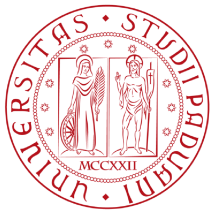


UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea in Fisica

Tesi di Laurea

Studio e applicazione di algoritmi quantistici ai dati dell'esperimento LHCb

Relatore

Prof. Donatella Lucchesi

Correlatore

Dr. Lorenzo Sestini

Laureando

Filippo Pra Floriani

Anno Accademico 2021/2022

Indice

Introduzione

Gli algoritmi di apprendimento automatico sono ampiamente utilizzati nella Fisica delle particelle elementari. La classificazione dei *jets*, ovvero la determinazione del sapore del quark che genera il jet, è una della applicazioni che ha ottenuto un notevole miglioramento tramite l'uso di algoritmi di apprendimento automatico. I *jets* sono flussi di particelle che si formano in seguito alla frammentazione e adronizzazione di quark e gluoni durante i processi di collisione negli accelatori, come accade al Large Hadron Collider (LHC) [1] di Ginevra. Data la mole di prodotti derivanti dalla collisione, risulta necessario servirsi di nuovi metodi per ricostruire le proprietà dei jets. L'obiettivo in questo settore della Fisica delle particelle elementari è lo sviluppo di algoritmi innovativi sempre più performanti, che abbiano tempi di esecuzione brevi e limitino l'uso di risorse informatiche in modo che queste non aumentino esponenzialmente con la crescita della mole di dati.

I computer quantistici sono un'evoluzione degli attuali sistemi informatici in grado di sfruttare le proprietà della meccanica quantistica per risolvere problemi. Conseguentemente alla nascita di tali calcolatori, è nata la computazione quantistica, necessaria per interfacciarsi con la nuova strumentazione e permettere di scrivere nuovi algoritmi per testare la prestanza delle macchine. Recentemente è nata una collaborazione tra CERN [2] e IBM [3] con lo scopo di utilizzare computer quantistici per l'analisi dei dati prodotti da LHC. Il primo esercizio per capire i possibili benefici di questa nuova tecnologia è l'elaborazione di modelli di apprendimento automatico applicati a temi di fisica delle particelle.

In questa tesi sono presentati i più recenti algoritmi di intelligenza artificiale quantistica applicati a problemi di Fisica delle Particelle. Successivamente viene presentato uno studio di intelligenza artificiale quantistica che effettua la classificazione dei *jets* prodotti da quark b e c analizzando eventi simulati dell'esperimento Large Hadron Collider beauty, LHCb [4]. L'algoritmo viene poi processato su un calcolatore quantistico IBM per la prima volta e si valutano i risultati da un punto di vista fisico e informatico.

La tesi ha la seguente struttura:

- Capitolo 1: breve introduzione del computer quantistico, e dei fondamenti della computazione quantistica, nello specifico della definizione di un registro di *qubits* e dell'utilizzo di porte logiche. Si presenta anche la nuova frontiera dell'intelligenza artificiale quantistica definendo la struttura dei modelli di appredimento automatico basati sulla computazione quantistica.
- Capitolo 2: rassegna degli algoritmi sviluppati, applicati o applicabili ad un reale problema di fisica delle particelle, che utilizzano un simulatore e/o un calcolatore quantistico.
- Capitolo 3: presentazione dello studio della classificazione di *jets* in base al sapore pesante dell'adrone iniziale (quark b vs quark c), mediante l'implementazione di un circuito quantistico. Analisi dei risultati ottenuti usando il computer quantistico IBM per fare la stessa classificazione. Quello descritto in questa tesi è il primo esercizio fatto su un computer reale per la classificazione dei *jets*.
- Capitolo 4: riassunto dei risultati ottenuti e discussione delle prospettive future.

Capitolo 1

Computazione Quantistica e Intelligenza Artificiale Quantistica

1.1 Computazione quantistica

Il recente sviluppo tecnologico ha aperto le porte ad una nuova frontiera di calcolatori, i computer quantistici. Essi nascono dall'idea del 1981 di R. Feynman che riteneva che per simulare i processi della fisica, come i sistemi quantistici, fosse necessario servirsi di strumenti che possedessero certe particolarità, i computer quantistici. Dal 1981 ad oggi, sono stati realizzati *hardware* di diverso tipo, basati su differenti proprietà della materia: fotoni singoli, cavità QED, ioni intrappolati, spin nucleari, superconduttività [5, 6]. Ad oggi sono due gli approcci predominanti: ioni intrappolati e superconduttività. Google [7] ha ideato e prodotto i processori "*Foxtail*" nel 2016,"*Bristlecone*" nel 2017 e "*Sycamore*" nel 2018, tutti fondati sulle proprietà dei superconduttori. D-Wave [8] ha costruito sistemi superconduttori basati sul *Quantum Annealing* [9]. Rigetti [10] ha realizzato diversi calcolatori, la serie "Aspen" e i computer "Acorn" e "Agave", fondati sui superconduttori. IBM ha costruito una serie di computer, e ne continuerà ad ideare, basati anch'essi sui superconduttori. Si riporta infine l'opera di Amazon [11] che ha portato alla luce "*Amazon Braket*", un servizio *cloud* per interfacciarsi con una schiera di computer quantistici basati su varie tecnologie quantistiche, come ioni intrappolati, superconduttori, *Quantum Annealing*.

Nel corso di questa tesi è stato utilizzato un calcolatore quantistico IBM, nello specifico il computer "Toronto" a 27 *qubits* e "Nairobi" a 7 *qubits*. Questi *hardware* sono tenuti a bassissima temperatura, circa un centesimo di grado sopra lo zero assoluto. A questa temperatura, i materiali utilizzati diventano superconduttori, ovvero gli elettroni circolano attraverso di essi senza resistenza. Attraversando i superconduttori, gli elettroni si legano condensando, nonostante la natura fermionica, formando delle "coppie di Cooper" [12]. Tali coppie trasportano carica elettrica attraverso le barriere, o materiali isolanti, per mezzo dell'effetto tunnel. Questi computer quantistici utilizzano come *qubits* (sez.1.1.1) le "giunzioni Josephson" [13], ovvero due superconduttori posti lateralmente a un isolante. Grazie all'invio di fotoni a microonde su queste giunzioni, è possibile controllarne il comportamento e modificare o leggere le unità di informazione.

In seguito a tale avanzamento tecnologico, è sopraggiunta la necessità di ideare una nuova tipologia di computazione, capace di sfruttare tali macchine e permettere un'evoluzione dei sistemi informatici. Nasce così la computazione quantistica.

1.1.1 *Qubits*

Il *qubit* [5, 6] è l'analogo quantistico del bit classico. È l'unità di informazione della computazione e può assumere i valori 0 o 1, come un bit, ma può rappresentare anche una qualsiasi combinazione di questi due valori, in accordo con il principio di sovrapposizione che vige per un oggetto quantistico. Tale proprietà consente dunque di gestire una quantità di informazione esponenzialmente maggiore

e permette con algoritmi quantistici di eleborare informazioni con un guadagno di risorse (tempo, memoria, energia) rispetto al caso classico.

Uno stato puro è rappresentativo di un sistema quantistico perfettamente conosciuto, descrivibile mediante una precisa funzione d'onda, e non rappresentabile con una miscela statistica di stati. Tale stato è esprimibile mediante un vettore a coefficienti complessi nello spazio di Hilbert o tramite la relativa matrice densità, l'operatore di proiezione nello stato del sistema [14].

Definendo con ψ il *qubit*, un generico stato puro è dato da

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad \alpha,\beta \in \mathbf{C} \quad |\alpha|^2 + |\beta|^2 = 1 \tag{1.1}$$

con la condizione di normalizzazione sui parametri complessi α, β. Lo stesso sistema si può inoltre descrivere, nella base computazionale $\{|0\rangle, |1\rangle\}$, mediante la matrice densità

$$\rho = |\psi\rangle\langle\psi| = \begin{pmatrix} \alpha\alpha^* & \alpha\beta^* \\ \beta\alpha^* & \beta\beta^* \end{pmatrix} \tag{1.2}$$

Equivalentemente, descrivendo i coefficienti complessi α, β come fasi, si ottiene la seguente espressione per lo stato del *qubit*, dove si inserisce un fattore di fase $e^{i\phi}$ per rendere il vettore il più generico possibile:

$$|\psi\rangle = \cos(\frac{\theta}{2})\,|0\rangle + e^{i\phi}\sin(\frac{\theta}{2})\,|1\rangle \qquad 0 \le \theta \le \pi, 0 \le \phi \le \pi \tag{1.3}$$

con la matrice densità ad esso associata

$$\rho = \begin{pmatrix} \cos^2(\frac{\theta}{2}) & \sin(\frac{\theta}{2})\cos(\frac{\theta}{2})e^{-i\phi} \\ \sin(\frac{\theta}{2})\cos(\frac{\theta}{2})e^{i\phi} & \sin^2(\frac{\theta}{2}) \end{pmatrix} \tag{1.4}$$

Le coordinate polari (θ, ϕ) permettono di individuare un vettore unitario con coordinate cartesiane:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \sin(\theta)\cos(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\theta) \end{pmatrix} \tag{1.5}$$

che può essere rappresentato nella sfera di Bloch, come in Fig. 1.1. Riscrivendo infatti la matrice definita in Eq. 1.4 in queste coordinate, sfruttando le regole di bisezione e duplicazione per le funzioni seno e coseno e le matrici di Pauli, si ottiene:

$$\rho = \frac{1}{2}(\mathbb{1} + x\hat{\sigma_x} + y\hat{\sigma_y} + z\hat{\sigma_z}) \tag{1.6}$$

Ponendo $\vec{r} = (x, y, z)$, si trova il vettore unitario che punta alla superficie della sfera di Bloch, concordemente con il fatto che si sta trattando uno stato puro. In generale, vettori di stato che differiscono per un fattore di fase globale sono rappresentativi dello stesso sistema, dunque gli stati dei *qubits* sono più precisamente raggi vettori.

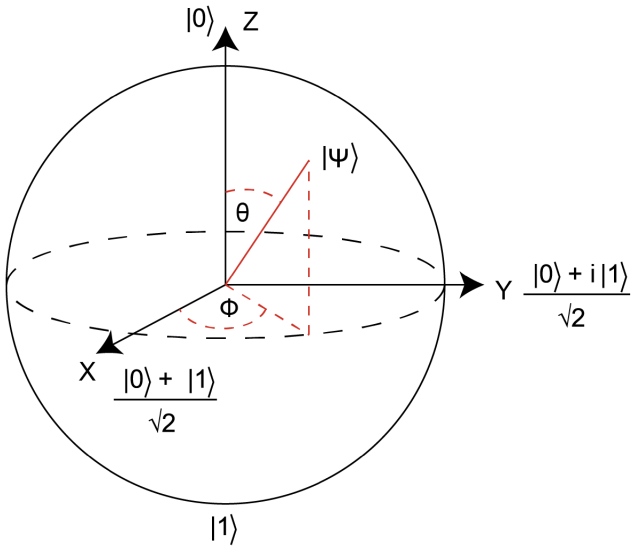


Figura 1.1: Rappresentazione del vettore di stato nella Sfera di Bloch

A livello computazionale, il *qubit* racchiude la stessa quantità di informazione di un bit classico: infatti nonostante possa assumere stati che sono la sovrapposizione dei valori binari, al momento della misura si verifica il collasso della funzione d'onda del bit quantistico, che conduce inevitabilmente a $|0\rangle$ oppure $|1\rangle$. Per ricavare tutta l'informazione disponibile nello stato del *qubit*, si potrebbe ricorrere a tecniche di tomografia, effettuando un gran numero di misure e ricostruendo gli esatti coefficienti della funzione d'onda. Tuttavia tale procedimento conduce a un'informazione proporzionale al numero di misurazioni e si ricadrebbe dunque ad un sistema classico a molti bit.

I *qubits* sono la parte fondante di un computer quantistico, poichè permettono di immagazzinare ed estrarre informazione. Un *computer quantistico* dunque deve essere in grado di possedere un registro di *qubits* e di poter agire su di essi. Per realizzare un tale calcolatore è necessario rispettare determinati criteri, ideati da "Di Vincenzo" [15] nel 2000 :

1. Scalabilità: realizzazione di *qubits* ben definiti e replicabili
2. Inizializzazione: possibilità di creare stabilmente stati iniziali del tipo $|000...0\rangle$
3. Tempi di coerenza lunga: definito il tempo di decoerenza dei *qubits* τ_d e la durata media necessaria per effettuare un'operazione con un *gate* τ_g, deve valere $\frac{\tau_d}{\tau_g} >> 1$. Ottimale sarebbe riuscire ad ottenere $\frac{\tau_d}{\tau_g} > 10^4$ per poter implementare anche codici di correzione degli errori.
4. Set universale di *gates*: insieme di porte logiche necessarie per eseguire una generica computazione logica
5. Misure efficienti: possibilità di misurare i *qubits* in maniera affidabile

È desiderabile inoltre riuscire a stabilire comunicazioni quantistiche, per cui le informazioni si trasmettono tramite *qubits* mobili (detti "volanti"), tipicamente fotoni. Si richiedono per tale motivo due criteri aggiuntivi:

6. Interfaccia tra *qubits* stazionari e *qubits* "volanti"
7. Capacità di trasmettere in modo affidabile *qubits* "volanti" tra due luoghi

Un computer quantistico si basa perciò su inizializzare un registro di n *qubits* e farlo evolvere tramite opportuni operatori unitari. In un generico istante, dunque, il registro si trova in uno stato dato dalla combinazione dei prodotti tensori della base computazionale $|k_i\rangle = \{|0\rangle, |1\rangle\}$:

$$|\psi\rangle = \sum_{\vec{k}} c_{k_1,...,k_n} \bigotimes_{i=1}^{n} |k_i\rangle = \sum_{\vec{k}}^{n} c_{k_1,...,k_n} |k_1, ..., k_n\rangle$$
$$c_{\vec{k}} \in \mathbf{C} \quad \sum_{\vec{k}} |c_{\vec{k}}|^2 = 1 \tag{1.7}$$

dove i coefficienti complessi $c_{\vec{k}}$ soddisfano alla condizione di normalizzazione.

