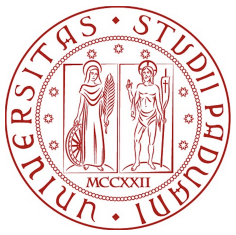

UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea in Fisica

Tesi di Laurea

Utilizzo di autoencoders variazionali per l'identificazione di parametri d'ordine in meccanica statistica

Relatore

Prof. **Samir Simon Suweis**

Correlatore

Dr. **Francesco Mambretti**

Laureando

Filippo Boni

Anno Accademico 2020/2021

Indice

1 Introduzione **2**

2 Introduzione ai Variational Autoencoders **4**
2.1 Premesse 4
2.2 I Variational Autoencoders(VAE) 6

3 Modello di Ising e l'Algoritmo di Clustering **11**
3.1 Introduzione al modello di Ising 11
3.2 Generazione Casuale di Configurazioni del Modello di Ising 13

4 Applicazione di un Variational Autoencoder a Configurazioni di un Modello di Ising 2D **17**
4.1 Generazione delle Configurazioni del Modello di Ising 2D e Creazione dei Dataset . . . 17
4.2 Il Variational Autoencoder utilizzato 18
4.3 I Risultati del Training 20
4.4 Magnetizzazione e First Latent Variable 23
4.5 Distribuzione della First Latent Variable 24
4.6 Generazione di Nuove Immagini 25
4.7 Conclusione 26

Capitolo 1

Introduzione

La fisica dei sistemi complessi tratta con sistemi governati da un elevato numero di gradi di libertà e quindi caratterizzati da variabili ad alta dimensionalità. In alcuni casi tali sistemi sono risolvibili analiticamente , ma spesso ciò non è possibile ed è necessario affidarsi a grandi quantità di dati per ottenere una descrizione del loro comportamento. Le moderne tecniche di machine learning possono infatti giocare un ruolo fondamentale nell'estrarre informazioni da questi dati e fornire un supporto per comprendere fenomeni estremamente complessi. Ad esempio una famosa applicazione di tecniche di machine learning alla fisica è il lavoro di Juan Carrasquilla e Roger G. Melko dal titolo "Unsupervised learning for local structure detection in colloidal systems" [1].
Un esempio di sistema in cui le unità interagiscono e possono dare luogo a comportamenti complessi è il modello di Ising [2], un modello fisico-matematico studiato in meccanica statistica sviluppato per descrivere il magnetismo nella materia. Il modello di Ising considera variabili discrete (con valori possibili pari a 1 o -1) chiamate spin. Gli spin interagiscono in coppie tramite interazioni che sono in genere limitate ai primi vicini. Un aspetto fondamentale che il modello di Ising riesce a catturare è la transizione dal ferromagnetismo a paramagnetismo quando la temperatura cresce al di sopra di una temperatura critica, detta di Curie. La temperatura viene in questo caso denominata parametro di controllo del sistema, mentre la magnetizzazione è il parametro d'ordine, in quanto descrive la simmetria (o il grado d'ordine) del sistema.

In generale non è semplice identificare quale siano i parametri di controllo e i parametri d'ordine di un sistema complesso. Recentemente alcune ricerche in ambito di machine learning [3, 4] hanno proprio come scopo quello di identificare le cosiddette variabili latenti del sistema, ossia un ridotto numero di quantità che racchiudono le caratteristiche fondamentali del sistema stesso e che quindi possono considerarsi come suoi parametri d'ordine. Un metodo per ottenere tale riduzione dimensionale è l'utilizzo dei Variational Autoencoders indicati con l'acronimo VAE [5]. Il VAE è un algoritmo di machine learning che ha come scopo quello di identificare la distribuzione di probabilità che meglio modella i dati disponibili, andando a definire delle variabili latenti che possano poi rappresentare le caratteristiche più significative di questi stessi dati, e da esse poterne generare di nuovi.

Seguendo un recente lavoro di S.J. Wetzel [4], il lavoro di questa tesi consiste nel costruire un variational autoencoder e cercare di identificare in un modello meccanico statistico come quello di Ising (in particolare il modello di Ising bidimensionale) una variabile latente che si comporti a tutti gli effetti come un parametro d'ordine del sistema. L'obiettivo è verificare se un algoritmo di machine learning come il VAE possa rappresentare un buon metodo per l'analisi e la descrizione di un sistema meccanico statistico come il modello di Ising.
Si cercherà quindi di appurare se esiste una correlazione col parametro d'ordine del sistema, che in questo caso è la magnetizzazione, allenando un VAE su un dataset costituito da configurazioni di Ising simulate a diverse temperature attraverso un algoritmo Monte Carlo noto come algoritmo di cluster [6].
La tesi sarà esposta nel seguente modo. Nel secondo capitolo si introdurranno dei concetti di base del

campo del machine learning per poi spiegare in maniera approfondita le basi teoriche e la struttura di un VAE. Nel terzo capitolo verranno trattati i concetti teorici fondamentali del modello di Ising e dell'algoritmo Monte Carlo utilizzato per generare il dataset. Nel quarto capitolo si mostreranno e analizzeranno i risultati dell'applicazione di un VAE ai dati creati, per poterne trarre delle conclusioni le quali saranno esposte alla fine del capitolo.

Capitolo 2

Introduzione ai Variational Autoencoders

2.1 Premesse[1]

Machine Learning
Il *Machine Learning* è lo studio e l'utilizzo di algoritmi che siano in grado di migliorare in modo autonomo le proprie prestazioni attraverso l'analisi dei dati. Gli algoritmi di machine learning possono essere di diversi tipi a seconda del tipo di scopo, detto *task*, per cui sono stati pensati. Se a questi algoritmi viene fornito un numero sufficiente di dati, attraverso processi di ottimizzazione, sono in grado di eseguire il task in modo sempre più efficiente, senza la necessità di un intervento esterno. Il processo di apprendimento consiste nel fatto che al crescere dei dati di input alla macchina, la performance del task, grazie all'ottimizzazione dell'algoritmo, cresce.

Deep Learning e Neural Networks
Il *Deep Learning* è un certo tipo di modelli di machine learning, che si basa sulle *Artificial Neural Networks* (ANN). Come ogni algoritmo di machine learning, un algoritmo di deep learning riceve in input un set di dati X, denominato training set, di cui impara iterativamente le caratteristiche ottimizzando il suo apprendimento per raggiungere l'obiettivo per cui è stato costruito, attraverso l'ottimizzazione di una certa funzione detta *loss function* di cui si tratterà nel prossimo paragrafo. L'efficacia dell'apprendimento, detto *training*, verrà poi verificato attraverso un secondo dataset dello stesso tipo di quello di training Y, detto test set, mai visto dall'algoritmo durante il training. Essendo tale set un insieme di dati su cui l'algoritmo non ha effettuato il training, questi gli sono completamente nuovi ed estranei. Se si osservano i risultati attesi mandando in input il test set, allora si può concludere di aver ottenuto un algoritmo in grado di produrre i risultati voluti su datasets diversi da quello di training. In questo caso si dice che l'algoritmo ha acquisito generalità.
Le Artificial Neural Networks, che da questo punto verranno indicate come ANN, sono algoritmi la cui struttura è ispirata alle reti neurali del cervello umano. Le ANN si basano su unità tra loro connesse chiamate nodi o *neuroni*. Tali neuroni sono organizzati in gruppi detti *layers* che possono essere rappresentati come strati adiacenti di neuroni artificiali [7]. I neuroni di un determinato layer sono collegati a tutti quelli del layer precedente e del layer successivo, ricevendo in input l'insieme degli output prodotti dai neuroni del layer precedente. I layers a loro volta sono classificati come input layer, hidden layers e output layer. L'input layer è costituito da neuroni che gestiscono l'input di tutta la ANN rappresentato da un dataset X. Gli hidden layers sono le serie di neuroni che eseguono il training. L'output layer produce il risultato finale del training. Nella figura 2.1 si riporta una semplice struttura di una ANN.

[1] i concetti presentati in questo paragrafo sono per la maggior parte ripresi dai capitoli della referenza [5]

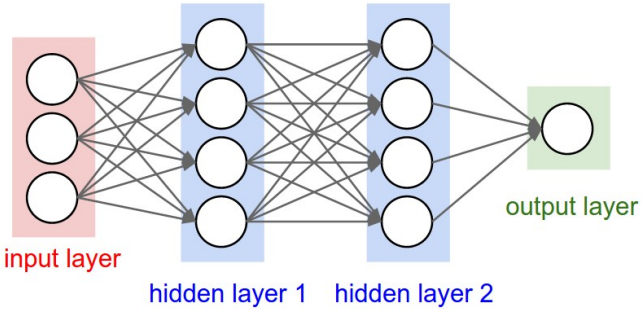

Figura 2.1: Schema di una struttura base di una Artificial Neural Network

Ottimizzazione, Loss Function e Stochastic Gradient Descent
Gli algoritmi di machine learning e deep learning prevedono un processo di ottimizzazione. Tale processo si basa sulla minimizzazione iterativa di una certa funzione $L(x;\theta)$ detta *loss function*, una funzione che si calcola a partire dai dati e che dipende dai parametri della rete θ. L'algoritmo consiste nella minimizzazione (o massimizzazione) di $L(x;\theta)$ (-$L(x;\theta)$) variandone il valore dei parametri θ. Tale loss function ha un'espressione diversa a seconda degli obiettivi dell'algoritmo.
La minimizzazione (o massimizzazione) della loss function si può raggiungere attraverso diversi metodi. Uno dei più noti è quello del "gradient descent" [2].

Un problema degli algoritmi del machine learning è che un training set grande (o comunque con elementi ad alta dimensionalità) è necessario per un apprendimento efficace, ma allo stesso tempo questo comporta costi computazionali elevati, in particolar modo nell'applicare la tecnica del gradient descent. Un possibile tentativo di risoluzione di questo problema consiste nell'implementazione di un'altra tecnica: lo *stochastic gradient descent*. Il punto fondamentale di tale tecnica è quello di considerare il gradiente della loss function come un valore di aspettazione e stimarla approssimativamente campionando un certo numero di punti del dataset di training, costruendo quello che viene chiamato minibatch, rappresentato da $B = [x_1, ..., x_m]$, con m un numero positivo minore della dimensione del set di training. Il vantaggio dello stochastic gradient descent è quindi quello di utilizzare lo stesso meccanismo del gradient descent senza però gli svantaggi computazionali annessi all'applicazione di tale tecnica ad un dataset dimensionalmente grande. Si può quindi raggiungere la minimizzazione della loss function calcolando il suo gradiente e quindi applicando la tecnica dello stochastic gradient descent senza elevati tempi computazionali.

