Università degli Studi di Padova

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea in Fisica

Machine-learned Coarse-graining per Sistemi Polimerici

Relatore:
Dr. Emanuele Locatelli

Laureando:
Giovanni Fantoni

Correlatore:
Dr. Francesco Mambretti

Anno Accademico 2022/2023

Indice

Introduzione

Le proprietà chimiche e fisiche di diverse molecole possono essere studiate per via computazionale attraverso simulazioni di dinamica molecolare, permettendo inoltre la scoperta di nuovi materiali. Tuttavia la richiesta computazionale per calcoli accurati di meccanica quantistica è elevata e ciò risulta un ostacolo nel caso in cui il sistema atomistico in esame manifesti proprietà su larghe scale spaziali e temporali. Per questo motivo vengono utilizzate delle tecniche per ridurre i costi computazionali delle simulazioni come il *coarse-graining*, una classe di modelli che consistono nel trovare una rappresentazione della molecola originale con un numero minore di pseudo-atomi (detti *siti coarse-grained*) e associare ad essa un'hamiltoniana. In questo modo è possibile trattare il sistema con meno gradi di libertà, diminuendo i costi computazionali e permettendo di simulare per tempi più lunghi e su regioni spaziali più vaste.
Utilizzando questi modelli possono essere studiate diverse proprietà di un sistema atomistico, come le proprietà topologiche. Ad esempio, dalla presenza di nodi che restringono notevolmente il numero di conformazioni che una molecola può assumere possono scaturire proprietà chimiche quali la chiralità e l'attività catalitica.
Negli ultimi anni il rapido sviluppo di tecniche di apprendimento automatico ha dimostrato come allenare modelli di machine learning a riprodurre i risultati ricavati da simulazioni di Dinamica Molecolare permetta di ottenere predizioni accurate delle proprietà chimiche. Nel 2017 Scütt et al. propongono *SchNet*[1], una rete neurale convoluzionale a filtro continuo volta ad apprendere le forze d'interazione agenti su ogni atomo di una molecola.
Anche il coarse-graining può essere effettuato con delle reti neurali. Nel loro articolo del 2019 *Coarse-graining auto-encoders for molecular dynamics*[2], W. Wang e R. Gomez-Bombarelli propongono l'architettura di un *auto-encoder* in grado di apprendere direttamente da un data-set la rappresentazione e il potenziale coarse-grained di una molecola.
In questa tesi sono descritte le architetture di queste due reti neurali, che sono state successivamente applicate al caso di polimeri ad anello composti da 70 monomeri. Inoltre è stato verificato se la procedura di coarse-graining rispettasse la topologia dei polimeri o introducesse dei nodi.

Capitolo 1

Auto-encoders e coarse-graining

Lo studio computazionale di sistemi atomistici permette di ottenere informazioni riguardo proprietà chimiche e fisiche di diverse molecole, accelerando la scoperta di nuovi materiali.
Lo sviluppo di tecnologie sempre più potenti a livello computazionale consente di effettuare simulazioni per decine di nanosecondi e diversi nanometri e con risoluzione rispettivamente del femtosecondo e dell'angstrom; tuttavia diversi fenomeni si manifestano su scale temporali di oltre tre ordini di grandezza superiori.
Per questo motivo è necessario introdurre delle tecniche di calcolo che permettano di simulare sistemi estesi per tempi sufficientemente lunghi senza richiedere tempi di calcolo eccessivi. Una di queste è il *coarse-graining* (CG) che prevede la rappresentazione del sistema atomistico attraverso un numero minore di pseudoatomi detti *siti di interazione* (o atomi/siti CG). Tuttavia tale modello deve sottostare a certe condizioni affinché i risultati delle simulazioni a bassa risoluzione (CG) siano coerenti con quelle basate su modelli atomistici: risulta necessario definire un framework statistico dal quale derivino tali condizioni.

1.1 Coarse-graining

Un modello CG si dice consistente con un modello atomistico se:

- coordinate e momenti CG sono definiti come combinazione lineare delle coordinate e dei momenti di alcuni atomi del sistema;
- all'equilibrio la distribuzione delle coordinate e dei momenti del modello CG è uguale a quella del modello atomistico.

Nel 2008 W.G. Noid et al propongono un modello statistico che permette di descrivere le condizioni sufficienti per la consistenza di un modello CG. [3]

L'hamiltoniana che descrive il sistema sarà data da

$$H(\mathbf{r},\mathbf{p}) = \sum_{i=1}^{n} \frac{1}{2m_i}\mathbf{p}_i^2 + u(\mathbf{r})$$

dove $\mathbf{r} = (\mathbf{r}_1, ..., \mathbf{r}_n)$ sono le coordinate degli n atomi, $\mathbf{p} = (\mathbf{p}_1, ..., \mathbf{p}_n)$ sono i rispettivi momenti ($\mathbf{r}_i, \mathbf{p}_i \in \mathbb{R}^3$) e $u(\mathbf{r})$ il potenziale d'interazione degli atomi. Possiamo descrivere la densità di probabilità degli stati nell'ensemble canonico attraverso il prodotto delle probabilità

$$p_r(\mathbf{r}) \propto \exp\left(-\frac{u(\mathbf{r})}{k_b T}\right) \quad \text{e} \quad p_p(\mathbf{p}) \propto \exp\left(-\sum_{i=1}^{n} \frac{1}{k_b T}\frac{\mathbf{p}_i^2}{2m_i}\right)$$

ovvero $p_{rp}(\mathbf{r},\mathbf{p}) = p_r(\mathbf{r})p_p(\mathbf{p})$, dove k_b, T e m_i sono rispettivamente la constante di Boltzmann, la temperatura del sistema considerato e le masse dei rispettivi atomi.
Si può dare una definizione analoga per l'hamiltoniana $\bar{H}(\mathbf{R},\mathbf{P})$, la densità degli stati $p_{RP}(\mathbf{R},\mathbf{P})$ e il potenziale $U(\mathbf{R})$ del modello CG con coordinate dei siti CG $\mathbf{R} = (\mathbf{R}_1, ..., \mathbf{R}_N)$ e rispettivi momenti $\mathbf{P} = (\mathbf{P}_1, ..., \mathbf{P}_N)$. Dalla definizione di modello consistente possiamo dunque imporre che

$$p_{rp}(\mathbf{r},\mathbf{p}) = p_r(\mathbf{r})p_p(\mathbf{p}) \stackrel{!}{=} p_R(\mathbf{R})p_P(\mathbf{P}) = p_{RP}(\mathbf{R},\mathbf{P}).$$