1.1.2 *Gates*

Un qualsiasi operatore che agisce su un tale stato di n *qubits* può essere scomposto nell'azione combinata di porte logiche quantistiche fondamentali, i *gates* [5, 6].

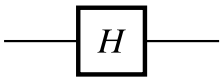
Esistono diversi *gates* disponibili, alcuni sono recuperabili dall'analogo classico come l'identità $\mathbb{1}$ e il NOT, altri sono invece descrivibili solo quantisticamente. Nella computazione quantistica non è possibile implementare la porta COPY, in quanto vige il "*No Cloning Theorem*" [16] che non permette di copiare uno stato generico a meno che non sia un elemento della base computazionale o un vettore ortogonale. Come le porte logiche classiche, anche quelle quantistiche presentano errori. Con l'attuale livello di tecnologia, la probabilità che si verifichi un errore non è sufficientemente piccola. Dunque algoritmi che presentano un'elevata quantità di porte sono maggiormente soggetti a sbagliare. Esistono codici di correzione degli errori, ma anche essi presentano *gates*, perciò il loro utilizzo deve essere contenuto. In alcuni casi si è raggiunta la *Fault tolerant quantum computing*, ossia un'architettura di computazione resistente ad errori generali.
I *gates* si suddividono in due categorie: porte logiche a 1 *qubit* e porte logiche a 2 *qubits*. Si riportano degli esempi per ciascuna categoria.

Porte a 1 *qubit*

Le porte a 1 *qubit* sono matrici unitarie 2×2 solitamente espresse nella base computazionale $\{|0\rangle, |1\rangle\}$ in notazione matriciale:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{1.8}$$

- Hadamard
  Permette di creare la sovrapposizione degli stati del *qubit* a pari coefficienti. L'azione del *gate* è quella di passare dalla base degli autostati di $\hat{\sigma_z}$, $\{|0\rangle, |1\rangle\}$, a quella di $\hat{\sigma_x}$, $\left\{|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}, |-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}\right\}$:

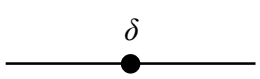
  

  $$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1.9}$$

  Applicando H ad un vettore della base computazionale si ottiene:

  $$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$$
  $$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \tag{1.10}$$

- Phase-shift
  Permette di aggiungere una fase relativa nella funziona d'onda del *qubit*. È equivalente ad una rotazione di un angolo δ attorno all'asse definito come z nella sfera di Bloch:

  $$R_z(\delta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\delta} \end{pmatrix} \tag{1.11}$$

  Applicando il *gate* ad un vettore generico $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ si ottiene:

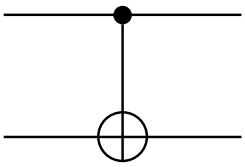
  $$R_z(\delta)|\psi\rangle = \alpha|0\rangle + e^{i\delta}\beta|1\rangle \tag{1.12}$$

Porte a 2 *qubits*

Le porte a 2 *qubits* sono matrici unitarie 4×4, solitamente espresse nella base computazionale $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ così definita:

$$|00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \quad |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix} \quad |10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \quad |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix} \tag{1.13}$$

- CNOT
  È un *gate* NOT controllato dallo stato di un altro *qubit*, detto per questo *qubit* "di controllo". Nella scrittura di uno stato il *qubit* di controllo usualmente ricopra le prima posizione.
  Se il *qubit* di controllo si trova nello stato $|0\rangle$ allora il CNOT agisce come identità sull'altro *qubit*.
  Se il *qubit* di controllo si trova nello stato $|1\rangle$ allora il CNOT agisce tramite un NOT sull'altro *qubit*.



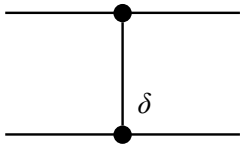
$$CNOT = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix} \tag{1.14}$$

L'applicazione del CNOT su uno stato di due *qubits* si realizza tramite:

$$\begin{aligned} CNOT(|0\rangle \otimes |0\rangle) = CNOT\,|00\rangle = |0\rangle \otimes \mathbb{1}\,|0\rangle = |00\rangle \\ CNOT(|1\rangle \otimes |0\rangle) = CNOT\,|10\rangle = |1\rangle \otimes NOT\,|0\rangle = |11\rangle \end{aligned} \tag{1.15}$$

Questo *gate* è la base per creare entanglement fra due *qubits*.

- CPHASE
  È un phase shift controllato dallo stato di un altro *qubit*. Permette di effettuare una rotazione di un angolo ϕ attorno all'asse z della sfera di Bloch, se il *qubit* di controllo è $|1\rangle$, altrimenti se è $|0\rangle$ agisce come identità.



$$CPHASE = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&1&e^{i\phi}\end{pmatrix} \tag{1.16}$$

Applicando CPHASE ad uno stato di due *qubits* si ottiene:

$$\begin{aligned} CPHASE\,|00\rangle = |0\rangle \otimes \mathbb{1}\,|0\rangle = |00\rangle \\ CPHASE\,|10\rangle = |1\rangle \otimes R_z(\delta)\,|0\rangle = e^{i\phi}\,|10\rangle \end{aligned} \tag{1.17}$$

1.1.3 *Entanglement*

L' *entanglement* è un fenomeno di natura quantistica, non presenta un analogo classico. Stati *entangled* possiedono delle specifiche correlazioni, quantistiche e di natura non locale, che non possono essere rappresentate da altre tipologie di stati, gli stati separabili.
Presi A e B due sistemi e considerati stati non correlati, $\{\rho_i^A, \rho_i^B\}$, presi con certe probabilità classiche $\{p_1, ..., p_k\}$, un generico stato del sistema composto è separabile se è descrivibile tramite la matrice densità:

$$\rho = \sum_{i=1}^{k} p_i \rho_i^A \otimes \rho_i^B \tag{1.18}$$

Tali stati presentano solo correlazioni classiche, ovvero producibili da *Local operations & Classical Communications* (LOCC). Considerati i sistemi A e B presso due laboratori, si generano le suddette correlazioni con una sequenza di operazioni locali sui sistemi e con la possbilità di comunicazioni classiche tra i due. Gli stati *entangled* invece presentano correlazioni di natura quantistica, che non sono replicabili con operazioni locali sui sottosistemi. Per riconoscere stati *entangled* si può ricorrere alla "Decomposizione di Schmidt" ed il "Rango di Schmidt" [5, 6], al "Criterio di Peres-Horodecki" [17, 18] oppure direttamente alle misure di *entanglement*. Esempi di tali stati sono gli stati di Bell, stati massimamente *entangled*:

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \qquad |\psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{1.19}$$

$$|\phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle \qquad |\psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \tag{1.20}$$

Per produrre *entanglement* tramite porte logiche, si ricorre al *gate* CNOT. Considerando inizialmente uno stato generico a 2 *qubits* $|\psi\rangle = (\alpha\,|0\rangle + \beta\,|1\rangle) \otimes |0\rangle$, si genera uno stato *entangled*:

$$|\psi\rangle \to CNOT(|\psi\rangle) = \alpha\,|00\rangle + \beta\,|11\rangle \tag{1.21}$$

In generale, come segue dalla decomposizione di Schmidt, uno stato *entangled* è esprimibile come una somma di prodotti tensore, mentre gli stati separabili come un unico prodotto.

L'*entanglement* entra in conflitto con l'idea di località ed evidenzia come la meccanica quantistica sia necessariamente una teoria non locale.

1.2 Intelligenza artificiale quantistica

Il *Quantum Machine Learning (QML)* è un ramo dell'intelligenza artificiale che riprende i passi dell'apprendimento automatico classico, sfruttando però le leggi della meccanica quantistica. Lo scopo è rendere la macchina, il computer, capace di imparare a risolvere un problema direttamente dai dati. Gli algoritmi di apprendimento automatico sono composti da due fasi, una prima detta di *training*, in cui il calcolatore parte da un grande insieme di dati e sfruttando le loro caratteristiche (dette *features*) impara a risolvere un determinato problema, e una seconda detta di *testing*, in cui vengono valutate le abilità di risoluzione del problema su un altro *dataset* con le stesse *features* ma differenti eventi. Se tra le *features* disponibili per l'allenamento sono presenti anche le soluzioni del problema, allora il modello viene detto "supervisionato", altrimenti si parla di modello "non supervisionato".

L'implementazione quantistica si fonda sulla nuova tipologia di computazione, costruendo un circuito basato su *qubits*, porte logiche parametrizzate e operatori di misura.

Tramite un registro di *qubits*, si codificano i dati x, le *features* scelte per risolvere il problema, in un insieme di parametri del circuito. Tale procedimento produce uno stato quantistico $|x\rangle$ rappresentativo dei dati in input. Esistono diversi metodi di codifica, si presentano i due utilizzati nel corso di questa tesi:

- *Amplitude Embedding*: il modello [19] consiste nel definire un circuito quantistico parametrizzato ed utilizzare le ampiezze del vettore di stato del *qubit* per immagazzinare le informazioni nei coefficienti della funzione d'onda.
  Si possono codificare fino a 2^n *features* nelle ampiezze di uno stato di n *qubits* o equivalentemente un vettore di N *features* può essere codificato utilizzando $\log_2 N$ *qubits*:

  $$|x\rangle = \sum_{i=1}^{2^n} x_i\,|k_i\rangle \qquad , \qquad \sum_{i=1}^{2^n} |x_i|^2 = 1 \tag{1.22}$$

  dove x_i è la i-esima *features*, $|k_i\rangle$ è un vettore della base computazionale e si è imposta la condizione di normalizzazione sul vettore $|x\rangle$.
  Se il numero di *features* non è una potenza di 2, le rimanenti ampiezze vengono poste costanti.

- *Angle Embedding*: il modello [19] permette di codificare le *features* negli angoli di rotazione di *gates* rotazionali. Tuttavia, è necessaria una corrispondenza uno a uno tra *qubits* e features in input, il che rende poco pratico questo metodo di codifica per collezioni di dati di grandi dimensioni.

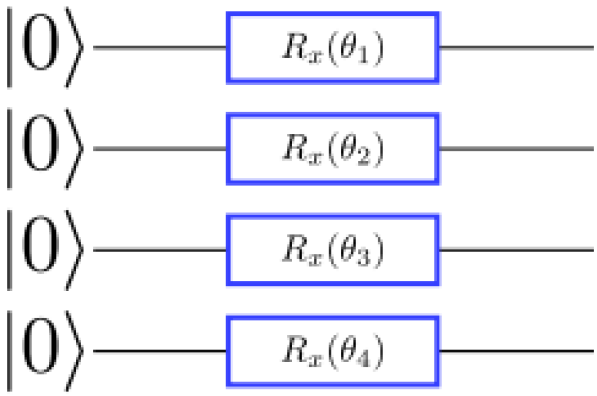


Figura 1.2: Circuito rappresentativo di un *Angle Embedding*. [20]

Successivamente, i dati codificati vengono passati a un circuito parametrizzato, composto da porte logiche dipendenti da parametri ottimizzabili nella fase di "allenamento". Tipicamente il circuito si ripete, realizzandosi in vari strati di *gates*, detti *layer*, ognuno con parametri modificabili nel corso del *training*. Esistono diverse tipologie di circuiti per la fase di ottimizzazione, nel corso di questa tesi è stato utilizzato un circuito variazionale parametrizzato (vedi capitolo 3).

L'ottimizzazione avviene ricercando i parametri che minimizzano un'opportuna funzione, detta *Loss Function*, la quale misura la discrepanza tra le predizioni e i valori conosciuti. Si spera di ottenere il minimo valore della funzione al fine di garantirsi la minor perdita sul *dataset* studiato. È complesso ricercare il minimo, in quanto non sempre si è in grado di trovare quello globale, ma piuttosto un minimo locale o una sella. Ciò conduce ad una non efficiente minimizzazione della funzione e la perdita del modello non è la minima come auspicato.
I parametri ottimizzati vengono infine utilizzati per fornire delle predizioni e verificare l'accuratezza dell'algoritmo su un nuovo *dataset* compredente eventi non ancora analizzati. Le predizioni vengono estratte misurando i *qubits* e collassando dunque la loro funzione d'onda su un valore definito.

Capitolo 2

Metodi di Intelligenza Artificiale Quantistica per la Fisica delle Alte Energie

Esistono differenti problemi nella Fisica delle Alte Energie per cui è stato possibile ricorrere a metodi di intelligenza artificiale classica. Ora l'obiettivo è affrontare tali problemi con la nuova tecnologia quantistica, esplorando la realizzabilità dei modelli di QML [21].

2.1 Problemi di Tracciamento

Negli accelatori di particelle, in seguito alle collisioni, vengono prodotte in ogni direzione molte particelle che vengono successivamente misurate dagli esperimenti tramite dei sensori che originano un segnale. Nel caso di particelle cariche, è possibile ricostruire la loro traccia.
Il tracciamento di una particella consiste nel ricostruire la sua traccia a partire dai segnali elettrici rilasciati nei sensori dell'esperimento. Dato l'elevato numero di segnali lasciati in una tipica collisione protone-protone, si deve ricorrere a metodi che permettano una ricostruzione delle tracce veloce e affidabile.