Validation Set
Il test set non ha solo il ruolo di verificare se, completato il training, l'algoritmo ha raggiunto risultati corretti, ma può essere coinvolto anche durante il processo di apprendimento. In questo caso è detto *validation set*. Questo è utilizzato durante il training ed è formato da dati dello stesso tipo del training set e del test set. Come il training set, il validation set è sottoposto all'algoritmo ad ogni iterazione, ma, a differenza del training set, non fornisce le informazioni al fine di migliorare il suo apprendimento. Sostanzialmente l'algoritmo non apprende dal validation set e quindi ad ogni iterazione rappresenta un insieme di dati completamente nuovo all'algoritmo. Il validation set è utile per controllare la generalità dell'algoritmo al crescere delle iterazioni. Tale valutazione si effettua attraverso la *validation loss function*, cioè una loss function con la stessa espressione matematica della loss function calcolata dal training set, ma che è computata utilizzando i dati del validation set. Come per il primo tipo di loss function, anche questa deve essere ottimizzata durante il training. La sua continua decrescita è sintomo di una buona generalità dell'algoritmo e quindi dello svolgimento di un buon training.
Inoltre l'osservazione dell'andamento della validation loss function aiuta a prevenire il fenomeno di

[2] Se consideriamo la funzione $f(x)$, e indichiamo la sua derivata come $f'(x)$, allora considerando una piccola variazione ϵ del valore di x (con $\epsilon > 0$), l'effetto che produce su $f(x)$ è essere descritto da $f(x+\epsilon) \approx f(x) + \epsilon \cdot f'(x)$. La derivata è utile quindi per minimizzare $f(x)$ in quanto descrive come cambiare x per variare leggermente $f(x)$. Si ha infatti che $f(x - \epsilon \cdot sign(f'(x))) < f(x)$ per un ϵ abbastanza piccolo. Si può ridurre $f(x)$ variando x a piccoli passi con il segno opposto della derivata.

overfitting. L'overfitting è un problema che può caratterizzare gli algoritmi di machine learning e indica la perdita di generalità dell'algoritmo che esegue l'apprendimento. Può accadere infatti che l'algoritmo impari a riconoscere le caratteristiche attribuibili ai soli dati del training set, perdendo la capacità di poter essere utilizzato su altri set di dati. Può essere visto come un eccesso di apprendimento sul training set da parte dell'algoritmo. Se si riscontra una crescita continua della validation loss function e una contemporanea decrescita della normale *loss function* all'aumentare delle epoche dopo il raggiungimento di un punto di minimo, si può affermare che il modello è affetto da overfitting.

I Modelli Generativi
Nel campo del machine learning esistono tipi di modelli classificati come *Generative Models* che sono caratterizzati dalla capacità di generalizzare e generare dati simulati simili a quelli di input utilizzati per il training.
Si considera un dataset $\vec{X} = x_1, x_2, ..., x_N$ definito in un certo spazio multi-dimensionale $\mathbf{X}$, e caratterizzato da una distribuzione di probabilità $p(\vec{X})$ ignota. L'obiettivo dei modelli generativi è quello di trovare la distribuzione di probabilità che caratterizza meglio i dati del dataset $\vec{X}$, imparando le caratteristiche e le relative dipendenze tra variabili. Lo scopo ultimo di un Generative Model è quello di generare in output nuovi dati in cui si possano riscontrare la maggior parte delle caratteristiche, o almeno quelle più significative, dei dati del dataset di input $\vec{X}$ senza però produrne di identici, basandosi sulla distribuzione di probabilità $p(\vec{X})$ imparata dal modello.

Gli Autoencoders[3]
Per comprendere appieno il funzionamento di un Variational Autoencoder è necessario conoscere un'altra tipologia di neural network da cui esso deriva: *l'Autoencoder*. Un autoencoder è un tipo di modello generativo progettato per avere una rappresentazione basso dimensionale del dataset di partenza, e da essa generare un output che abbia le stesse caratteristiche dei dati di input. Un autoencoder infatti deve essere in grado di trovare le caratteristiche più significative degli elementi di un dataset tralasciando quelle marginali, per generare un output simile ai dati di input e in cui si possano riscontrare le proprietà degli elementi di input. Analizzando il risultato dell'autoencoder si riesce a capire quali sono tali proprietà e quindi ad ottenere informazioni sul set di partenza. Internamente è formato da uno o più hidden layers h che formano un codice utilizzato per rappresentare l'input. La rete può essere vista come composta da due parti: una *encoder function* $h = f(x)$ e una *decoder function* che produce una ricostruzione $r = g(h)$.
La struttura più semplice di un autoencoder è costituita da un input layer, un hidden layer ed un output layer. Tra l'input layer e l'hidden layer avviene l'encoding mentre tra l'hidden layer e l'output layer avviene il decoding. Questo tipo di struttura può essere raffinata aggiungendo più layers che agiscano come encoder o decoder.

2.2 I Variational Autoencoders(VAE)[4]

Un Variational Autoencoder, che da questo punto verrà indicato con l'acronimo VAE, è un modello generativo facente parte della famiglia degli algoritmi del deep learning che si basa sull'apprendimento di particolari variabili dette *variabili latenti*. Queste sono variabili che il VAE deve produrre e si ottengono campionandole da una distribuzione di probabilità $q(x)$ del dataset X che il VAE impara a costruire durante il training. Una volta campionate, sono utilizzate per costruire un'altra distribuzione di probabilità dipendente da esse $p(z)$ con cui si produce l'output attraverso un processo di campionamento. Un'importante caratteristica di tali variabili è che il loro numero è molto inferiore alla dimensione del dataset X.
La struttura del VAE riprende quella dell'autoencoder semplice, ma, al posto di imparare una funzio-

[3]la trattazione di questo paragrafo riprende la trattazione della referenza [5], capitolo 14
[4]La trattazione di questa sezione segue i passaggi della referenza numero [8] della bibliografia. Anche i passaggi matematici sono esposti seguendo il formalismo e l'esposizione della referenza [8].

ne per riprodurre l'input, il VAE impara a costruire delle distribuzioni di probabilità. Attraverso il campionamento dalla distribuzione $p(z)$ è possibile generare dei dati simili a quelli di input.

Le variabili latenti

In statistica, le variabili latenti sono variabili che non vengono osservate direttamente ma sono piuttosto dedotte attraverso un modello matematico da altre variabili che vengono osservate (misurate direttamente). Per poter dire che il modello è rappresentativo del set di dati, bisogna assicurasi che per ogni punto x nel set X, ci siano una o più vie per generare un output molto simile a x attraverso l'uso delle variabili latenti. Formalmente, si considera un vettore di variabili latenti z in uno spazio $\mathbf{Z}$ ad alta dimensione che si può facilmente campionare secondo una funzione di densità di probabilità $p(z)$ definita su Z. Quindi, si considera una famiglia di funzioni $f(z;\theta)$, parametrizzate da un vettore θ in un certo spazio Θ, tale che sia $f : \mathbf{Z} \times \Theta \to \mathbf{X}$. f è deterministica, ma se z è casuale ed è fisso, allora $f(z;\theta)$ è una variabile casuale nello spazio $\mathbf{X}$. L'obiettivo è ottimizzare θ in modo tale che sia possibile campionare z da $p(z)$ per ottenere con alta probabilità una $f(z;\theta)$ che sia simile alle x nel set di dati. Formalizzando tale concetto matematicamente, si cerca di massimizzare la probabilità di ottenere ogni punto x del training set X nel processo di generazione di nuovi dati. Data la dipendenza del campionamento di nuovi dati dalla densità di probabilità $p(z)$ e applicando la definizione di probabilità condizionata si ha che la probabilità da massimizzare di generare x del training set X è esprimibile come:

$$p(x) = \int p(x|z;\theta)p(z)dz \quad [5] \tag{2.1}$$

Generalmente si ipotizza che la $p(x|z;\theta)$ sia una gaussiana della forma $N(x|f(z;\theta), \sigma^2 \cdot I)$ avente quindi media pari a $f(z;\theta)$ e la matrice di covarianza come una matrice diagonale con elementi tutti uguali a σ^2, che rappresenta un parametro di tipo scalare da determinare. Tale aspetto si approfondirà più avanti nella trattazione dell'ottimizzazione del processo di training, ma si specifica che la scelta di una Gaussiana è solo quella più comune e che a seconda dei casi è possibile scegliere la distribuzione di probabilità che meglio si adatta al risultato atteso. Il fatto che si tenti di massimizzare la densità di probabilità $p(x)$ cercando di intuire la densità di probabilità $p(x|z;\theta)$ indica che nel generare nuovi dati non è possibile ottenere gli stessi dati di partenza x, ma il risultato sarà qualcosa di simile ad x. La massimizzazione di $p(x)$ si raggiunge cercando nel training di ridurre la distanza tra $f(z;\theta)$ e x.

La nuova loss function

Il concetto matematico espresso dalla formula (2.1) esprime efficacemente l'obiettivo di un VAE, ma presenta un ostacolo di non facile risoluzione: il calcolo dell'integrale. Risulta infatti difficile tentare di calcolare tale integrale, in quanto è caratterizzato da un'alta dimensionalità dovuta al numero di dati. É evidente che, anche se questo compito venisse svolto da un computer, i tempi per ottenere un risultato sarebbero molto dilatati e la potenza richiesta al computer usato elevata. Il problema si risolve definendo una loss function adatta agli obiettivi di un VAE. Ridurre il valore della loss function tentando di portarla a convergenza durante il training, produrrà l'effetto desiderato della massimizzazione di $p(x)$.

Il primo passo consiste nel notare che per la maggior parte degli elementi del vettore z, cioè per la maggior parte delle variabili latenti, la $p(x|z)$ è prossima allo zero e che quindi contribuiscono poco alla massimizzazione di $p(x)$. Si può quindi ignorarle e considerare solo alcune variabili latenti per calcolare $p(x)$. A questo punto è possibile definire una nuova distribuzione di probabilità $q(z|x)$ che considera il dataset X e restituisce una distribuzione di probabilità sui valori di z che hanno più probabilità di produrre in output x. Il vantaggio di queste considerazioni è che il numero di variabili latenti è sensibilmente minore e quindi lo spazio delle variabili latenti che sono considerate dalla distribuzione $q(z|x)$ ha una dimensione minore rispetto al numero di elementi totali del vettore delle

[5] dove $p(x)$ è la distribuzione di probabilità di generare x dal VAE, $p(x|z;\theta)$ è la distribuzione di probabilità di generare x dal VAE considerando le variabili latenti e $p(z)$ è la distribuzione delle variabili latenti

variabili latenti z. Questo permette di poter calcolare il valor medio $E_{z-q}[p(x|z)]$[6]. Per spiegare la relazione tra $E_{z-q}[p(x|z)]$ e la $p(x)$ che si vuole massimizzare è necessario introdurre il concetto della divergenza di Kullback-Leibler[7] (indicata con D) tra una certa distribuzione $p(z|x)$ ed un'arbitraria distribuzione di probabilità $q(z)$ che potrebbe non dipendere da x.

