SAPIENZA
UNIVERSITÀ DI ROMA

Modello di Diffusione Quantistico per la Simulazione di Immagini Galattiche

Facoltà di Scienze Matematiche, Fisiche e Naturali
Laurea Triennale in Fisica

Angiolino Pio Oriente
Matricola 2024001

Relatore
Prof. Stefano Giagu

Anno Accademico 2023/2024

Tesi discussa il 19/12/2024
di fronte a una commissione esaminatrice composta da:

Prof. Carlo Mariani (presidente)
Prof. Ernesto Placidi
Prof. Matteo Paoluzzi
Prof. Alessandro Coppolecchia
Prof.ssa Sibilla Di Pace
Prof. Francesco Renga
Prof. Fabio Riccioni

Modello di Diffusione Quantistico per la Simulazione di Immagini Galattiche
Tesi di Laurea Triennale in Fisica. Sapienza Università di Roma

Questa tesi è stata composta con LaTeX e la classe Sapthesis.

Email dell'autore: apo.angelopio@gmail.com

A Mamma e Papà,
per avermi insegnato il valore della curiosità e della perseveranza.
Ai miei Nonni, per il loro affetto costante e supporto incondizionato.
A Tata, per avermi aiutato a trovare la calma nei momenti di caos.

Indice

Introduzione

La presente dissertazione si propone di studiare il machine learning e la computazione quantistica, che costituiscono due dei campi più all'avanguardia nella teoria dell'informazione.
Il machine learning è una disciplina che si occupa della progettazione e implementazione di algoritmi in grado di riconoscere pattern complessi celati all'interno di grandi quantità di dati, con l'obiettivo di fare previsioni accurate.
Sebbene noi esseri umani siamo naturalmente abili in questo tipo di compito e siamo capaci di trasmettere le conoscenze acquisite di generazione in generazione, la quantità di dati che una singola persona può analizzare è limitata. Il fascino del machine learning risiede proprio nella capacità dei computer di elaborare dataset estremamente vasti. Per raggiungere tale obiettivo, si parte da un modello matematico molto generale, che viene poi adattato ai dati specifici attraverso un processo di addestramento (*training* in inglese).
Tuttavia, va sottolineato come il modello finale sia spesso una "scatola nera" che, pur generando risultati affidabili, non offre informazioni dirette sui meccanismi fisici che governano il fenomeno studiato.

D'altro canto, la computazione quantistica, nata nei primi anni '80, utilizza i fenomeni della meccanica quantistica, come la sovrapposizione e l'entanglement, per elaborare dati tramite algoritmi specifici eseguiti su macchine quantistiche; a differenza dei computer classici, che processano informazioni in unità binarie - i bit - i computer quantistici utilizzano i qubit.
I potenziali vantaggi di questo nuovo paradigma nella teoria dell'informazione sono ancora oggetto di approfondito studio, ma sono già stati sviluppati diversi algoritmi quantistici per i quali è stata dimostrata una complessiva superiorità rispetto agli algoritmi classici. Tra questi, spiccano l'algoritmo di Shor [1] per la fattorizzazione di numeri interi e l'algoritmo di Grover [2] per la ricerca non strutturata all'interno di database.

Uno dei campi in cui la computazione quantistica può mostrare i suoi vantaggi è proprio quello del machine learning in una disciplina che fonde i metodi di entrambe: il machine learning quantistico.
Questa giovane disciplina affermatasi definitivamente a partire dal 2013 con l'articolo di Lloyd Mohseni e Rebentrost [3] oggi può contare su solide basi in numerosi ricercatori, aziende e startup disponendo anche di diversi framework come TensorFlow Quantum [4], Yao [5] e Pennylane [6], quest'ultima libreria è stata utilizzata nel presente elaborato.

L'obiettivo di questa dissertazione è quello di analizzare un modello di diffusione ibrido finalizzato alla generazione di immagini di galassie; ci si propone quindi di implementare un connubio tra modelli di diffusione classici e la computazione quantistica.

A tal fine si studiano dapprima le caratteristiche fondamentali di un modello di diffusione classico il quale è costituito da due passaggi principali: la diffusione diretta tramite una catena di Markov, e il processo inverso di denoising, che permette di generare nuovi dati sulla base di quelli iniziali.

Si sono successivamente analizzate le basi della computazione quantistica studiandone gli elementi costitutivi, ovvero qubit e porte logiche, fino all'analisi dell'applicazione di questa disciplina al machine learning.

E' stato infine possibile applicare tale modello al caso specifico della simulazione delle immagini galattiche; attraverso il linguaggio Python e la libreria Pennylane si è progettata un'implementazione che permette la generazione di immagini di galassie a partire da un dataset reale, ottenendo risultati non indistinguibili dalle immagini originali ma comunque soddisfacenti.
Per fare ciò, come anticipato, si è utilizzato un approccio ibrido che fondesse le due discipline descritte a livello teorico, utilizzando un autoencoder classico per comprimere le immagini, una catena di Markov classica, ed infine un circuito quantistico parametrico che svolgesse il compito di denoising del processo di diffusione.

Capitolo 1

La Simulazione di Immagini di Galassie

1.1 Motivazioni

Sebbene il machine learning quantistico sia una disciplina ancora agli albori, le sue applicazioni sono già state ipotizzate in numerosi ambiti della fisica e non solo.
Potenziali impieghi sono ad esempio nel campo della fisica delle alte energie [7] o per l'anomaly detection delle LLP (Long-Lived Particles) [8].
In particolare un'ulteriore applicazione esplorata nella presente dissertazione del machine learning quantistico è la generazioni di immagini a partire da dati reali.

L'impiego di simulazioni è infatti una delle tecniche più utilizzate nell'astronomia moderna; oltre alle complesse simulazioni dinamiche, anche quelle relative alle immagini rivestono un ruolo di grande importanza.
Nello specifico, una possibile applicazione della simulazione di immagini di galassie, è la generazione rapida di numerosi dati realistici necessari per l'addestramento e validazione delle più disparate reti neurali, le quali richiedono un enorme quantitativo di immagini per svolgere compiti come la classificazione galattica, il completamento di dati osservativi reali, l'identificazione di oggetti anomali, e così via.
Inoltre va sottolineato come il metodo utilizzato in questa dissertazione è teoricamente applicabile, al netto dei costi computazionali, per la generazione di una qualsiasi immagine, offrendo in base al contesto un vastissimo numero di applicazioni diverse. Sarebbe ad esempio possibile, in frazioni di secondo, generare immagini a partire da dati derivanti dalle simulazioni N-body, conducendo a rappresentazioni realistiche e riducendo considerevolmente i costi computazionali.

1.2 Dataset

Le immagini che si intendono generare in tale dissertazione sono galassie appartenenti a dieci classi distinte; i dati reali su cui si basa il modello sono le immagini del dataset Galaxy10 DECaLS contenente circa 18000 galassie categorizzate in classi, distinte sulla base delle loro caratteristiche morfologiche.

L'obiettivo del modello è pertanto quello di cogliere le caratteristiche tipiche, comuni a galassie facenti parte della stessa categoria, mantenendo tuttavia una elevata variabilità interna rendendo quindi le galassie uniche e simili a quelle reali.
Nel dettaglio, il dataset Galaxy10 DECaLS presenta le immagini fornite dal sondaggio DESI Legacy Imaging Surveys (DECaLS) [9]; in figura 1.1 sono riportati esempi di galassie per ciascuna classe del dataset.