La definizione di coordinate e momenti CG saranno invece date da degli operatori lineari di mappatura $M_{\mathbf{R}}(\mathbf{r}) = \{M_{\mathbf{R}_1(\mathbf{r})}, ..., M_{\mathbf{R}_N(\mathbf{r})}\}$ e $M_{\mathbf{P}}(\mathbf{p})) = \{M_{\mathbf{P}_1(\mathbf{p})}, ..., M_{\mathbf{P}_N(\mathbf{p})}\}$ tali che

$$M_{\mathbf{R}_I}(\mathbf{r}) = \sum_{i=1}^{n} c_{Ii}\mathbf{r}_i, \quad M_{\mathbf{P}_I}(\mathbf{p}) = M_I \sum_{i=1}^{n} c_{Ii}\frac{\mathbf{p}_i}{m_i} \quad \text{per} \quad I = 1, ..., N$$

dove $c_{Ii} \in \mathbb{R}$ e le masse M_I dei siti CG saranno da definire in funzione delle masse m_i degli atomi associati al sito I. Inoltre si può dimostrare che la relazione tra $U(\mathbf{R})$ e $u(\mathbf{r})$ è data univocamente, a meno di una costante, dal potenziale atomistico e dagli operatori di mappatura $M_{\mathbf{R}}(\mathbf{r})$.
Infine per sistemi che non presentino vincoli intra-molecolari rigidi possiamo definire il seguente insieme di condizioni sufficienti per la consistenza tra i modelli CG e atomistico nello spazio delle fasi:

- ogni atomo deve essere assegnato a uno e un solo sito CG;
- la posizione di ogni sito CG è definita dagli operatori di mappatura $M_{\mathbf{R}}(\mathbf{r})$ definiti sopra;
- le forze agenti sul singolo sito CG devono essere definite come

  $$F_I(\mathbf{R}) = \langle \mathcal{F}_I(\mathbf{r})_{\mathbf{R}} \rangle \quad \text{con} \quad \mathcal{F}_I(\mathbf{r})_{\mathbf{R}} = \sum_{j \in S_I} f_j(\mathbf{r})\frac{d_{Ij}}{c_{Ij}}$$

  dove il set di coefficienti $\{d_{Ij}\}$ è tale che $d_{Ij} \neq 0$ solo per $I = j$ e $\sum_{j\in S_I} d_{Ij} = 1$ con $S_I = \{i | c_{Ij} \neq 0 \ \text{ e } \ c_{Ji} = 0 \ \text{ per } \ J \neq I\}$ ovvero S_I è l'insieme di tutti gli atomi associati

ad un sito CG. I coefficienti costanti d_{Ij} per come sono definiti fanno sì che ogni atomo sia associato ad un solo sito CG. Le masse di tali siti sono definiti come

$$M_I = \left(\sum_{i \in S_I} \frac{c_{Ii}}{m_i} \right)^{-1} .$$

In caso il potenziale sia calcolato a partire da intuizioni chimico-fisiche il modello è detto *top-down*; da questi modelli è difficile estrapolare informazioni quantitative e con capacità predittive in quanto si concentrano sul rispettare le proprietà che il sistema manifesta. Una seconda possibilità consiste nel calcolare il potenziale come un potenziale efficace di forza media a molti corpi (PMF). Esso è definito dal modello atomistico e dalla mappa di assegnazione ed è interpretabile come una funzione di energia libera il cui gradiente è uguale alle medie delle forze atomistiche. Un approccio di questo tipo è detto *bottom-up* e la difficoltà in questo caso risiede nel trovare approssimazioni del PMF trattabili ed efficienti dal punto di vista computazionale e sufficientemente accurate per descrivere un dato fenomeno.

1.2 Auto-encoder

Risulta quindi necessario definire gli operatori di mappatura $M_{\mathbf{R}}(\mathbf{r})$. Al posto che darne una definizione a priori è possibile utilizzare delle reti neurali dette *auto-encoders* al fine di imparare direttamente dai dati simulati tali mappe, dette di *encoding*.
Un auto-encoder è una rete neurale addestrata per produrre un output simile all'input presentato. In generale un input $\mathbf{x} \in \mathbb{R}^n$ può essere descritto attraverso una funzione parametrizzata di encoding (*encoder*) $h = f(\mathbf{x}; \theta)$ che mappa l'input in uno spazio detto *latente* ed una di decoding (*decoder*) $r = g(h; \phi)$ dove θ e ϕ sono i vettori dei parametri delle rispettive funzioni. L'apprendimento di encoder e decoder viene effettuato attraverso la minimizzazione di una loss function $L\left(\mathbf{x}, g(f(\mathbf{x}))\right) \in \mathbb{R}$ che risulta maggiore più l'output si discosta dall'input, ad esempio attraverso lo scarto quadratico medio (MSE)

$$L\left(\mathbf{x}, g(f(\mathbf{x}))\right) = \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{x}_i - g(f(\mathbf{x})_i)\right)^2$$

dove N è la dimensione di input e output.
Iterazione dopo iterazione i valori di θ e ϕ vengono aggiornati fino a portare la loss function a convergere ad un minimo locale o, in un caso ideale, a un minimo globale. Tuttavia se l'auto-encoder imparasse una funzione identità copiando quindi l'input nell'output non rappresenterebbe una compressione dei dati e quindi non catturerebbe le loro caratteristiche salienti. Un metodo per ovviare a ciò consiste nel rendere la dimensione dello spazio latente inferiore a quella di input e output, creando così una forma a "collo di bottiglia" (un *bottleneck*). La scelta di tale dimensione può modificare drasticamente l'esito dell'apprendimento: se fosse troppo

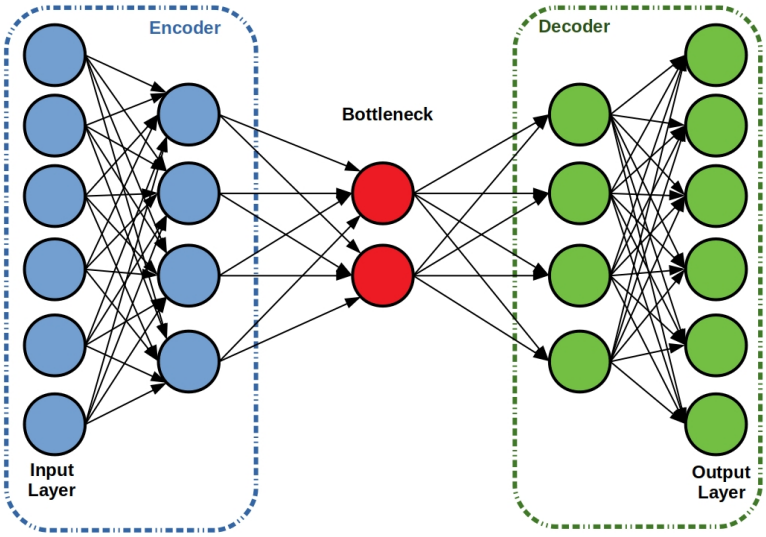

Figura 1.1: Esempio di auto-encoder undercomplete. (https://starship-knowledge.com/autoencoder)

piccola non permetterebbe di cogliere tutte le caratteristiche salienti, viceversa se fosse troppo vicina a quella dell'input può imparare ad effettuare una semplice copia dei dati.
Ulteriori metodologie per affinare l'apprendimento consistono nell'aggiunta di termini di regolarizzazione alla loss function, come nel caso di auto-encoders sparsi[1] per cui si aggiunge un termine $\Omega(h)$ di penalizzazione di sparsità che tiene conto di statistiche uniche dei dati affinché l'auto-encoder venga regolarizzato come *sparso*. Un'altra tecnica di regolarizzazione costringe il modello ad imparare una funzione che restituisca valori simili per piccole variazioni dell'input. Ciò è realizzato aggiungendo un termine alla loss function che dipenda dal gradiente delle variabili dello spazio latente h_i:

$$\Omega(h, \mathbf{x}) = \lambda \sum_i ||\nabla_{\mathbf{x}} h_i||^2.$$

Infine nel caso di *feedforward neural networks* come gli auto-encoders, a seconda della classe di funzioni che si vogliono rappresentare l'addestramento di reti a più layers permette un'approssimazione dell'encoder arbitrariamente accurata, una riduzione dei costi computazionali, un decremento esponenziale dei dati necessari al training e una miglior compressione rispetto agli auto-encoder a singolo layer.

