando le risorse utilizzate e il tempo(dunque computazioni lunghe vanno bene) e si crede che sia più grande di N e NP ma non è stato provato.
Infine c'è la classe BPP(Bounded-error Probabilistic Polynomial) caratterizzata da pro-

blemi che possono essere risolti tramite algoritmi probabilistici in tempo polinomiale con una certa tolleranza, data dalla probabilità d'errore, sul risultato.
Per quanto riguarda invece le classi di computazione quantistica si definisce la classe BQP(Bounded-error Quantum Polynomial)[15] con caratteristiche analoghe alla classe BPP. Non è ancora chiaro come posizionare questa classe nello spazio dei problemi in quanto finora si sa che i computer quantistici sono in grado di risolvere efficientemente i problemi P ma non ci sono problemi al di fuori di PSPACE che riescano a risolvere altrettanto velocemente. Dunque BQP giace da qualche parte tra P e PSPACE, ma in parte questa incertezza è data anche dal fatto che non si sa se PSPACE sia più grande di P o meno.

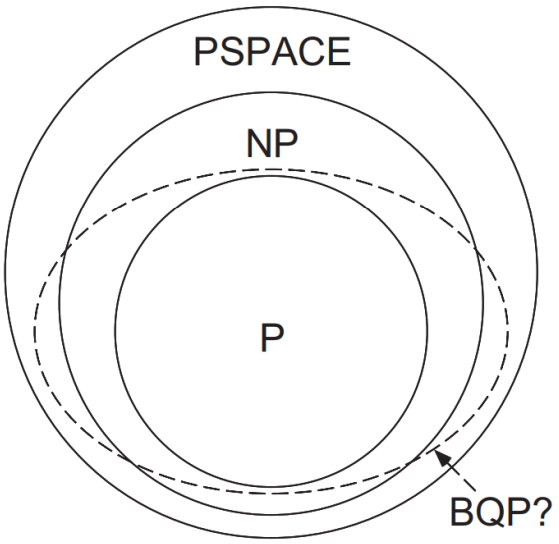


Figura 2.3: Rappresentazione insiemistica delle classi di complessità

2.7 Legge di Moore e i nuovi limiti

L'hardware si sviluppa ad una velocità sempre crescente ma allo stesso tempo anche le dimensioni si stanno riducendo, infatti Gordon Moore, co-fondatore di Intel, nel 1975 ipotizzò che i transistor in un'unità di spazio sarebbero raddoppiato ogni 2 anni creando così circuiti più efficienti e riducendo anche le dimensioni dei computer.
Però questa riduzione delle dimensioni ha un limite in quanto non si può ridurre ulteriormente le dimensioni dei circuiti perché si arriverà in un punto in cui si consumerà più energia nel raffreddare i transistor dell'effettiva energia che usano questi ultimi. Purtroppo si è sempre più vicini a questo limite perché le dimensioni stanno raggiungendo la scala del nanometro e andando ancora più a fondo le regole della fisica classica ini-

ziano a non valere più perché si entra nel mondo delle particelle, infatti Feynman[24] ipotizzò che in futuro le porte logiche e i fili consisteranno in semplici atomi e quindi la meccanica quantistica giocherà un ruolo fondamentale.
Qui sotto è riportato il grafico in scala semilogaritmica, per poter visualizzare meglio l'andamento esponenziale, della crescita del numero di transistor negli anni.

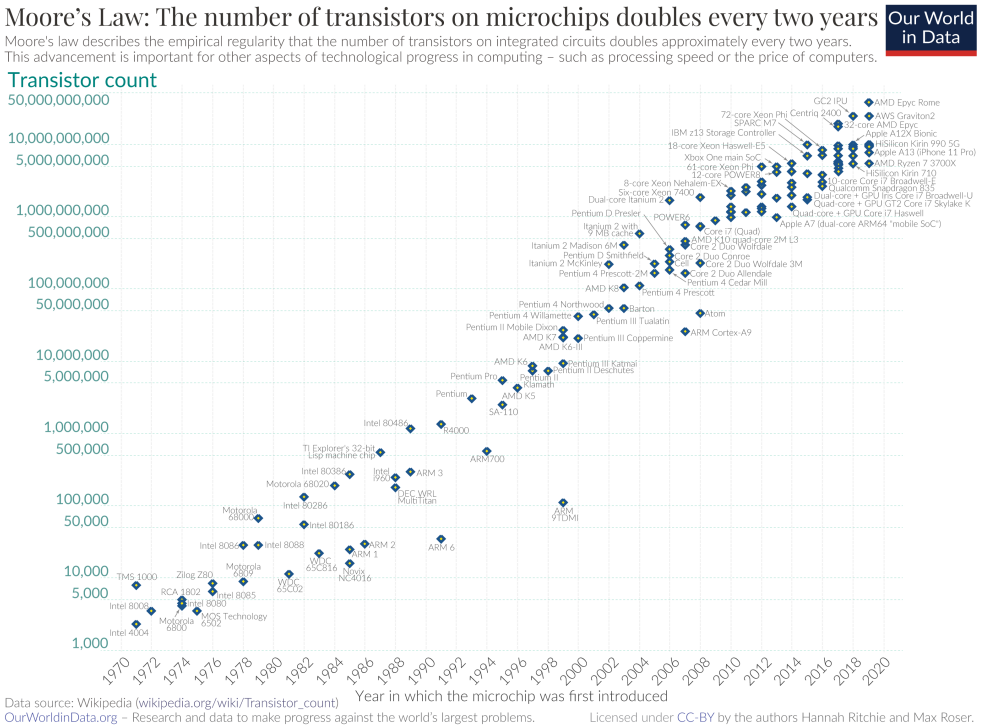


Figura 2.4: Crescita del numero di transistor nei vari modelli di processori negli anni

Infatti una delle soluzioni risiede nei computer quantistici, riprendendo la formula secondo cui $Bit = 2^{Qubit}$[18] arriviamo alla conclusione per cui per raddoppiare la potenza di una macchina quantistica, ci basterebbe aggiungere un qubit e seguendo la legge di Moore dovremmo avere l'aumento di 1 qubit ogni 2 anni però si può vedere in figura 2.5 come l'andamento sia quasi esponenziale nonostante ciò.