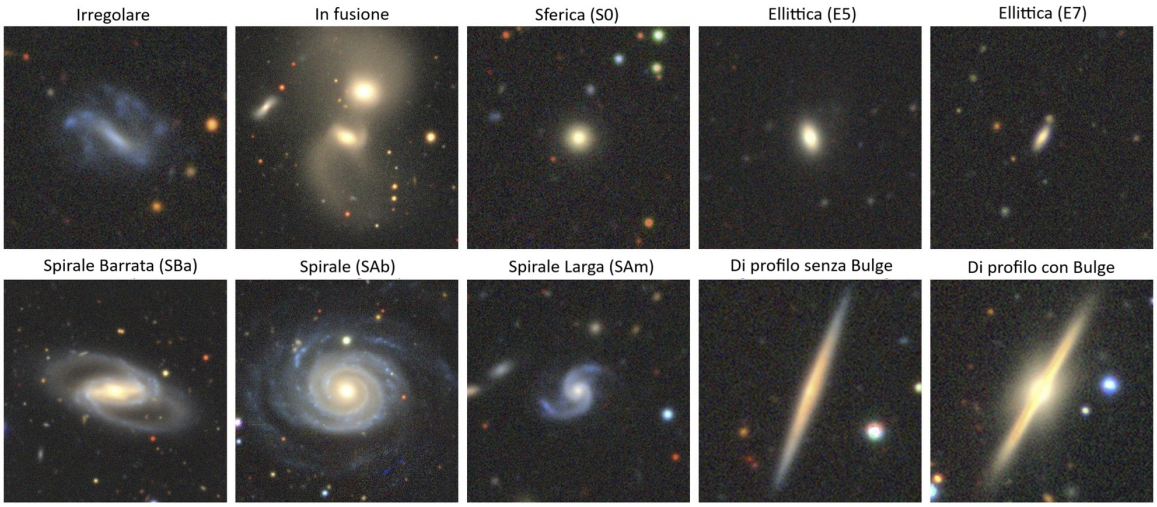

Figura 1.1 Classi galattiche in Galaxy10 DECaLS. In parentesi è riportata la rispettiva classificazione di Hubble.

1.3 Alcuni metodi esistenti

La capacità di generare immagini è uno dei compiti più utili e studiati nell'ambito del machine learning. Due dei modelli generativi più noti in grado quindi di apprendere e riprodurre strutture complesse all'interno dei dati sono le GAN [10] (Generative Adversarial Networks) e i VAE [11] (Variational Autoencoders) che rappresentano due approcci distinti per la generazione di dati sintetici.

GAN (Generative Adversarial Networks)

Le GAN sono architetture costituite da due reti neurali convoluzionali dette CNN (Convolutional Neural Networks).
Il funzionamento di una CNN si basa sulla conversione dei dati di input in vettori e l'elaborazione di questi; la CNN apprende varie caratteristiche visive tramite i suoi strati convoluzionali: nei primi strati la rete impara a riconoscere pattern semplici come bordi e angoli mentre nei livelli successivi la rete riconosce strutture più complesse come forme e oggetti. Così procedendo si arriva alla creazione di mappe di caratteristiche (*feature maps*).
All'interno di una GAN si impiegano due di queste reti neurali: una detta *generatore* e l'altra *discriminatore*, che vengono messe in contrapposizione. Entrambe le reti vengono addestrate su milioni di immagini in linea con quelle che l'algoritmo dovrà

produrre, successivamente il generatore inizierà a produrre nuove rappresentazioni mentre il discriminatore cercherà di distinguere i dati reali da quelli sintetici ottenuti dal generatore; il processo continua finché il discriminatore approva l'output fittizio riconoscendolo come reale.
Questo meccanismo molto semplice, in cui le due reti neurali competono costantemente a livelli sempre più elevati, consente di ottenere dati sintetici sempre migliori. Tuttavia questo modello presenta dei limiti: poiché il generatore e il discriminatore lavorano in modo collaborativo e competitivo, il processo di addestramento tende a diventare instabile, portando a risultati potenzialmente non ottimali.

VAE (Variational Autoencoders)

I VAE sono un sottoinsieme della più ampia categoria degli autoencoder, ovvero un'architettura di reti neurali costituita da due elementi fondamentali:

- **L'encoder**, il cui obiettivo è codificare i dati di input comprimendoli in uno spazio latente di ridotta dimensionalità; ciò è fatto gradualmente attraverso gli strati dell'encoder mediante i cosiddetti *filtri convoluzionali*.
- **Il decoder**, il cui obiettivo è quello di ricostruire il più fedelmente possibile i dati originali a partire dai vettori nello spazio latente generati dall'encoder; per fare ciò agisce in modo analogo all'encoder andando tuttavia ad aumentare gradualmente la dimensione dei vettori di input.

La differenza tra gli autoencoder tradizionali e i VAE è che i primi sono modelli deterministici che codificano l'input in un punto nello spazio latente; i secondi invece sono modelli probabilistici che codificano l'input nello spazio latente come una distribuzione di probabilità.
In particolare, l'encoder del VAE esprime ogni input attraverso due vettori, uno delle medie (μ) e uno delle deviazioni standard (σ), che definiscono una distribuzione gaussiana multivariata $\mathcal{N}(\mu, \sigma)$ nello spazio latente.
La scelta di una funzione regolare come prior per lo spazio latente permette al modello di campionare punti da questa distribuzione, mantenendo la coerenza con i dati di addestramento.
Attraverso il campionamento di un punto casuale nello spazio latente, il decoder può ricostruire un dato simile, ma non identico, all'input originale. Questo approccio probabilistico rende il VAE capace di generare varianti realistiche dei dati di addestramento esplorando lo spazio latente in modo continuo, permettendo la creazione di dati sintetici nuovi.

Per entrambi questi modelli sono state sviluppate anche le rispettive controparti quantistiche [12][13],[14].
Si inseriscono in questo contesto i modelli di diffusione, che offrono una valida alternativa ai modelli appena citati grazie alla loro superiore qualità e variabilità dei dati generati.

Capitolo 2

Modelli di Diffusione e Quantum Machine Learning

Per la simulazione delle immagini è stata scelta una delle più recenti tecniche generative: il modello di diffusione. Nel seguente capitolo si analizzano dapprima i modelli di diffusione classici e la loro logica fondante; a seguire, si trattano a livello teorico le basi della computazione e del machine learning quantistico.

2.1 Modelli di Diffusione Classici

I modelli stocastici di diffusione, introdotti per la prima volta nel 2015 e ispirati ai processi di diffusione nella termodinamica del non-equilibrio [15], sono alla base di alcuni dei più avanzati generatori di immagini, come Stable Diffusion [16] e DALL-E [17]. Questi modelli si articolano concettualmente in due processi principali:

- Processo di diffusione diretto
- Denoising o processo di diffusione inverso

Nel processo di diffusione diretto i dati di addestramento vengono progressivamente disturbati dall'aggiunta di rumore gaussiano attraverso una catena di Markov; l'obiettivo del modello sarà quindi quello di imparare a ricostruire l'immagine invertendo il processo di diffusione.
Dopo la fase di training il modello sarà in grado di generare nuove immagini, simili a quelle di partenza mediante il processo di denoising di un rumore gaussiano campionato casualmente.

Diffusione diretta

Lo scopo della diffusione diretta è quello di utilizzare una catena di Markov per mappare la distribuzione arbitraria che descrive l'immagine di partenza $q(\mathbf{x}_0)$ in una distribuzione gaussiana multivariata $\pi(\mathbf{x}_T)$, dove T è il numero di transizioni nella catena di Markov.
La singola transizione da $t-1$ a t, con $t \in \{1, ..., T\}$ può essere rappresentata con la distribuzione condizionata $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ la quale è comunemente scelta come:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) \coloneqq \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{2.1}$$

dove la sequenza di varianze $\beta_1, ..., \beta_T$ per ogni passo, è un iperparametro[1] del modello.
Una delle proprietà più utili nel definire in questo modo il processo di diffusione diretto è la possibilità di calcolare in forma chiusa la distribuzione di probabilità di qualsiasi passo t a partire dalla sola distribuzione iniziale:

$$q(\mathbf{x}_t|\mathbf{x}_0) \coloneqq \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}), \tag{2.2}$$

dove al fine di alleggerire la notazione si è definita: $\bar{\alpha}_t \coloneqq \prod_{s=1}^{t}(1-\beta_s)$.
Si può inoltre rendere più esplicito il concetto dell'*aggiunta di rumore* gaussiano esplicitando la variabile aleatoria x_t in funzione di distribuzioni normali standardizzate:

$$\begin{cases} \mathbf{x}_t(\mathbf{x}_{t-1}, \varepsilon) = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\varepsilon} \\ \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \end{cases} \qquad \begin{cases} \mathbf{x}_t(\mathbf{x}_0, \varepsilon) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon} \\ \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \end{cases} \tag{2.3}$$

In questo modo risulta particolarmente evidente la relazione tra i dati iniziali e quelli ottenuti nei passi successivi della catena di Markov attraverso l'aggiunta del rumore.
In figura 2.1 è schematizzato il processo di diffusione diretta:

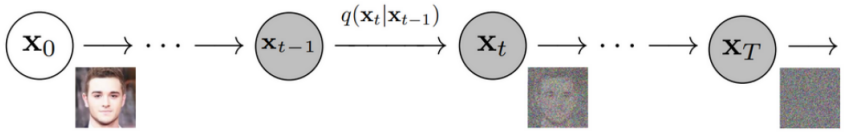

Figura 2.1 Schema della diffusione diretta. Fonte: [18]

[1] In questa dissertazione le varianze vengono fissate a priori, ma in linea di principio possono essere apprese dal modello con la riparametrizzazione [11].