[1] In questo contesto la sparsità di un auto-encoder indica che per alcuni parametri della rete il loro valore ottimale deve essere 0.

Capitolo 2

Apprendimento di energie e forze

La procedura di coarse-graining richiede la definizione di siti CG e il calcolo di un potenziale efficace di interazione per la dinamica del sistema. Nel capitolo precedente sono stati introdotti gli auto-encoders, una tipologia di reti neurali che possono essere utilizzati per apprendere la mappa di encoding necessaria per la definizione dei siti CG. Per imparare il potenziale efficace si può utilizzare una rete neurale convoluzionale (CNN) a filtro continuo.

2.1 Reti neurali convoluzionali

Le CNN [4] sono un tipo di rete neurale tipicamente utilizzata per analizzare dati che abbiano una topologia "a griglia". Un classico esempio sono le immagini, interpretabili come una griglia 2-dimensionale.
Su tale griglia viene applicata una convoluzione che nel caso più generale può essere definita come

$$s(t) = (x * w)(t) = \int x(a)w(t-a)\mathrm{d}a \quad \text{oppure} \quad s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

a seconda che le funzioni x e w siano definite su uno spazio continuo o discreto. In particolare w è detta *kernel* o *filtro* e l'output del layer che effettua la convoluzione è detto *feature map*. L'applicazione di una convoluzione ad un input n-dimensionale richiede la definizione di un filtro di n dimensioni. Un filtro 2-D può essere definito come una funzione a due variabili $K(i,j)$ e la convoluzione ad esso associata risulta essere

$$S(i,j) = (I*K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \overset{(a)}{=} (K*I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

dove (a) evidenzia la proprietà commutativa della convoluzione. Ciò permette di utilizzare indifferentemente $(I*K)$ o $(K*I)$ nell'implementazione a computer di una CNN. Si osservi che tale proprietà deriva dal fatto che gli indici del kernel sono invertiti rispetto a quelli dell'input, ovvero se nel calcolo della convoluzione una delle due funzioni è indicizzata m, n (indici delle

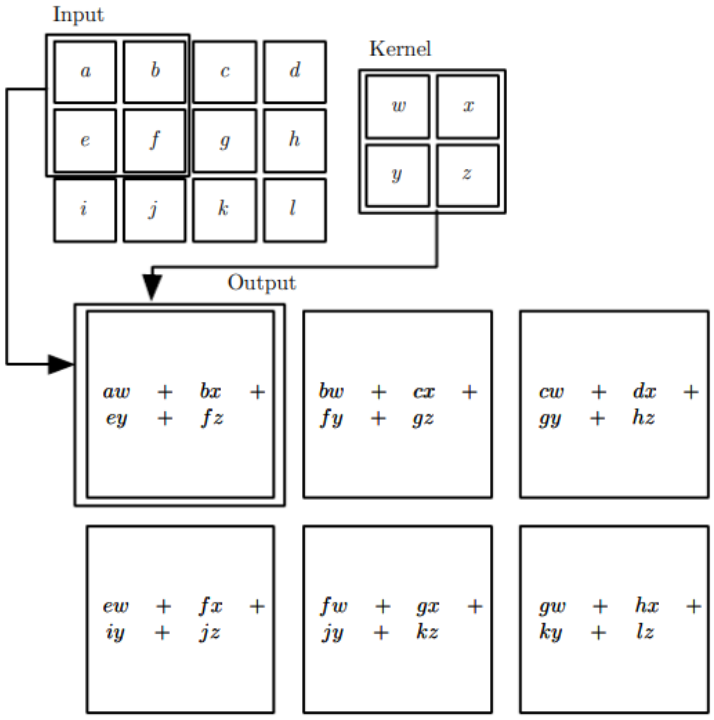

Figura 2.1: Esempio di convoluzione 2-D con *cross-correlation*.

due serie) l'altra sarà indicizzata da $-m, -n$ shiftati del valore degli indici di S ovvero i e j, e da ciò scaturisce la commutatività. Nella pratica molte librerie come pytorch[1] e tensorflow[2] utilizzano la *cross-correlation*

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i+m, j+n)$$

al posto della convoluzione. Di fatto cross-correlation e convoluzione sono equivalenti a meno dell'inversione degli indici $i, j \to -i, -j$; inoltre i valori dei filtri sono inizializzati casualmente e successivamente ottimizzati attraverso la minimizzazione di una *loss function* e per questo motivo se la rete imparasse una cross-correlation basterebbe invertire gli indici del risultato per ottenere una convoluzione.

La possibilità di utilizzo della cross-correlation come convoluzione implica che si possano dare diverse definizioni di convoluzione in diverse situazioni (per quanto riguarda l'apprendimento automatico), come vedremo nei prossimi paragrafi.

Una rappresentazione grafica di quanto appena descritto si può trovare in figura 2.1: è applicata una convoluzione con una matrice 2×2 (filtro) ad un input 3×4. In particolare viene calcolata la somma del prodotto *element-wise* del filtro con un blocco 2×2 dell'input e l'output diventa l'elemento di matrice $(1, 1)$ della feature map; successivamente si effettua la stessa operazione con tutti gli altri blocchi 2×2 estraibili dall'input e questo procedimento porta ad un output

[1] `https://pytorch.org/`

[2] `https://www.tensorflow.org/`

di dimensioni 2×3.

Questo esempio mostra un tipo di interazione tra neuroni diversa da quello delle reti neurali comuni. Considerando un classico layer denso (detto anche *fully connected layer*) che mappa un input $\mathbf{x}^0 \in \mathbb{R}^n$ in $\mathbf{x}^1 \in \mathbb{R}^m$ tale interazione sarà descritta da

$$\mathbf{x}^1 = A\mathbf{x}^0 + b$$

dove $A \in M^{m \times n}(\mathbb{R})$ e $b \in \mathbb{R}^m$; ogni elemento di matrice di A rappresenta l'interazione del singolo neurone d'uscita con il singolo neurone d'ingresso.