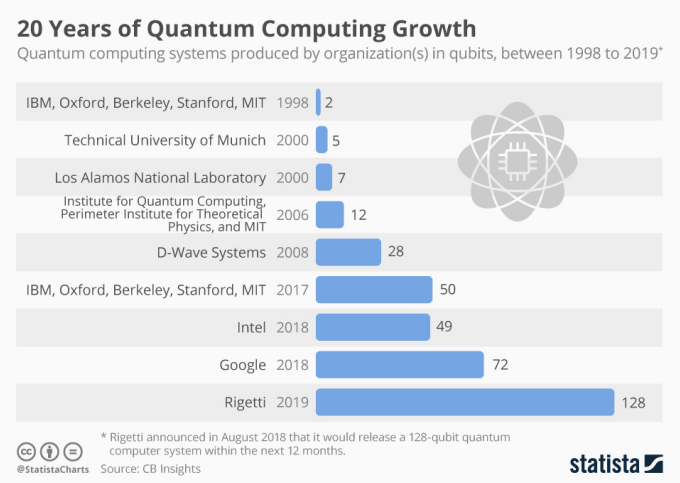


Figura 2.5: Grafico che mostra il raddoppio del numero di qubit in sistemi quantistici computazionali ogni 2 anni

3

Reti neurali quantistiche

Con l'avanzare della tecnologia verso il mondo quantistico è possibile fare un passo in avanti nelle reti neurali. Inoltre uno dei motivi principali per cui è stato concepito il collegamento tra mondo quantistico e reti neurali risiede in una possibile spiegazione di alcuni fenomeni del cervello umano, come l'apprendimento e la coscienza[25], tuttora inspiegabili date le odierne teorie. Sarebbe possibile anche poter capire meglio il cervello in se, le sue funzioni e la sua abilità di processare le informazioni. Possiamo riscontrare alcune analogie tra i due mondi, per esempio un neurone ha la stessa funzione della funzione onda e le connessioni tra di essi possono essere rappresentante tramite il concetto di superposizione.
Nonostante si pensi che la meccanica quantistica, ancora da esplorare, possa aiutare nello studio del cervello, ci sono elementi che potrebbero rendere questo collegamento fuorviante: se prendiamo in considerazioni i neuroni delle reti neurali, essi operano seguendo funzioni non lineari mentre la meccanica quantistica si basa su funzioni lineari. Però grazie al fenomeno della decoerenza è possibile avere reti neurali quantistiche in quanto questo fenomeno viene utilizzato per poter leggere i risultati prodotti in un sistema quantistico e rende possibile l'evoluzione della rete neurale per arrivare ad uno stato stabile dopo tanti cicli di addestramento.
Dobbiamo la grande potenza computazionale delle reti neurali alla loro capacità di elaborare i dati in maniera altamente parallela e distribuita. D'altro canto il parallelismo quantistico offre uno strumento ancora più potente nell'elaborare enormi quantità di dati processando parallelamente tutte le configurazioni degli stati di n qubit per poi leggere solo un singolo risultato dopo la decoerenza. Inoltre grazie all'entanglement è possibile conoscere leggere il valore di tutti gli altri qubit senza effettivamente avere il risultato.

3.1 Pattern recognition e QNN

Per poter capire come unire questi due concetti proviamo a considerare l'applicazione più semplice di una rete neurale: pattern recognition che ha come obiettivo l'apprendimento[4] di un classificatore di dati in base a conoscenze pregresse fornite o informazioni statistiche sui pattern così da poter imparare a riconoscere le varie classi di input. Uno degli approcci più comuni è quello di classificare diversi pattern in vari template cosicché sia possibile classificare gli input scegliendo il template che più ci assomiglia. Però per essere efficiente questo processo dev'essere eseguito in parallelo in quanto i template sono numerosi e vanno confrontati singolarmente con l'input. Con le reti neurali è possibile immagazzinare una grande quantità di dati in un singolo sistema, infatti grazie all'apprendimento è possibile conservare diverse classi e poter classificare gli stimoli forniti alla rete. Per funzioni più complesse potrebbe diventare necessario l'implementazione di più reti neurali con l'approccio multi modulare dove ogni rete è allenata con pattern di una singola classe però questa soluzione comporta un elevato utilizzo di risorse.
La soluzione secondo cui si immagazzinano più pattern in una singola rete può riscontrare problemi nel caso in cui si cerchi di raccogliere più pattern di natura molto diversa tra di loro assieme poiché questi possono creare interferenze e portare ad un risultato sbagliato. Dunque un approccio possibile richiede l'intervento di elementi della meccanica quantistica, considerando l'interpretazione "Many worlds"[28] secondo cui la decoerenza non crea una distruzione della superposizione ma anzi crea tanti universi paralleli quanti i diversi risultati possibili della misurazione[7], i qubit nei circuiti esistono in più mondi paralleli e l'elaboratore quantistico è in grado di operare tra queste "dimensioni" facendo collassare l'osservatore su una delle possibili dimensioni. Pertanto per risolvere il problema possiamo procedere considerando i vari pattern come degli stati di una superposizione anche così però ci si imbatte in risultati non corretti tuttavia in questo caso, a differenza delle reti neurali classiche, non sono dati dall'interferenza tra pattern però almeno in questa maniera è possibile ideare un sistema di reti neurali dove ognuna è allenata con un singolo pattern però senza il bisogno di implementare più reti ma ne basta una sola in quanto le singole coesistono in un'unica QNN.
Questo approccio può essere realizzato[30] utilizzando MLP allenate con pattern diversi e formando una QNN i cui pesi sono le superposizioni delle varie MLP esistenti negli universi paralleli. Le reti neurali classiche si basano sul cervello umano per il quale è difficile gestire un numero così elevato di pattern contemporaneamente, però con questo metodo è possibile velocizzare di molto l'apprendimento.