Diffusione inversa e training

Il cuore del modello è, come già accennato, il processo di diffusione inversa.
Il modello parametrico è infatti addestrato cercando di stimare la probabilità condizionata $p_\theta(\mathbf{x}_{t-1}|x_t)$, dove θ sono i parametri del modello da apprendere durante l'addestramento.
Fatto ciò la distribuzione finale del processo inverso sarà poi computabile con la legge delle probabilità totali a partire dalla distribuzione $\pi(\mathbf{x}_T)$:

$$p_\theta(\mathbf{x}_0) = \int \pi(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|x_t)\, d\mathbf{x}_{1:T} \tag{2.4}$$

Quando β_t è sufficientemente piccolo la distribuzione di probabilità condizionata inversa sarà formalmente identica alla distribuzione diretta [15]:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I}) \tag{2.5}$$

I parametri del modello sono quindi relativi alle medie delle distribuzioni normali delle probabilità condizionate, essi sono stimati ottimizzando la funzione di loss.
Nel caso specifico dei modelli di diffusione, l'obiettivo è trovare i parametri θ che massimizzano la verosimiglianza tra la distribuzione parametrizzata e i dati iniziali x_0. Si sta in questo modo richiedendo che il modello riesca a ricreare i dati x_0 con una probabilità elevata, suggerendo che la distribuzione generativa approssimi bene i dati reali.
Questo ragionamento si traduce nella volontà di minimizzare la seguente verosimiglianza logaritmica negativa: $\mathbb{E}[-\log p_\theta(\mathbf{x}_0)]$. Si può infine sfruttare la disuguaglianza di Jensen andando a minimizzare in fase di addestramento il limite superiore della verosimiglianza negativa:

$$\mathbb{E}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] =: L \tag{2.6}$$

È spesso comodo riscrivere la loss L appena definita in termini della divergenza di Kullback-Leibler, ma non essendo di particolare utilità nel seguito della dissertazione si rimanda a [18] per una trattazione più approfondita.
In figura 2.2 è schematizzato il processo di diffusione inversa:

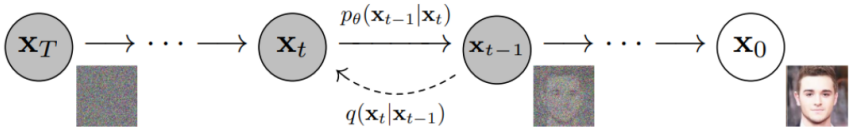

Figura 2.2 Schema della diffusione inversa. Fonte: [18]

2.2 La Computazione Quantistica

I computer quantistici sono dispositivi che eseguono calcoli sfruttando le leggi della meccanica quantistica. Tali macchine funzionano attraverso algoritmi, ovvero manipolazioni controllate del sistema quantistico. Il linguaggio in cui sono formulati tali algoritmi è il modello a circuito, che si basa sul concetto di qubit, unità che sostituisce il bit classico, e sulle porte logiche quantistiche utili per eseguire calcoli su tali qubit.

Qubit

È concezione comune l'idea che il potere dei computer quantistici risieda nella capacità dei qubit di esistere in una combinazione lineare degli stati 0 e 1 consentendo loro di assumere un qualsiasi "stato intermedio". Tuttavia tale affermazione non è del tutto corretta: l'informazione emerge infatti dal processo di misura del qubit che restituisce uno stato deterministico 0 o 1 ma la reale potenza deriva dalla sua natura quantistica e da fenomeni quali la sovrapposizione e l'entanglement.
In tal senso si può pensare ad un qubit come la realizzazione di un sistema quantistico a due livelli essendo misurabile in due stati di base. Utilizzando la notazione di Dirac si denotano gli stati di base con i ket $|0\rangle$ e $|1\rangle$ che formano la *base computazionale*; l'analogia con il caso classico è quindi immediata: lo stato $|0\rangle$ corrisponde ad un bit che assume il valore 0 e lo stato $|1\rangle$ corrisponde ad un bit che assume il valore 1.
I ket formano pertanto una base ortonormale nello spazio di Hilbert bidimensionale e il più generico ket rappresentante un qubit lo si può scrivere come:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle \tag{2.7}$$

dove $\alpha_0, \alpha_1 \in \mathbb{C}$ e tali che $|\alpha_0|^2 + |\alpha_1|^2 = 1$.
La generalizzazione ad n qubit introduce il fenomeno dell'*entanglement*. Quando i qubit sono non-entangled possono essere descritti separatamente l'uno dall'altro e lo stato collettivo $|\psi\rangle$ è dato dal prodotto tensoriale degli n qubit $|q_1\rangle ... |q_n\rangle$:

$$|\psi\rangle = |q_1\rangle \otimes ... \otimes |q_n\rangle \tag{2.8}$$

Quando i qubit sono entangled essi sono intrinsecamente legati tra loro; una misurazione di un qubit influisce sullo stato degli altri e pertanto lo stato $|\psi\rangle$ non è separabile nel prodotto tensoriale, quindi:

$$|\psi\rangle = \alpha_0 |0...00\rangle + \alpha_1 |0...01\rangle + ... + \alpha_{2^n-1} |1...11\rangle \tag{2.9}$$

Questa notazione può essere alleggerita sostituendo i ket della base computazionale di n qubit $\{|0...00\rangle, ..., |1...11\rangle\}$ con $\{|0\rangle, |1\rangle, ..., |n\rangle\}$ interpretando le sequenze di 0 e 1 come se fossero rappresentazioni binarie. Così facendo si riscrive:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle \tag{2.10}$$

dove $\alpha_i \in \mathbb{C}$ e $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$

Rappresentazione geometrica

Risulta spesso utile fornire una rappresentazione geometrica dei qubit. La descrizione dello stato puro (2.7) utilizza quattro parametri reali, tuttavia dato che solo la fase relativa tra i coefficienti dei ket di base ha significato fisico, e data l'imposizione della normalizzazione, tale descrizione risulta ridondante.
È infatti sufficiente una rappresentazione dei qubit su una 2-sfera unitaria: la $sfera$ *di Bloch*, i cui poli nord e sud corrispondono rispettivamente ai ket di base $|0\rangle$ e $|1\rangle$. Scegliendo quindi il coefficiente di $|0\rangle$ reale e non negativo si può parametrizzare uno qubit generico come:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \quad \text{con } 0 \leq \theta \leq \pi \text{ e } 0 \leq \phi < 2\pi \tag{2.11}$$

In questo modo lo spazio di Hilbert dei ket $|\psi\rangle$ può essere visualizzato come la superficie di una sfera in $\mathbb{R}^3$ parametrizzata in coordinate sferiche mediante il vettore $(\sin\theta\cos\phi,\ \sin\theta\sin\phi,\ \cos\theta)$ come illustrato in figura 2.3:

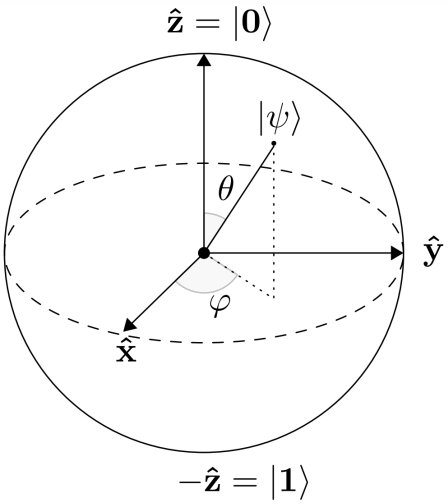

Figura 2.3 Sfera di Bloch in $\mathbb{R}^3$. Fonte: [19]

Gate Quantistici

Un teorema fondamentale nell'informazione quantistica afferma che qualsiasi evoluzione quantistica fondante un algoritmo possa essere approssimata da una sequenza di poche manipolazioni elementari, chiamati gate quantistici (porte logiche quantistiche), che agiscono solo su uno o due qubit alla volta [20].
Le porte logiche sono trasformazioni unitarie che si applicano allo stato $|\psi\rangle$; i gate applicabili ad un qubit sono quindi descritti da matrici 2x2 unitarie. A differenza del caso classico, il formalismo complesso della meccanica quantistica permette che tali matrici possano presentare anche entrate negative e complesse.
Nel seguito si presentano alcuni tra i più importanti gate.