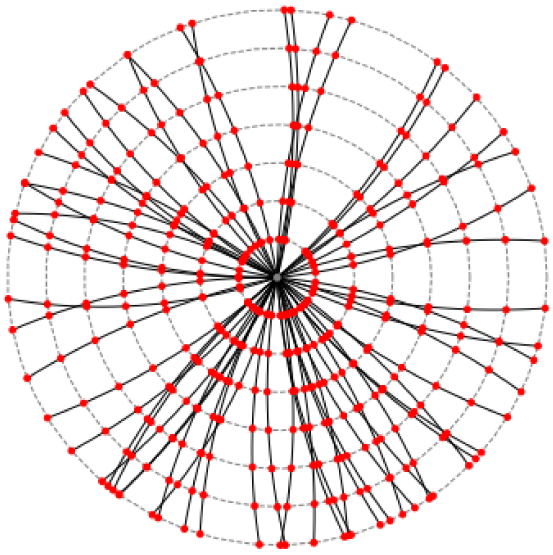


Figura 2.1: Problema di Tracciamento: visione trasversa del rivelatore tracciante a strati cilindrici (cerchi grigi tratteggiati) in un tipico evento simulato. I segnali lasciati dalle particelle sono rappresentati dai pallini rossi, la traiettoria (linea nera) si ricostruisce raggruppando i segnali corrispondenti alla stessa particella. [21]

Tra i vari metodi di tracciamento, il metodo combinatorio di Kalman Filter [22] è alla base dei programmi di ricostruzioni in Fisica delle Alte Energie. L'articolo [23] ha analizzato tale tecnica e ne ha individuato quattro routines principali: *seeding*, ricostruzione della traccia, pulizia e selezione. Propone inoltre un'implementazione quantistica.
Nella prima fase si identificano delle tracce grezze, con pochi segnali rilasciati nei sensori, e le si propongono come candidate tracce. Successivamente nella "ricostruzione della traccia", si estrapola il percorso della particella cercando i segnali compatibili e aggiungendo man mano quelli dei detector successivi. Per evitare di avere tracce multiple riferite alla stessa particella, nella fase di pulizia si scartano determinati candidati. Infine si sceglie la migliore, ovvero quella che rispecchia determinati criteri basati sul fit della traiettoria e dei corrispondenti segnali.
Viste le problematiche sorte dagli studi con algoritmi classici, come la complessità del problema e le notevoli risorse computazionali richieste, è stata proposta un'implementazione quantistica di questo metodo, basata su una routine quantistica conosciuta come *Amplitude amplification* [24], ovvero una generalizzazione dell'algoritmo di "Grover" [25]. Questa permette di risolvere problemi di ricerca non strutturati in tempo polinomiale e nei problemi di tracciamento può essere utilizzata per trovare segnali con specifiche proprietà molto più velocemente e con un grande risparmio di risorse computazionali.

Un ulteriore metodo proposto consiste nell'utilizzare il *Quantum Annealing* (Sez. 2.4), ovvero una tecnica che permette di risolvere problemi di ottimizzazione combinatori codificando la soluzione del problema nello stato fondamentale di una certa Hamiltoniana.
Per quanto riguarda il problema di ricostruzione delle tracce, è stato formulato un algoritmo [21] che utilizza un'ottimizzazione binaria non limitata, la quale può essere ricondotta ad un *Quantum Annealer*.

2.2 Problemi di classificazione

In Fisica delle Particelle, una volta raccolti i dati provenienti dai rivelatori, si effettua la fase di analisi dati. È cruciale sapere operare distinzioni nei dati, come per esempio distinguere un segnale d'interesse dal fondo oppure discriminare gli eventi per assegnare il nome alle particelle rivelate. Gli strumenti di classificazione sono dunque una parte fondamentale nel processo di studio di fenomeni di Fisica delle Particelle.

2.2.1 Circuiti variazionali quantistici

I *Variational Quantum Circuits (VQC)* sono algoritmi ibridi classico-quantistici che mirano a sfruttare la potenza e la scalabilità di entrambi i paradigmi. Calcolatori classici vengono infatti utilizzati per risolvere problemi di ottimizzazione, mentre computer quantistici sono usati per compiti specifici, come il calcolo di determinati valori di aspettazione.
Un VQC viene utilizzato per classificare dati, dopo aver appreso da un insieme di dati di allenamento (*dataset di training*), oppure per modellizzare correlazioni nei dati stessi. La verifica delle prestazioni e il computo della relativa efficienza avviene nella fase detta di *testing*.

Il circuito si compone di una parte di codifica e una parte parametrizzata, in figura sotto la parte di codifica è rappresenta dal primo blocco con $U_\Phi(\vec{x})$, la parte parametrizzata dal riquadro blu con $U_{rot}(\vec{\theta})$ e U_{ent}.

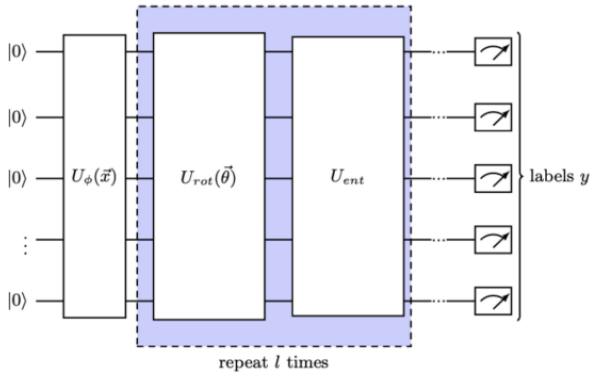


Figura 2.2: Circuito variazionale quantistico: a sinistra il blocco di matrici unitarie per la codifica dei dati $U_\Phi(\vec{x})$, nel riquadro blu *layers* ripetibili di porte logiche dipendenti da parametri $U_{rot}(\vec{\theta})$ e generanti entanglement U_{ent}, a destra operatori di misura. [21]

Tramite opportuni blocchi di porte logiche i dati sono inizialmente codificati in stati quantistici, i quali vengono poi passati ad insiemi di *gates*, dipendenti da parametri da ottimizzare nella fase di "allenamento", che avviene minimizzando un'opportuna *loss function*. I VQC presentano dunque uno stadio variazionale, ovvero un circuito quantistico parametrizzato, contenente porte logiche a singolo e doppio *qubit* (in fig.2.2 è rappresentato dal riquadro blu). L'ottimizzazione può avvenire tramite metodi differenti, particolarmente usato è il metodo del gradiente, come l'ADAM [26]. Infine, per estrarre un risultato, si misura un'osservabile, per esempio l'operatore di Pauli σ_Z, sugli stati dei *qubits* collassandone la funzione d'onda.
Esistono varie geometrie di circuiti utilizzati per codificare uno stato quantistico, tra cui l'*Angle Embedding* e l'*Amplitude Embedding* (sez.1.2).
Nella fase di *training*, i dati vengono utilizzati per allenare l'algoritmo. Tipicamente i *datasets* vengono suddivisi in piccoli lotti (detti *batch*) e in ognuno si applica un processo di ottimizzazione. Con il gradiente della *loss function*, vengono aggiornati i parametri del modello, un'epoca di *training* termina quando sono stati analizzati tutti i lotti. La procedura completa risulta conclusa una volta che tutte le epoche di addestramento sono state eseguite, fino ad ottenere una sufficiente minimizzazione della *loss function*. Nella fase di *testing*, si applica il modello ad un nuovo insieme di dati e si estrae un risultato dal classificatore.

Nella maggior parte dei casi le prestazioni di un circuito variazionale quantistico sono valutate tramite una funzione detta ROC, *Receiver Operating Characteristics curve*, e l'area sotto la curva associata, *Area Under the Curve*, AUC. La ROC rappresenta il rateo di "veri positivi" contro il rateo di "falsi positivi", in un problema di classificazione binaria.

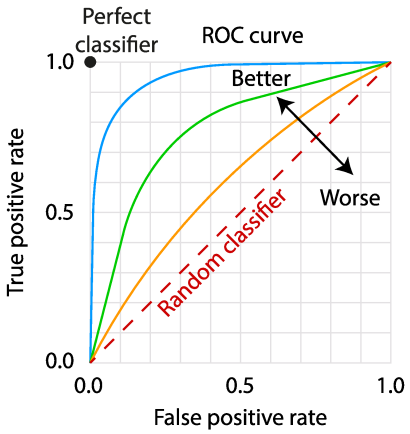


Figura 2.3: Esempi di ROC

Il tasso di veri positivi (TPR, *True Positive Rate*) è conosciuto come "sensibilità", il tasso di falsi positivi (FPR, *False Positive Rate*) è detto "*fall-out*" e si calcola come $1 - specificità$ (dove la specificità è la capacità di identificare correttamente):

$$TPR = \frac{TP}{TP + FN} \tag{2.1}$$

$$FPR = \frac{FP}{FP + TN} \tag{2.2}$$

Nelle formule, TP, *True Positive*, sono i veri positivi; FP, *False Positive*, sono i falsi positivi; TN, *True Negative*, sono i veri negativi; FN, *False Negative*, sono i falsi negativi.

Lo scenario migliore si verifica quando sia la sensibilità che la specificità tendono a 1, ovvero quando la curva si avvicina al vertice $(0, 1)$.

2.2.2 Applicazioni

In Fisica delle particelle, l'impiego più comune per questo tipo di algoritmo di intelligenza artificiale è la classificazione di *datasets* provenienti da esperimenti. Nella rassegna dell'articolo [21] si illustrano le molteplici applicazioni, a titolo d'esempio si riporta l'applicazione inerente all'argomento di questa tesi:

- Classificazione di produzione di coppie di chargino tramite un bosone di Higgs contro fondo del modello standard (SM) [27]
- Classificazione di $t\bar{t}H$ in cui vengono prodotti una coppia di quark top con un bosone di Higgs [28]
- Tecniche di ri-caricamento dei dati [29]
- Classificazione di eventi $t\bar{t}$ contro il resto dei processi del SM usando solo due *features* cinematiche [30]
- Analisi della produzione di bosoni di Higgs in associazione con coppie di quark top e decadimento del bosone di Higgs in due muoni [31]
- Identificazione del sapore dei *jets* prodotti nelle collisioni protone-protone nell'esperimento LHCb [20].

  Si intende definire il tipo di quark che origina il jet: b vs $\bar{b}$.
  Si utilizza un VQC per l'identificazione del quark e la sua *performance* viene comparata con una rete neurale standard e con un particolare algoritmo sviluppato ad LHCb, l'algoritmo *muon tagging*, che deduce il sapore del flusso di particelle per mezzo della carica del muone che si trova al suo interno. I dati simulati sono analizzati e informazioni sulla sottostruttura del jet sono utilizzate come *features* in input per il classificatore. Per ogni *jet* cinque tipi di particelle con il maggior impulso trasverso p_{T} sono scelte: muone, kaone, elettrone, pione e protone. Per ogni particella si considerano tre variabili (vedi fig.3.1): la quantità di moto trasversale rispetto all'asse del *jet* p_{T}^{rel}; la carica q della particella; la distanza ΔR rispetto all'asse del *jet* nel piano η-ϕ, con η pseudo-rapidità definita come $\eta = -\log[\tan(\frac{\theta}{2})]$ in cui θ è l'angolo tra il momento e l'asse z, e con ϕ definito come l'angolo azimutale tra la proiezione del momento nel piano xy e l'asse y. Si considera inoltre una variabile globale, la carica ponderata del flusso di particelle Q.
  Il circuito presenta un registro di n *qubits* in cui si immagazzinano n *features* tramite due geometrie, l'*Angle Embedding* e l'*Amplitude Embedding*, seguite da *layers* generanti un forte entanglement, composti da porte logiche rotazionali e CNOT. Al fine di comprendere l'importanza dei dati in input si sono utilizzati due distinti dataset: il *muon dataset*, che utilizza le variabili del muone all'interno del flusso e la carica ponderata Q, e il *dataset* completo che utilizza tutte le variabili.
  I risultati [20] mostrano che la geometria con l'*Angle Embedding* funziona meglio di quella con l'*Amplitude Embedding*, raggiungendo le stesse prestazioni del DNN per il *muon dataset*, mentre per la collezione completa di dati funziona ancora meglio il DNN. Si nota inoltre che l'accuratezza della classificazione cresce all'aumentare del numero di *layers* parametrizzati fino a saturare

ad un certo valore. In base ad analisi sul numero di eventi di *training*, i classificatori quantistici hanno migliori prestazioni con pochi eventi rispetto a quelli classici, mentre raggiungono simili *performance* con un numero di eventi più elevato. Da simulazioni con rumore si nota infine che l'accuratezza non peggiora, lasciando immaginare che questi circuiti siano sufficientemente robusti da poter girare anche su reali calcolatori.

2.2.3 *Quantum support vector machine* (Q-SVM)

Questa tipologia di modelli condivide molte caratteristiche con i circuiti variazionali quantistici. In entrambi i paradigmi, stati classici di input sono codificati in spazi di Hilbert tramite opportune unitarie. La principale differenza risiede in come gli stati sono trattati una volta inizializzati.
In questa architettura il problema di classificazione dei dati avviene separando gli input tramite opportuni iperpiani ("*kernel*") autonomamente ottimizzati dal calcolatore.
Le QSVM mappano un determinato vettore di input $\vec{x}$ in uno spazio delle *features* a dimensione maggiore, in cui si misurano le somiglianze tra due qualunque istanze, dette "*kernel entries*" $k(\vec{x_i}, \vec{x_j})$. L'algoritmo dunque ottimizza i parametri di un iperpiano che separa i dati e li distingue in due categorie. Una grossa limitazione delle QSVM è la valutazione di queste *kernel entries* in un grande spazio delle *features* poichè risulta particolarmente dispensiosa dal punto di vista computazionale. Per ovviare al problema, si può utilizzare direttamente lo spazio degli stati quantistici come rappresentazione dello spazio delle *features*, definendo funzioni *kernel* difficili da calcolare classicamente. Risulta che le mappe delle *features* potrebbero condurre potenzialmente a esiti di classificazione migliori.

2.2.4 Applicazioni

Nella rassegna [21] si riportano discusse tutte le applicazioni:

- Classificazione di $t\bar{t}H$ in cui vengono prodotti una coppia di quark top con un bosone di Higgs [28].
- Discriminazione degli eventi $t\bar{t}H$ nel canale di decadimento $H \to \gamma\gamma$ da eventi di produzione non risonante di due fotoni [32].
- Classificazione dei segnali di fondo nel decadimento dei mesoni B [33].
- Classificazione di dati 67 dimensionali di una supernova con *hardware* da 17 *qubits* [34].