3.2 Percettroni quantistici e struttura

Nel percettrone classico si prendono i vari input x_i con i rispettivi pesi w_i producendo un output $z = \sum_i x_i w_i + b$ e successivamente viene applicata la funzione di attivazione ottenendo così $a = f(z)$ che è l'output finale. Il percettrone quantistico ha il medesimo funzionamento ma viene realizzata con un circuito Repeat-Until-Success(RUS). L'idea di questo circuito consiste nel ripetere continuamente la misura di un qubit ausiliario finché non si ottiene quella desiderata nel qubit di output in quanto non è possibile creare una copia esatta di uno stato quantico e che ci permette di realizzare una funzione di attivazione non lineare applicando la porta $R_y(2\theta) = e^{-iY\frac{2\theta}{2}}$ che effettua una rotazione lungo l'asse Y di un angolo 2θ.

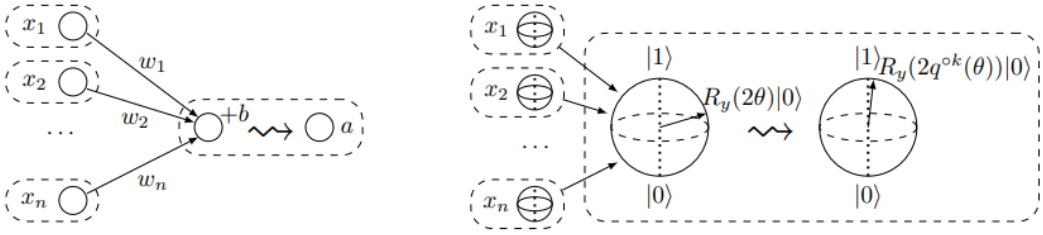


Figura 3.1: Confronto tra un percettrone classico e quantistico

In figura 3.1 si può vedere la controparte quantistica[2], che unisce il concetto di linearità della meccanica quantistica e la non linearità delle reti neurali, dove si possono vedere le sfere di Bloch prima e dopo l'applicazione di RUS, rispettivamente corrispondono all'applicazione della funzione lineare e non lineare, quest'ultima realizzata dalla porta $R_y(2q(\varphi))$ dove $q(\varphi) = arctan(tan^2\varphi)$ è l'angolo utilizzato per effettuare la rotazione e ha un comportamento simile alla sigmoide.

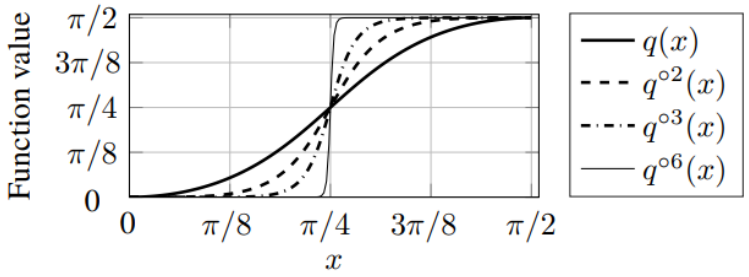


Figura 3.2: Funzione $q^{ok}(x) = \arctan(\tan^{2^k} x)$

Nelle reti neurali classiche abbiamo $a = f(\sum_i x_i w_i + b)$ e applicando una funzione non lineare otteniamo un certo output, per esempio con una funzione gradino ottenia-