Gate di Pauli

I gate di Pauli ($\mathbf{X}, \mathbf{Y}, \mathbf{Z}$) sono le porte logiche descritte dalle tre matrici di Pauli:

$$\mathbf{X} = \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \mathbf{Y} = \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \mathbf{Z} = \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.12}$$

Si noti come $\mathbf{X}$ è l'equivalente della porta NOT nella computazione classica, infatti:

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

Similmente $\mathbf{X}\,|0\rangle = |1\rangle$

Gate di Hadamard

Il gate di Hadamard ($\mathbf{H}$) è definito dalla seguente matrice unitaria:

$$\mathbf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.13}$$

La sua azione sui ket di base è quindi la seguente:

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

Pertanto questo gate permette portare il qubit in uno stato di sovrapposizione dei ket della base computazionale.

Gate CNOT

Il gate CNOT è una porta logica che si applica a due qubit ed è un esempio di gate controllato. In particolare la sua funzione è quella di applicare una porta $\mathbf{X}$ (NOT classico) al secondo qubit qualora il primo fosse nello stato $|1\rangle$. la sua matrice è:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.14}$$

Pertanto l'azione sarà:

$$|00\rangle \to |00\rangle\,, \quad |01\rangle \to |01\rangle\,, \quad |10\rangle \to |11\rangle\,, \quad |11\rangle \to |10\rangle\,.$$

È interessante notare come in un sistema a due qubit nello stato iniziale $|00\rangle$, l'applicazione successiva del gate di Hadamard sul primo qubit e del gate CNOT permette di creare uno stato entangled chiamato *stato di Bell*:

$$CNOT((\mathbf{H} \otimes \mathbb{I})(|0\rangle \otimes |0\rangle))$$

Applicando l'Hadamard gate si ottiene:

$$\text{CNOT}\left(\frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|10\rangle\right) = \frac{1}{\sqrt{2}}\,(|00\rangle + |11\rangle)$$

Gate Rotazionali

Particolarmente utili per il seguito, sono i gate rotazionali. Queste porte effettuano delle rotazioni parametrizzate con un angolo θ sui qubit intorno ad uno dei tre assi (X, Y, Z) sulla sfera di Bloch. Le tre matrici che definiscono tali gate si ricavano a partire dalle matrici di Pauli e sono:

$$\mathbf{R}_x(\theta) = e^{-i\frac{\theta}{2}\sigma_x} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{2.15}$$

$$\mathbf{R}_y(\theta) = e^{-i\frac{\theta}{2}\sigma_y} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{2.16}$$

$$\mathbf{R}_z(\theta) = e^{-i\frac{\theta}{2}\sigma_z} = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} \tag{2.17}$$

L'importanza di questi gate nel machine learning quantistico risiede nella loro parametrizzazione che permette la creazione di circuiti i cui parametri sono addestrabili proprio come nel machine learning classico.

Codifica dei dati - Amplitude Encoding

Un aspetto fondamentale dell'elaborazione delle informazioni quantistiche riguarda la rappresentazione di dati classici mediante stati quantistici. I metodi utilizzabili per perseguire questo obiettivo sono molteplici, nel seguito si tratta l'Amplitude Encoding[2] essendo questo il metodo utilizzato nel modello di diffusione.
L'Amplitude Encoding consiste nel codificare i dati iniziali come le ampiezze dei ket di base dello spazio di Hilbert. L'utilità del metodo è chiara: grazie alla sovrapposizione quantistica mediante n qubit si può rappresentare un vettore di $2^n - 1$ componenti:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_{2^n} \end{pmatrix} \quad \longleftrightarrow \quad |\psi_{\mathbf{x}}\rangle = \sum_{j=0}^{2^n-1} x_j |j\rangle. \tag{2.18}$$

Tuttavia vanno sottolineate anche le limitazioni del metodo: è infatti necessario che il vettore $\mathbf{x}$ abbia norma 1 affinché sia codificabile nelle ampiezze dello stato quantistico $|\psi_{\mathbf{x}}\rangle$; inoltre per costruire la sovrapposizione è necessario un numero di porte CNOT che cresce esponenzialmente con il numero di dati da codificare.

Un esempio di interferenza con il gate di Hadamard

L'obiettivo del seguente paragrafo è far intravedere come il formalismo della meccanica quantistica possa essere sfruttato per ottenere vantaggi significativi rispetto alla computazione classica. Per fare ciò si propone una semplice applicazione del gate di Hadamard su un sistema di due qubit comparandola all'operazione equivalente su due classici bit.

[2] Altri metodi sono il Basis Encoding, Hamiltonian Encoding, Angle Encoding [21].

Si immaginino due bit classici entrambi nello stato 0 e si supponga di effettuare un'operazione che permetta di rendere completamente casuale il primo bit (proprio come il lancio di una moneta). Il risultato di tale operazione sarà, con ugual probabilità, ancora lo stato 00 oppure lo stato 10. Applicando una seconda volta la medesima operazione (sempre sul primo bit) la situazione nel caso classico resta invariata: i due stati possibili, con ugual probabilità, sono ancora 00 o 10.
Formalmente si può rendere quanto detto attraverso il vettore di probabilità $\mathbf{p}$ che descrive lo stato dei due bit; il risultato dell'applicazione della prima operazione si può pertanto scrivere:

$$\mathbf{p} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \rightarrow \mathbf{p}' = \frac{1}{2} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix},$$

Dove la prima componente di $\mathbf{p}$ rappresenta la probabilità dello stato 00, la seconda dello stato 01, la terza dello stato 10 e l'ultima dello stato 11. La trasformazione che implementa l'operazione nel formalismo matriciale è quindi data dalla seguente matrice stocastica:

$$\mathbf{S} = \frac{1}{2} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

Risulta pertanto evidente quanto detto sopra: $\mathbf{Sp} = \mathbf{p}'$, tuttavia una seconda applicazione conduce a $\mathbf{Sp}' = \mathbf{p}'$.
Tale esempio cambia radicalmente nel formalismo quantistico con l'applicazione del gate di Hadamard, che può essere considerato la controparte quantistica che implementa l'operazione poc'anzi descritta.
Nell'approccio quantistico i vettori di probabilità diventano i vettori delle ampiezze $\boldsymbol{\alpha}$ e la probabilità di osservare uno stato sarà il modulo quadro dell'ampiezza corrispondente; pertanto lo stato iniziale risulta essere:

$$\boldsymbol{\alpha} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

l'operatore che implementa il gate di Hadamard sul primo qubit in uno stato con due qubit si può scrivere attraverso il prodotto tensoriale: $\mathbf{H}_2^{(q_1)} = \mathbf{H} \otimes \mathbb{I}$; pertanto la matrice che implementa $\mathbf{H}_2^{(q_1)}$ è data da:

$$\mathbf{H}_2^{(q_1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}.$$

Moltiplicando quindi $\mathbf{H}_2^{(q_1)}$ per il vettore $\boldsymbol{\alpha}$ si ottiene il seguente vettore $\boldsymbol{\alpha}'$:

$$\boldsymbol{\alpha}' = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix},$$