2.2.5 *Quantum Convolutional Neural Networks* (QCNN)

È lo spazio di lavoro che utilizza un VQC per eseguire dei calcoli di convoluzione in una classica rete neurale.
I filtri di convoluzione o i *kernel* sono rimpiazzati da VQC per sfruttare la potenzialità dell'*entanglement*. Tramite *kernel* quantistici di convoluzione che effettuano misurazioni, i dati vengono trasformati in un vettore di rappresentazione con dimensione minore e blocchi VQC assicurano di ottenere vettori di *features* di varie scale di lunghezza in differenti *layers*.

Le QCNN sono particolarmente utilizzate in problemi di classificazione di immagini.

- Classificazioni di eventi di neutrino rivelati in una camera a proiezione temporale ad argon liquido [35].
- Confronto tra modelli su dati sparsi [35, 36].

2.3 Generazione di dati

Un ruolo fondamentale in Fisica delle Alte Energie è ricoperto dalle simulazioni dei processi e delle interazioni che avvengono negli accelatori. Queste permettono di studiare gli eventi, prevedere le risposte del detector e pianificare eventuali aggiornamenti e miglioramenti da apportare agli apparati di rivelazione e misurazione. In Fisica delle particelle elementari, le simulazioni si articolano su diverse

fasi, necessarie alla corretta descrizione dei fenomeni sub-nucleari: descrizione della Lagrangiana, previsione dell'*hard scattering* e *parton shower*, creazione di un modello di adronizzazione basato su misurazioni affidabili, computo della completa risposta del *detector*.

Tipicamente, le simulazioni sono prodotte tramite tecniche Monte-Carlo, tuttavia sono molto dispendiose dal punto di vista computazionale, occupando una significativa parte delle risorse disponibili, come l'uso della CPU e lo spazio su disco. Attualmente le migliori previsioni richiedono diversi minuti di attesa per una singola collisione, con numerosi eventi che devono essere generati in media per ogni campagna di simulazione presso LHC.
Modelli di generazione di campioni hanno l'obbiettivo di generare nuovi campioni di dati senza ricorrere alle misurazioni. Essi devono essere in grado di apprendere dai dati, rappresentare e campionare da distribuzioni di probabilità a grande dimensione. Molti di questi modelli usati su macchine classiche si basano su reti neurali e sono parametrizzate con pesi e *bias*: macchine di Boltzmann, auto-codificatori variazionali, *Generative Adversarial Network* (GAN). Questa tipologia di problemi si presta particolarmente bene alla computazione quantistica, grazie alla capacità di rappresentare vettori in uno spazio n-dimensionale utilizzando $log(n)$ *qubits* e di gestire matrici di basso rango in tempo $O(poly(log(n)))$.
Ognuno dei modelli sopra citati presenta un analogo quantistico.

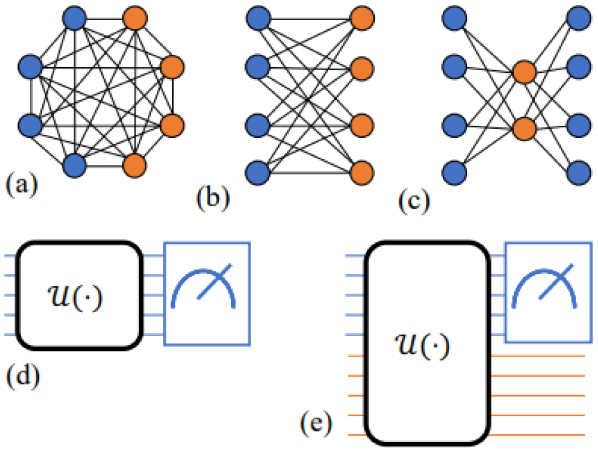


Figura 2.4: Modelli generativi classici che utilizzano una rete neurale con unità visibili (blu) e nascoste (arancioni): (a): Macchina di Boltzmann, (b): Macchina di Boltzmann ristretta, (c): Auto-codificatore (Variazionale). Modelli generativi quantistici che utilizzano unitarie parametrizzate ($U(\cdot)$): (d): Circuito unitario parametrizzato (e): Circuito unitario parametrizzato con misure parziali. [21]

2.3.1 Macchine di Boltzmann

Algoritmi capaci di generare campioni di dati simili a quelli utilizzati nella fase di allenamento.
Una rete composta da spin è interconnessa con pesi sui bordi e *bias* in loco, che sono ottimizzati nella fase di *training* per trovare lo stato fondamentale dell'Hamiltoniana del sistema.
Le macchine di Boltzmann calzano in modo naturale con la tecnologia basata sul *Quantum Annealing* e mostrano un guadagno con tempi polinomiali rispetto al caso classico [21]. I neuroni della rete vengono rimpiazzati con i *qubits* e l'Hamiltoniana viene tradotta in un circuito unitario o in un modello di Ising.

2.3.2 Auto-codificatori variazionali

Un auto-codificatore variazionale (*Variational auto-encoder*, VAE) ha l'obiettivo di mappare un vettore di dati in input in un vettore in uno spazio compresso, il quale tipicamente viene detto "spazio latente". La rete che comprime l'input è il codificatore, al contrario la rete che permette di ricostruire il vettore iniziale viene detto decodificatore. La *loss function* è l'errore di ricostruzione del vettore iniziale e deve essere minimizzata. Non ci sono restrizioni alla tipologia di variabili che possono appartenere allo spazio latente. Per questo motivo tale tecnologia è ampiamente utilizzata, nonostante non permetta

di avere una struttura, un'organizzazione dei dati. Gli auto-codificatori variazionali mirano a risolvere questo problema, restituendo una distribuzione semplice dallo spazio latente, generalmente Gaussiana aggiungendo termini aggiuntivi nella *loss function*.
L'implementazione quantistica, auto-codificatori variazionali quantistici (in ing. qVAE), è costituita da modelli che tramite porte logiche modificano la funzione d'onda della rete e permette di generare campioni di dati efficientemente.

2.3.3 *Quantum Generative Adversarial Network* (qGAN)

L'architettura si articola attraverso delle componenti di base: un generatore, un discriminatore e una procedura di addestramento "avversario".
Il generatore trasforma campioni di dati da qualche dataset di partenza in nuovi campioni di dati, generati dalla macchina e per questo detti "finti". Il generatore dunque mappa la distribuzione di probabilità $p_{iniziale}$ a una nuova distribuzione p_{finta} di dati generati. Il discriminatore prende in input campioni di dati, dalla distribuzione iniziale, e cerca di distinguerli dai dati generati precedentemente. La procedura di addestramento "avversaria" si basa su un "gioco tra due avversari" in cui due reti si affrontano e il guadagno di uno è la perdita dell'altro. Questa tecnica permette di generare un set di dati con la stessa statistica di quello di allenamento. In questa rete, il generatore che propone campioni di dati si alterna al discriminatore che cerca di riconoscere i campioni generati. Il processo si conclude quando si è raggiunto un equilibrio stabile durante le epoche di addestramento ed il generatore propone un nuovo e robusto campione di dati. Tipicamente la *loss function* è la *cross-entropy* binaria.
Il potenziale di questa tecnologia fusa con la computazione quantistica è notevole e rappresenta un'interessante prospettiva per il futuro. In Fisica delle particelle elementari, due applicazioni spiccano per l'elevato dispendio di risorse: la simulazione di rivelatori (e dei calorimetri) e la generazione con tecniche Monte-Carlo degli eventi nei vari processi fisici.

- Simulazione dei rivelatori

  Attualmente, le simulazioni tramite Monte-Carlo richiedono molto tempo e spesa di svariate risorse computazionali, si è cercato dunque di utilizzare tecniche quantistiche per velocizzare i processi di simulazione.
  Prototipi di algoritmi basati su una rete generativa avversaria (GAN) raggiungono alti livelli di accuratezza e producono dati realistici. Un esempio sono gli ibridi quantistici GAN, in cui un generatore quantistico è allenato contro un discriminatore classico.

  Dual Parametrized Quantum Circuit (PQC) GAN [37, 38, 39]

  È composto da un discriminatore classico e due generatori quantistici, sotto forma di circuiti quantistici parametrizzati, che sono in grado di riprodurre immagini a pixel di dimensioni ridotte dei volumi dei calorimetri. Il modello è stato allenato per riprodurre distribuzioni energetiche monodimensionali, generate da singoli elettroni nel calorimetro. I risultati sono stabili sia nelle simulazione che nella vera macchina.
  Con la stessa tecnica, è stato testato anche un *hardware* che sfrutta la computazione quantistica basata sui fotoni, la quale ha il vantaggio di permettere una rappresentazione naturale delle variabili continue negli stati quantistici.

- Generazione di eventi Monte-Carlo

  L'uso di modelli di generazione di eventi per le simulazioni basati su reti avversarie è stato l'obiettivo principale nell'apprendimento automatico classico. L'implementazione quantistica è ora la nuova frontiera. Si riporta un'applicazione di questa tecnologia: processo di produzione $pp \to t\bar{t}$ presso LHC con $\sqrt{s} = 13$ TeV [40]

  È stato inoltre notato attraverso l'aggiustamento di iperparametri che la rete in cui ogni dimensione di input è rappresentata da un singolo *qubit*, *entangled* con quelli a fianco, e un singolo *layer* è sufficiente per allenare un GAN.

L'uso di un qGAN non è limitato alla generazione di dati, bensì può essere utilizzato come un codificatore intermedio delle funzioni di generazioni del processo. È stato utilizzato in questa maniera codificando informazioni alla base del processo nei *qubits* ed in seguito un algoritmo di *Quantum Amplitude Estimation (QAE)* [41] ha effettuato l'integrazione in una e due dimensioni del processo elementare su questi *qubits*. Nel futuro, ci si aspetta che i circuiti siano costituiti da diversi algoritmi messi insieme, ognuno con specifici compiti e che riescano a portare da dati grezzi a risultati raffinati e accurati.

Quantum circuit Born Machine (QCBM) [42]

Modello implicito per l'apprendimento generativo. Permette di generare dati misurando il sistema.
Diversamente dalle Macchine di Bohr, il circuito unitario non fa riferimento ad una specifica Hamiltoniana, un *QCBM* è un'unitaria parametrizzata $U(\langle 0|\,\theta\,|0\rangle)$ che prepara n *qubits* in un determinato stato $|\psi_\theta\rangle = U(\vec{\theta})\,|\psi_0\rangle)$. Questi modelli possono essere allenati utilizzando metodi non "avversari", per esempio il metodo di ottimizzazione del gradiente. L'epoca di addestramento avviene minimizzando una *loss function* che misura la somiglianza tra la distribuzione di partenza e quella prodotta dal circuito.

2.4 Algoritmi basati sul *Quantum Annealing*

I *Quantum Annealers*, macchine che implementano il *Quantum Annealing*, sono fortemente limitati dalle mansioni che possono compiere, in quanto sono disegnati per risolvere problemi di ottimizzazione binaria quadratica non vincolata (QUBO, *Quadratic Unconstrained Binary Optimization*). Il *Quantum Annealing* è una tecnica che permette di risolvere problemi di ottimizzazione combinatori codificando la soluzione nello stato fondamentale di una certa Hamiltoniana. La soluzione dello stato fondamentale è poi raggiunta inizializzando il sistema quantistico in uno stato fondamentale di un'altra Hamiltoniana ben conosciuta. Si lascia poi evolvere il sistema cambiando lentamente Hamiltoniana verso quella di interesse. Il teorema adiabatico quantistico [43] garantisce che esista un tempo minimo di evoluzione per cui lo stato finale è arbitrariamente vicino alla soluzione.

L'obbiettivo di questi nuovi algoritmi è tradurre un problema generico in un problema di ottimizzazione QUBO. Il tempo richiesto varia da problema a problema, in generale scala in modo peggiore del quadrato della dimensione dei dati in input. Inoltre, un altra difficoltà si riscontra nel cercare il minimo globale, fattore principale per sfruttare tale tecnologia. Infine, se l'algoritmo prevede un elevato grado di connessione tra le variabili binarie (*qubits* logici), risulta complesso codificare i dati negli stati fisici dei *qubits* e l'operazione può richiedere un tempo esponenzialmente lungo.

Capitolo 3

Applicazione dell'Intelligenza Artificiale Quantistica ai dati di LHCb

3.1 Introduzione

Come discusso nei capitoli precedenti, i problemi di classificazione sonoimportanti e giocano un ruolo fondamentale in Fisica delle Alte Energie, nelle applicazioni sperimentali dei metodi di apprendimento automatico e della loro versione quantistica.
In questo studio, l'oggetto della classificazione è l'identificazione del sapore dei quark pesanti che originano i *jets*. Sono stati analizzati quelli prodotti dai quark b e c e il problema è stato tradotto in una classificazione binaria. È stato implementato un algoritmo quantistico di apprendimento automatico per distinguere tra *jets* originati da quark b dai quark c, e valutare le prestazione su *hardware* reali IBM.

L'esperimento LHCb studia le proprietà dei quark b e c attraverso l'analisi dei prodotti delle collisioni protone-protone a LHC. Quest'ultimo è un acceleratore di particelle circolare lungo 27 km e posto 100 m sotto terra presso il CERN di Ginevra. I fasci di protoni circolano in tubi dove è stato realizzato il vuoto e sono suddivisi in pacchetti di circa 10^{11} protoni, lunghi 30 cm e di dimensione trasversale dell'ordine di 1 mm che viene compresso fino a 16 μm nei punti di collisione. Ogni fascio possiede 2808 pacchetti, equispaziati temporalmente di 25 ns, permettendo di raggiungere una frequenza di collisione di 40 MHz ed una luminosità di $\mathcal{L} = 2 \cdot 10^{34}\ cm^2 s^{-1}$. I due fasci si intersecano in 4 punti di collisione, dove sono localizzati gli esperimenti: CMS, ATLAS, ALICE e LHCb [4].