mo 1 se $\sum_i x_i w_i + b > 0$ altrimenti -1, oppure con altre funzioni continue come quelle discusse nel primo capitolo. Invece con una rete neurale quantistica, prendiamo come esempio un qubit il cui stato è $R_y(a\frac{\pi}{2} + \frac{\pi}{2})|0\rangle = \cos(a\frac{\pi}{4} + \frac{\pi}{4})|0\rangle + \sin(a\frac{\pi}{4} + \frac{\pi}{4})|1\rangle$ dove $a \in [-1, 1]$ è uno scalare e $R_y(\theta)$ è la porta di rotazione vista prima. Avremo quindi i casi limiti $a = -1$ e $a = 1$ che corrispondono agli stati $|0\rangle$ e $|1\rangle$ analogamente al caso con output binario, però per qualsiasi $a \in (-1, 1)$ non c'è nessuna analogia.
Il motivo principale per cui viene usato il circuito RUS risiede nel fatto che vogliamo ricreare un valore di soglia che permetta l'attivazione o meno del percettrone. Infatti in questo sistema abbiamo un qubit ausiliario(così come il valore z nell'esempio prima) a cui vengono applicate le rotazioni e se la misura ritorna $|0\rangle$ sappiamo che la rotazione di $2q^{ok}(\theta)$ è stata applicata successivamente al qubit che andrà in uscita. Se invece otteniamo $|1\rangle$ dal qubit ausiliario significa che c'è stata una rotazione $R_y(\frac{\pi}{2})$ e quindi bisogna applicare $R_y(-\frac{\pi}{2})$ per correggere e ripetere nuovamente le rotazioni finché non si ottiene $|0\rangle$. Il meccanismo del RUS è stato introdotto in quanto è impossibile creare una copia identica di uno stato quantico e sarebbe difficile quindi propagare il corretto output di un layer al successivo, ma così facendo si continua a ripetere la misurazione finché il qubit output non sarà uguale al qubit ausiliario. Inoltre comportamento del circuito introduce un valore di soglia che risiede nella scelta dell'angolo θ: applicando $R_y(2q^{ok}(\theta))$, se $\theta > \frac{\pi}{4}$ vorremmo che il qubit di output sia più vicino possibile a $R_y(\pi)|0\rangle = |1\rangle$ altrimenti che sia più vicino a $R_y(0)|0\rangle = |0\rangle$.
Per quanto riguarda la struttura della rete invece non c'è differenza rispetto alla controparte classica, infatti è formata sempre dai soliti input, output e hidden layer però con le attuali tecnologie non è possibile avere troppi qubit per via dell'alta interferenza e della incapacità dei dispositivi odierni di compensare.

3.3 Addestramento

Così come nella rete neurale classica, anche in quella quantistica è necessario il processo di addestramento per raggiungere i valori dei pesi, che in questo caso non sono altro che operatori unitari, ottimali. L'obiettivo finale[12] quindi è quello di trovare un operatore unitario V dato un training set formato dalle coppie $(|\phi_i^{in}\rangle, |\phi_i^{out}\rangle)$ dove $i = 1, 2, .., N_T$, N_T è il numero di coppie usate per l'addestramento e $|\phi^{out}\rangle = V|\phi^{in}\rangle$.
Per poter valutare il funzionamento della rete quindi dobbiamo confrontare l'output desiderato $|\phi_{out}\rangle$ con l'output fornito dalla rete ρ^{out}.

È necessaria quindi una funzione dei costi definita come

$$C_T = \frac{1}{N_T} \sum_{i=1}^{N_T} \langle \phi_i^{out} | \rho_i^{out} | \phi_i^{out} \rangle$$

che varia da 0 a 1(caso in cui è ottima).
I vari parametri della rete sono inizializzati con valori casuali e ad ogni epoca i valori vengono aggiornati così da massimizzare il costo. I nuovi parametri sono calcolati come

$$p_{t+1} = p_t + \eta \nabla C(p_t)$$

Anche in questo caso è necessario controllare che i parametri trovati siano validi, infatti introduciamo il validation set formato dalle coppie $(|\phi_i^{in}\rangle, |\phi_i^{out}\rangle)$ dove $i = 1, 2, .., N_V$, abbiamo quindi il costo

$$C_V = \frac{1}{N_V} \sum_{i=1}^{N_V} \langle \phi_i^{out} | \rho_i^{out} | \phi_i^{out} \rangle$$

Ricapitolando l'addestramento può essere suddiviso 3 fasi: **inizializzazione** dove si forniscono un set di coppie di addestramento e un set di coppie di validazione, si impostano gli iperparametri e si inizializzano i parametri con valori randomici; successivamente si procede con l'**addestramento** in cui grazie alla funzione dei costi e la discesa del gradiente si aggiustano i pesi dei parametri con più iterazioni del processo ed infinite la **validazione** in cui si verificano le capacità di generalizzazione della rete verificando che i parametri ottenuti siano validi.

3.4 Confronto con una rete neurale classica

In questo paragrafo analizziamo i risultati di alcuni esempi pratici di applicazioni di reti neurali classiche e quantistiche confrontate tra di loro per poter visualizzare le differenze tra i due tipi di modelli. Il primo esempio riportato è un benchmark di un problema classico[17] delle reti neurali: le porte logiche. Gli altri due esempi trattano un problema di regressione e di classificazione di immagini riportati in [23].

3.4.1 Benchmark di porte logiche

In questo benchmark vengono confrontate le prestazioni di una rete neurale creata su NeuralWorks[16] usata come baseline, una Real-Valued Neural Network(RVNN), una Complex-Valued Neural Network(CVNN) e una Quantum Neural Network(QNN).

Le RVNN sono le reti neurali classiche in grado di approssimare una qualsiasi funzione continua capace di mappare un input in un output e le loro funzioni di attivazione sono del tipo $f = \mathbb{R}^n \rightarrow \mathbb{R}$.
Mentre le CVNN utilizzano valori complessi invece dei reali e infatti possono fare tutte le operazioni possibili in $\mathbb{R}^n$ in quanto $\mathbb{R}^n \subset \mathbb{C}^n$. Quindi le CVNN possono risolvere ogni problema risolvibile anche dalle RVNN però sono più efficienti: un chiaro esempio è il problema XOR, un problema non lineare che una CVNN è in grado di risolvere utilizzando un solo layer, mentre la RVNN ha bisogno di un almeno un hidden layer.
I segnali di input sono del tipo $z = \sum_i x_i w_i$ dove $x, w \in \mathbb{C}^n$ e la funzione di attivazione $P(z) = e^{iarg(z)}$ prende l'input e lo mappa in un cerchio unitario nel piano complesso per poi passarlo ai neuroni successivi.
Le reti sono costituite da un singolo strato e sono allenate con i 4 possibili input {(00), (01), (10), (11)} e si è verificato quante epoche sono necessarie per allenare la rete applicando le varie porte logiche(AND, NAND, OR, NOR, XOR, XNOR). Si noti che non è possibile avere dati riguardanti le RVNN dei problemi XOR e XNOR in quanto richiedono più di un singolo layer.