Analogamente al caso classico una misura dello stato finale produrrà con ugual probabilità sia lo stato 00 che 10. Tuttavia, applicando la matrice $\mathbf{H}_2^{(q_1)}$ una seconda volta, il risultato sarà profondamente diverso: $\mathbf{H}_2^{(q_1)}\boldsymbol{\alpha}' = \boldsymbol{\alpha}$ e non $\boldsymbol{\alpha}'$ come nel caso classico. In modo controintuitivo ciò vuol dire che, nel caso quantistico, l'operazione di rendere casuale il primo qubit attraverso il gate di Hadamard applicata due volte allo stato 00, conduce con certezza allo stesso stato iniziale.
Tutto ciò è ovviamente generalizzabile al caso di un sistema con n qubit; la matrice che implementa il gate di Hadamard sul primo qubit sarà data da:

$$\mathbf{H}_n^{(q_1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbb{I} & \mathbb{I} \\ \mathbb{I} & -\mathbb{I} \end{pmatrix} \tag{2.19}$$

dove $\mathbb{I}$ è la matrice identità di dimensioni $\frac{2^n}{2}\text{x}\frac{2^n}{2}$.
Denotando con a la prima metà delle componenti del vettore delle ampiezze e con b la seconda metà, l'applicazione del gate di Hadamard restituirà il vettore $(a+b, a-b)^T$. Si noti che per quanto il gate venga applicato solamente sul primo qubit la trasformazione agisce su tutte le 2^n ampiezze, inoltre il formalismo complesso fa in modo che tali ampiezze possano essere soggette ad interferenza costruttiva o distruttiva portando a risultati profondamente diversi dai casi classici, proprio come nell'esempio appena descritto. È in questo fenomeno (oltre che in quello dell'entanglement) che risiedono i potenziali vantaggi della computazione quantistica.

Vantaggi e Stato dell'Arte

Esistono algoritmi quantistici il cui tempo di esecuzione cresce più lentamente con la dimensione dell'input rispetto agli algoritmi classici noti che risolvono lo stesso problema.
Inoltre ogni algoritmo classico può essere implementato su un computer quantistico con un costo aggiuntivo polinomiale, quindi un computer quantistico è almeno tanto efficiente quanto un computer classico dal punto di vista della complessità asintotica. Tuttavia costruire un computer del genere è un compito complesso in quanto richiede un controllo accurato che non disturbi la fragile coerenza quantistica; questo fa si che la correzione degli errori per tali sistemi sia molto più difficile dell'analogo classico essendo pertanto una delle principali sfide ingegneristiche nello sviluppo di questi computer su larga scala.
Sono stati fatti molti progressi in tal senso con lo sviluppo dei cosiddetti dispositivi quantistici NISQ [22] (Noisy Intermediate-Scale Quantum) che attualmente contano un numero dell'ordine di 10-100 qubit e che sono i primi prototipi di quello che un giorno potrebbe essere un computer quantistico completo.
Tuttavia essi presentano delle problematiche: non permettono infatti di effettuare la correzione degli errori e le computazioni dell'hardware producono alti livelli di rumore che rendono impossibile l'implementazione di circuiti particolarmente complessi.

2.3 Quantum Machine Learning

Il Quantum Machine Learning è un'area di ricerca che unisce organicamente le idee della computazione quantistica con quelle del machine learning.
Tali modelli si basano su circuiti quantistici variazionali (o parametrizzati) schematizzabili attraverso una struttura di porte quantistiche parametriche (come i gate rotazionali). In questo modo i circuiti sono ottimizzabili tramite la minimizzazione di una funzione di loss.
Va sottolineato che le limitazioni dell'hardware quantistico attuale richiedono l'uso di algoritmi ibridi, che combinano calcoli quantistici brevi con operazioni classiche, questo è l'approccio utilizzato nell'implementazione del modello di diffusione [Capitolo 3].

Circuito Quantistico come Modello di Machine Learning

Un modello di machine learning classico può essere interpretato in duplice maniera: come una funzione $f_\theta : X \to Y$ che mappa un input di dati su un dominio di output oppure come una distribuzione di probabilità $p_\theta : X \otimes Y \to [0, 1]$, entrambe le scelte dipendono da un insieme di parametri addestrabili θ.
Analogamente i circuiti quantistici variazionali possono essere interpretati sia come modelli di machine learning deterministici che probabilistici.

Modello Quantistico Deterministico

Si definisce un modello quantistico deterministico, un circuito parametrizzato che, dato un insieme di input X e dei parametri θ, restituisce un output ottenuto attraverso una funzione deterministica. Più formalmente, denotando il circuito quantistico con $U(x, \theta)$ si può definire il modello deterministico come una funzione:

$$f_\theta(x) = \langle \psi(x, \theta)| \, \mathcal{M} \, |\psi(x, \theta)\rangle \tag{2.20}$$

dove $\mathcal{M}$ è l'osservabile quantistica e $|\psi(x, \theta)\rangle$ è uno stato preparato tramite il circuito quantistico e le sue porte logiche ovvero $|\psi(x, \theta)\rangle = U(x, \theta) \, |0\rangle$.

Modello Quantistico Probabilistico

Un modello quantistico probabilistico sfrutta la natura aleatoria delle misurazioni quantistiche, quindi l'output non è un singolo valore deterministico bensì è una distribuzione di probabilità sui possibili risultati della misurazione. Più formalmente se X e Y sono rispettivamente i domini di input e output e $U(x, \theta)$ rappresenta il circuito quantistico, si definisce un modello quantistico probabilistico la distribuzione:

$$p_\theta(y|x) = | \, \langle y|\psi(x, \theta)\rangle \, |^2 \tag{2.21}$$

dove ancora una volta lo stato $|\psi(x, \theta)\rangle = U(x, \theta) \, |0\rangle$.
Questo è di fatto il modello utilizzato nel seguito per l'implementazione del modello di diffusione.

Circuito Quantistico come Reti Neurali

Le reti neurali quantistiche sono circuiti parametrizzati che imitano il comportamento delle reti neurali classiche.
La loro peculiarità è la loro architettura stratificata; sono infatti formate da layer di gate parametrici e ottimizzabili, simili ai pesi dei modelli di reti neurali classiche.
La non linearità delle reti neurali quantistiche (possibile in quelle classiche con le funzioni di attivazione) è resa indirettamente attraverso processi di misurazione dei qubit. È pertanto la stessa evoluzione quantistica complessa, attraverso numerosi gate rotazionali e operazioni di entanglement e misurazioni, a garantire un' ampia espressività del modello.
L'addestramento avviene in modo del tutto analogo al caso classico, ovvero minimizzando una funzione di loss.

Capitolo 3

Implementazione

In questo capitolo si discute l'implementazione del modello di diffusione per la simulazione di immagini di galassie.
Come già accennato si è optato per l'utilizzo di un algoritmo ibrido. L'implementazione è stata eseguita in Python, sfruttando la libreria Pennylane per il circuito variazionale.

3.1 Algoritmo ibrido

Data l'enorme quantità di dati presenti nel dataset, è impensabile la trattazione di un simile problema a livello puramente quantistico; il numero di qubit richiesti sarebbe troppo elevato per essere gestiti su una macchina classica che simuli il comportamento quantistico.
Si è pertanto optato per un modello ibrido costituito da un autoencoder che permettesse la compressione delle immagini in uno spazio latente 8x8. Su questi elementi nello spazio latente è stata applicata la catena di Markov precedentemente discussa. Pertanto solo il processo di diffusione inverso è stato trattato quantisticamente attraverso un circuito parametrizzato che ha svolto la funzione di denoising.
Nonostante l'approccio ibrido adottato, non è stato possibile addestrare il modello direttamente sulle immagini del dataset, composto da immagini RGB di dimensione 256x256 pixel. Si è infatti tentato di comprimere direttamente tali immagini utilizzando un encoder composto da 8 layer convoluzionali, con l'obiettivo di codificarle in uno spazio latente sufficientemente ridotto da essere gestito con un numero limitato di qubit.
Tuttavia, la profondità dell'autoencoder ha reso lo spazio latente eccessivamente complesso e altamente non lineare, ostacolando l'ottimizzazione efficace dei parametri del circuito quantistico. Inoltre, si sono riscontrati ulteriori problemi, tra cui il rischio di overfitting.
Per ovviare a tali problematiche le immagini del dataset sono state ridimensionate ad una grandezza di 28x28 pixel e trasformate in scala di grigi.