Nei layer convoluzionali delle CNN si ha invece quella che è detta *interazione sparsa*, ovvero il filtro ha dimensioni inferiori rispetto all'input. Assieme al fatto che i parametri del kernel non sono utilizzati solo per mappare da un neurone di input ad uno di output (utilizzati dunque una sola volta) ma ogni elemento del filtro è moltiplicato almeno una volta per ognuno di quelli dell'input (caratteristica nota come *parameter sharing*), tale layer richiede meno memoria per immagazzinare i parametri ed effettua meno operazioni per calcolare l'output, portando nel complesso ad un miglioramento di efficienza computazionale.

Supponiamo infatti di avere gli input $\mathbf{x}^0$ e output $\mathbf{x}^1$ definiti sopra. Siccome A è una matrice $m \times n$ gli algoritmi tipicamente utilizzati per effettuare moltiplicazioni tra matrici e vettori hanno tempo di esecuzione $O(m \times n)$. Tuttavia se volessimo che ogni neurone d'uscita fosse connesso a k neuroni d'ingresso (con k tipicamente svariati ordini di grandezza inferiore rispetto ad m) potremmo definire un filtro $W \in M^{k \times n}(\mathbb{R})$, diminuendo notevolmente il runtime a $O(k \times n)$ per la singola moltiplicazione $W\mathbf{x}^0$.

Il *parameter sharing* inoltre impatta il numero di parametri da tenere in memoria: mentre per un *fully connected layer* sarebbe necessario immagazzinare gli $m \times n$ parametri di A, in un layer convoluzionale sono sufficienti i $k \times n$ parametri di W; essendo $k << m$ questo porta ad una richiesta di memoria nettamente inferiore.

In un layer di una tipica CNN all'applicazione della convoluzione otteniamo un insieme di attivazioni lineari a cui dovranno essere applicate delle funzioni di attivazione non-lineari come la softplus definita come $softplus(x) = \ln(1 + e^x)$ oppure un rettificatore $f(x) = \max(0, x)$ dove $x \in \mathbb{R}$; mentre il primo è detto *stadio convoluzionale* il secondo è a volte chiamato *stadio rivelatore*.

Un ultimo stadio detto *pooling* sostituisce alcuni neuroni dell'output dello stadio rivelatore applicando una funzione che permetta di ricavare una statistica riassuntiva. Alcuni esempi possono essere il max pooling (restituisce il valore più alto), il sum pooling (restituisce la somma dei valori dei neuroni) o l'average pooling (restituisce la media di tali valori).

2.2 Schrödinger Network (SchNet)

Le CNN possono essere utilizzate anche per imparare le forze di un sistema atomistico e il modello Schrödinger Network (SchNet) permette di fare ciò [5, 1].
Un sistema atomistico composto da n atomi può essere descritto in maniera univoca attraverso gli insiemi dei loro numeri atomici $Z = (Z_1, ..., Z_n)$ e dalle loro posizioni $R = (\mathbf{r}_1, ..., \mathbf{r}_n)$ dove $\mathbf{r}_i \in \mathbb{R}^3$. In SchNet tale sistema viene descritto attraverso una tuple di features $X^l = (\mathbf{x}_1^l, ..., \mathbf{x}_n^l)$ con $\mathbf{x}_i^l \in \mathbb{R}^F$, dove l indica il layer in considerazione e F indica il numero di feature maps (intesi come nodi o neuroni) di ogni layer.
Nel suo complesso la rete sfrutta dei *blocchi d'interazione* che applicano una convoluzione a filtro continuo seguiti da dei layer di output per ottenere la stima della proprietà intensiva o estensiva desiderata. L'embedding iniziale è dato dalla rappresentazione X^0 le cui features $\mathbf{x}_i^0$ sono inizializzate randomicamente a valori $\mathbf{a}_{Z_i}$ successivamente ottimizzati dalla rete. Al termine del processo di *training* della rete gli $\mathbf{a}_{Z_i}$ dipenderanno dal numero atomico degli atomi Z_i.

2.2.1 Blocchi d'interazione

Nella rete vengono utilizzati diversi tipi di layer. Uno di questi, e il primo ad essere utilizzato, è un layer denso detto *atom-wise* in quanto agisce sulle singole features rappresentanti gli atomi come

$$\mathbf{x}_i^{l+1} = A^l \mathbf{x}_i^l + \mathbf{b}^l$$

dove A^l e $\mathbf{b}^l$ sono rispettivamente la matrice dei pesi e il vettore degli offset del dense layer.
Poiché l'*atom-wise* layer è applicato all'atomo individualmente necessitiamo di un layer convoluzionale per farli interagire tra di loro. Tuttavia i filtri definiti nelle comuni CNN non sono applicabili in questa situazione in quanto gli atomi non sono disposti su una griglia ma le loro posizioni possono assumere qualsiasi valore in $\mathbb{R}^3$. Per questo è necessario definire una rete che generi dei filtri continui in base alle posizioni dei singoli atomi, ovvero vogliamo creare una funzione

$$W^l : \mathbb{R}^3 \to \mathbb{R}^F$$

che prenda in input le posizioni di due atomi $\mathbf{r}_{i,j} \in \mathbb{R}^3$ e restituisca un filtro dato da un vettore della stessa dimensione delle features $\mathbf{x}_i \in \mathbb{R}^F$. Ciò permette di ottenere un panorama di energia libera continuo e non discreto come rappresentato in figura 2.2.
Tale rete calcola le distanze $d_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$ e successivamente le espande in base gaussiana

$$e_k(\mathbf{r}_i - \mathbf{r}_j) = \exp\left(-\gamma(\|\mathbf{r}_i - \mathbf{r}_j\| - \mu_k)^2\right)$$

dove i centri μ_k sono scelti su una griglia uniforme unidimensionale tra lo zero e la distanza di cutoff massima oltre la quale le distanze sono considerate trascurabili; γ è un iperparametro

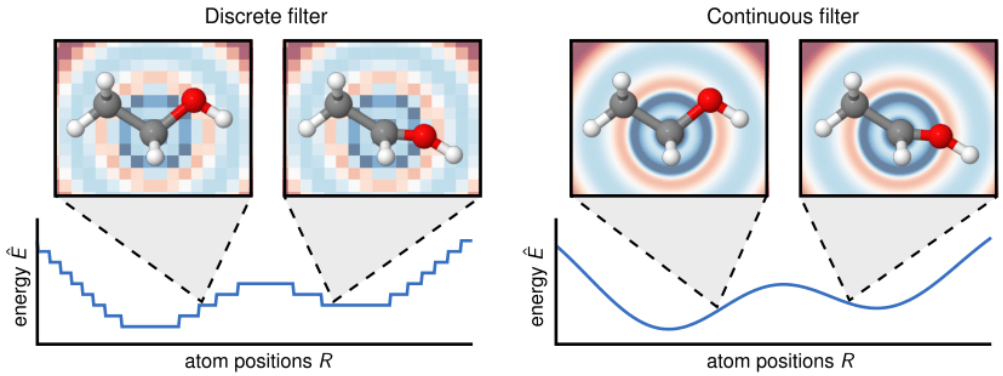

Figura 2.2: Rappresentazione della differenza tra filtro discreto e continuo in termini in termini di energia.[1]