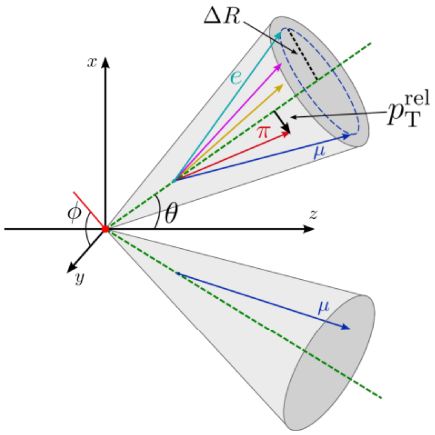


Figura 3.1: Visione schematica di un *jet*, sono riportate varia tipologie di particelle e le variabili ΔR, la distanza rispetto all'asse del jet nel piano η-ϕ, e p_T^{rel}, la quantit'a di moto trasversale rispetto all'asse del jet. [21]

LHCb è il rivelatore progettato per studiare la *Fisica del Sapore* per la quale la regione di massima produzione è quella in avanti. ATLAS e CMS grazie a strutture cilindriche attorno al punto di collisione studiano la regione d'interazione centrale. L'apparato sperimentale di LHCb è formato da uno spettrometro a singolo braccio in avanti, con una copertura angolare rispetto all'asse del fascio compresa tra circa 10 e 300 *mrad* corrispondente ad una copertura in pseudo-rapidità (vedi sez. 2.2.2) tra 1.8 e 4.9. I continui miglioramenti ed innovazioni del rivelatore hanno condotto l'esperimento ad avere ottimi risultati anche nello studio della Fisica dei *jets*. Il rivelatore puòessere suddiviso in: *detector* per il tracciamento e *detector* per l'identificazione delle particelle (PID, *Particle IDentification*). Si cita a titolo d'esempio per i primi, il *VErtex LOcator* (VELO), composto da sensori a *microstrip* di silicio che permettono di individuare i vertici primari e secondari di interazione attorno al punto di collisione dei fasci. Per i secondi si riportano i sistemi di rivelatori Čherenkov, *Ring Imaging ČHerenkov detector (RICH)*, atti a rivelare i coni di luce tipici del fenomeno, e i calorimetri elettromagnetico e adronico, "ECAL" e "HCAL", necessari ad assorbire l'energia depositata da fotoni, elettroni e adroni. I calorimetri sono composti rispettivamente da fogli di piombo con pannelli scintillatori e da strati di ferro alternati con materiale scintillatore [44, 45].

I *jets* sono il risultato della frammentazione ed adronizzazione dei quark e dei gluoni generati nelle collisioni protone-protone. I *jets* appaiono come un flusso collimato di particelle cariche nei sistemi di tracciamento e come insiemi concentrati di energia nei calorimetri. Lo studio di fenomeni come per esempio i decadimenti del bosone di Higgs in quark $b\bar{b}$ e $c\bar{c}$ richiede di distinguere ed identificare il tipo di *jet* [46, 47], cioè identificare il sapore del quark che lo ha originato. Esistono due tipologie di algoritmi per adempiere a tale compito:

- *Jet Flavour tagging*: permette di identificare il sapore del quark pesante che ha originato il flusso di particelle, distinguendo tra *jets* creati da b, da c e *jets* prodotti da quark leggeri, con bassa probabilità di identificazione errata[48].
- *Jet Charge tagging*: permette di identificare la carica elettrica del quark pesante che ha prodotto il *jet* , discriminando *jet* b da *jet* $\bar{b}$ oppure *jet* c da *jet* $\bar{c}$ [48].

La tipologia di algoritmo studiato in questo capitolo è la prima, operante però soltanto su eventi riconducibili ad adroni con quark b o c. I *jets* orginati da quark leggeri son di facile classificazione rispetto agli altri. Gli adroni con quark b o c hanno una vita media dell'ordine di 10^{-12} s, ovvero tale da viaggiare per un tratto significatamente diverso da zero prima di decadere, originando un vertice secondario distante dal vertice primario. Grazie a tale peculiarità si può dunque ricorrere ad una versione specifica di algoritmi di *flavour tagging*: il *Secondary Vertex tagging*. Questo modello permette di distinguere tra *jets* prodotti da quark b piuttosto che c, utilizzando caratteristiche relative al vertice secondario.
Le *features* riferite al vertice secondario (SV, *Secondary Vertex*) sono [49, 50, 48]:

- la massa M definita come la massa invariante delle tracce associate al vertice;
- la massa corretta, definita come la minima massa che l'adrone a vita lungo può avere coerentemente con la direzione di volo, $M_{cor} = \sqrt{M^2 + p^2 \sin^2(\theta)} + p \sin(\theta)$, dove $p \sin(\theta) = |p^{\mathrm{T}}_{\mathrm{mancante}}|$ è la componente mancante del momento rispetto alla direzione di volo della particella;
- la distanza nel piano trasverso delle due tracce più vicine al vertice primario;
- il momento trasverso del SV, ovvero la componente della quantità di moto trasversale al *jet*, p_{T};
- la frazione di p_{T} del *jet* associata al vertice secondario: $(\frac{p_{\mathrm{T}}(\mathrm{SV})}{p_{\mathrm{T}}(\mathrm{jet})})$;
- la distanza rispetto all'asse del jet nel piano η-ϕ (vedi sez. 2.2.2): ΔR;
- il numero di tracce nel SV;
- il numero di tracce del SV con $\Delta R < 0.5$;
- la carica netta delle tracce che formano il SV;

- il χ^2 della distanza di volo, definito come il χ^2 ottenuto dal fit sul vertice primario se le tracce del *Secondary Vertex tagging* vengono aggiunte al risultato del fit;
- la somma di tutti i χ^2_{IP} delle tracce del SV, dove il χ^2_{IP} è definito come la variazione del χ^2 del fit sul vertice primario quando la traccia viene rimossa (IP, *Impact Parameter*, minima distanza della traccia dal vertice primario);
- la vita media dell'adrone che decadendo origina il SV;
- la coordinata z del SV.

Al fine di massimizzare l'efficienza di identificazione dei *jets* vengono esclusi quelli meno rilevanti nelle misure di fisica o mal ricostruiti. Si richiede [20]:

$$20 < p_{\text{T}} < 100 \quad \text{GeV/c} \qquad (3.1) \qquad\qquad 2.2 < \eta < 4.2 \qquad (3.2)$$

Le richieste sul momento trasverso p_{T} e sulla pseudo-rapidità η sono necessarie per assicurarsi che i *jets* siano ricostruiti in una regione dello spazio delle fasi dove l'efficienza di ricostruzione ed identificazione siano uniformi rispetto a p_{T} e η e per rimuovere eventi non desiderati, come *jets* provenienti da quark leggeri. Inoltre, la condizione su η garantisce che i *jets* siano interamente dentro l'accettanza della strumentazione.

Una volta che i vertice secondario e le *features* sono disponibili, il problema diviene una classificazione binaria. Per tale tipologia di compiti, si utilizzano sistemi multivariati, come ad esempio un BDT, *Boosted Decision Trees* [20], che nel presente caso devono determinare se il *jet* è originato da un quark b o c. Questo tipo di problema si presta bene per l'utilizzo di modelli di intelligenza artificale quantistica.

Il prototipo del circuito utilizzato è un circuito variazionale quantistico con uno *stage* di codifica basato sugli angoli di porte logiche rotazionali. I circuiti sono stati disegnati e tradotti in linguaggio di programmazione tramite "*Qiskit*" [51], un kit di sviluppo *software open source* per lavorare con computer quantistici a livello di circuiti e algoritmi. Basandosi sui precedenti studi di classificazione [20, 49], in cui sono stati scritti algoritmi tramite "*Pennylane*" [19], un analogo di "Qiskit", è risultato che una particolare scelta di circuito variazionale performa in modo eccellente in questa tipologia di problemi, almeno a livello di simulazione. Questa rete è dunque stata tradotta da linguaggio "Pennylane" a linguaggio "Qiskit", al fine utilizzare quest'ultimo per svolgere i calcoli.

3.2 Qiskit

Libreria per implementare *software* lavorando direttamente sui circuiti e sugli algoritmi. Comprende un set completo di porte logiche e di circuiti pre costruiti per realizzare algoritmi di ricerca e sviluppi di applicazioni. Presenta inoltre la funzione *Transpiler* [51], la quale permette di riscrivere il circuito disegnato ottimizzandolo per lo specifico computer mediante una riduzione del numero di *gates* e delle connessioni tra *qubits* e sostituendo le porte logiche con quelle native del calcolatore. Utilizzando inoltre le funzioni di "Pytorch" [52], permette di unire le reti tensoriali nella creazione del circuito e di accedere ai simulatori o ai calcolatori reali in *runtime*. Il codice è stato scritto in linguaggio "Python" [53] importando per l'appunto le librerie "Qiskit" e le librerie "Pytorch".

3.3 Circuito

Lo schema di base del circuito è un *TwoLayerQNN* [51], una rete neurale quantistica a due strati che agisce su n *qubits*, con una prima fase per la codifica dei dati negli stati dei *qubits* e una seconda variazionale per l'epoca di addestramento, con *gates* rotazionali dipendenti da parametri ottimizzabili e *gates* generanti entanglement. Il risultato si ottiene misurando l'osservabile di parità $Z^{\otimes n}$ e collassando la funzione d'onda di un *qubit*.

3.3.1 Codifica dei dati: *Z feature map*

È uno schema circuitale [51] nativo delle librerie di Qiskit. Sfrutta porte logiche Hadamard H e *gates* di rotazione $U1(2.0 \cdot x[n])$ attorno all'asse della sfera di Bloch definito come z. Le *features* vengono trascritte negli angoli di rotazioni delle porte logiche. Sono stati utilizzati 4 *qubits* per 4 *features*. Quest'ultime sono state scelte in base alla loro importanza fornite da una *features importance map* di un modello BDT [20] calcolata tramite una simulazione [49]. Esse sono, in ordine: la massa corretta M_{cor}, la frazione del momento trasverso portato dal SV $\frac{p_{\text{T}}(\text{SV})}{p_{\text{T}}(\text{jet})}$, la distanza della direzione di volo del SV dall'asse del jet ΔR, la massa M.

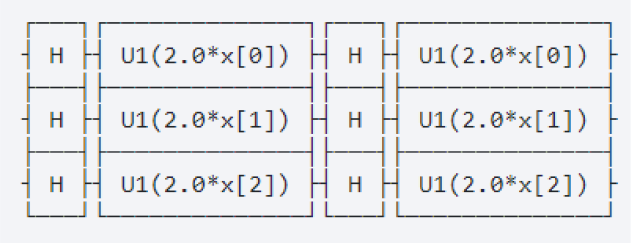


Figura 3.2: *Z feature map* (figura nel linguaggio "Qiskit"). [51]

Le porte logiche di rotazione in figura sono matrici unitarie del tipo:

$$U1(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix} \tag{3.3}$$

3.3.2 Circuito variazionale (Ansatz): *Strongly Entangling Layers*

Il circuito [19] appartiene alla libreria "Pennylane" e per le sue prestazioni è stato scelto e tradotto nel linguaggio "Qiskit". Consiste di un numero modificabile di *layers* ognuno composto da *gates* di generica rotazione e porte logiche generanti entanglement. Ogni strato permette di definire dei pesi iniziali che vengono aggiornati nella fase di allenamento mediante l'ottimizzazione della *loss function*. In questo circuito sono stati utilizzati un registro di 4 *qubits* e due strati di *gates*, per un totale di 24 parametri da affinare. L'entanglement viene realizzato mediante porte CNOT, definendo una struttura entangled circolare.

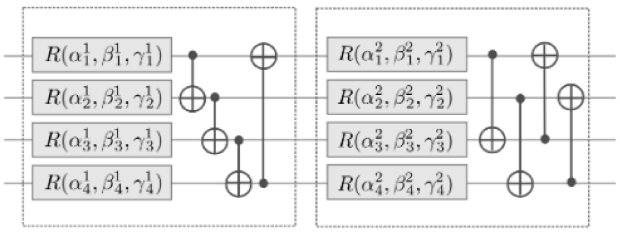


Figura 3.3: *Strongly Entangling Layers* (figura nel linguaggio "Pennylane", le matrici di rotazione sono rappresentate con $R(\alpha, \beta, \gamma)$). [19]

Le matrici di rotazione generica sono:

$$R(\alpha, \beta, \gamma) = \begin{pmatrix} cos(\frac{\alpha}{2}) & -e^{i\gamma} sin(\frac{\alpha}{2}) \\ e^{i\beta} sin(\frac{\alpha}{2}) & e^{i(\beta+\gamma)} cos(\frac{\alpha}{2}) \end{pmatrix} \tag{3.4}$$

3.3.3 *Loss Function*

L'epoca di addestramento mira a minimizzare la *loss function*, nel presente studio è stato scelto l'errore quadratico medio:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(P_b^i(\theta) - T^i)^2 \tag{3.5}$$

in cui N è il numero di *jets* d'allenamento, P_b^i è la probabilità predetta e T^i è la probabilità detta di *target*, ovvero la soluzione del problema di classificazione già presente nel *dataset* di allenamento (problema di apprendimento automatico supervisionato) e può assumere i valori 1 o -1. Il processo di minimizzazione mira ad ottimizare i pesi, parametri, dello *stage* variazionale del circuito e l'ottimizzatore utilizzato è l'ADAM [26] con un tasso di apprendimento pari a 0.01.
ADAM è un metodo stocastico di *gradient descent* basato sulla stima del primo e del secondo momento. Il tasso di apprendimento di un algoritmo di ottimizzazione è solitamente definito come il fattore di scala applicato al gradiente della *loss function* durante l'aggiornamento dei parametri e può essere regolato come parametro del processo di apprendimento.
Le probabilità predette tendono a spostarsi verso 1 e -1, identificando rispettivamente i flussi di particelle originati da quark b e da quark c. A causa del livello attuale della tecnologia utilizzata, le predizioni non giungono ai valori estremali, ma occupano posizione intermedie più centrate sullo 0 pur evidenziando una separazione.

3.4 Risultati

L'algoritmo di classificazione è stato implementato per essere utilizzato su un reale calcolatore quantistico. Data l'attuale efficienza di tali macchine, si è preferito pre allenare il circuito tramite una simulazione dello stesso circuito scritto in "Pennylane". Questo linguaggio permette di eseguire il *training* con molti dati ad una maggiore velocità rispetto alle simulazioni sui dispositivi forniti da IBM con "Qiskit".