No. of Epochs to RMS error $\leq$1%				
Gate	N.Works	RVNN	CVNN	QNN
AND	11000	11146	222	1
NAND	11000	11145	127	1
OR	6000	5672	114	1
NOR	6000	5671	172	1
XOR	n/a	n/a	110	1
XNOR	n/a	n/a	78	1

Figura 3.3: Tabella riassuntiva dei risultati ottenuti

Nella tabella è possibile notare il numero di epoche necessarie per ottenere un RMSE$\leq$ 1% e si vede come una rete neurale a valori complessi sia molto più efficiente dal punto di vista computazionale di una a valori reali ed è anche in grado di risolvere problemi non lineari rapidamente. La rete neurale quantistica invece strutturalmente può essere vista come una CVNN però, grazie alle proprietà della meccanica quantistica, hanno delle prestazioni molto migliori: infatti in tutte le tipologie di porte logiche sottoposte a test, sono riuscite ad arrivare ad un RMSE minore dell'1% in una singola epoca di

addestramento e questo risultato è notevole in quanto questa fase è quella che richiede più risorse nell'ambito delle reti neurali.

3.4.2 House sale regression

Il dataset[9] comprende un record di 21613 vendite di case del King County, Washington, USA dal 02/05/2014 al 27/05/2015. Questo dataset è composto da 21 colonne con informazioni come il numero di stanze, metri quadri, periodo di costruzione, ultima ristrutturazione ed ulteriori classificazioni in base alle condizioni, la vista dalla casa, la qualità della costruzione, etc. L'obiettivo di questa regressione è trovare il prezzo a cui è stata venduta la casa date tutte queste informazioni. La versione quantistica di questo problema utilizza l'asse Z del qubit per distribuire la probabilità tra i suoi stati $|0\rangle$ e $|1\rangle$ e viene utilizzata la funzione di attivazione $tanh$ e la rete è realizzata da 3 qubit che vengono manipolati da una porta R_x.