che determina la spaziatura della griglia e quindi la risoluzione del filtro.
A seguire due layer densi con la shifted softplus come funzione d'attivazione, definita come $ssp(x) = \ln(0.5\exp(x) + 0.5)$ per $x \in \mathbb{R}$, restituiscono il filtro continuo per la convoluzione con le features risultanti dal primo *atom-wise* layer. Essa è definita come

$$\mathbf{x}_i^{l+1} = \left(X^l * W^l\right) = \sum_j \mathbf{x}_j \circ W^l(\mathbf{r}_i - \mathbf{r}_j)$$

dove $\circ$ rappresenta la moltiplicazione *element-wise*. Questa convoluzione con i filtri definiti dalla mappa W^l rappresentano il layer convoluzionale (*CfConv*).
Si noti che i filtri così definiti rispettano solamente le invarianze per rotazione e scambio di indici degli atomi ma possono essere modificati per tenere in considerazione ulteriori caratteristiche chimiche. Ad esempio è possibile implementare le periodic boundary conditions (PBC): nel caso di un sistema con PBC si ha che date due celle unitarie a e b le cui ripetizioni del feature vector $\mathbf{x}_i$ sono rispettivamente $\mathbf{x}_{ia}$ e $\mathbf{x}_{ib}$ e dato un filtro $\tilde{W}^l(\mathbf{r}_{jb} - \mathbf{r}_{ia})$ su tutti gli atomi tali che $\|\mathbf{r}_{jb} - \mathbf{r}_{ia}\| < r_{cut}$ per linearità della convoluzione otteniamo che

$$\mathbf{x}_i^{l+1} = \mathbf{x}_{im}^{l+1} = \frac{1}{n_{\text{neighbors}}} \sum_{j,n} \mathbf{x}_{jn}^l \circ \tilde{W}^l\left(\mathbf{r}_{jn} - \mathbf{r}_{im}\right) = \frac{1}{n_{\text{neighbors}}} \sum_j \mathbf{x}_j^l \circ \underbrace{\left(\sum_n \tilde{W}^l(\mathbf{r}_{jn} - \mathbf{r}_{im})\right)}_{\text{W}}$$

e il nuovo filtro W così definito, sommando su tutte le copie periodiche del feature vector $\mathbf{x}_i$, dipende dalle PBC.
Questi due layer (*Atom-wise* e *CfConv*) assieme a due ulteriori *Atom-wise* layer di cui solo il primo con funzione d'attivazione shifted-softplus costituiscono il *residual mapping*[6] del blocco d'interazione $\mathbf{v} = (\mathbf{v}_1, ..., \mathbf{v}_n)$ e l'output risulta essere

$$X^{l+1} = X^l + \mathbf{v}.$$

2.2.2 Output della rete e training

Data una proprietà P l'obiettivo di SchNet è fornirne una stima $\hat{P}$. In particolare dopo ulteriori blocchi di interazione otteniamo i contributi $\hat{P}_i$ dei singoli atomi alla stima $\hat{P}$ della proprietà cercata attraverso due *Atom-wise* layers di cui il secondo riduce la dimensione delle features $\mathbf{x}_i$ a 1 cosicché un pooling su X^l permetta di calcolare $\hat{P}$. In particolare se P è intensiva o estensiva ne si calcola la media dei $\hat{P}_i$ o ne si fa la somma rispettivamente.
Nel caso specifico di nostro interesse effettuiamo una somma dei singoli contributi $\hat{E}_i$ all'energia ottenendo così una funzione dei numeri atomici e delle posizioni

$$\hat{E} = \hat{E}\left(Z_1, ..., Z_n, \mathbf{r}_1, ..., \mathbf{r}_n\right).$$

Ciò permette di predire le forze differenziando la SchNet dell'energia rispetto alle posizioni:

$$\hat{\mathbf{F}}_i\left(Z_1, ..., Z_n, \mathbf{r}_1, ..., \mathbf{r}_n\right) = -\frac{\partial \hat{E}}{\partial \mathbf{r}_i}\left(Z_1, ..., Z_n, \mathbf{r}_1, ..., \mathbf{r}_n\right).$$

Per ogni proprietà d'interesse P alleniamo una rete SchNet minimizzando la loss

$$l(\hat{P}, P) = \|P - \hat{P}\|^2$$

e in particolare per allenare le energie e le forze per le traiettorie di dinamica molecolare usiamo una loss combinata

$$l((\hat{E}, \hat{\mathbf{F}}_1, ..., \hat{\mathbf{F}}_n), (E, \mathbf{F}_1, ..., \mathbf{F}_n)) = \rho\|E - \hat{E}\|^2 + \frac{1}{n_{atoms}} \sum_{i=0}^{n_{atoms}} \left\| \mathbf{F}_i - \left(-\frac{\partial \hat{E}}{\partial \mathbf{R}_i}\right)\right\|^2,$$

dove ρ è un iperparametro che regola il trade-off tra le loss di energie e forze.

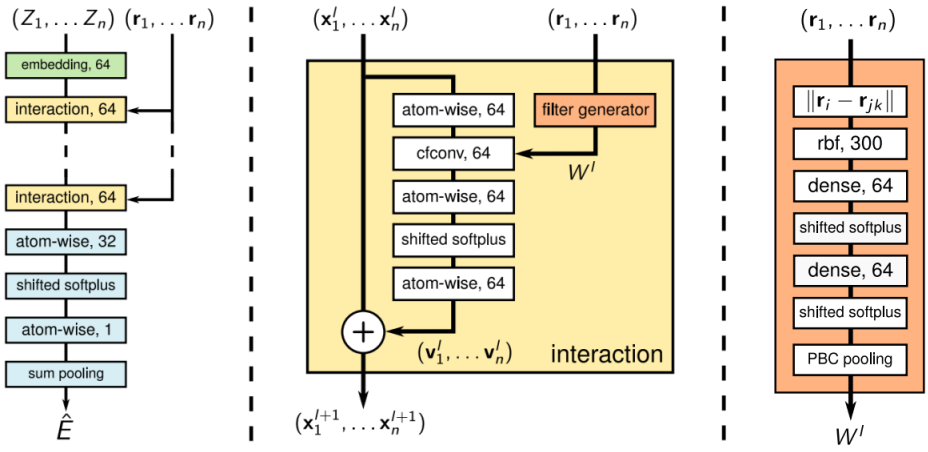

Figura 2.3: Struttura della rete: da sinistra sono rappresentati l'architettura complessiva della rete, la struttura del blocco d'interazione e della rete genera-filtri. Accanto ai nomi dei layer sono riportati il numero di neuroni per layer utilizzati in [5].

Capitolo 3

Coarse-graining auto-encoders

Nell'articolo di W. Wang e G. Bombarelli *Coarse-graining auto-encoders for molecular dynamics*[2] del 2019 gli autori sfruttano le reti indicate nei capitoli precedenti per effettuare il coarse-graining automatico di diversi sistemi.
In particolare le tecniche da loro proposte permettono di:

1. imparare la rappresentazione CG delle molecole e la ricostruzione a partire dai siti CG della molecola originale;

2. sfruttare una loss function composta da due termini: uno di ricostruzione L_{rec} per apprendere le caratteristiche salienti della molecola e uno di regolarizzazione L_{reg} per ottenere una superficie di energia libera regolare;

3. apprendere un potenziale per i siti CG.