La definizione è la seguente:

$$D(q(z)||p(z|x)) = E_{z-q}[log(q(z)) - log(p(z|x))] \tag{2.2}$$

Da questa, applicando la regola di Bayes sulla probabilità condizionata, si ottiene:

$$D(q(z)||p(z|x)) = E_{z-q}[log(q(z)) - log(p(x|z)) - log(p(z)) + log(p(x))] \tag{2.3}$$

Dato che $p(x)$ è indipendente da z si può scrivere:

$$D(q(z)||p(z|x)) = E_{z-q}[log(q(z)) - log(p(x|z)) - log(p(z))] + log(p(x)) \tag{2.4}$$

Riconoscendo la definizione di divergenza di Kullback-Leibler e spostando i termini dell'equazione si ottiene:

$$log(p(x)) - D(q(z)||p(z|x)) = E_{z-q}[log(p(x|z))] - D(q(z)||p(z)) \tag{2.5}$$

Essendo la distribuzione $q(z)$ arbitraria si può scegliere una distribuzione dipendente da x definita come in precedenza $q(z|x)$.

Si può riscrivere quindi l'equazione precedente come:

$$log(p(x)) - D(q(z|x)||p(z|x)) = E_{z-q}[log(p(x|z))] - D(q(z|x)||p(z)) \tag{2.6}$$

La parte sinistra dell'equazione comprende un termine da massimizzare, la $p(x)$, e un termine da minimizzare, la $D(q(z|x)||p(z|x))$. Quest'ultimo termine è la distanza tra la distribuzione teorica non deducibile analiticamente $p(z|x)$, che descrive i valori di z che hanno più probabilità di generare dati simili ad x, e la distribuzione $q(z|x)$, che agisce come la $p(z|x)$, ma è la distribuzione effettivamente calcolata per l'ottimizzazione. L'obiettivo è diminuire attraverso il training la differenza tra di esse rendendola più piccola possibile in modo da massimizzare direttamente la $p(x)$.

Processo di Ottimizzazione
Si tratta ora del processo di ottimizzazione, nello specifico si analizza il termine a destra dell'equazione (2.6). Innanzitutto è necessario definire le due distribuzioni di probabilità $q(z|x)$ e $p(z)$. Normalmente si ipotizza che entrambe abbiano la forma di una gaussiana, rispettivamente descritte da $N(z|\mu_0(x;\theta), \Sigma_0(x;\theta))$ e $N(z|\mu_1(x;\theta), \Sigma_1(x;\theta))$ [8]. I parametri μ_0 e Σ_0 sono dedotti attraverso il training con Σ_0 forzata ad essere una matrice di covarianza diagonale. La $p(z)$ si ipotizza essere una gaussiana di media μ_1=0 e Σ_1=I. Definite le distribuzioni di probabilità è possibile calcolare la divergenza di Kullback-Leibler.

$$D(q(z|x)||p(z|x)) = D(N(z|\mu_0(x;\theta), \Sigma_0(x;\theta))||N(z|\mu_1(x;\theta), \Sigma_1(x;\theta))) \tag{2.7}$$

[6] con E_{z-q} si intende il valor medio calcolato campionando z da $q(z)$
[7] tale definizione è tratta dalla referenza [8]
[8] con μ e Σ si indicano rispettivamente la media e la varianza della distribuzione

Si ottiene che tale divergenza, se è calcolata tra due distribuzioni gaussiane, è pari a

$$D(N(z|\mu_0(x;\theta),\Sigma_0(x;\theta))||N(z|\mu_1(x;\theta),\Sigma_1(x;\theta))) = \frac{1}{2}(tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1-\mu_0)^T\Sigma_1^{-1}(\mu_1-\mu_0) + log\left(\frac{det(\Sigma_1)}{det(\Sigma_0)}\right) - k) \quad [9] \tag{2.8}$$

con k la dimensione delle distribuzioni. Nel caso trattato, con $p(z)$=$N(0,I)$, si ha:

$$D(N(z|\mu_0(x;\theta),\Sigma_0(x;\theta))||N(z|0,I)) = \frac{1}{2}(tr(\Sigma_0) + (\mu_0)^T(\mu_0) + log\left(\frac{det(I)}{det(\Sigma_0)}\right) - k) \tag{2.9}$$

è importante sottolineare che tutti i termini dell'equazione, tranne la dimensione k, sono dedotti dal dataset di training X attraverso l'apprendimento. A questo punto è possibile applicare la tecnica di ottimizzazione dello *stochastic gradient descent* a questo termine dell'equazione. Trattare il primo termine $E_{z-q}[log(p(x|z))]$ è più complicato in quanto per avere una buona stima del valor medio sarebbe necessario campionare un numero elevato di z ed è ovvio che sarebbe molto dispendioso dal punto di vista computazionale. Quindi si applica la seguente approssimazione:

$$E_{z-q}[log(p(x|z))] \approx log(p(x|z)) \tag{2.10}$$

Complessivamente si utilizza lo stochastic gradient descent sul termine:

$$log(p(x|z)) - D(N(z|\mu_0(x;\theta),\Sigma_0(x;\theta))||N(z|0,I)) \tag{2.11}$$

campionando dati dal dataset di training X.

Conclusioni sulla struttura di un VAE
Come per il semplice autoencoder , nel VAE si possono riconoscere un *input*, un *encoder*, un *decoder* e un *output*. Il training set X rappresenta l'input, che passa attraverso l'encoder dato dalla distribuzione di probabilità $q(z|x)$. A questo punto si può calcolare la divergenza $D(N(z|\mu_0(x;\theta),\Sigma_0(x;\theta))||N(z|0,I))$. Si campiona z dalla distribuzione $N(z|\mu_0(x;\theta),\Sigma_0(x;\theta))$ e si utilizza per computare $p(x|z)$ che costituisce il decoder. Si stima la funzione $f(z;\theta)$ producendo l'output e calcolando la distanza dai valori del dataset X. Tale processo si ripete ciclicamente fino alla convergenza della loss function sopra definita ed è schematizzato chiaramente nella figura 2.2.
Nell'equazione (2.6), su cui si basano le considerazioni appena esposte, è presente un problema. Il termine $E_{z-q}[log(p(x|z))]$ non dipende solamente dai parametri di $p(x|z)$, ma indirettamente anche dai parametri di $q(z|x)$. Tale dipendenza è però scomparsa nei termini dell'equazione. Per avere un buon apprendimento è necessario che l'encoder $q(z|x)$ produca dei risultati, basati sull'input x, che $p(x|z)$ possa decodificare in modo corretto. Questo problema è ovviato dal processo di ottimizzazione che si basa sulla tecnica dello *stochastic gradient descent*, che può gestire elementi stocastici in input, ma non all'interno della rete, come succede nel campionamento di z dopo il processo di encoding. Il metodo per aggirare questo limite è chiamato *reparameterization trick* il quale consiste nello spostare il campionamento in input. Date $\mu(x)$ e $\Sigma(x)$ si può campionare z dalla distribuzione $q(z|x)$ campionando un parametro ϵ da una gaussiana $N(0,I)$ per poi calcolare $z = \mu(x) + \Sigma(x)^{\frac{1}{2}} \cdot \epsilon$. La variazione nella struttura del VAE è esposta nelle figura 2.3.

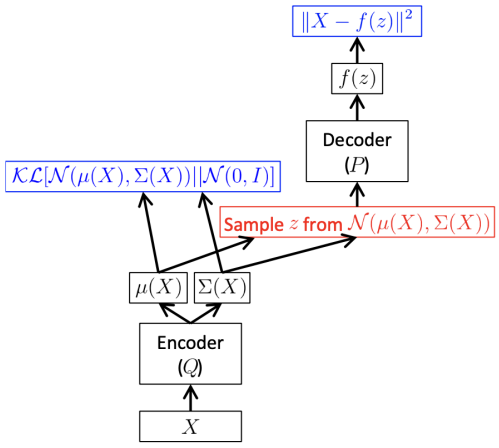

Figura 2.2: Schema della struttura di base di un Variational Autoencoder

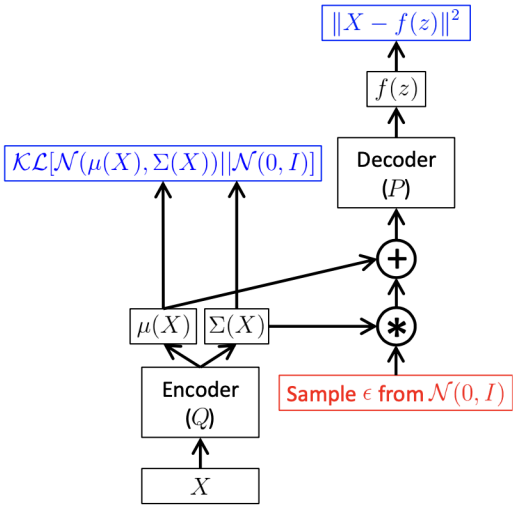

Figura 2.3: Schema della struttura di un Variational Autoencoder a cui è applicato il reparameterization trick

Capitolo 3

Modello di Ising e l'Algoritmo di Clustering

3.1 Introduzione al modello di Ising[1]

Il ferromagnetismo
Uno dei fenomeni più interessanti dello studio dei materiali nella fisica è il ferromagnetismo, cioè la proprietà di alcuni materiali (specialmente i metalli come ferro e nichel) di formare magneti permanenti (oggetti formati da un materiale che crea un proprio e persistente campo magnetico) o di essere attratti dai magneti. L'origine del ferromagnetismo e dei fenomeni di magnetismo in generale risiedono in una delle proprietà fondamentali degli elettroni che è possedere un momento di dipolo magnetico[2]. Un elettrone si comporta come un minuscolo magnete, producendo un campo magnetico. Questo momento di dipolo deriva dal fatto che l'elettrone sia dotato di spin quantomeccanico. Le due grandezze sono legate dalla seguente espressione:

$$\vec{\mu} = -g_s \mu_B \frac{\vec{S}}{\hbar} \tag{3.1}$$

Si ha in questa (3.1) che $\vec{\mu}$ è il momento magnetico, $\vec{S}$ lo spin dell'elettrone, μ_B è una costante detta momento magnetone di Bohr pari a $9.274 \times 10^{-24}\ \frac{J}{T}$ (con J si indica l'unità di misura Joule e T l'unità di misura Tesla), $\hbar$ la costante di Planck divisa per 2π e g_s una costante adimensionale pari a 2,002319. Per la sua natura quantistica, lo spin dell'elettrone può trovarsi in uno dei due stati possibili. L'elettrone può quindi avere spin "up" o "down". Quando questi dipoli magnetici in una zona del materiale considerato sono allineati, cioè puntano nella stessa direzione, i loro campi magnetici individualmente minuscoli si sommano per creare un campo macroscopico molto più grande dando vita ad un campo magnetico interno e al fenomeno del ferromagnetismo.

Il Modello di Ising
In alcuni metalli come il Ferro (Fe) e il Nichel (Ni) gli spin possono assumere uno stato fisso ("up" o "down") comune in modo spontaneo ottenendo quindi una zona in cui gli elettroni puntano tutti nella stessa direzione, creando quindi un campo elettromagnetico e permettendo di osservare il fenomeno del ferromagnetismo. Questo succede solo per temperature minori di una certa temperatura caratteristica detta temperatura critica o di Curie (T_c). Oltre tale temperatura gli spin assumono delle direzioni che complessivamente sono randomiche.