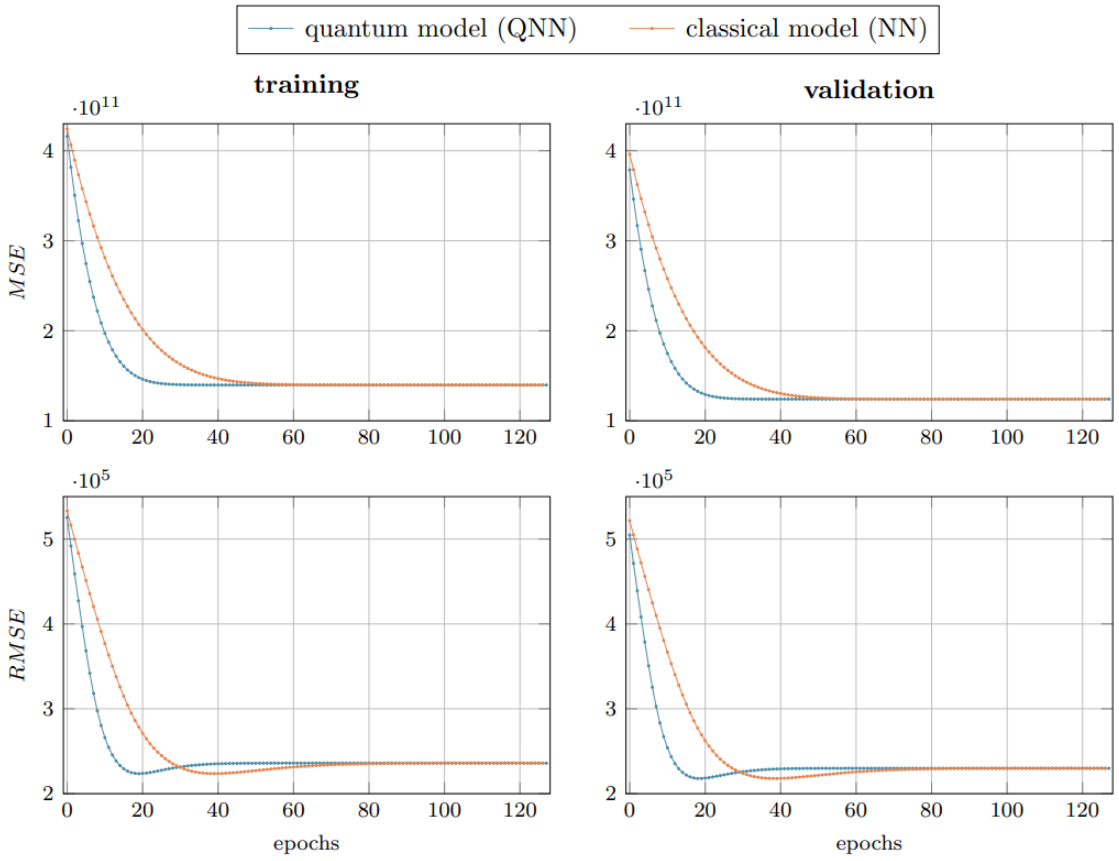


Figura 3.4: Confronto della MSE e RMSE tra una QNN e NN in una regressione

In questi grafici è possibile osservare i risultati finali dove vengono messi a confronto MSE(Mean Squared Error) e RMSE(Root Mean Squared Error) degli step di addestramento e validazione. Si può notare come una rete neurale quantistica e una rete neurale classica possano performare alla stessa maniera per un problema di regressione. Nonostante entrambi i modelli presentino un RMSE finale di 2.5×10^5, a parità di struttura della rete e parametri è possibile osservare come la rete quantistica stabilizzi il suo MSE all'incirca dopo 22 epoche, mentre la rete neurale classica ce ne mette quasi il doppio: da questi dati evidenziano come una QNN sia più veloce di una NN ad addestrare i suoi parametri.

3.4.3 MNIST CLASSIFICATION

Il dataset della Modified National Institute of Standards and Technology(MNIST) contiene delle immagini con 70000 sfumature di grigio ed è un database di cifre scritte a mano salvate come immagini 28x28 con i pixel con valore da 0(bianco) a 255(nero). Per poter lavorare con queste immagini avremmo bisogno di 784 qubit, che è ben oltre la capacità degli elaboratori quantistici odierni, oppure un computer classico con 2^{784} bit di memoria, un numero ben oltre il numero di particelle nell'universo. Dunque la soluzione è un down-scale delle immagini, infatti vengono ridotte in 4x4, grandezza più che sufficiente per avere un'accuratezza dell'85%.

Figura 3.5: Un esempio di immagini presenti nel dataset MNIST

I risultati trovati mettono in confronto i grafici della loss function di tipo cross-entropy, una funzione utile per valutare risultati che comprendono delle probabilità, e dell'accuratezza che ci fornisce quante delle predizioni della rete sono state corrette.

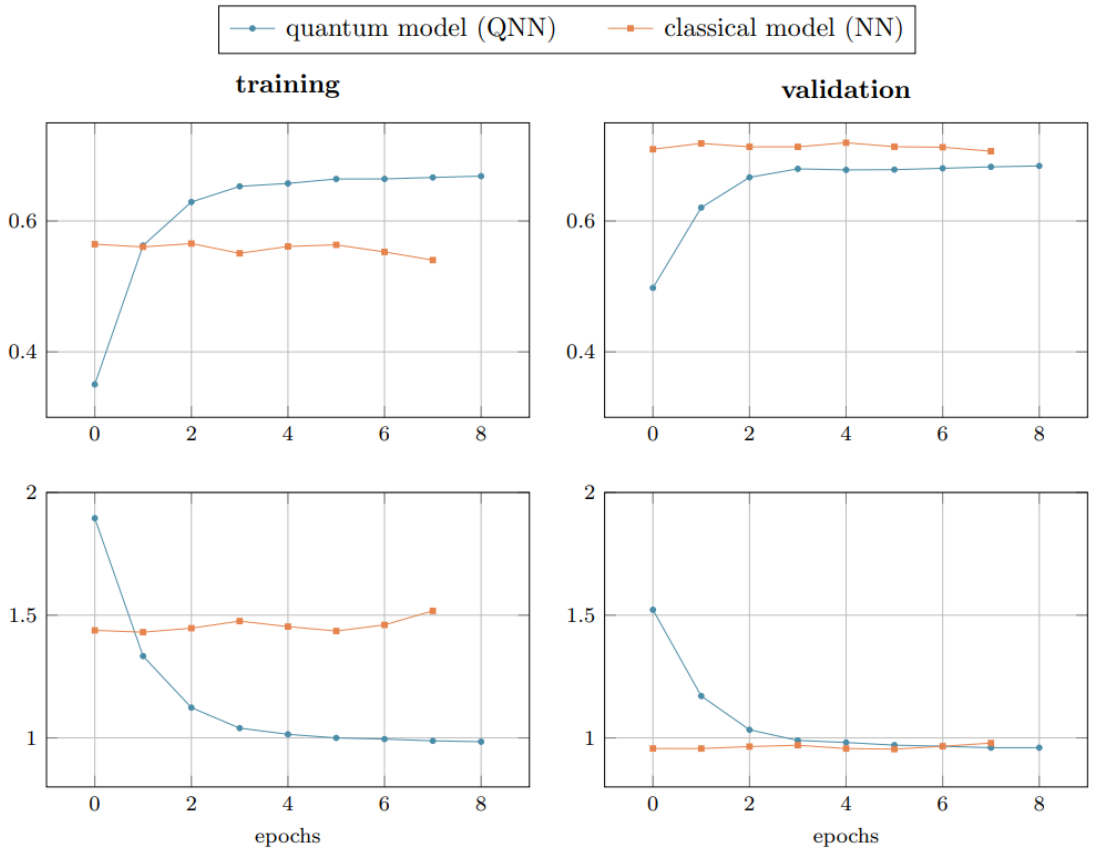


Figura 3.6: Confronto della loss function e accuracy tra una QNN e NN in una classificazione

In questi grafici si può notare come la NN abbia un comportamento quasi lineare e ciò implica che il modello non stia imparando nelle epoche, probabilmente per via di una scelta di η bassa. D'altro canto la QNN, con lo stesso learning rate, riesce ad imparare e raggiungere un valore limite, ovvero un'accuratezza di circa 80%. Infatti in questo esempio si può osservare che la rete neurale quantistica sia più performante di una rete neurale classica nonostante quest'ultima abbia un numero maggiore di parametri da addestrare(650 contro i 218 della QNN). Ma questo non implica per forza la superiorità delle QNN sulle NN in quanto nell'esempio precedente si è potuto vedere come abbiano avuto un comportamento simile però la differenza sostanziale è stata riscontrata nel tempo di addestramento.