Prima di effettuare prove su *hardware* quantistici, sono stati realizzati degli studi sugli algoritmi quantistici applicati a macchine classiche. Nel linguaggio "Pennylane" è stato eseguito un *training* con molti eventi, sull'ordine delle centinaia di migliaia, sulle GPU di un computer classico [49], tramite i simulatori quantistici di "Pennylane" per poi compararlo a quello effettuato su un simulatore di un computer quantistico IBM. La scelta di utilizzare le GPU di un calcolatore classico è stata presa in considerazione per velocizzare la procedura di addestramento potendo sfruttare la presenza di una RAM, da cui leggere i dati, che su un computer quantistico non è presente. I risultati della valutazione del modello allenato su GPU e su simulatore IBM sono stati confrontati: le prestazioni risultano compatibili. Ciò ha permesso di utilizzare la velocità delle GPU per effettuare delle operazioni intermedie nella procedura di allenamento e valutazione dei modelli di apprendimento automatico applicati ad un reale calcolatore.

La rete quantistica di questa tesi è stata dunque allenata con "Pennylane" e le GPU di un computer classico utilizzando tutte e 13 le *features* ed usando un campione di 100 000 dati, opportunatamente selezionati tramite applicazione di tagli (vedi sez. 3.1). I parametri ottimizzati sono stati quindi riformattati per essere utilizzati sul circuito di "Qiskit".
I dati vengono campionati in modo casuale dal *dataset* di riferimento, ovvero sia per la fase di allenamento che per la fase di valutazione. Per tale motivo, i risultati dipendono dalle fluttuazioni statistiche con cui gli eventi sono selezionati.

Accedendo al simulatore o *computer* quantistico, si eseguono le fasi di addestramento e di valutazione delle prestazioni. Il circuito ora presenta 4 *qubits* per le 4 *features* più importanti e sono state utilizzate collezioni di dati di dimensioni variabili per studiare la diversa risposta del circuito. La fase di *training* viene rieseguita su un simulatore IBM per raggiungere un migliore set di parametri, ovvero un insieme di parametri che converga maggiormente ai valori ottimali. La rete allenata viene infine valutata su un nuovo *dataset* e viene computata l'accuratezza della classificazione.

Le predizioni dell'algoritmo mirano a fornire un'identificazione assegnando una *label* pari a 1 per i *jets* originati da quark b e pari a -1 per i *jets* da quark c. Nei fatti le predizioni calcolate tendono a tali valori, senza mai effettivamente uguagliarli.

A seconda del numero di dati selezionati per la fase di allenamento, le *performance* subiscono dei cambiamenti: al crescere della dimensione dell'insieme di dati, cresce l'accuratezza dell'algoritmo. Nello studio [20, 49] è stato dimostrato come per un algoritmo quantistico siano necessari meno eventi rispetto all'analogo classico per raggiungere la convergenza dei parametri dei modelli e poter fornire predizioni esaustive. Tale particolarità è stata verificata confrontando le *performance* di un algoritmo classico con quelle di un modello allenato tramite GPU classiche su un simulatore quantistico [49]: le prestazioni sono comparabili, evidenziando nuovamente la solidità dei risultati ottenuti tramite GPU e la concretezza dei risultati ottenibili con un computer quantistico. Nello specifico, in questa tesi, confrontando le prestazioni tra calcolatore reale IBM e simulatore IBM, si notano nuovamente comportamenti paragonabili.

Il modello è stato utilizzato dapprima su di un simulatore IBM senza rumore, il "*qasm simulator*", con *dataset* per la fase di allenamento e di valutazione di varie dimensioni. I parametri ottenuti dalla simulazione su GPU classiche tramite "Pennylane" sono stati utilizzati per pre-alleanare la rete, inizializzando dunque i pesi delle porte logiche parametrizzate a valori più vicini a quelli ottimali.
Si riportano dei risultati:

- 100 eventi per l'epoca di addestramento e 1000 eventi per l'epoca di valutazione

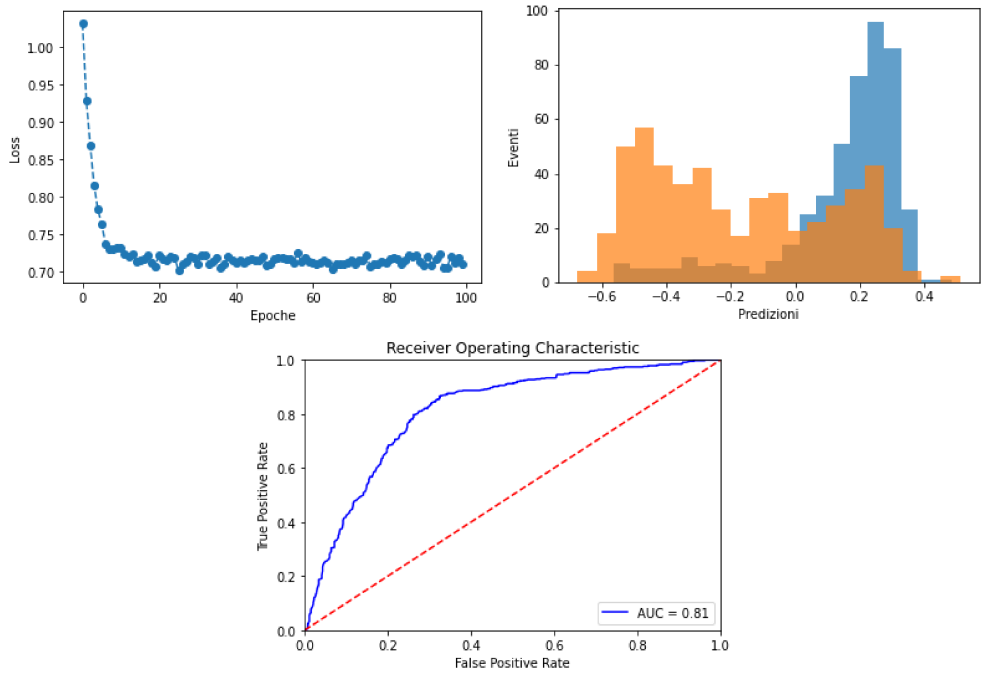


Figura 3.4: *Loss Function*, predizioni e ROC per *dataset* con 100 dati per il *training* e 1000 dati per il *testing*

L'algoritmo risulta performante. La *loss function* scende, a dimostrazione del fatto che la minimizzazione è avvenuta correttamente e i parametri sono stati ottimizzati. La distribuzione delle predizioni evidenzia una separazione sufficientemente marcata da poter distinguere la tipologia dei *jets* analizzati. Le predizioni tendono a discostarsi dal centro, dallo 0, spostandosi verso i valori 1 e -1. Dal computo della ROC, si nota come il circuito risponda bene, mostrando che l'algoritmo funziona e sta apprendendo le correlazioni presenti nei dati. L'area sotto la curva, AUC, tende al valore ottimale di 1, tuttavia non è abbastanza elevata da essere comparabile alle reti classiche.

- 100 eventi per l'epoca di addestramento e 5000 eventi per l'epoca di valutazione

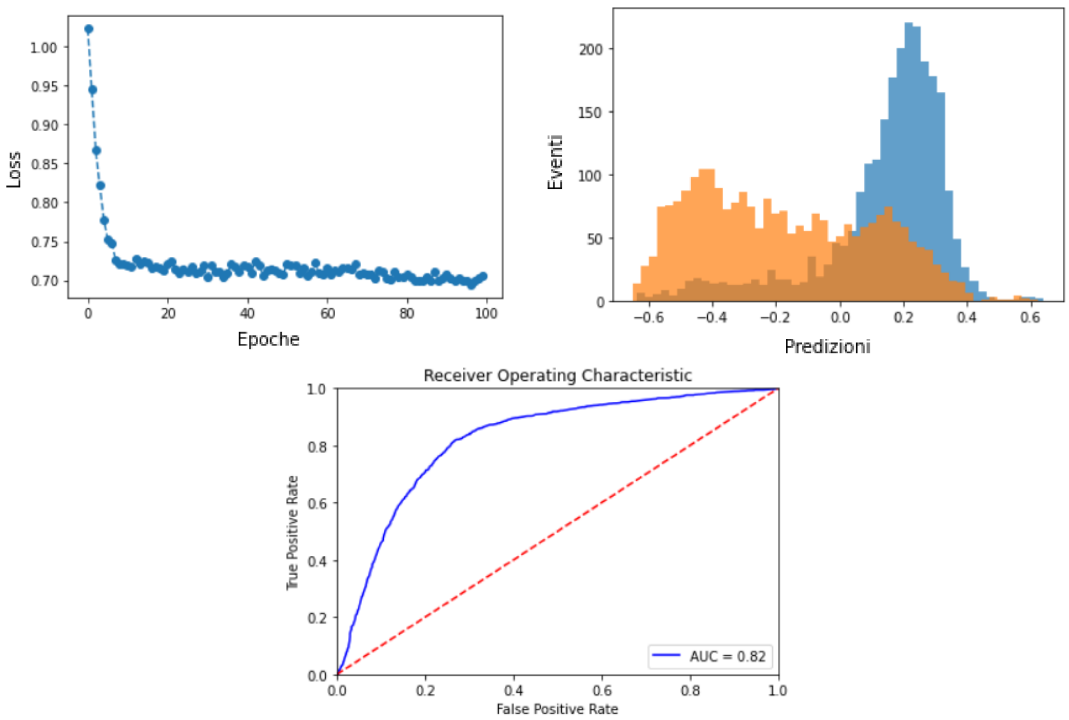


Figura 3.5: *Loss Function*, predizioni e ROC per *dataset* con 100 dati per il *training* e 1000 dati per il *testing*

L'algoritmo risponde ancora bene. Grazie alla maggiore dimensione del *dataset* utilizzato per la fase di *testing*, si nota una più netta seperazione nella distribuzione delle predizioni. La ROC appare anch'essa migliore, più liscia e con una AUC più elevata. La *loss function* dimostra ancora che il circuito sta ottimizzando i propri parametri.

In seguito, è stata valutata la *performance* del modello tramite un *testing* su reali calcolatori, "Toronto" privato del CERN e "Nairobi" pubblico.
L'algoritmo è stato sempre pre allenato tramite "Pennylane", in seguito una fase di *training* è stata eseguita sul simulatore "qasm" di IBM e i parametri del modello sono stati esportati per utilizzarli in un reale computer quantistico valutandone le prestazioni. Tale procedura è stata necessaria per garantire che il circuito riesca a minimizzare efficientemente la *loss function* e non ricada in un qualche minimo errato o sella. I parametri allenati si riferiscono al modello che utilizza 5000 dati per la fase di *testing* (fig.3.5).
Si ottengono i seguenti risultati:

- 500 eventi per la fase di valutazione sul computer Toronto

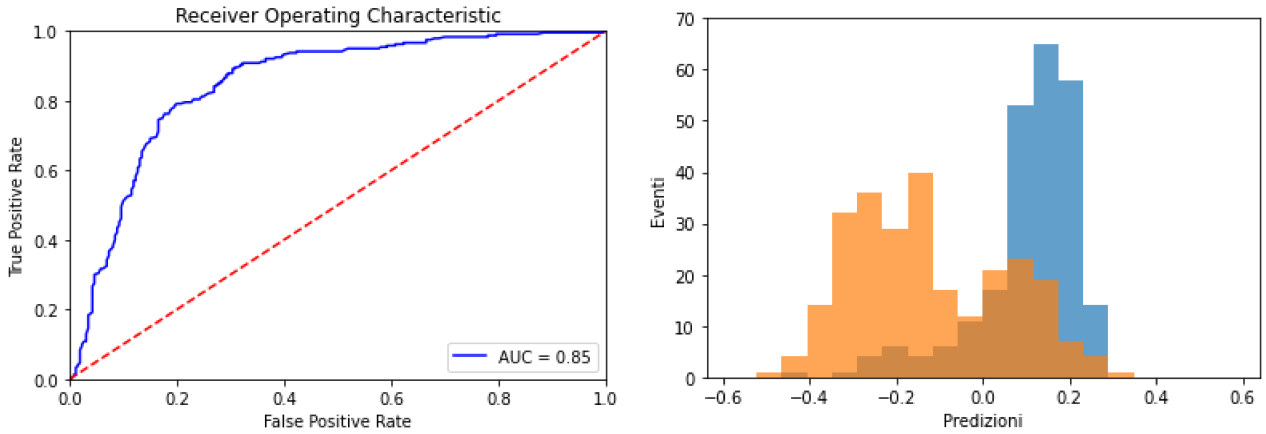


Figura 3.6: Predizioni e ROC per *dataset* con 500 dati per il *testing*

- 1000 eventi per la fase di valutazione sul computer Toronto

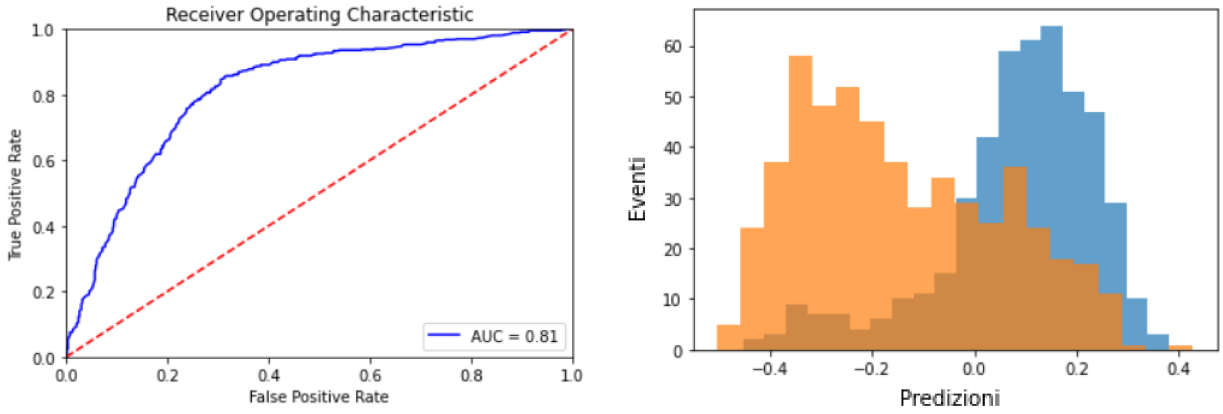


Figura 3.7: Predizioni e ROC per *dataset* con 1000 dati per il *testing*

Il calcolatore riporta ottime prestazioni. Le distribuzioni risultano separate con evidente distinzione tra *jets* originati da quark b piuttosto che c. La ROC assume la sua forma caratteristica e la AUC è buona, tendente al valore ottimale. Si nota che l'area sotto la curva è minore per il modello a 1000 eventi rispetto al modello a 500 eventi, la causa può essere ricondotta a fluttuazioni statistiche nella scelta dei dati appartenenti al *dataset* di valutazione. Una differente selezione di eventi condurrebbe a diversi valori, nel limite di tutti gli eventi appartenenti alla collezione ci si aspetta che l'area sia maggiore.