[1] La trattazione di questa sezione segue i passaggi delle referenze numero [2] e [9] della bibliografia. Anche i passaggi matematici sono esposti seguendo il formalismo e l'esposizione della referenza [2]

[2] il dipolo magnetico è un magnete ottenuto considerando una spira percorsa da corrente. Il momento di dipolo magnetico è una grandezza vettoriale definita come il prodotto tra l'area della spira e la corrente in essa, con direzione e verso del vettore normale all'area della spira e determinato dalla regola della mano destra

Si possono quindi riconoscere due fasi:

$$\begin{cases} \mathrm{T}<\mathrm{T}_c & fase\ ordinata \\ \mathrm{T}>\mathrm{T}_c & fase\ disordinata \end{cases}$$

Il *Modello di Ising* è un modello meccanico statistico che si pone l'obiettivo di modellizzare e simulare questo tipo di fenomeno. Nel modello di Ising si considera un sistema formato da un array di N punti chiamati siti reticolari che formano un reticolo periodico n-dimensionale con $n = 1, 2, 3$. La forma del reticolo può essere ad esempio quadrata nel caso bidimensionale oppure cubica in quello tridimensionale. Ad ogni sito è associata una variabile s_i (con $i = 1, ..., N$) che rappresenta lo spin. Normalmente per un elettrone gli stati di spin possono essere $+\frac{1}{2}$ oppure -$\frac{1}{2}$, ma per semplicità nel modello si associa un valore di s=+1 se lo spin è "up" e s=-1 se lo spin è "down". Una data serie $[s_i]$ di valori degli spin nei siti è detta *configurazione del sistema*. Nella figura 3.1 si rappresenta una configurazione di un reticolo bidimensionale quadrato.

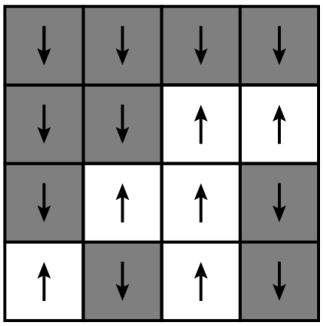

Figura 3.1: Esempio di una configurazione del modello di Ising bidimensionale quadrato

Come in molti altri modelli fisici, l'evoluzione temporale di questo sistema è regolata da una hamiltoniana, la quale può essere considerata come somma di due termini:

$$H = H_{ij} + H_i \tag{3.2}$$

in cui H_{ij} rappresenta il contributo dato dall'interazione tra gli spin, mentre H_i è il contributo all'hamiltoniana che si ottiene dall'interazione con un eventuale campo magnetico esterno.
L'espressione del primo termine è:

$$H_{ij} = -\sum_{\langle ij \rangle} J_{ij} s_i s_j \tag{3.3}$$

dove $\langle ij \rangle$ rappresenta che la somma deve essere effettuata considerando un sito con spin s_i e i suoi primi vicini s_j. Il numero dei primi vicini varia a seconda della struttura del reticolo. Ad esempio è facile intuire che se si ha una struttura bidimensionale quadrata il numero dei primi vicini corrisponde a 4, mentre nel caso di una struttura cubica semplice si ha un numero di siti primi vicini pari a 6. Il parametro J_{ij} è detto termine di coupling e regola l'intensità del termine H_{ij} dell'hamiltoniana. Tendenzialmente questo termine viene considerato costante su tutto il reticolo, ottenendo una nuova forma:

$$H_{ij} = -J \sum_{\langle ij \rangle} s_i s_j \tag{3.4}$$

L'energia di due spin vicini $h_{ij} = J s_i s_j$ è J se gli spin sono paralleli e $+J$ se sono antiparalleli. Quindi se $J > 0$ il modello favorisce gli spin paralleli. Si dice che l'interazione è ferromagnetica. A basse temperature si riscontrano configurazioni in cui la totalità o la maggior parte degli spin puntano nella

stessa direzione di una delle due disponibili, formando una $fase\ ferromagnetica$. Se J <0 si chiama interazione antiferromagnetica. Gli spin tendono ad allinearsi in una $fase\ antiferromagnetica$ a basse temperature. Ad alte temperature oltre la T_c, indipendentemente dal segno di J, ci si aspetta una fase disordinata. Gli spin fluttuano in una $fase\ paramagnetica$.
L'espressione del secondo termine è:

$$H_i = -h \sum_i s_i \tag{3.5}$$

Il parametro h regola l'intensità del termine H_i dell'hamiltoniana.
Esiste un'altra osservabile importante per la descrizione del sistema considerato: la magnetizzazione. Tale grandezza è definita come:

$$M_i = \sum_i s_i \tag{3.6}$$

La magnetizzazione è utile per capire la configurazione del sistema. Il suo valore varia infatti a seconda degli spin. Da essa è quindi possibile capire a quale temperatura si trova il sistema. Questo fatto è particolarmente rilevante in quanto la magnetizzazione segue un preciso andamento al variare della temperatura. Si ha che se la magnetizzazione è divisa per il numero N di siti allora si ottiene la magnetizzazione media.

3.2 Generazione Casuale di Configurazioni del Modello di Ising[3]

Il Metodo Monte Carlo e Breve Descrizione dell'Algoritmo di Metropolis
Il metodo Monte Carlo è un insieme di algoritmi che si basano sul campionamento casuale per calcolare risultati numerici. L'obiettivo è quello di ottenere risultati da simulazioni che hanno alla base la generazione di una serie di numeri casuali attraverso il campionamento da una specifica distribuzione di probabilità che si suppone descriva il fenomeno da analizzare. Un esempio di impiego del metodo Monte Carlo è utilizzarlo per trovare le soluzioni di problemi matematici a molte variabili e che non possono essere risolti facilmente, per esempio il calcolo integrale. Un tipo molto utilizzato di metodo Monte Carlo è $l'algoritmo\ di\ Metropolis$. Definita una distribuzione di probabilità $p(x)$, i passaggi dell'algoritmo per generare numeri casuali $[x_i]$ che seguano la distribuzione sono i seguenti:

1. generare casualmente un punto x_i
2. generare casualmente un altro punto x_{trial}
3. se $p(x_{trial}) > p(x_i)$ si aggiunge x_{i+1}=x_{trial} alla serie di numeri $[x_i]$, si pone $i = i + 1$ e si ritorna al punto 2
4. se non è soddisfatta la condizione al punto 3, si genera casualmente, utilizzando una distribuzione uniforme, un numero $r \in$[0,1]
5. se $\frac{p(x_{trial})}{p(x_i)} > r$ si aggiunge x_{i+1}=x_{trial} alla serie di numeri $[x_i]$, si pone $i = i + 1$ tornando successivamente al punto 2
6. se non è soddisfatta la condizione al punto 5 si aggiunge x_i alla serie di numeri $[x_i]$, si pone x_{i+1}=x_i e $i = i + 1$ e si torna al punto 2

L'algoritmo di Metropolis sarà implementato nel codice e utilizzato per la generazione di configurazioni del modello di Ising esposta nei prossimi paragrafi.

[3]La trattazione di questa sezione riprende quella della referenza numero [6] della bibliografia. Il formalismo esposto è quello utilizzato nella referenza numero [6]

Bilancio Dettagliato e Probabilità a Priori

L'approccio del Monte Carlo non costruisce uno stato ben definito del sistema, ad esempio minimizzandone l'energia, ma tenta di generare un numero di configurazioni rappresentative statisticamente indipendenti a, con probabilità $p(a)$. Nella fisica statistica classica dell'equilibrio, $p(a)$ è dato dalla distribuzione di Boltzmann, descritta da $p = Ae^{\frac{-H}{k_B T}}$ [4]. Per generare queste configurazioni in modo che rappresentino bene il sistema considerato (e a velocità ottimale), l'algoritmo Monte Carlo si sposta in un'iterazione dalla configurazione a alla configurazione b con probabilità $P(a \to b)$. Questa probabilità di transizione è scelta per soddisfare la condizione fondamentale detta di *bilancio dettagliato*:

$$p(a)P(a \to b) = p(b)P(b \to a)^5 \tag{3.7}$$

che è implementata usando l'algortimo di Metropolis in cui:

$$P(a \to b) = min\left(1, \frac{p(a)}{p(b)}\right) \tag{3.8}$$

Un modo molto utilizzato per passare dalla configurazione a alla configurazione b consiste nel cambiare lo spin di un singolo sito passando dallo stato "up" allo stato "down" o viceversa. Nel caso del modello di Ising si utilizza la distribuzione di probabilità di Boltzmann considerando come energia:

$$H_{ij} = -J\sum_{\langle ij \rangle} s_i s_j - h\sum_i s_i \tag{3.9}$$

Un concetto molto importante nel cambio di configurazione e quindi nell'algoritmo del metodo Monte Carlo è il fatto che la probabilità $P(a \to b)$ sia composta a sua volta da due probabilità: la probabilità di considerare il passaggio dalla configurazione a alla configurazione b, simboleggiata da $A(a \to b)$ e la probabilità di accettazione del passaggio da a a b, indicata da $B(a \to b)$. Si ottiene quindi:

$$P(a \to b) = A(a \to b)B(a \to b) \tag{3.10}$$

Date queste definizioni si può riscrivere la legge di bilancio dettagliato:

$$\frac{B(a \to b)}{B(b \to a)} = \frac{p(a)}{A(a \to b)}\frac{A(b \to a)}{p(b)} \tag{3.11}$$

e, se implementato dall'algoritmo di Metropolis, dall'equazione (3.8) si ottiene:

$$B(a \to b) = min\left(1, \frac{p(a)}{A(a \to b)}\frac{A(b \to a)}{p(b)}\right) \tag{3.12}$$

Si ha che la probabilità $A(a \to b)$ deve soddisfare alcune caratteristiche:

1. Deve condurre lo stato del sistema da una configurazione a ad una configurazione b, in modo

[4] dove A è una costante, H è l'energia del sistema, k_B è la costante di Boltzmann e T è la temperatura del sistema

[5] si indica con $P(a \to b)$ la probabilità di passare dalla configurazione a a quella b. $P(b \to a)$ rappresenta il passaggio inverso

tale che, eventualmente, si possano raggiungere tutte le possibili configurazioni

2. Deve essere possibile calcolare $\frac{p(a)}{p(b)}$. Nel caso del modello di Ising questa condizione è soddisfatta in quanto le distribuzioni di probabilità sono date dalle distribuzioni di probabilità di Boltzmann, le quali dipendono dall'energia del sistema considerato.

3. Si può calcolare $A(a \to b)$ e $A(b \to a)$ per ogni possibile transizione $a \to b$

L'Algoritmo di Clustering per il Modello di Ising
Passare tra le varie configurazioni variando lo spin di un singolo sito è un processo lungo e dispendioso computazionalmente. Inoltre, data la piccola variazione tra la configurazione iniziale e la configurazione finale, si otterrebbe che configurazioni successive sono correlate tra loro. Un modo per ovviare al problema è quella di implementare un algoritmo detto di *clustering*. L'idea consiste nel variare da una configurazione all'altra gli spin di più siti. Questo può essere fatto iniziando la costruzione di un cluster con uno spin campionato casualmente e aggiungendo in modo iterativo siti vicini della stessa direzione con una probabilità p. Uno spin per essere aggiunto al cluster deve soddisfare la seguente condizione: se il sito i è nel cluster e un sito vicino j non lo è, e se $s_i = s_j$, allora si dovrebbe aggiungere il sito j con probabilità p. Un esempio del passaggio da una configurazione a ad una configurazione b utilizzando la tecnica del clustering è mostrata nella figura 3.2.