3.1 Struttura dell'auto-encoder

Siano $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \in \mathbb{R}^{3n}$ le coordinate degli n atomi di massa m_j e $\mathbf{z} = \{\mathbf{z}_1, ..., \mathbf{z}_N\} \in \mathbb{R}^{3N}$ le coordinate degli N siti CG ($\mathbf{x}_j, \mathbf{z}_i \in \mathbb{R}^3$).

3.1.1 Encoder

L'encoder sarà dato da una funzione $E : \mathbb{R}^{3n} \to \mathbb{R}^{3N}$ tale che $\mathbf{z} = E(\mathbf{x})$. In particolare devono valere le seguenti affermazioni:

1. $z_{ik} = \sum_{j=1}^{n} E_{ij} x_{jk}, \quad i = 1, ..., N, \quad j = 1, ..., n,$

2. $\sum_{j=1}^{n} E_{ij} = 1$ e $E_{ij} \geq 0,$

3. ogni atomo contribuisce al più ad un sito CG,

dove E_{ij} rappresenta l'elemento di matrice di assegnamento dell'atomo j al sito i e k indica la coordinata cartesiana considerata (ovvero $k = x, y$ o z). Le masse dei siti CG vengono definite come $M_i = \left(\sum_{j=1}^{n} \frac{E_{ij}}{m_j}\right)^{-1}$ in accordo con quanto scritto nel primo capitolo.
I parametri dell'encoder $E(\mathbf{x})$ sono inizializzati casualmente a dei vettori $\vec{\phi}_i \in \mathbb{R}^n$ i cui elementi ϕ_{ij} saranno ottimizzati al fine di minimizzare la loss function dell'auto-encoder. Siccome gli elementi ϕ_{ij} rappresentano la probabilità di assegnazione dell'atomo j al sito CG i, alla fine del processo di ottimizzazione tutti i vettori $\vec{\phi}_i$ dovranno essere *one-hot encoded*, ovvero avranno solo un elemento pari a 1 mentre tutti gli altri saranno nulli. In questo modo infatti ogni atomo sarà assegnato ad un solo sito CG e varrà $\sum_{j=1}^{n} \phi_{ij} = 1$.
Per rispettare queste proprietà dopo l'inizializzazione i parametri ϕ_{ij} sono riparametrizzati attraverso la Gumbel-softmax

$$C_{ij} = \frac{e^{(\log(\phi_{ij}) + g_{ij})/\tau}}{\sum_{j=1}^{n} e^{(\log(\phi_{ij}) + g_{ij})/\tau}}$$

dove i g_{ij} sono estratti da una distribuzione di Gumbel come $g_{ij} = -\log(-\log(u_{ij}))$ e gli u_{ij} sono a loro volta estratti da una distribuzione uniforme tra 0 e 1, mentre τ è una variabile che diminuisce all'aumentare del numero di epoche del training. Questa parametrizzazione tende al vettore dei parametri *one-hot encoded* $\vec{C}_i = \text{one\_hot}(\arg\max_j \log \phi_{ij})$ nel limite $\tau \to 0$.
I pesi dell'encoder, ovvero i suoi elementi di matrice E_{ij}, saranno dati dalla normalizzazione di C_{ij}:

$$E_{ij} = \frac{C_{ij}}{\sum_{j=1}^{n} C_{ij}}.$$

3.1.2 Decoder

Per generare delle coordinate atomistiche a partire da quelle CG possiamo definire una mappa $D : \mathbb{R}^{3N} \to \mathbb{R}^{3n}$ tale che $\hat{\mathbf{x}}_j = \sum_{i=1}^{N} D_{ji} z_{ik}$.
Si noti che $\mathbf{x}$, $\mathbf{z}$, $E(\mathbf{x})$ e $D(\mathbf{z})$ sono tutte matrici e le loro dimensioni sono rispettivamente $n \times 3, N \times 3, N \times n$ e $n \times N$. In questo modo $\hat{x} = D(E(\mathbf{x}))$ restituisce una matrice delle stesse dimensioni dell'input, rappresentando la ricostruzione delle coordinate atomistiche.

3.2 Loss function

La loss function dell'auto-encoder L_{ae} è data dai due termini di ricostruzione e di regolarizzazione sopra citati come

$$L_{ae} = L_{rec} + \rho L_{reg}$$

dove ρ rappresenta un iperparametro di *trade-off* tra i due addendi della loss function totale.

3.2.1 Loss di ricostruzione

Gli obiettivi dell'auto-encoder sono quelli di trovare la rappresentazione CG di una molecola e di poter ricostruire le posizioni originali degli atomi a partire dai siti CG. Per questo motivo la loss function da minimizzare dovrà penalizzare il discostarsi dell'output del decoder dall'input dell'encoder.
A partire dalla descrizione della struttura dell'auto-encoder data sopra la minimizzazione della loss function avverrà secondo la seguente espressione:

$$\min_{\phi,D} L_{rec} = \min_{\phi,D} \mathbb{E}\left[(\mathbf{x} - \hat{\mathbf{x}})^2\right] = \min_{\phi,D} \mathbb{E}\left[(\mathbf{x} - D\left(E(\mathbf{x})\right))^2\right]$$

dove sono esplicitate le funzioni ϕ e D i cui parametri andranno ottimizzati attraverso l'algoritmo di backpropagation.

3.2.2 Loss di regolarizzazione

La componente di regolarizzazione della loss function L_{reg} ha invece l'obiettivo di garantire una corrispondenza tra la distribuzione di probabilità delle coordinate CG $P_{CG}(\mathbf{x})$ e quelle atomistiche $P(\mathbf{x})$ date rispettivamente dalle espressioni

$$P_{CG}(\mathbf{z}) = \frac{1}{Z}\int e^{-\beta V(\mathbf{x})}\delta(E(\mathbf{x}) - \mathbf{z})\mathrm{d}\mathbf{x} \quad \text{e} \quad P(\mathbf{x}) = \frac{1}{Z}e^{-\beta V(\mathbf{x})}$$

dove Z è la funzione di partizione del sistema atomistico, $\beta = (k_b T)^{-1}$, $V(\mathbf{x})$ è il potenziale atomistico e $E(\mathbf{x})$ è l'encoder. Il potenziale a molti corpi di forza media associato a $P_{CG}(\mathbf{z})$ è

$$A(\mathbf{z}) = -\frac{1}{\beta}\ln P_{CG}(\mathbf{z})$$

e l'opposto della sua derivata rispetto a $\mathbf{z}$ restituisce le forze medie agenti sui singoli siti CG. Per questo motivo una trattazione basata sulle forze medie al posto che le funzioni di distribuzione delle coordinate risulta equivalente e di più facile implementazione a computer oltre a permettere l'apprendimento di un potenziale CG.