- 500 eventi per la fase di valutazione sul computer Nairobi

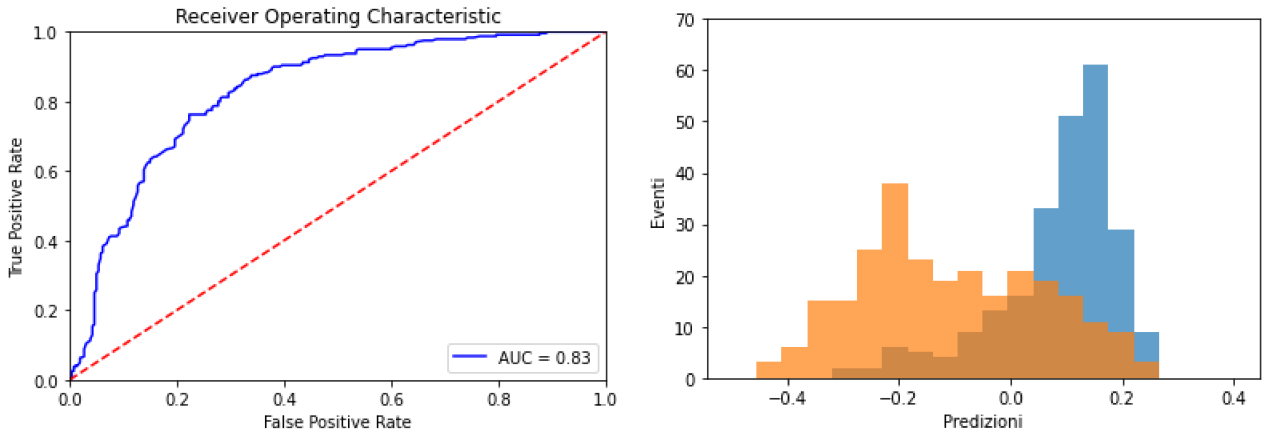


Figura 3.8: Predizioni e ROC per *dataset* con 500 dati per il *testing*

Il computer Nairobi mostra comportamenti simili: distribuzione delle predizioni separata e ROC crescente. L'AUC è anch'essa buona, con un valore confrontabile con l'altra macchina.

La differenza principale fra i due calcolatori risiede nell'insieme di porte logiche native. Ogni computer possiede una ben specifica topologia di elementi circuitali, l'algoritmo viene dunque adattato all'*hardware* riscrivendo la rete con l'utilizzo di tali porte. Consegue che la profondità del circuito, e quindi la sua complessità, possa aumentare richiedendo una maggiore spesa di risorse, principalmente il tempo di esecuzione. Grazie alla funzione "*Transpiler*" di "Qiskit", è possibile vedere come il circuito venga tradotto dallo specifico calcolatore e di come cambi la natura della rete.

3.4.1 *Transpiler*

Il processo di modifica del circuito è tutt'altro che semplice [51]. La rete deve essere ottimizzata per utilizzarla su computer che presentano rumore e non è dunque accettabile che il circuito risulti troppo sensibile ad errori. Il flusso di riscrittura di un circuito non è lineare, può contenere *sub-loop* iterativi, rami condizionali e altri procedimenti complessi. In linea generale, lo schema è il seguente:

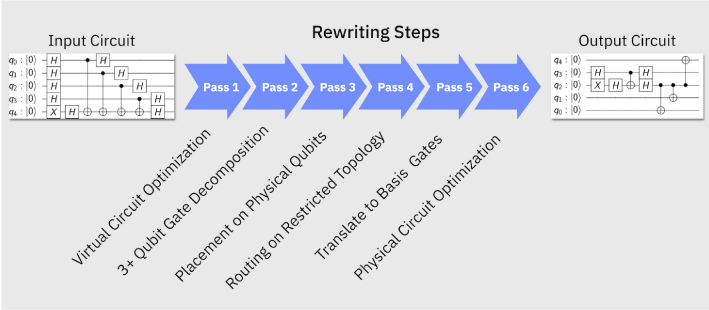


Figura 3.9: Schema del procedimento di riscrittura con la funzione *Transpiler* di "Qiskit". [51]

Un processo fondamentale nella riscrittura della reta è la minimizzazione del numero di SWAP, porte logiche che scambiano i *qubits*, necessarie per implementare *gates* a due *qubits*, come i CNOT. Ogni SWAP viene infatti decomposto in tre CNOT nei dispositivi IBM e questo rappresenta un'operazione dispensiosa e originante molto rumore. Tuttavia trovare questo minimo è un problema di ottimizzazione molto complesso, per ovviare a ciò si utilizza un algoritmo euristico chiamato *Qiskit.transpiler.passes.StochasticSwap*. È un metodo stocastico per generare il circuito ed in quanto tale non è garantito di ottenere lo stesso risultato su varie ripetizioni. Il risultato di molteplici applicazioni è dunque l'ottimizzazione della rete con un numero differente di porte logiche utilizzate e differenti livelli di ottizzazione, da 0 a 3, dove più alto è il livello, migliore è il grado di ottimizzazione (ma non è garantito che la profondità sia anche minore).

Il circuito utilizzato viene riscritto dai calcolatori "Toronto" e "Nairobi" nella seguente maniera:

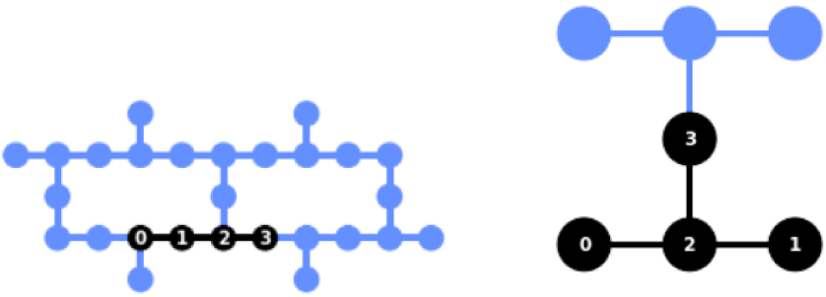


Figura 3.10: Circuiti riscritti in visualizzazione stilizzata con la funzione *Transpiler* : in nero i *qubits* con le connessioni reciproche. A sx il computer "Toronto", a dx il computer "Nairobi". [51]

La profondità originale del circuito in numero di *gates* è pari a 11. Una volta che la *routine* di ottimizzazione del circuito viene applicata più volte, la profondità del circuito decomposto diviene di 38 per "Toronto" e 39 per "Nairobi". Per entrambi il livello di ottimizzazione scelto è il livello 2.

3.4.2 Tempo di esecuzione

Il tempo impiegato dal simulatore piuttosto che dal computer fisico è differente a seconda dello strumento utilizzato e dalla dimensione del *dataset* scelto. Si evince che la fase di allenamento richiede una spesa in termini di tempo molto maggiore, ordini di grandezza, rispetto a quella di valutazione

e che il simulatore è nettamente più veloce. Tale fatto è in linea con ciò che ci si aspetta da questa tipologia di esperimenti: la fase di addestramento è infatti complessa dal punto di vista computazionale, in quanto la macchina deve ricercare correlazione nei dati minimizzando la specifica *loss function* senza ricadere in minimi non utili e rappresentare dunque stati non fisici; il simulatore d'altronde è più veloce poichè possiede un grande set di porte logiche native, al contrario dei calcolatori reali che ne posseggono solo alcune. Si può notare inoltre come la macchina "Toronto" richieda più tempo di esecuzione di "Nairobi", questo è sempre dovuto all'insieme di *gates* nativi del calcolatore, che a seconda di come riscrive il circuito può aumentarne la complessità. Si riportano i tempi di esecuzione per la varie epoche e per lo strumento utilizzato:

Tabella 3.1: Tempi di esecuzione

Strumento	*Training*	*Testing*
Simulatore 100-1000	1887.2 s	21.5 s
Simulatore 100-5000	3751.5	42.3 s
Toronto 500	-	8704.1 s
Toronto 1000	-	14886 s
Nairobi 500	-	3526.2 s

A tale tempo è da sommare anche l'attesa per utilizzare la macchina, in quanto tramite le funzioni "*runtime*" si può accedervi da remoto, ma bisogna attendere il proprio turno e il lavoro viene messo in coda agli altri.

Per abbreviare i tempi di esecuzioni effettevi si potrebbero implementare delle RAM quantistiche, "qRAM" [ref due articoli], che permetterebbero il recupero dei dati molto più velecemente, senza ricorrere a specifiche GPU e RAM classiche.

3.4.3 Fluttuazioni

Sono stati verificati i commportamenti del computer quantistico in diverse condizioni rispetto a quanto riportato precedentemente. Dapprima è stata rieffettuata la fase di valutazione senza cambiare il *dataset* da *testing* e senza cambiare i parametri allenati del modello, si intende vedere se i risultati ottenibili coincidono oppure appaiono delle fluttuazioni. È stato analizzato il modello con 500 eventi per la fase di valutazione su "Toronto" (fig. 3.6) e "Nairobi" (fig. 3.8):

- Toronto

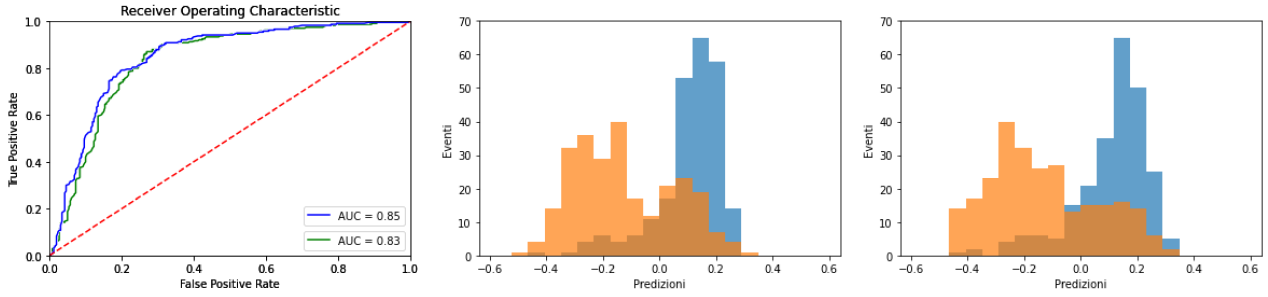


Figura 3.11: Confronto tra modelli con gli stessi parametri allenati e stesso *dataset* di valutazione. Nella figura delle ROC, in blu il modello originale (fig. 3.6) e in verde il nuovo modello. A sx modello originale (fig. 3.6), a dx modello ri-valutato

- Nairobi

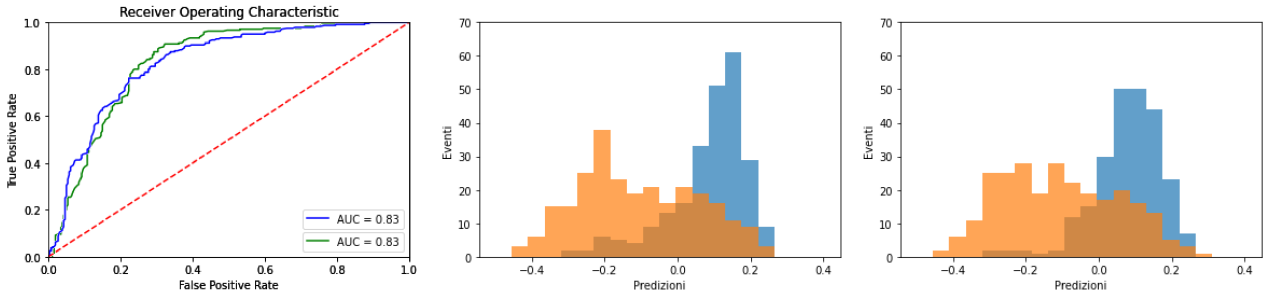


Figura 3.12: Confronto tra modelli con gli stessi parametri allenati e stesso *dataset* di valutazione. Nella figura delle ROC, in blu il modello originale (fig. 3.8) e in verde il nuovo modello. A sx modello (fig. 3.8), a dx modello ri-valutato

Nonostante l'utilizzo dello stesso *dataset* e i parametri dello stesso modello, la risposta del circuito applicato al reale *hardware* tende a cambiare fluttuando. Si nota inoltre che per il computer "Toronto" l'AUC è diminuita confermando l'ipotesi che la risposta del calcolatore sia soggetta a fluttuazioni nel computo della valutazione.

Successivamente, è stata studiata la risposta del circuito cambiando gli eventi da valutare e mantenendo i parametri allenati del modello. Si intende verificare se l'algoritmo ha potere predittivo anche su altri campioni di dati.