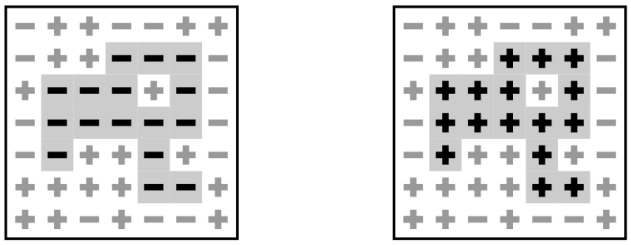

Figura 3.2: Rappresentazione dell'applicazione dell'algoritmo di Clustering ad una configurazione del modello di Ising

Definiamo come *link* una coppia di siti con spin dello stesso tipo, ma con un sito all'interno del cluster e con un sito all'esterno. Si può notare che nella configurazione a (l'immagine a sinistra della figura 3.2) la formazione del cluster si è stoppata lasciando 9 links del tipo $--$. Quindi si ha che 9 siti potevano essere inclusi nel cluster con probabilità p, ma non sono stati accettati. Questo genera un termine $(1-p)^9$ nella probabilità $A(a \to b)$. Cambiando lo spin dei siti nel cluster si otterrà la configurazione b (l'immagine a destra della figura 3.2). Se si volesse tornare dalla configurazione b alla configurazione a si dovrebbe considerare lo stesso cluster e cambiare lo spin dei siti nel cluster. Come nel passaggio da a a b sono presenti dei links del tipo $++$. In questo caso si ha che 19 siti potevano essere accettati nel cluster ma sono stati rigettati, aggiungendo un termine $(1-p)^{19}$ alla probabilità $A(b \to a)$. Si possono quindi calcolare le probabilità:

$$A(a \to b) = A_{interior}(1-p)^9 \tag{3.13}$$

$$A(b \to a) = A_{interior}(1-p)^{19} \tag{3.14}$$

Con *interior* si indica la parte del cluster che non tocca il bordo. Si possono anche scrivere le espressioni delle energie delle due configurazioni (supponendo di non avere un'interazione con un campo magnetico esterno):

$$E_a = E_{interior} + E_{exterior} - 9J + 19J \tag{3.15}$$

$$E_b = E_{interior} + E_{exterior} + 9J - 19J \tag{3.16}$$

e di conseguenza delle distribuzioni di Maxwell associate alle due configurazioni:

$$p_a \propto e^{-\beta E_a} \tag{3.17}$$

$$p_b \propto e^{-\beta E_b} \tag{3.18}$$

Si ha per costruzione che le due $A_{interior}$ e le energie $E_{interior} + E_{exterior}$ sono uguali per due configurazioni connesse da un singolo cambio di cluster. A questo punto si può scrivere la probabilità di accettazione (imponendo J=1):

$$B(a \to b) = min\left(1, \frac{e^{9\beta}\ e^{-19\beta}}{(1-p)^9} \quad \frac{(1-p)^{19}}{e^{19\beta}\ e^{-9\beta}}\right) \tag{3.19}$$

e quindi:

$$B(a \to b) = min\left(1, \left[\frac{e^{-2\beta}}{(1-p)}\right]^9 \left[\frac{(1-p)}{e^{-2\beta}}\right]^{19}\right) \tag{3.20}$$

Un caso particolare riguarda il caso in cui p sia $p = 1 - e^{-2J\beta}$ in cui si ha che $B(a \to b)$ è sempre 1 per ogni possibile mossa.

Capitolo 4

Applicazione di un Variational Autoencoder a Configurazioni di un Modello di Ising 2D

4.1 Generazione delle Configurazioni del Modello di Ising 2D e Creazione dei Dataset

Si tratta ora di come sono state generate le configurazioni del modello di Ising, di come sono state trasformate per renderle adatte ad un processo di training di un VAE e di come poi sono stati costruiti i dataset.
Sono necessarie però alcune premesse. Sono state studiate delle configurazioni di un modello di Ising bi-dimensionale con reticolo quadrato con area 28×28, intendendo che su ogni lato sono presenti 28 siti con i relativi spin, per un totale di 784 siti. Le temperature di cui si sono considerate le configurazioni sono presenti nella serie: $[0.6, 0.8, 1.0, 1.2, 1.4, 1.5, 1.6, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 3.0, 3.1, 3.2, 3.3, 3.5, 3.6, 3.8]$ ed ognuna di esse rappresenta un dataset.
É importante sottolineare come in generale sia gli Autoencoders che i Variational Autoencoders sono stati costruiti per imparare da dataset formati da immagini. Per capirlo basta pensare che il MNIST, un dataset pubblico tra i più utilizzati al mondo per studiare il comportamento di autoencoders e VAE e per verificare il loro funzionamento, è costituito da immagini 28×28 in scala di grigi raffiguranti cifre da 0 a 9 scritte a mano. Inoltre utilizzare le immagini è stato necessario per poterle caricare nel codice del VAE.

L'Algoritmo di Clustering e la Generazione di Immagini
L'algoritmo di Clustering è stato applicato utilizzando il linguaggio di programmazione Python e alcune sue librerie tra cui Numpy [10] . Si inizia definendo: la dimensione del reticolo, le temperature di cui si vuole calcolare la magnetizzazione e il numero di spin totali di cui variare lo stato prima che il processo si concluda. Si crea quindi un array di 784 spin con valori che possono essere +1 o -1 scelti casualmente grazie ad una funzione della libreria Random. Si implementa un ciclo *while* in cui ad ogni iterazione corrisponde alla creazione di un cluster. Le iterazioni sono dette *step* ed ad ogni step è calcolata la magnetizzazione media della configurazione. Ogni 50 steps l'array degli spin corrispondente alla configurazione generata viene trasformato in una matrice 28×28. Da questa matrice, attraverso la libreria PIL [11] di Python si genera un'immagine su cui viene applicato un *resizing* creando un'immagine di pixel 28×28, in cui ogni pixel corrisponde ad un sito, e trasformata in bianco e nero. In particolare se il pixel è bianco lo spin del sito ha valore 1 mentre se è nero ha valore -1. Bisogna precisare che nella generazione delle immagini è stato necessario convertire tutti i termini dell'array di spin uguali a -1 in termini uguali a 0. Questo per ottenere delle immagini bianche e nere. Tale fatto non ha inficiato però i risultati che si sono ottenuti, ha solamente riparametrizzato

il valore degli spin e delle magnetizzazioni nel processo di training del VAE. Nei risultati finali si sono ristabiliti i valori iniziali di -1 e 1. Si riportano degli esempi di configurazioni generate nella figura 4.1.

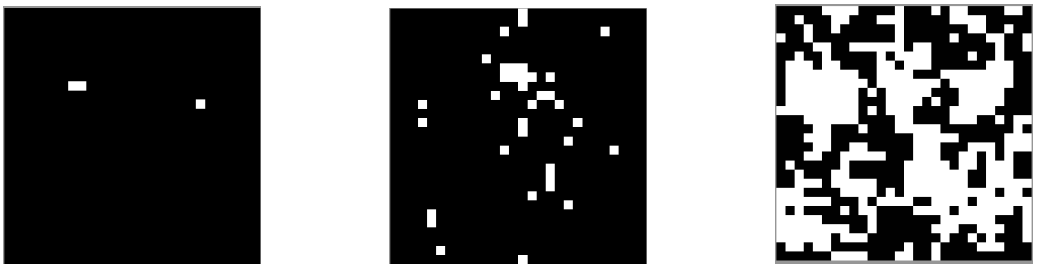

Figura 4.1: Esempi di configurazioni del modello di Ising generate con l'algoritmo di Clustering. Le prime due immagini fanno riferimento a configurazioni ordinate, la terza a configurazioni disordinate.

Struttura dei Datasets
I datasets, come specificato in precedenza, sono formati da immagini bianche e nere di dimensioni in pixel 28×28. Queste sono state pescate durante l'esecuzione del ciclo *while* che ha generato le configurazioni, in particolare ogni 50 steps. La scelta di considerare le configurazioni ogni 50 steps si basa su due motivi tra loro legati. Il primo è cercare di estrarre configurazioni correlate il meno possibile tra loro e quindi distanziate più del tipico tempo di correlazione in termini di steps. Il secondo motivo è collegato ai costi computazionali e temporali di generare dei datasets abbastanza grandi, infatti generare i datasets utilizzati ha richiesto tempi abbastanza dilatati per gli scopi che si volevano raggiungere. Quindi 50 steps è stato identificato come un giusto compromesso tra la non correlazione delle immagini e l'ottimizzazione dei costi computazionali della loro generazione.
Ovviamente si è scelta anche la dimensione per ogni dataset. Questa corrisponde a 1100 immagini. Nel codice di generazione era presente un comando che stoppasse l'esecuzione del codice nel momento in cui la dimensione del dataset avesse raggiunto quella predefinita. Anche qui la dimensione scelta è frutto di un compromesso. Da un lato avere un dataset molto grande è molto utile ai fini dell'apprendimento, ma avrebbe richiesto una grande quantità di tempo, dall'altro ridurre esageratamente la dimensione del dataset a favore di una maggiore velocità avrebbe reso il training inefficace. In conclusione la quantità di immagini scelta per ogni dataset è stata ritenuta la più corretta, anche perché con essa si sono raggiunti gli obiettivi prefissati.
Si analizza ora la scelta delle temperature che corrispondono ai dataset. Dall'elenco di temperature mostrato nel primo paragrafo si può notare innanzitutto che non si parte da temperature molto vicine a 0. Fino a circa $T = 1.0$ le configurazioni generate in un dataset sono quasi tutte uguali per la tendenza degli spin ad allinearsi a basse temperature. Risulta quindi molto più facile per il VAE trovare delle regolarità nel dataset. Tale comportamento diventa meno evidente avvicinandosi alla temperatura critica, in quanto le configurazioni diventano più disordinate. Quindi campionare frequentemente dati corrispondenti a temperature tra $T = 0$ e $T = 1$ non è sensato, in quanto si otterrebbero dei datasets che descrivono lo stesso fenomeno. Si è inoltre scelto invece di aumentare il numero dei datasets per $T \in [1.8, 2.5]$ per accrescere le informazioni sul comportamento del sistema intorno alla temperatura critica T_c, che, nel caso di un modello di Ising 2D è stata dedotta analiticamente: $T_c = \frac{2}{ln(1+\sqrt{2})} \approx 2.269$ [12]. Infine l'alto numero di datasets per temperature elevate è da attribuirsi al fatto che a queste temperature corrispondono delle configurazioni disordinate, con poche regolarità e pattern. Per questo si tenta di incrementare l'informazione formando una buona quantità di datasets per temperature elevate.