4

Conclusioni

I computer quantistici offrono molti vantaggi e possono portare miglioramenti in molti algoritmi già esistenti, come per esempio l'algoritmo di fattorizzazione di Shor[20] dove c'è stato un miglioramento nell'esecuzione da un tempo esponenziale a polinomiale grazie all'utilizzo delle proprietà della meccanica quantistica. Altrettanto importante è un miglioramento nell'algoritmo di ricerca di un database non ordinato di Grover[13] dove si è riscontrato un miglioramento da $O(N)$ a $O(\sqrt{N})$. In termini di velocità tra computer quantistico e computer classico generalmente sono alla pari, però il primo è esponenzialmente più veloce del secondo se si utilizzano gli algoritmi adatti, come negli esempi citati.
Per capire effettivamente i vantaggi di queste due tipologie di computazione bisogna valutare le loro applicazioni. Infatti una computazione quantistica eccelle nella ricerca di tutti gli input possibili e nella ottimizzazione, degli esempi di applicazioni possono essere la decrittazione di RSA[26](grazie alla facilità di esecuzione dell'algoritmo di Shor per la fattorizzazione dei primi, su cui si basa tutto il mondo della crittografia), ottimizzazione delle reti neurali in termini di addestramento oppure elaborazioni in campo chimico o biologico dove la complessità molecolare può essere gestita senza problemi da un elaboratore quantistico. Però nonostante si pensi che la computazione quantistica sia meglio, ci sono ancora campi in cui prevale la classica come per le operazioni sequenziali in quanto gli algoritmi quantistici non offrono nessun vantaggio nello svolgimento. È più probabile che gli elaboratori quantistici svolgano la funzione di server in ambiti di ricerca e sviluppo anche data la difficoltà della loro implementazione, quindi dubito che questa tipologia di computer diventi d'uso comune.
Però sicuramente presentano delle sfide per il futuro: la prima sicuramente il limitato

numero di qubit utilizzati, un'altra invece riguarda la creazione di nuovi algoritmi in grado di sfruttare appieno le proprietà quantistiche ma la sfida più grande consiste nella riduzione dell'errore nei circuiti quantistici, in quanto abbiamo visto che è possibile inserire solo un numero limitato di porte quantistiche in un circuito per via delle interferenze tra particelle. Infatti queste macchine sono realizzate in un ambiente vicino allo zero assoluto per cui è possibile controllare singolarmente gli atomi così da gestire gli spin ma ad un livello così microscopico si è più proni ad errori e rumori nelle operazioni e misurazioni, infatti per ora si riscontra circa un errore dell'1% per ogni operazione elementare. Per quanto un'accuratezza del 99% sembri buona, un semplice errore potrebbe corrompere completamente il risultato finale.
Le potenzialità delle QNN sono sicuramente un buon motivo per proseguire la ricerca in questo campo non ancora molto conosciuto. Anche perché si pensa che i meccanismi della meccanica quantistica possano spiegare fenomeni legati alla mente umana ancora sconosciuti, infatti si pensa che ci sia una correlazione tra i processi quantistici e le funzioni cerebrali, come la memoria a breve termine e la consapevolezza cosciente. Inoltre si crede che l'emisfero sinistro sia responsabile dei processi logici ed elaborazione e infatti è stato possibile replicarlo con una macchina di Von Neumann, però l'emisfero destro è responsabile di processi più astratti come il pensiero e l'intuizione per cui non è ancora stato possibile creare un modello in grado di astrarre queste funzioni ma può darsi che la risposta stia nella computazione quantistica. Una probabile correlazione è stata discussa nella ricerca[3] dove si trova che facendo una risonanza magnetica agli spin dei protoni del liquido cerebrale si riscontri dei segnali che assomigliano ai potenziali prodotti dal battito cardiaco e questo risultato non è spiegabile in quanto questi ultimi non sono rilevabili con una risonanza magnetica e quindi si pensa che l'entanglement sia l'unica possibile spiegazione di questo fenomeno in quanto significherebbe che i processi cerebrali hanno qualche sorta di interazione con gli spin dei nuclei atomici.
Questo potrebbe potrebbe portare a nuove teorie sul fatto che tuttora il nostro cervello possa superare di gran lunga i supercomputer quando si tratta di processi decisionali, circostanze impreviste oppure l'apprendimento di qualcosa di nuovo. Dunque non è da escludere che le QNN possano spiegare il funzionamento del cervello umano.
Come visto nelle simulazioni viste nel capitolo precedente, le QNN hanno capacità migliori delle NN nel risolvere i problemi, però questo dipende dalla struttura dell'algoritmo quantistico e vale solo per alcuni tipi di problemi, ma comunque c'è un notevole vantaggio nell'apprendimento da parte delle QNN che a parità di struttura e parametri imparano molto più rapidamente.