- Toronto

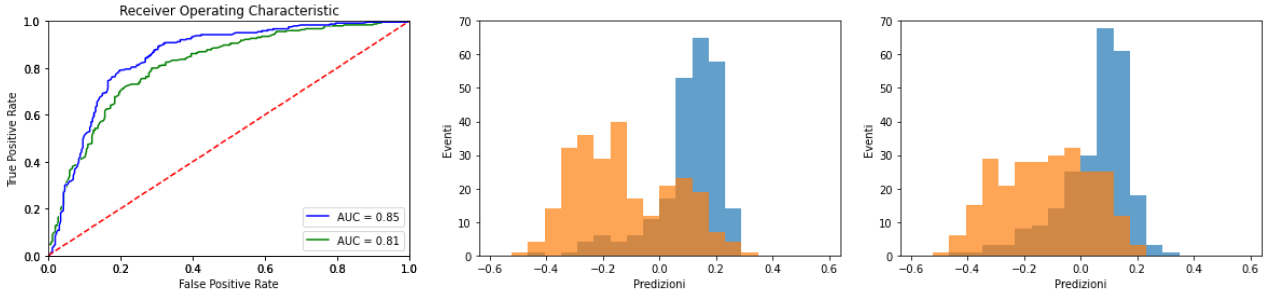


Figura 3.13: Confronto tra modelli con gli stessi parametri allenati, ma differente *dataset* di valutazione. Nella figura delle ROC, in blu il modello originale (fig. 3.6) e in verde il nuovo modello. A sx modello originale (fig. 3.6), a dx modello con nuovi dati

- Nairobi

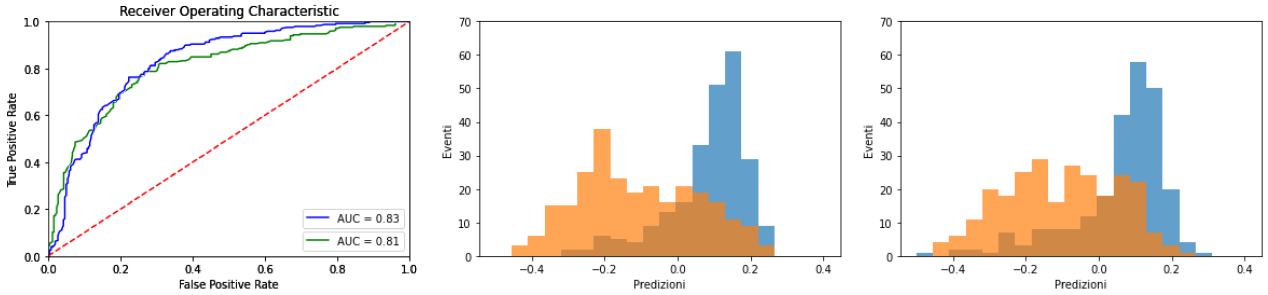


Figura 3.14: Confronto tra modelli con gli stessi parametri allenati, ma differente *dataset* di valutazione. Nella figura delle ROC, in blu il modello originale (fig. 3.8) e in verde il nuovo modello. A sx modello originale (fig. 3.8), a dx modello con nuovi dati

Si evince che l'algoritmo risponde ancora efficacemente, riuscendo ad operare la classificazione dei *jets*. L'AUC è per entrambi poco diminuita, ma questo è in linea con la selezione stocastica degli eventi dalla collezione complessiva.

Capitolo 4

Conclusioni

4.1 Risultati

In questa tesi, è stato implementato un algoritmo di QML per l'identificazione del sapore dei quark pesanti che originano i *jets* all'esperimento LHCb. Il problema era stato tradotto, in un prededente lavoro, in una classificazione binaria e risolto tramite un circuito variazionale quantistico (VQC, sez. 2.2.1). Le prestazioni del modello sono state valutate sul simulatore IBM, "*qasm-simulator*", e su due computer quantistici IBM, "Toronto" e "Nairobi", con differenti *dataset* per la fase di *testing*. Per Toronto sono stati processati 500 e 1000 eventi per il *testing*. L'algoritmo ha risposto con buone prestazioni, fornendo per 500 eventi una AUC = 0.85 e per 1000 eventi una AUC = 0.81.
Nairobi ha processato 500 eventi sempre per il *testing* dando una AUC = 0.83.
Le esecuzioni su *hardware* hanno confermato i risultati ottenuti con la simulazione. Questi risultati, su computer quantistico reale, sono stati ottenuti per la prima volta all'interno di LHCb e si è confidenti di poter espandere le attività usando circuiti più complessi. Per completare, però l'esercizio è necessario riuscire a portare a termine la fase di *training* su *hardware*, attualmennte non possibile a causa dei limiti fisici delle macchine reali. Da notare che gli sviluppi in questo senso sono molto veloci, un anno fa questo esercizio era impossibile.

4.2 Considerazioni finali

La ricerca e lo sviluppo tecnologico nel settore dei computer quantistici stanno progrendo rapidamente. Attualmente, il livello tecnologico non ha raggiunto le prestazioni che l'analogo classico è in grado di offrire. Ci si attende che nei prossimi anni tale tecnologia possa migliorare notevolmente; IBM ha già presentato i proprio futuri computer introducendo numeri di *qubits* sempre in aumento e archittetture più robuste agli errori. È previsto per il 2022 il computer "*Osprey*" a 433 *qubits*, per il 2023 "*Condor*" a 1121 *qubits* e dal 2026 in poi altri calcolatori con un numero scalabile di *qubits* compreso tra 1000 e 1 milione e capaci di gestire comunicazione classiche e quantistiche.

I computer quantistici odierni presentano dei problemi:

- Sensibilità al rumore: l'interazione tra il sistema di *qubits* e l'ambiente può condurre al fenomeno della decoerenza, distruggendo le correlazioni quantistiche e perdendo il vantaggio dato dall'utilizzare le proprietà quantistiche, come la sovrapposizione e l'entanglement.

- Codifica dei dati negli stati dei *qubits*: tale operazione è limitata dal numero di eventi codificabili e dall'efficienza della procedura

- Affidabilità dei *gates*: le porte logiche presentano probabilità di errori troppo elevate per condurre ad una archietettura *Fault Tolerant* e per poter implementare codici di correzione.

- Stabilità dei computer: l'implementazione di algoritmi richiede di lavorare con piccoli *dataset* e semplici modelli per riuscire a raggiungere una convergenza nei parametri da ottimizzare, a causa dell'instabilità delle macchine nelle procedure di ottimizzazione.
- Tempo di utilizzo: attualmente i calcolatori sono accessibili da remoto e la grande richiesta per utilizzarli origina code con conseguente attesa prima di potervi accedere

Tuttavia si nota che, nonostante le difficoltà tecnologiche, anche ridotti algoritmi performino in modo comparabile a reti neurali classiche riuscendo a fornire predizioni esaustive.

Bibliografia

[1] Cern. *Cern-Large Hadron Collider*. URL: https://home.cern/science/accelerators/large-hadron-collider.

[2] Cern. *Cern*. URL: https://home.cern/.

[3] IBM. *IBM Quantum*. URL: https://www.ibm.com/quantum.

[4] A Augusto Alves Jr et al. «The LHCb detector at the LHC». In: *Journal of instrumentation* 3.S08005 (2008).

[5] Michael A Nielsen e Isaac Chuang. *Quantum computation and quantum information*. American Association of Physics Teachers, 2002.

[6] Giuliano Benenti, Giulio Casati e Giuliano Strini. *Principles of Quantum Computation and Information-Volume II: Basic Tools and Special Topics*. World Scientific Publishing Company, 2007.

[7] GoogleAI. *GoogleAI*. URL: https://quantumai.google/.

[8] D-Wave Systems. *D-Wave Systems*. URL: https://www.dwavesys.com/.

[9] Gilles Brassard et al. «Quantum amplitude amplification and estimation». In: *Contemporary Mathematics* 305 (2002), pp. 53–74.

[10] Rigetti. *Rigetti*. URL: https://www.rigetti.com/.

[11] Amazon. *Amazon Braket-Quantum Computing Research*. URL: https://aws.amazon.com/it/braket/quantum-computing-research/?pg=ln&sec=uc.

[12] Leon N Cooper. «Bound electron pairs in a degenerate Fermi gas». In: *Physical Review* 104.4 (1956), p. 1189.

[13] Brian D Josephson. «The discovery of tunnelling supercurrents». In: *Reviews of Modern Physics* 46.2 (1974), p. 251.

[14] Claude Cohen-Tannoudji, Bernard Diu e Frank Laloe. *Quantum Mechanics, Volume 1*. Vol. 1. 1986, p. 898.

[15] David P DiVincenzo. «The physical implementation of quantum computation». In: *Fortschritte der Physik: Progress of Physics* 48.9-11 (2000), pp. 771–783.

[16] William K Wootters e Wojciech H Zurek. «A single quantum cannot be cloned». In: *Nature* 299.5886 (1982), pp. 802–803.

[17] Asher Peres. «Separability criterion for density matrices». In: *Physical Review Letters* 77.8 (1996), p. 1413.

[18] M Horodecki, P Horodecki e R Horodecki. *Separability of mixed quantum states: necessary and sufficient conditions Phys*. 1996.

[19] Pennylane. *Pennylane*. URL: https://pennylane.ai/.

[20] Alessio Gianelle et al. «Quantum Machine Learning for b-jet charge identification». In: *Journal of High Energy Physics* 2022.2202.13943 (2022), pp. 1–24.

[21] Andrea Delgado et al. «Quantum Computing for Data Analysis in High-Energy Physics». In: *arXiv preprint arXiv:2203.08805* (2022).

[22] Ruslan Leontyevich Stratonovich. «Optimum nonlinear systems which bring about a separation of a signal with constant parameters from noise». In: *Radiofizika* 2.6 (1959), pp. 892–901.

[23] Duarte Magano et al. «Quantum speedup for track reconstruction in particle accelerators». In: *Physical Review D* 105.7 (2022), p. 076012.

[24] Gilles Brassard et al. «Quantum amplitude amplification and estimation». In: *Contemporary Mathematics* 305 (2002), pp. 53–74.

[25] Lov K Grover. «Quantum mechanics helps in searching for a needle in a haystack». In: *Physical review letters* 79.2 (1997), p. 325.
[26] Diederik P Kingma e Jimmy Ba. «Adam: A method for stochastic optimization». In: *arXiv preprint arXiv:1412.6980* (2014).
[27] Koji Terashi et al. «Event classification with quantum machine learning in high-energy physics». In: *Computing and Software for Big Science* 5.1 (2021), pp. 1–11.
[28] Vasilis Belis et al. «Higgs analysis with quantum classifiers». In: *EPJ Web of Conferences*. Vol. 251. EDP Sciences. 2021, p. 12.
[29] Adrián Pérez-Salinas et al. «Data re-uploading for a universal quantum classifier». In: *Quantum* 4 (2020), p. 226.
[30] Andrew Blance e Michael Spannowsky. «Quantum machine learning for particle physics using a variational quantum classifier». In: *Journal of High Energy Physics* 2021.2 (2021), pp. 1–20.
[31] Sau Lan Wu et al. «Application of quantum machine learning using the quantum variational classifier method to high energy physics analysis at the LHC on IBM quantum computer simulator and hardware with 10 qubits». In: *Journal of Physics G: Nuclear and Particle Physics* 48.125003 (2021).
[32] Sau Lan Wu et al. «Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the LHC». In: *Physical Review Research* 3.033221 (2021).
[33] Jamie Heredge et al. «Quantum support vector machines for continuum suppression in B meson decays». In: *Computing and Software for Big Science* 5.1 (2021), pp. 1–9.
[34] Evan Peters et al. «Machine learning of high dimensional data on a noisy quantum processor». In: *npj Quantum Information* 7.1 (2021), pp. 1–5.
[35] Samuel Yen-Chi Chen et al. «Quantum convolutional neural networks for high energy physics data analysis». In: *Physical Review Research* 4.013231 (2022).
[36] Samuel Yen-Chi Chen et al. «Hybrid quantum-classical graph convolutional network». In: *arXiv preprint arXiv:2101.06189* (2021).
[37] Su Yeon Chang et al. «Dual-parameterized quantum circuit GAN model in high energy physics». In: *EPJ Web of Conferences*. Vol. 251. 03050. EDP Sciences. 2021.
[38] Su Yeon Chang et al. «Running the Dual-PQC GAN on noisy simulators and real quantum hardware». In: *arXiv preprint arXiv:2205.15003* (2022).
[39] Su Yeon Chang et al. «Quantum generative adversarial networks in a continuous-variable architecture to simulate high energy physics detectors». In: *arXiv preprint arXiv:2101.11132* (2021).
[40] Carlos Bravo-Prieto et al. «Style-based quantum generative adversarial networks for Monte Carlo events». In: *Quantum* 6 (2022), p. 777.
[41] Tadashi Kadowaki e Hidetoshi Nishimori. «Quantum annealing in the transverse Ising model». In: *Physical Review E* 58.5355 (1998).
[42] Brian Coyle et al. «The Born supremacy: quantum advantage and training of an Ising Born machine». In: *npj Quantum Information* 6.1 (2020), pp. 1–11.
[43] Max Born e Vladimir Fock. «Beweis des adiabatensatzes». In: *Zeitschrift für Physik* 51.3 (1928), pp. 165–180.
[44] Marco Ceoletta. «Jet reconstruction with the LHCb calorimeters in future upgrades Ricostruzione dei jet con i futuri calorimetri di LHCb».
[45] Christophe Salzmann. «LHCb Spectrometer Alignment and Verification of its Performance using the Decay $B_d^0 \to K^{0*} J\psi$». Tesi di dott. Zurich U., 2011.
[46] CMS Collaboration et al. «Observation of Higgs boson decay to bottom quarks». In: *Physical Review Letters* (2018).
[47] CMS Collaboration et al. «Search for Higgs boson decay to a charm quark-antiquark pair in proton-proton collisions at $\sqrt{s} = 13$ TeV». In: *arXiv preprint arXiv:2205.05550* (2022).
[48] LHCb Collaboration et al. «Identification of beauty and charm quark jets at LHCb». In: *Journal of Instrumentation* 10.06 (2015), P06013.

[49] Carlos Eduardo Cocha Toapaxi. ≪Study of b-and c-jets identification with quantum machine learning algorithms and application to the Higgs reconstruction≫.
[50] Davide Nicotra. ≪Study of b-quark production asymmetry with quantum machine learning techniques at the LHCb experiment≫.
[51] Qiskit. *Qiskit*. URL: `https://qiskit.org/`.
[52] Pytorch. *Pytorch*. URL: `https://pytorch.org/`.
[53] Python. *Python*. URL: `https://www.python.org/`.