4.2 Il Variational Autoencoder utilizzato

In questa sezione viene esposta la struttura del VAE utilizzato. Innanzitutto si dichiara che tale struttura è stata ripresa da un tutorial presente sul sito dell'high-level package Keras[1]. Tale tutorial

[1] tale tutorial si trova alla referenza numero [13]

utilizza le immagini del dataset del MNIST. Anche per questo motivo si è scelto di basarsi sulle immagini per eseguire il training.
Prima di trattare della struttura effettiva del VAE è necessario esplicitare la densità di probabilità che ci si aspetta rappresenti i dati e di cui bisogna stimare i parametri cioè le variabili latenti. Come in molti altri algoritmi, anche in questo caso si è scelta come distribuzione di probabilità una gaussiana. Di conseguenza le variabili latenti sono la media e la varianza della distribuzione, per essere precisi si usa il logaritmo naturale di tale grandezza. Questi parametri sono stati indicati con z_{mean} e $z_{log\ sigma}$. Come trattato in precedenza z_{mean} e $z_{log\ sigma}$ permettono di definire un altra variabile, indicata con z, data dalla formula:

$$z = \mu(x) + \Sigma(x)^{\frac{1}{2}} \cdot \epsilon. \tag{4.1}$$

che, riscritta secondo la definizione dei parametri z_{mean} e $z_{log\ sigma}$, è esprimibile come:

$$z = z_{mean} + e^{0.5 \cdot z_{log\ sigma}} \cdot \epsilon. \tag{4.2}$$

Innanzitutto si sono trasformate le immagini in array di dimensione 784. Ogni dataset è stato diviso in training set e validation set, in particolare questo è costituito dal 10% del dataset.
É stato poi costruito un encoder, formato da: un hidden layer h che riducesse la dimensione delle immagini da 784 a 256, un layer che stimasse le variabili latenti z_{mean} e $z_{log\ sigma}$ ricevesse in input il risultato di h e un ultimo layer per ottenere la variabile latente z. L'obiettivo dell'encoder è creare un set di array che comprendano le variabili z_{mean}, $z_{log\ sigma}$ e z. Il decoder è invece strutturato nel seguente modo: un layer che ricevesse le variabili latenti, ricostruendo un output di dimensione 256, e un layer di output che producesse l'immagine (sotto forma di array) dedotta dalla distribuzione di probabilità imparata dal VAE. É stata poi definita un'appropriata loss function, seguendo la teoria esposta nelle prime pagine. L'insieme di questi elementi è andato a costituire il VAE usato, il quale è schematizzato nella figura 4.2.

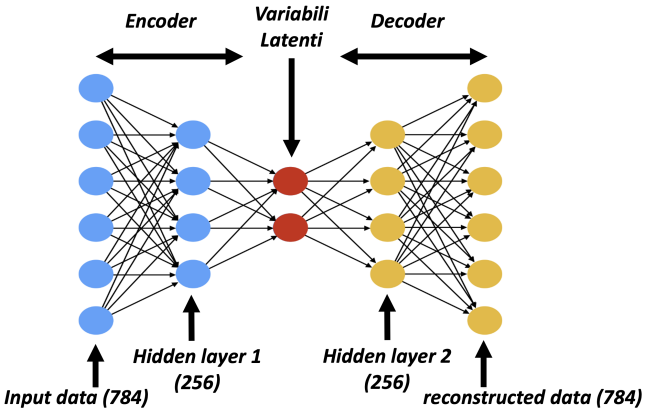

Figura 4.2: Schema rappresentativo del VAE utilizzato

Il training di quest'ultimo si basa sui seguenti parametri: le *epoche* e la *batch size*. Le epoche sono il numero di volte in cui i dati di input sono inseriti nel VAE per produrre un output. Ovviamente ad ogni epoca l'efficacia del VAE aumenta in quanto riesce a raccogliere più informazioni dai dati di input, effettuando il training. Sostanzialmente le epoche possono essere identificate come il numero di iterazioni che compongono il processo di training. Ad ogni epoca i dati di input vengono divisi in insiemi detti *batch* i quali vengono poi inseriti in modo consequenziale nel VAE. La batch size è la dimensione dei batch.

Modifiche del codice dal tutorial
É importante segnalare che rispetto al VAE presentato nel tutorial della referenza [13] si sono apportate

alcune modifiche. La prima è quella di aver posto una varianza Σ=1 nella distribuzione gaussiana utilizzata per campionare la variabile ϵ nel reparameterization trick. É facile verificare infatti che nel tutorial invece viene usata una deviazione standard σ=0.1. Concettualmente non è un cambio drastico perché è mantenuta l'idea alla base del reparameterization trick, ma si è scelto di cambiarla per rimanere fedeli a quanto esposto nelle prime pagine. La seconda modifica è stata apportata perché si è notata la presenza di un errore nel codice del tutorial. Analizzando il codice si può vedere che la variabile latente $z_{log\,sigma}$ è utilizzata in due formule diverse nel codice, ma legate tra di loro nel training, come varianza della distribuzione gaussiana in una delle due formule e come deviazione standard nell'altra formula. Questo ovviamente porta ad una mancanza di coerenza nella definizione della variabile. Per ovviare al problema è stato necessario moltiplicare per un fattore di 0.5 la variabile $z_{log\,sigma}$ come fatto nella formula 4.2.

4.3 I Risultati del Training

Il training è stato effettuato gestendo i datasets in due modi differenti. Il primo si basa sul considerare separati i datasets, completando per ognuno di essi un training, per poi unire i risultati ottenuti. Nel secondo invece sono stati uniti i datasets prima dell'inizio del training formando un dataset molto più grande dei singoli, contenente 27500 immagini.
Per entrambi i metodi il limite di epoche è pari a 100. La batch size per il primo metodo è 495 e per il secondo 100.

Minimizzazione delle Loss Function
Esistono svariati modi per verificare la bontà del training di un modello durante il processo di apprendimento. Il più basilare e fondamentale è analizzare l'andamento della loss function al crescere delle epoche. Come spiegato in precedenza, l'obiettivo del training è minimizzarla per ottenere una convergenza ad un certo valore della loss function utilizzata. In particolare però esistono due tipi di loss functions, le quali sono uguali per espressione matematica, ma differiscono per il dataset a cui si riferiscono. In questo caso l'espressione della formula matematica della loss function utilizzata nel training deriva dalla formula 2.11.
Nei training effettuati per i dataset separati si sono raggiunte nella maggior parte dei casi delle convergenze nelle rispettive loss functions, caratterizzate però da oscillazioni che sono più o meno forti a seconda del dataset. Si può quindi asserire che i training sono risultati abbastanza efficaci, ma non possono essere considerati completamente affidabili. La loss function del secondo metodo presenta una marcata convergenza e quindi il training si è concluso in modo positivo. Si riportano i grafici delle loss functions di alcuni datasets a titolo esemplificativo nella figura 4.3. Le quattro immagini mostrate rappresentano alcuni risultati del primo metodo. Il grafico mostrato in figura 4.4 è invece l'andamento delle due loss functions nel secondo metodo. Sebbene nel primo metodo la maggioranza dei datasets presenti delle loss functions buone, in alcuni datasets ci sono delle criticità evidenti nell'apprendimento. In particolare nei datasets corrispondenti alle temperature T=1.6 e T=2.0 malgrado ci sia una decrescita delle loss functions le oscillazioni di queste sono troppo forti per essere considerate buone. Nel dataset corrispondente alla temperatura T=1.0 invece non è neanche presente la diminuzione della loss function. Vengono comunque inclusi tutti i datasets nei risultati finali per completezza. Si riporta il grafico del dataset a T=1.6 nella figura 4.5.

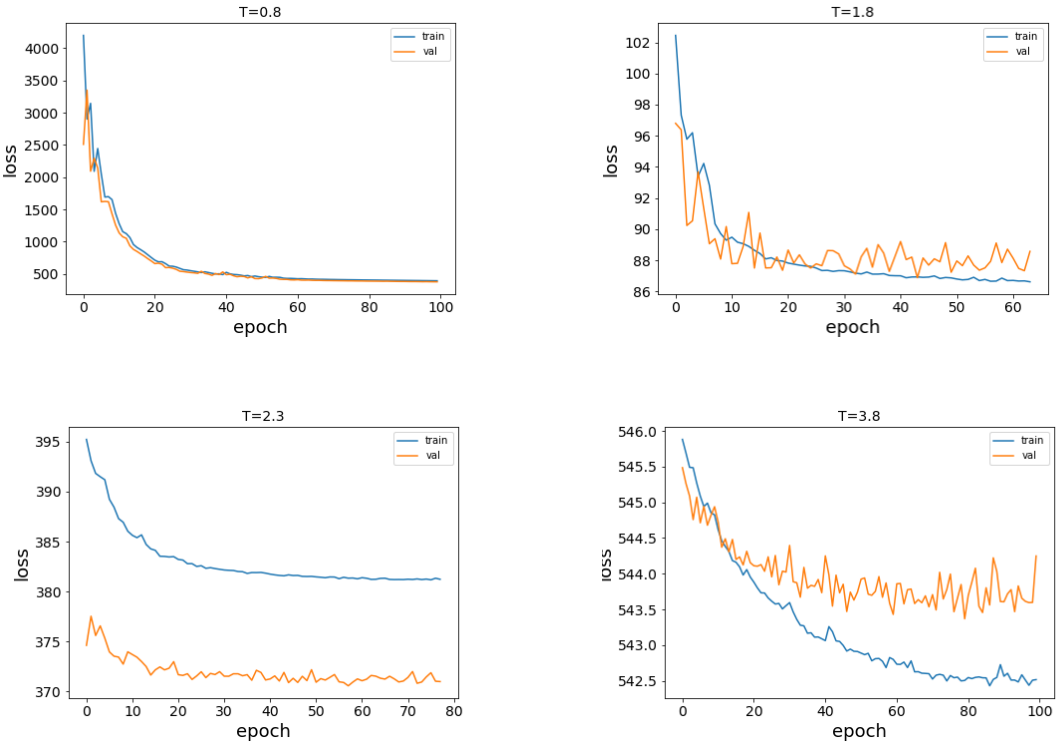

Figura 4.3: Grafici delle loss functions col metodo dei datasets separati corrispondenti alle temperature T=0.8, 1.8, 2.3, 3.8. Il numero di epoche nei training è diverso perché è stato implementato un comando che stoppasse il training prima di incorrere in un rischio di overfitting.