3.2 Autoencoder

L'autoencoder implementato presenta un architettura composta da 3 layer convoluzionali seguito da uno strato completamente connesso.
Come funzione di attivazione tra i layer è stata scelta la Leaky ReLU la quale, a differenza della ReLU standard, mitiga il problema dello spegnimento progressivo dei neuroni che ricevono in input dati con valori negativi.
La peculiarità dell'autoencoder implementato consiste nell'imposizione di una normalizzazione $\mathbf{L}_2$ sui vettori dello spazio latente. Tale strategia è stata necessaria al fine di addestrare il decoder a decomprimere vettori già normalizzati come quelli che avrebbe ricevuto in input dal circuito quantistico. Quest'ultimo infatti codificando dati con l'amplitude encoding, avrebbe dovuto restituire interamente gli stati quantistici, ovviamente normalizzati.
L'addestramento dell'autoencoder è avvenuto con l'ottimizzazione della loss SSIM [23] che ha dato risultati migliori rispetto la classica MSE. In particolare tale loss fornisce una misura di similarità tra le immagini basandosi su luminanza, contrasto e struttura. restituendo un valore pari a 1 in caso di immagini identiche, 0 se non vi è alcuna somiglianza.
In figura 3.1 è possibile visionare il funzionamento dell'autoencoder appena discusso su un set di immagini del dataset ridimensionato.

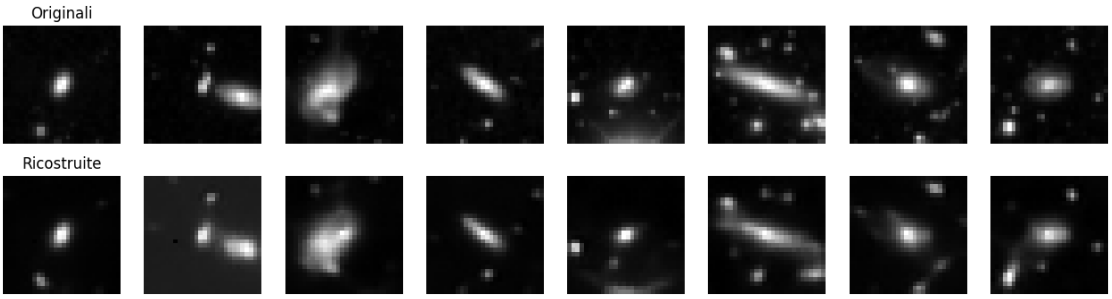

Figura 3.1 Esempi del funzionamento dell'autoencoder.

Un'attenzione particolare è stata dedicata a prevenire il fenomeno di overfitting da parte dell'autoencoder. In particolare, è stato garantito che il decoder non fosse in grado di interpretare vettori dello stato latente completamente rumorosi, decodificandoli in immagini di galassie. In tal modo, pur raggiungendo l'obiettivo finale, non si farebbe alcun ricorso al circuito quantistico, il quale non apporterebbe alcun beneficio al processo di denoising, che verrebbe eseguito esclusivamente dal decoder (similmente ad un VAE). Questa problematica, ad esempio, si manifesta nel tentativo discusso nel paragrafo precedente, dove un autoencoder significativamente più profondo campiona le immagini di input in uno spazio latente di dimensioni eccessivamente ridotte.

3.3 Catena di Markov

Per l'applicazione della catena di Markov si è scelto un numero di transizioni pari a $T = 15$.
Per quanto riguarda la scelta della sequenza delle varianze, sono state condotte due prove: una con una progressione lineare di β_t come suggerito da [18] e l'altra con una variazione di $\bar{\alpha}_t$ basata sul coseno al quadrato [24]. In quest'ultima configurazione, è stato osservato un lieve miglioramento delle performance del modello.
In particolare nel caso della sequenza cosinusoidale si è scelto:

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, f(t) = \cos\left(\frac{\frac{t}{T} + s}{1+s} \cdot \frac{\pi}{2}\right)^2, \text{ con } \quad s = 0.008 \tag{3.1}$$

Le motivazioni di tale scelta risiedono nei due diversi andamenti di $\bar{\alpha}_t$, infatti in riferimento al grafico 3.2, la progressione lineare di β_t si ripercuote in un andamento di $\bar{\alpha}_t$ che produce, verso la fine del processo diretto, stadi troppo rumorosi che non contribuiscono all'allenamento del modello, non portando informazione su come ricostruire l'immagine originale.
D'altra parte la sequenza delle varianze col coseno permette una discesa che aggiunge rumore in modo più graduale evitando la distruzione dell'informazione più velocemente del necessario. È importante sottolineare come ciò implichi che β_t, durante la

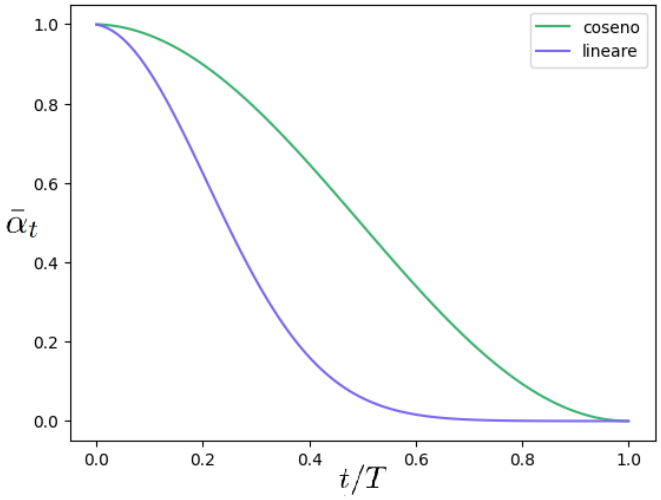

Figura 3.2 Andamento di $\bar{\alpha}_t$ nelle due sequenze delle varianze.

sequenza, non rimanga significativamente inferiore a uno, come indicato nella teoria per garantire che anche il processo di denoising segua una distribuzione gaussiana [15]. Tuttavia, questo non incide sul funzionamento del modello, poiché il circuito quantistico non verrà addestrato per apprendere i parametri della gaussiana, ma piuttosto è progettato per ricostruire direttamente le immagini.
Infine è importante evidenziare che, a causa della presenza del circuito quantistico, i valori delle ampiezze degli stati divengono complessi anche partendo da un rumore reale; pertanto, la catena di Markov è stata leggermente modificata nel seguente

modo, al fine di aumentare le prestazioni del modello:

$$\begin{cases} \varepsilon = \varepsilon_r + i\varepsilon_i \\ \varepsilon_r, \varepsilon_i \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \end{cases} \tag{3.2}$$

Mediante l'utilizzo delle relazioni (2.3) ogni immagine del dataset, codificata dall'encoder, è stata portata ad uno specifico passo $t \in \{0, ..., T-1\}$ della catena di Markov. Per tutte le immagini così ottenute si è considerato poi il passo successivo $t+1$ ottenendo due insiemi di immagini: rispettivamente il target e lo stato iniziale, necessari per addestrare la rete neurale quantistica per il denoising da $t+1$ a t.

3.4 Circuiti Quantistici Parametrici

I circuiti quantistici parametrici si occupano del denoising delle rappresentazioni latenti rumorose. Si può quindi intendere il t-esimo circuito come un operatore $U_t(\boldsymbol{\theta})$ che agisce sullo stato $|x_t\rangle$ nel seguente modo: $U_t(\boldsymbol{\theta})\,|x_t\rangle = |x_{t-1}\rangle$.
I circuiti implementati presentano sei qubit in modo tale da rappresentare tutti i pixel delle immagini rumorose nello spazio latente; come già anticipato infatti, per mezzo dell'amplitude encoding è stato possibile rappresentare tutti i 64 valori dei dati di input.
L'ansatz per il singolo layer di ogni circuito è lo *Strongly Entangling Layer* [25] mostrato in figura 3.3 per tre qubit.