Possiamo sintetizzare i vantaggi offerti dalle QNN in:

- elevate prestazioni con meno neuroni rispetto alle NN
- addestramento più rapido
- maggiore stabilità e affidabilità data dalla minor interferenza tra pattern
- velocità di esecuzione
- risoluzione di problemi nonlineari son un singolo layer

Può anche essere che le QNN possano superare le restrizioni imposte dalla tesi di Church-Turing[31]: tutti gli algoritmi eseguibili nel nostro cervello possono essere realizzati in una macchina di Turing e questo limite vale sia per le NN più recenti che per i computer quantistici ma per le QNN non è ancora verificato in quanto sono ancora da ricercare ulteriormente. Sicuramente è una sfida per il futuro della teoria delle reti neurali quantistiche, che potrebbero addirittura affrontare problemi irrisolvibili in principio.

Bibliografia

[1] Benjamin Schumacher (1995). *Quantum coding*. URL: https://journals.aps.org/pra/abstract/10.1103/PhysRevA.51.2738.

[2] Cao Yudong, Gian Giacomo Guerreschi, Alán Aspuru-Guzik (2017). *Quantum Neuron: an elementary building block for machine learning on quantum computers*. URL: https://arxiv.org/pdf/1711.11240.pdf.

[3] Christian M. Kerskens, David L. Pérez (2022). *Experimental indications of non-classical brain functions*. URL: https://iopscience.iop.org/article/10.1088/2399-6528/ac94be/pdf.

[4] Christopher M. Bishop (2006). *Pattern Recognition and Machine Learning*. URL: https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf.

[5] Ciaran Hughes, Joshua Isaacson, Anastasia Perry, Ranbel F. Sun, Jessica Turner (2021). *Quantum Computing for the Quantum Curious*. URL: https://link.springer.com/book/10.1007/978-3-030-61601-4.

[6] *Crivello dei campi di numeri generale*. URL: https://it.wikipedia.org/wiki/Crivello_dei_campi_di_numeri_generale.

[7] David Deutsch (1998). *The Fabric of Reality*. URL: https://www.ibs.it/fabric-of-reality-libro-inglese-david-deutsch/e/9780140146905.

[8] Frank Rosenblatt (1958). *The perceptron: A probabilistic model for information storage and organization in the brain.* URL: http://dx.doi.org/10.1037/h0042519.

[9] *House Sales in King County USA*. URL: https://www.kaggle.com/datasets/harlfoxem/housesalesprediction.

[10] John Preskill (2018). *Quantum Computing in the NISQ era and beyond*. URL: https://arxiv.org/abs/1801.00862v3.

[11] Kerstin Beer (2022. *Quantum neural networks*. URL: https://arxiv.org/abs/2205.08154.

[12] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J. Osborne, Robert Salzmann, Ramona Wolf (2019). *Efficient Learning for Deep Quantum Neural Networks*. URL: https://arxiv.org/abs/1902.10445.

[13] Lov K. Grover (1996). *A fast quantum mechanical algorithm for database search*. URL: https://arxiv.org/abs/quant-ph/9605043.

[14] Maria Schuld, Francesco Petruccione (2021). *Machine Learning with Quantum Computers*. URL: https://link.springer.com/book/10.1007/978-3-030-83098-4.

[15] Michael A. Nielsen, Isaac L. Chuang (2010). *Quantum Computation and Quantum Information*. DOI: 10.1017/CBO9780511976667.

[16] *NeuralWorks*. URL: https://www.neuralware.com.

[17] Nhung H. Nguyen, Elizabeth C. Behrman, Mohamed A. Moustafa, James E. Steck (2018). *Benchmarking neural networks for quantum computation*. URL: https://arxiv.org/pdf/1807.03253.pdf.

[18] Nicholas Yoder (2021). *Moore's Law of Moore's Law of Quantum Computing*. URL: https://nickyoder.com/moores-law-quantum-computer/.

[19] Peter W. Shor (1996). *Fault-tolerant quantum computation*. DOI: 10.1109/SFCS.1996.548464.

[20] Peter W. Shor (1999). *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. URL: http://dx.doi.org/10.1137/s0036144598347011.

[21] *Problema delle somme parziali*. URL: https://it.wikipedia.org/wiki/Problema_delle_somme_parziali.

[22] *Qubit*. URL: https://it.wikipedia.org/wiki/Qubit.

[23] Rafał Potempa (2021). *Simulation of quantum neural network with evaluation of its performance*. DOI: 10.13140/RG.2.2.28877.97765.

[24] Richard P. Feynman (1986). *Quantum mechanical computers*. URL: https://link.springer.com/article/10.1007/BF01886518.

[25] Roger Penrose (1995). *Shadows of the Mind*. URL: https://www.ibs.it/shadows-of-mind-search-for-libro-inglese-roger-penrose/e/9780099582113.

[26] *RSA*. URL: https://it.wikipedia.org/wiki/RSA_(crittografia).

[27] Scott Aaronson (2008). *I limiti del computer quantistico*. URL: https://www.lescienze.it/archivio/articoli/2008/05/01/news/i_limiti_del_computer_quantistico-549196/.

[28] Sean Carroll (2019). *The parallel worlds of quantum mechanics*. URL: https://www.sciencefocus.com/science/the-parallel-worlds-of-quantum-mechanics/.

[29] *Spazio di Hilbert*. URL: https://en.wikipedia.org/wiki/Hilbert_space.

[30] Tammy Menneer, Ajit Narayanan (1995). *Quantum artificial neural networks*. URL: https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.286530.

[31] *Tesi di Church-Turing*. URL: https://it.wikipedia.org/wiki/Tesi_di_Church-Turing.

[32] Warren S. McCulloch, Walter Pitts (1943). *A Logical Calculus of the Ideas Immanent in Nervous Activity(1943)*. URL: https://link.springer.com/article/10.1007/BF02478259.