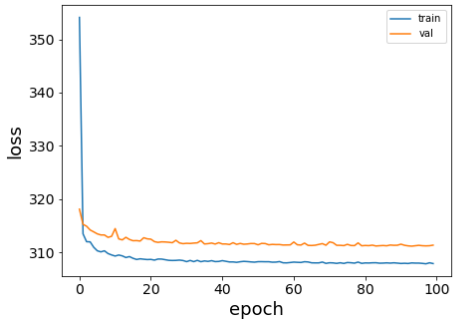

Figura 4.4: Grafico delle loss functions del metodo a datasets uniti. Il grafico dell'andamento delle loss functions in questo caso è unico in quanto è unico il dataset e quindi unico il training

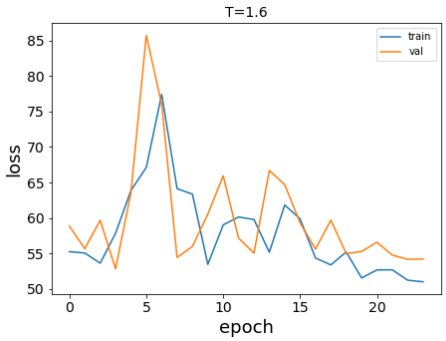

Figura 4.5: Grafico delle loss functions del dataset corrispondente alla temperatura T=1.6

Analisi di Magnetizzazione e First Latent Variable per ogni Dataset
Si introduce ora uno dei risultati che si volevano raggiungere in questo lavoro: capire il legame tra la prima variabile latente del VAE(la media della gaussiana) e la magnetizzazione delle configurazioni. In questo paragrafo si mostra il prodotto del training effettuato singolarmente sui datasets separati, in quanto il comportamento complessivo verrà confrontato con quello del secondo metodo nella prossima sezione. Per ogni dataset si è considerato il test set di 110 immagini. Per ogni immagine è stata calcolata la magnetizzazione media. A training concluso, per ogni immagine del test set è stato stampato il valore della prima variabile latente, attraverso l'encoder del VAE. Infine si sono plottati i risultati dei due processi. Si mostrano i grafici di quattro temperature diverse per dare un'idea dei risultati ottenuti nella figura 4.6.

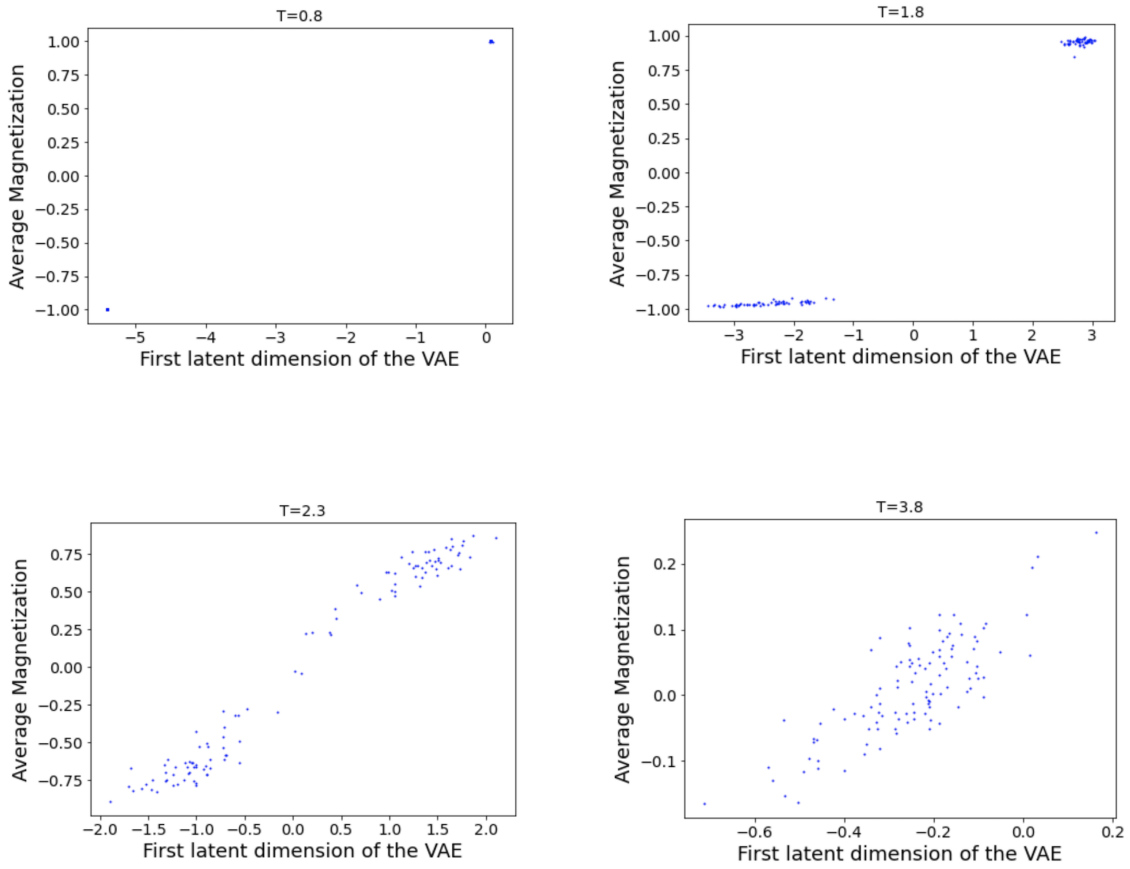

Figura 4.6: Grafici che mettono in relazione la prima variabile latente con la magnetizzazione media singolarmente per ogni dataset, ottenuti col metodo a dataset separati

Già da questi grafici si può intuire quello che sarà il comportamento complessivo al variare della temperatura. Innanzitutto si nota come a temperature basse il valore della magnetizzazione media è polarizzato verso due valori possibili: -1 e 1. Questo è dovuto al fatto che nella fase ordinata gli spin di una data configurazione tendono ad allinearsi nella stessa direzione e ad avere quindi lo stesso valore. Ad alte temperature invece il valore della magnetizzazione media si trova in un intorno di 0 dato che il numero di spin aventi valori rispettivamente -1 e 1 è quasi uguale essendo in configurazioni disordinate. Inoltre è evidente come i punti dei grafici all'aumentare della temperatura si dispongano in maniera meno ordinata rispetto a quelli di più basse temperature. In ogni caso è importante ricordare che questi sono prodotti del primo metodo, in cui si è spiegato che i training sono risultati buoni, ma non totalmente affidabili. Inoltre bisogna notare una disposizione dei punti nel grafico corrispondente alla T=2.3 che ha una forma lineare. Questo è un'anticipazione della relazione tra la prima variabile latente e la magnetizzazione media.

4.4 Magnetizzazione e First Latent Variable

In questa sezione viene mostrata la relazione tra magnetizzazione media e la prima variabile latente. Verranno esposti gli esiti dei due metodi utilizzati.

Datasets separati

Il risultato del primo metodo è mostrato nel grafico della figura 4.7:

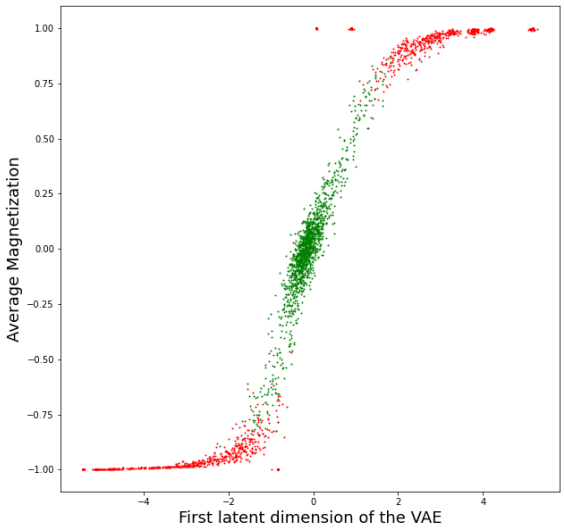

Figura 4.7: Andamento della magnetizzazione media rispetto alla prima variabile latente ottenuto col metodo dei datasets separati. Si precisa che i punti verdi si riferiscono a dati di temperature oltre la temperatura critica, mentre quelli rossi a dati di temperature inferiori a quella critica.

Datasets uniti

Il risultato del secondo metodo è mostrato nel grafico della figura 4.8:

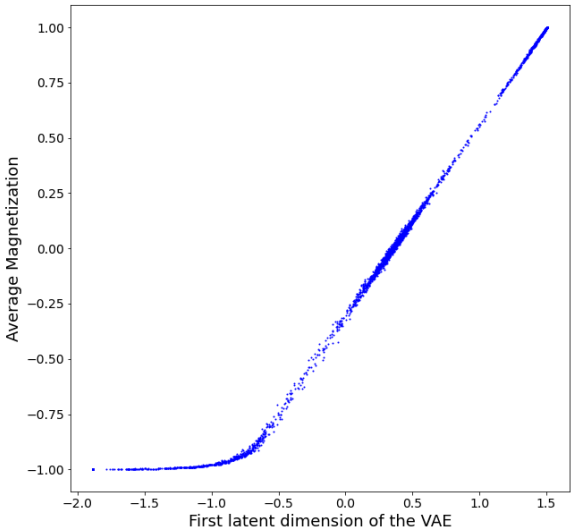

Figura 4.8: Andamento della magnetizzazione media rispetto alla prima variabile latente ottenuto col metodo dei datasets uniti.

Dal grafico del primo metodo in figura 4.7 si intravede un andamento lineare dei punti, soprattutto se si escludono i punti provenienti da basse temperature. Non è però ben definito.
Nel plot della figura 4.8 corrispondente al secondo metodo la linearità della disposizione dei punti è molto più marcata, specialmente se si ignora l'andamento lineare orizzontale che si nota all'inizio del grafico, attribuibile alle configurazioni a basse temperature. In questo caso quindi si può ipotizzare una correlazione tra magnetizzazione media e prima variabile latente.
Ci si potrebbe chiedere quindi perché ci si sia concentrati anche sul primo metodo. Innanzitutto per avere una conoscenza più approfondita sul funzionamento del VAE è utile confrontare più metodi. La seconda ragione, più importante, è che col secondo metodo non si potrebbe raggiungere l'obiettivo ultimo di un VAE: generare immagini che risultino simili a quelle di input.

4.5 Distribuzione della First Latent Variable

Si presenta ora, attraverso un istogramma, la distribuzione della prima variabile latente ottenuta dal secondo metodo. L'istogramma è quindi mostrato nella figura 4.9:

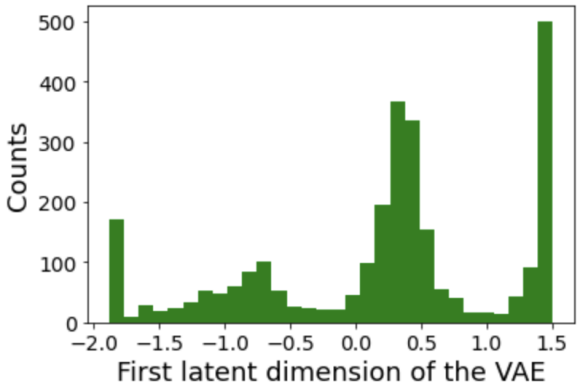

Figura 4.9: Istogramma della distribuzione dei valori della prima variabile latente

In questo istogramma è facile notare che ci sono tre picchi. Due agli estremi dell'istogramma e uno al centro. Per spiegare questo andamento è necessario fare riferimento alla prima figura della pagina precedente, che mostra la relazione tra variabile latente e magnetizzazione media. Malgrado tutte le precisazioni riguardanti il training del primo metodo, questo mette in luce come i dati provenienti da configurazioni ordinate rappresentino la maggioranza agli estremi del grafico, mentre quelli che rappresentano configurazioni disordinate popolino maggiormente la zona centrale del grafico. Si può concludere che i valori estremi della variabile latente provengano dai datasets corrispondenti a temperature al di sotto di quella critica, mentre i valori intermedi si riscontrano in datasets corrispondenti a temperature oltre quella critica. É ora possibile spiegare l'istogramma. Da quanto appena affermato, è chiaro che i due picchi estremi dell'istogramma provengano da configurazioni ordinate. Si può affermare che la prima variabile latente a basse temperature abbia una distribuzione di tipo bimodale. Tale risultato è conseguenza del fatto che le configurazioni ordinate abbiano i siti con valori di spin tutti o per la maggior parte uguali. É quindi evidente che si otterrà non solo una polarizzazione del valore della magnetizzazione, ma anche di quello della prima variabile latente. La parte centrale costituisce l'effetto dei dati derivati da temperature oltre quella critica. L'istogramma mostra che la distribuzione è di tipo gaussiano. Anche questo era prevedibile dato che all'aumentare della temperatura si ottengono configurazioni sempre più disordinate. La magnetizzazione media si avvicina sempre di più ad un valore medio tra -1 e 1. Ovviamente il fenomeno è accentuato ad alte temperature, mentre appena oltre la temperatura critica è più facile avere anche valori più vicini agli estremi, oltre a valori intermedi. Tale comportamento è evidente dai grafici delle pagine precedenti e si ripropone nei valori ottenuti della prima variabile latente.