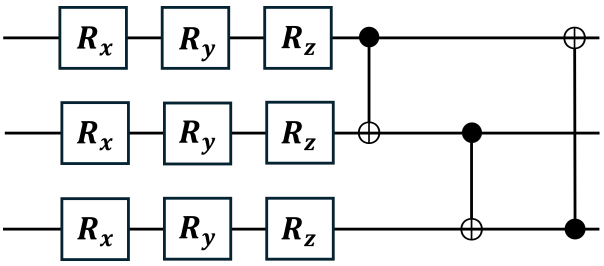

Figura 3.3 Schema dell'ansatz utilizzato nel circuito parametrico per 3 qubit.

Dove con R_x, R_y, R_z si indicano i gate rotazionali parametrici descritti nel capitolo precedente, mentre con i simboli successivi sono indicati una serie di CNOT applicati ciclicamente a coppie di qubit che permettono l'entanglement.
L'architettura complessiva dei circuiti è quella suggerita da [26] chiamata $reverse - bottleneck$. Essa è costituita da tre blocchi parametrici, ognuno dei quali presenta un numero variabile di layer costituiti come sopra.
In particolare per il primo e terzo blocco si sono scelti 30 layer ciascuno, per il secondo invece 40, per un totale di 1920 parametri. La peculiarità di questa architettura è la presenza di un cosiddetto qubit *ancilla* che permette di aumentare il potere espressivo del modello[1]. Una schematizzazione del circuito quantistico per tre qubit è mostrata in figura 3.4

[1]Si può infatti pensare che il qubit ancilla con il suo entanglement con gli altri qubit e successiva misurazione agisca come una funzione non lineare aumentando l'espressività della rete. [26]

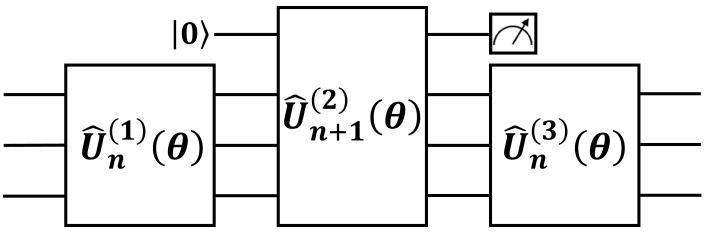

Figura 3.4 Schema del circuito reverse-bottleneck.

Per ogni passo della catena di Markov è stato addestrato uno specifico circuito che si occupasse del denoising del dato passo. L'addestramento di ogni circuito è stato effettuato attraverso un'ottimizzazione del tutto classica della seguente funzione di loss:

$$L(\boldsymbol{\theta}) = 1 - \mathbb{E}(\langle x_{t+1}| U^{\dagger}(\boldsymbol{\theta}, t) |x_t\rangle). \tag{3.3}$$

Nonostante l'elevato numero di layer non si sono riscontrate problematiche legate alla presenza dei *Barren Plateaus* [27].
La loss scelta inoltre è in linea di principio calcolabile anche su un hardware quantistico, resta però impraticabile, allo stato attuale, la realizzazione di questo esatto circuito su tali macchine a causa dell'elevata complessità dello stesso.

3.5 Risultati

La generazione di nuove immagini è stata possibile applicando in ordine i quindici circuiti quantistici ad un campionamento del tutto casuale di rumore gaussiano. Successivamente, l'immagine compressa di dimensioni 8x8 è stata elaborata dal decoder, che ne ha effettuato la decodifica per ricostruire un'immagine di una galassia in scala di grigi con dimensioni 28x28.
In Figura 3.5 sono riportate alcune immagini ricostruite mediante questo processo.

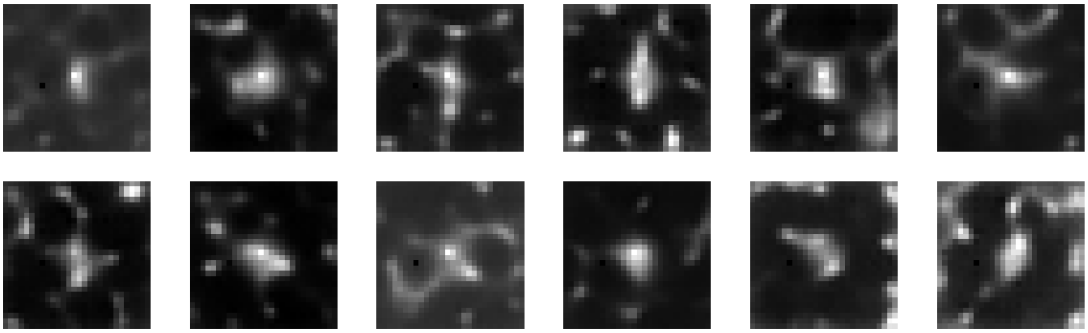

Figura 3.5 Galassie generate dal modello.

Per ottenere immagini più definite e compatibili con il formato del dataset, è stato progettato e addestrato un decoder in grado di trasformare immagini in scala di grigi con risoluzione 28x28 in immagini RGB 256x256. Questo decoder è stato addestrato utilizzando sempre il dataset Galaxy10 DECaLS: le immagini originali del dataset sono state impiegate come target, mentre le rispettive versioni ridimensionate hanno costituito gli input.
La struttura del decoder prevede un livello completamente connesso seguito da quattro strati deconvoluzionali, che incrementano progressivamente la risoluzione dell'immagine. La funzione di attivazione adottata è stata ancora la Leaky ReLU, ad eccezione dell'ultimo strato, dove è stata utilizzata una funzione sigmoide per limitare l'output all'intervallo $[0, 1]$.
L'applicazione del decoder alle immagini precedentemente generate produce i risultati in figura 3.6:

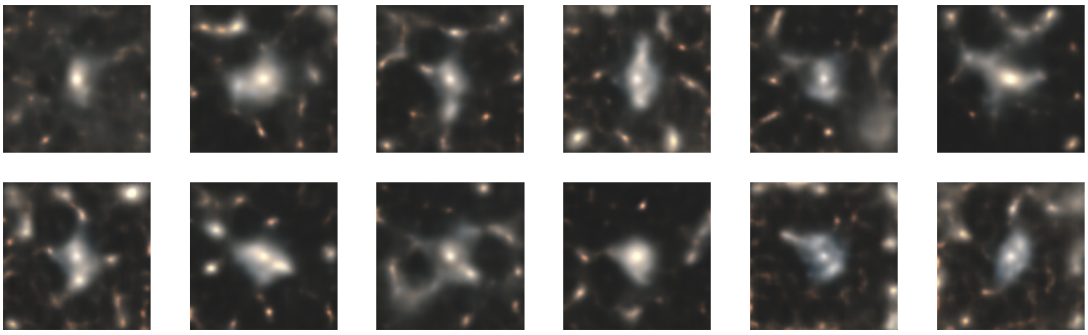

Figura 3.6 Galassie RGB 256x256 generate dal modello.

Dalle figure 3.5 e 3.6 è chiaro che le immagini generate non siano particolarmente definite e risultino ancora abbastanza rumorose rispetto le immagini iniziali. Le caratteristiche peculiari delle galassie, come prevedibile, si sono dimostrate troppo difficili da apprendere conducendo ad un modello che genera galassie molto simili tra loro con una forma irregolare e sfocata.
Tuttavia, è importante sottolineare che, nonostante queste limitazioni, il modello rappresenta un risultato soddisfacente, soprattutto se si considerano le risorse computazionali limitate a disposizione durante lo sviluppo.
La capacità di generare immagini che, seppur non perfettamente definite, presentano strutture riconducibili a galassie dimostra che il modello riesce comunque a catturare alcune delle caratteristiche fondamentali del dataset.
In un contesto di risorse computazionali più elevate, miglioramenti significativi potrebbero essere ottenuti aumentando la complessità del modello, il numero di qubit (e quindi di parametri addestrabili) o il volume di dati di addestramento. Nonostante ciò, i risultati ottenuti forniscono una solida base per futuri sviluppi e dimostrano l'efficacia dell'approccio adottato nel generare immagini coerenti con il dominio delle galassie.