Force matching

Il *force matching* si basa sull'ipotesi che la forza media agente su un sito CG calcolata a partire da un potenziale *coarse-grained* $V_{CG}(\mathbf{z})$ debba riprodurre la forza calcolata come una media su tutte le forze agenti sugli atomi associati al sito CG, ovvero

$$-\frac{\mathrm{d}A(\mathbf{z})}{\mathrm{d}\mathbf{z}} = F(\mathbf{z}) = \langle -\mathbf{b}\nabla V(\mathbf{x})\rangle_{E(\mathbf{x})=\mathbf{z}} \quad \text{con} \quad \mathbf{b} = \frac{\mathbf{w}}{\mathbf{w}^T\nabla E(\mathbf{x})}$$

dove $\mathbf{b}$ rappresenta la famiglia di possibili vettori tali che $\mathbf{w}^T\nabla E(\mathbf{x}) \neq 0$. La scelta adottata dagli autori per $\mathbf{w}$ è $\mathbf{w} = \nabla E(\mathbf{x})$ facendo sì che $\mathbf{b} = \mathbf{C}$ dove $\mathbf{C}$ è la matrice degli N vettori $\vec{C}_i$

one-hot encoded.

Secondo Zhang et al [7], $F(\mathbf{z})$ può essere interpretata come il valor medio di un'osservabile istantanea $F_{inst}(\mathbf{z})$ con fluttuazioni attorno alla media $\epsilon(E(\mathbf{x}))$ date da un'approssimazione dell'equazione generale di Langevin, con la condizione dunque che $\mathbb{E}_{\mathbf{z}}[F_{inst}] = F(\mathbf{z})$. Si può dimostrare che detta L la loss function associata a $F(\mathbf{z})$ e L_{inst} quella associata a $F_{inst}(\mathbf{z})$ esse sono legate dalla relazione

$$L_{inst} = L + \mathbb{E}\left[\epsilon(E(\mathbf{x}))^2\right].$$

Il problema si traduce dunque nella minimizzazione di L_{inst} rispetto ai parametri di ϕ e D. Nell'articolo è stato scelto $L_{inst} = \mathbb{E}\left[(F_{inst}(\mathbf{z}))^2\right] \equiv L_{reg}$ dove si identifica in L_{inst} il termine di regolarizzazione della loss function L_{reg}, e dunque si avrà

$$\min_{\phi,D} L_{reg} = \min_{\phi,D} \mathbb{E}\left[(F_{inst}(\mathbf{z}))^2\right] = \min_{\phi,D} \mathbb{E}\left[(F(\mathbf{z}))^2 + \epsilon(\mathbf{z})^2\right].$$

La loss function complessiva dell'auto-encoder risulta essere

$$L_{ae} = \frac{1}{N}\mathbb{E}\left[(\mathbf{x} - D(E(\mathbf{x})))^2 + \rho F_{inst}(E(\mathbf{x}))^2\right]$$

e la sua minimizzazione, oltre a permettere l'apprendimento di $E(\mathbf{x})$ e $D(\mathbf{z})$ di modo che $\hat{\mathbf{x}} \approx \mathbf{x}$, garantisce una superficie di energia libera regolare oltre all'equivalenza tra $P_{CG}(\mathbf{z})$ e $P(\mathbf{x})$.
Definita L_{ae} possiamo dunque fare il training dell'auto-encoder con il seguente algoritmo:

Algorithm 1 Addestramento *auto-encoder*

Inizializzazione di $\phi_{ij}, D_{ij}, \tau, \Delta\tau$
repeat
  $\mathbf{x} \leftarrow$ mini-batch casuale su frame degli atomi
  $g_{ij} \leftarrow$ Gumbel(0, 1)
  $C_{ij} \leftarrow \frac{e^{(\log(\phi_{ij})+g_{ij})/\tau}}{\sum_{j=1}^{n} e^{(\log(\phi_{ij})+g_{ij})/\tau}}$
  $E_{ij} \leftarrow \frac{C_{ij}}{\sum_{j}^{n} C_{ij}}$
  $g \leftarrow \nabla_{\phi_{ij},D_{ij}} L_{ae}(\mathbf{x}, \phi_{ij}, D_{ij}; \tau, g_{ij})$
  $\phi_{ij}, D_{ij} \leftarrow$ aggiornamento parametri attraverso i gradienti g
  $\tau \leftarrow \tau - \Delta\tau$
until L_{ae} converge al minimo

Potenziale CG

La tecnica del *force-matching* permette inoltre di imparare il potenziale CG $V_{CG}(\mathbf{z})$ associato alla configurazione dei siti CG $\mathbf{z}$.

Infatti, sulla base di quanto introdotto da Izvekov et al [8, 9] è sufficiente definire una loss

function che penalizzi il discostarsi delle forze prodotte dal potenziale CG $F_{CG}(\mathbf{z}) = -\nabla_{\mathbf{z}} V_{CG}(\mathbf{z})$ da $F_{inst}(z)$ e minimizzarla rispetto ai parametri θ di V_{CG}:

$$\min_{\theta} L_{V_{CG}} = \min_{\theta} \mathbb{E}\left[\left(F_{inst}(\mathbf{z}) - F_{CG}(\mathbf{z})\right)^2\right] = \min_{\theta} \mathbb{E}\left[\left(F_{inst}(\mathbf{z}) + \nabla_{\mathbf{z}} V(\mathbf{z})\right)^2\right]$$

3.3 Risultati

Al fine di verificare l'efficacia della metodologia introdotta, l'auto-encoder è stato allenato sulle traiettorie atomistiche in condizioni di equilibrio (ciascuna di migliaia di frame) di diverse molecole.
I dataset di orto-terfenile (OTP) e di anilina usano 3000 frames, mentre quello di alanina dipeptide 5000 e 10000 per quelli di alcuni alcani liquidi, ovvero etano (C2H6), propano (C3H8) e tetracosano (C24H50). Per l'alanina dipeptide e gli alcani liquidi non è stato utilizzato il termine di regolarizzazione in quanto nel primo caso non erano disponibili le forze e nel secondo la topologia era relativamente semplice. In tutti i casi i parametri sono stati ottimizzati utilizzando l'algoritmo di back-propagation con l'ottimizzatore Adam[10].