4.6 Generazione di Nuove Immagini

In questa sezione si presentano i risultati della generazione di nuove immagini utilizzando il VAE. Per fare questo bisogna partire da un dataset, che in questo caso è il test set. Una volta effettuato il training si manda in input il test set al VAE. L'output prodotto deve permettere la generazione di immagini che siano simili a quelle del test set.
Per spiegare il risultato ottenuto bisogna innanzitutto trattare di come sono determinati i colori dei pixel in un immagine. Si ricorda che le immagini che si trattano sono in scala di grigi. I colori dei pixel corrispondono ad un array contenente valori compresi tra 0 e 1. Più un elemento dell'array è vicino ad 1, più il pixel corrispondente a quell'elemento è di un tonalità di grigio vicina al bianco. Più un elemento dell'array è vicino a 0, più il pixel corrispondente a quell'elemento è di una tonalità di grigio vicina al nero. Ovviamente le immagini del test set, essendo immagini in bianco e nero, sono array contenenti solo 0 ed 1.
L'output del VAE è un array avente 784 elementi, ognuno corrispondente ad un pixel di un'immagine di dimensione 28×28. Al contrario delle immagini in input, gli elementi dell'array non sono solo 0 e 1, ma valori compresi tra 0 e 1. Questo è il risultato di un'analisi da parte del VAE delle immagini del test set, da cui ha ricavato una distribuzione di probabilità e da cui ha campionato dei valori per produrre un output che seguisse tale distribuzione. É quindi chiaro che il valore di un pixel specifico non sarà sempre lo stesso per ogni immagine del test set, ma potrà essere o 0 o 1. Ovviamente se ci si trova in una fase ordinata a basse temperature è probabile che per quasi tutte le immagini il valore di un dato pixel sia fisso e questo influenza il risultato del VAE. Ad esempio se per la quasi totalità delle immagini del test set il valore di un determinato pixel è 1, il valore che si otterrà nell'array di output del VAE sarà molto prossimo ad 1, ad esempio 0.992. Nel caso invece si utilizzasse un test set proveniente da un set preso ad alte temperature, la frequenza con cui uno specifico pixel assume il valore di 0 o 1 è la stessa, essendo in una fase disordinata. Questo porta il VAE a produrre un output che sia la media dei due effetti e quindi si ottiene un pixel con valore corrispondente di circa 0.5.
Essendo i valori dell'array di output una rappresentazione della frequenza con cui i valori di 0 e 1 appaiono, si possono interpretare come probabilità del pixel corrispondente di essere nero o bianco.
Per generare nuove immagini si è utilizzata una funzione della libreria *random* che campionasse valori da una distribuzione binomiale. In particolare 784 valori, per poi poter ottenere un'immagine 28×28. Ad ogni valore è associata una diversa distribuzione binomiale con diverse probabilità di successo. Per ottenere delle immagini che rispecchiassero le immagini del test, si è utilizzato l'array di output del VAE e ogni suo elemento è stato usato come probabilità di successo per le diverse distribuzioni binomiali. Si ottiene quindi che ad ogni distribuzione di probabilità è associata una probabilità di successo rappresentata da un elemento dell'array di output del VAE e quindi ad ogni campionamento corrisponde un elemento dell'array di output del VAE.

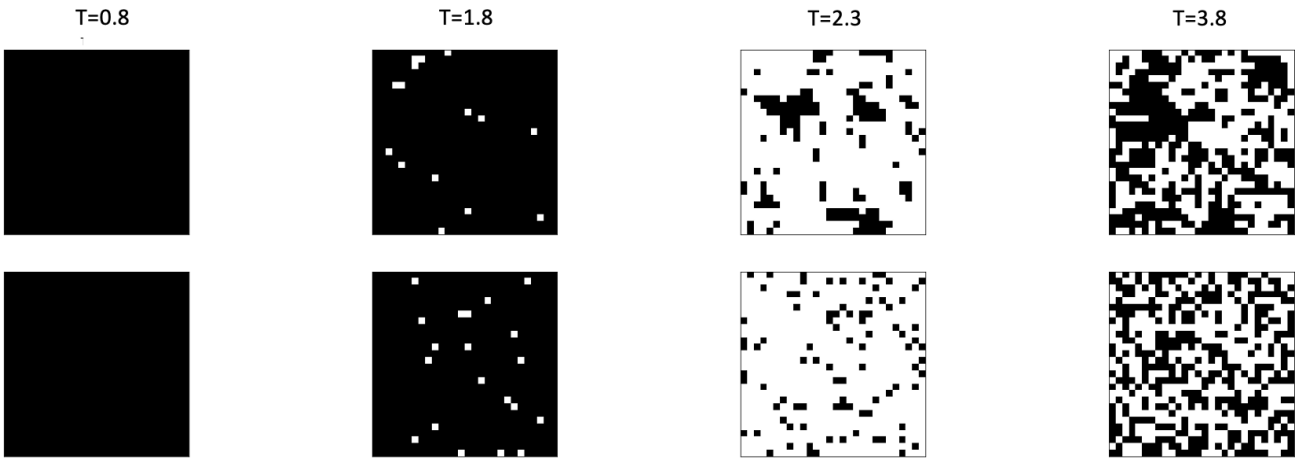

Figura 4.10: Si riportano nella figura immagini di alcuni test set con quelle risultanti dal processo esposto.

Nella figura 4.10 sono riportati dei risultati che confrontano alcune immagini di alcuni test set con quelle risultanti dal processo esposto. L'immagine superiore di ogni coppia è un'immagine del test set, mentre l'immagine inferiore è un'immagine generata attraverso il VAE. Si precisa che le immagini mostrate sono state pescate dai datasets corrispondenti alle temperature: T=0.8, T=1.8, T=2.3 e T=3.8.

4.7 Conclusione

Sulla base del lavoro esposto nelle precedenti sezioni si possono trarre conclusioni significative sul funzionamento di un VAE applicato al modello di Ising. Per farlo è necessario confrontare i due metodi seguiti. Anzitutto bisogna ricordare che nel modello di Ising il parametro d'ordine, che descrive l'evoluzione del grado di ordine di un sistema al variare di una certa grandezza, è la magnetizzazione e che la grandezza è la temperatura. Il primo metodo utilizzato è stato quello di effettuare separatamente i training su datasets acquisiti in corrispondenza a singoli valori della temperatura, mentre il secondo è stato quello di effettuare un unico training sul dataset ottenuto unendo i datasets corrispondenti alla singole temperature. Ai fini dell'apprendimento da parte del VAE della relazione tra parametro d'ordine e grandezza analizzata, si può ritenere che il primo metodo porti a risultati con una minore significatività rispetto al secondo, dato il fatto che col primo metodo si analizzano separatamente i datasets corrispondenti a singole temperature mentre col secondo si lavora contemporaneamente sulla totalità dei dati acquisiti avendo, perciò, la possibilità di metterli più efficacemente in relazione tra loro e ottenendo una comprensione da parte del VAE più completa dell'evoluzione del parametro d'ordine al variare della temperatura. Questa considerazione viene confermata confrontando i grafici delle loss functions (figure 4.3 e 4.4), e i grafici 4.7 e 4.8 dai quali emerge come il secondo metodo sia caratterizzato da un training più efficace e di come si sia rivelato maggiormente adatto ad individuare una correlazione tra il parametro d'ordine (la magnetizzazione media) e la prima variabile latente. Si può comunque ipotizzare che con dei datasets più grandi si sarebbero ottenuti dei miglioramenti negli andamenti complessivi delle loss functions del primo metodo e quindi una qualità migliore dei training. In ogni caso, il primo metodo ha consentito di comprendere meglio i risultati del secondo ed è, come spiegato, l'unico in grado di generare nuovi dati, che è lo scopo ultimo di un VAE. Si conclude che per temperature non troppo basse (una soglia minima non è stato possibile individuarla) la prima variabile latente di un VAE, costruito come descritto in questo lavoro e con un dataset strutturato come nel caso del secondo metodo, può rappresentare una valida alternativa alla magnetizzazione media come parametro d'ordine del sistema descritto dal modello di Ising 2D.

Bibliografia

[1] R. G. M. Juan carrasquilla, "Unsupervised learning for local structure detection in colloidal systems," *The journal of Chemical Physics*, 2017.

[2] K. Huang, *Statistichal Mechanics*. John Wiley & Sons.

[3] M. R. Carrasquilla J., "Machine learning phases of matter," *Nature Phys 13*, 2017. DOI: https://doi.org/10.1038/nphys4035.

[4] W. S.J., "Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders," *Phys. Rev. E 96*, 2017. DOI: https://doi.org/10.1103/PhysRevE.96.022140.

[5] I. Goodfellow, Y. Bengio e A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[6] W. Krauth, *Cluster Monte Carlo algorithms*. CNRS-Laboratoire de Physique Statistique Ecole Normale Supéerieure, 2008.

[7] W. S. McCulloch e W. Pitts., *A logical calculus of the ideas immanent in nervous activity*. The bulletin of mathematical biophysics, Vol 5, pages 115-133, 1943.

[8] C. Doersch, *Tutorial on Variational Autoencoders*. Carneige Mellon/UC Berkley, 2016.

[9] J. P. Sethna, *Statistichal Mechanics, Entrpoy, Order Parameters and Complexity*. Claredon Press, 2017.

[10] (2021), indirizzo: https://numpy.org/doc/stable/.

[11] (2021), indirizzo: https://pypi.org/project/Pillow/.

[12] L. Onsager, "Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition," *American Physical Society*, 1944. DOI: https://doi.org/10.1103/PhysRev.65.117.

[13] F. Chollet. (2016). "Building Autoencoders in Keras," indirizzo: https://blog.keras.io/building-autoencoders-in-keras.html.

Lingua: Italiano