Informazioni per la riproducibilità dei risultati

Il dataset utilizzato nella dissertazione è pubblico e disponibile al seguente indirizzo:
`https://astronn.readthedocs.io/en/latest/galaxy10.html`
Il codice Python utilizzato è disponibile contattando l'autore o analogamente sulla seguente pagina GitHub:
`https://github.com/Paddolo/QDM_galassie.git`

Capitolo 4

Conclusioni

Nella presente dissertazione è stato analizzato ed implementato un modello di diffusione ibrido in grado di generare immagini di galassie.
Pur utilizzando un numero di parametri inferiori rispetto le sue controparti classiche, il modello sviluppato presenta una discreta capacità nella riproduzione delle caratteristiche generali delle galassie per quanto le immagini non abbiano una qualità paragonabile a quelle reali.

Una possibile evoluzione al presente studio sarebbe l'esecuzione del circuito su un hardware quantistico. Va sottolineato come lo stato dell'arte degli attuali dispositivi NISQ imporrebbe numerose semplificazioni al modello, il quale risulta non trattabile con le attuali macchine quantistiche.
I principali ostacoli derivano dagli elevati tassi di errore legati ai gate CNOT, necessari per l'entangling, e dal ridotto tempo di rilassamento T_1 [28], ovvero il tempo entro il quale un qubit può essere utilizzato efficacemente prima di essere soggetto ad errori dovuti alla decoerenza.
Un ulteriore problema scaturisce dall'architettura delle attuali macchine, come la IBM-hanoi, che consentono l'applicazione delle porte CNOT solo tra qubit vicini rendendo impossibile la realizzazione della topologia ad anello per i sei qubit utilizzati nel circuito parametrico precedente.
Inoltre, l'addestramento del circuito quantistico dovrebbe essere ancora effettuato in un ambiente simulativo, poiché i problemi sopra esposti renderebbero impossibili i calcoli dei gradienti direttamente sulle macchine NISQ. In ogni caso, le semplificazioni necessarie per rendere il circuito implementabile (per il solo campionamento) su hardware quantistico, sarebbero enormemente impattanti sull'espressività del modello, rendendo particolarmente complessa l'efficace generazione di immagini galattiche.

Bibliografia

[1] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. DOI: 10.1137/S0097539795293172. eprint: https://doi.org/10.1137/S0097539795293172. URL: https://doi.org/10.1137/S0097539795293172.

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

[3] Seth Lloyd, Masoud Mohseni e Patrick Rebentrost. *Quantum algorithms for supervised and unsupervised machine learning*. 2013. arXiv: 1307.0411 [quant-ph]. URL: https://arxiv.org/abs/1307.0411.

[4] Michael Broughton et al. *TensorFlow Quantum: A Software Framework for Quantum Machine Learning*. 2021. arXiv: 2003.02989 [quant-ph]. URL: https://arxiv.org/abs/2003.02989.

[5] Xiu-Zhe Luo et al. "Yao.jl: Extensible, Efficient Framework for Quantum Algorithm Design". In: *Quantum* 4 (ott. 2020), p. 341. ISSN: 2521-327X. DOI: 10.22331/q-2020-10-11-341. URL: http://dx.doi.org/10.22331/q-2020-10-11-341.

[6] Ville Bergholm et al. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. 2022. arXiv: 1811.04968 [quant-ph]. URL: https://arxiv.org/abs/1811.04968.

[7] Sau Lan Wu e Shinjae Yoo. "Challenges and opportunities in quantum machine learning for high-energy physics". In: *Nature Reviews Physics* 4.3 (2022), pp. 143–144. ISSN: 2522-5820. DOI: 10.1038/s42254-022-00425-7. URL: https://doi.org/10.1038/s42254-022-00425-7.

[8] Simone Bordoni et al. "Long-Lived Particles Anomaly Detection with Parametrized Quantum Circuits". In: *Particles* 6.1 (2023), pp. 297–311. ISSN: 2571-712X. DOI: 10.3390/particles6010016. URL: https://www.mdpi.com/2571-712X/6/1/16.

[9] Arjun Dey et al. "Overview of the DESI Legacy Imaging Surveys". In: *The Astronomical Journal* 157.5 (apr. 2019), p. 168. ISSN: 1538-3881. DOI: 10.3847/1538-3881/ab089d. URL: http://dx.doi.org/10.3847/1538-3881/ab089d.

[10] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML]. URL: https://arxiv.org/abs/1406.2661.

[11] Diederik P Kingma e Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML]. URL: https://arxiv.org/abs/1312.6114.

[12] Pierre-Luc Dallaire-Demers e Nathan Killoran. "Quantum generative adversarial networks". In: *ArXiv* abs/1804.08641 (2018). URL: https://api.semanticscholar.org/CorpusID:13739672.

[13] Carlos Bravo-Prieto et al. "Style-based quantum generative adversarial networks for Monte Carlo events". In: *Quantum* 6 (ago. 2022), p. 777. ISSN: 2521-327X. DOI: 10.22331/q-2022-08-17-777. URL: https://doi.org/10.22331/q-2022-08-17-777.

[14] Amir Khoshaman et al. "Quantum variational autoencoder". In: *Quantum Science and Technology* 4.1 (set. 2018), p. 014001. DOI: 10.1088/2058-9565/aada1f. URL: https://dx.doi.org/10.1088/2058-9565/aada1f.

[15] Jascha Sohl-Dickstein et al. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. arXiv: 1503.03585 [cs.LG]. URL: https://arxiv.org/abs/1503.03585.

[16] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV]. URL: https://arxiv.org/abs/2112.10752.

[17] Aditya Ramesh et al. *Zero-Shot Text-to-Image Generation*. 2021. arXiv: 2102.12092 [cs.CV]. URL: https://arxiv.org/abs/2102.12092.

[18] Jonathan Ho, Ajay Jain e Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG]. URL: https://arxiv.org/abs/2006.11239.

[19] Antonio Tudisco. "Encoding Techniques for Quantum Machine Learning". Master's Thesis. Politecnico di Torino, dic. 2022. URL: https://webthesis.biblio.polito.it/25437/1/tesi.pdf.

[20] Adriano Barenco et al. "Elementary gates for quantum computation". In: *Physical Review A* 52.5 (nov. 1995), pp. 3457–3467. ISSN: 1094-1622. DOI: 10.1103/physreva.52.3457. URL: http://dx.doi.org/10.1103/PhysRevA.52.3457.

[21] Maria Schuld e Francesco Petruccione. *Machine Learning with Quantum Computers*. Springer, 2021.

[22] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (ago. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: http://dx.doi.org/10.22331/q-2018-08-06-79.

[23] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[24] Alex Nichol e Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672 [cs.LG]. URL: https://arxiv.org/abs/2102.09672.

[25] Maria Schuld et al. “Circuit-centric quantum classifiers”. In: *Physical Review A* 101.3 (mar. 2020). ISSN: 2469-9934. DOI: 10.1103/physreva.101.032308. URL: http://dx.doi.org/10.1103/PhysRevA.101.032308.

[26] Andrea Cacioppo et al. *Quantum Diffusion Models*. 2023. arXiv: 2311.15444 [quant-ph]. URL: https://arxiv.org/abs/2311.15444.

[27] Jarrod R. McClean et al. “Barren plateaus in quantum neural network training landscapes”. In: *Nature Communications* 9.1 (2018), p. 4812. ISSN: 2041-1723. DOI: 10.1038/s41467-018-07090-4. URL: https://doi.org/10.1038/s41467-018-07090-4.

[28] Malcolm Carroll et al. *Dynamics of superconducting qubit relaxation times*. 2022. arXiv: 2105.15201 [quant-ph]. URL: https://arxiv.org/abs/2105.15201.

Elenco delle figure