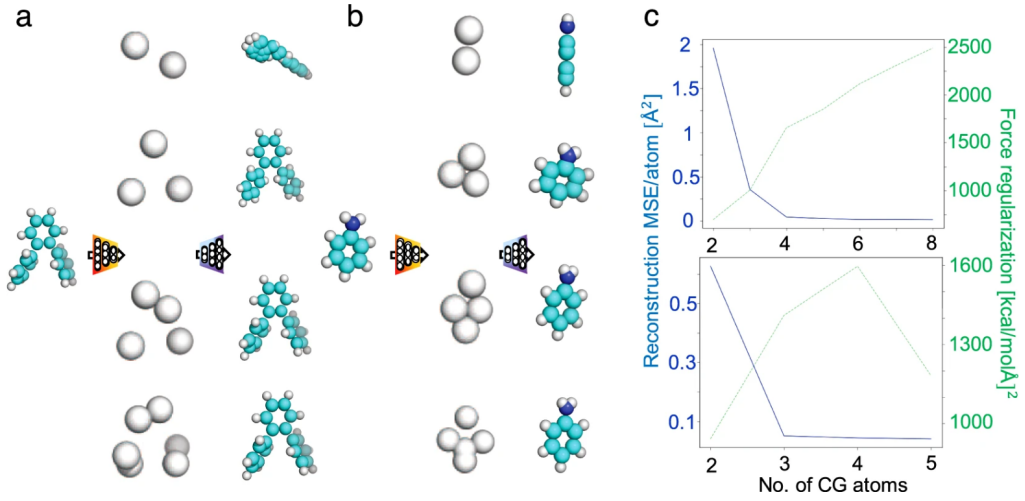

Figura 3.1: a-b Visualizzazione coarse-graining di OTP e anilina. **c** Grafici di L_{rec} e L_{reg} in funzione del numero di siti CG. Il grafico in alto rappresenta il caso di OTP mentre quello in basso il caso di anilina.

La figura 3.1 c mostra l'andamento delle componenti di ricostruzione e di regolarizzazione della loss function al variare del numero di siti CG nell'allenamento dell'auto-encoder su OTP e anilina. Come si può notare la loss di ricostruzione diminuisce all'aumentare nel numero di siti CG; ad esempio per l'OTP non sono sufficienti 2 o 3 siti per catturare tutte le informazioni caratteristiche della molecola come gli angoli tra i diversi anelli e infatti la loss L_{rec} non raggiunge un valore di convergenza. Con un numero di siti maggiore o uguale a 4 si riesce invece

a decodificare la rappresentazione CG in maniera più accurata. Viceversa la loss di regolarizzazione aumenta in quanto la superficie di energia libera diventa più irregolare.
In figura 3.2 b sono rappresentate le mappe di Ramachandran, ovvero grafici riportanti le possibili distribuzioni energeticamente permesse di angoli tra i residui amminoacidi di una struttura proteica, ottenute a partire dalle configurazioni ricostruite e da quelle simulate di alanina dipeptide. Nonostante la rete non sia in grado di determinare in maniera dettagliata la distribuzione degli atomi leggeri come l'idrogeno, con un numero adeguato di siti CG (in questo caso 8) è possibile determinare accuratamente quella degli atomi pesanti.
Si osservi inoltre come risulti necessario (figura 3.3) diminuire l'iperparametro ρ all'aumentare dei siti CG, altrimenti alcuni gradi di libertà disponibili potrebbero non essere utilizzati.

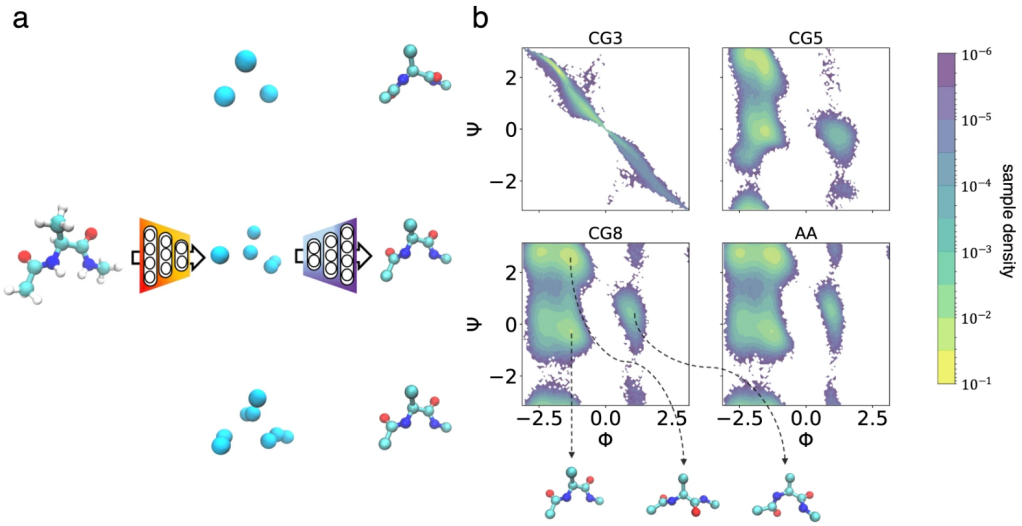

Figura 3.2: **a** Visualizzazione del coarse-graining di una molecola di alanina dipeptide. **b** Mappe di Ramachandran per le ricostruzioni dalla rappresentazione CG a 3, 5 e 8 siti e per le simulazioni atomistiche di alanina dipeptide.

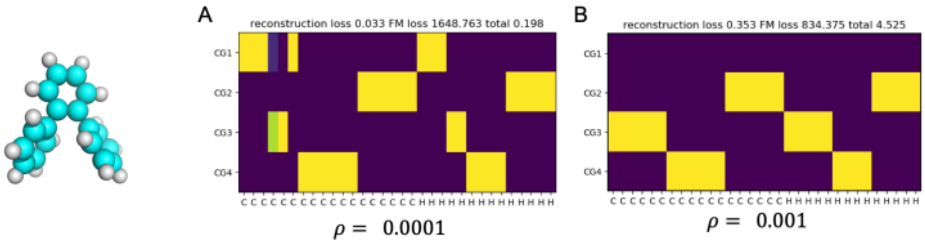

Figura 3.3: Visualizzazione della mappa di encoding per l'alanina dipeptide con 4 siti CG: si osservi che per $\rho = 0.001$ al sito CG1 non è associato alcun atomo, mentre diminuendolo a 0.0001 tutti i gradi di libertà sono sfruttati.

Il training della rete è stato effettuato anche su dataset di etano, propano e C24H50. In figura 3.4 sono riportati come esempio i risultati ottenuti per il propano. Dopo aver allenato l'auto-encoder per ottenere le posizioni CG è stato utilizzato il pacchetto NFF[1] che implementa SchNet per imparare il potenziale CG. Gli iperparametri di tale rete saranno dunque il numero di convoluzioni (T), le dimensioni dei filtri ($n_{filters}$), le dimensioni degli embedding atomici ($n_{atombasis}$), il numero di gaussiane da utilizzare nell'espansione in base gaussiana ($n_{gaussians}$) e la distanza di cutoff oltre la quale gli atomi sono considerabili non interagenti (d_{cutoff}).
Si può notare un buon accordo tra le pair correlation function calcolate dalla traiettoria atomistica e dalla ricostruzione a partire dalla rappresentazione CG (**a-d**). Anche le distribuzioni delle lunghezze di legame risultano simili, tuttavia la varianza della distribuzione CG è minore in quanto nella procedura di coarse-graining vengono persi i dettagli atomistici locali diminuendo la variabilità delle lunghezze di legame. Infine il grafico del mean square displacement mostra come esso sia sistematicamente maggiore per la dinamica CG rispetto a quella atomistica: questo perché i siti CG sono soggetti in maniera minore all'attrito atomistico.

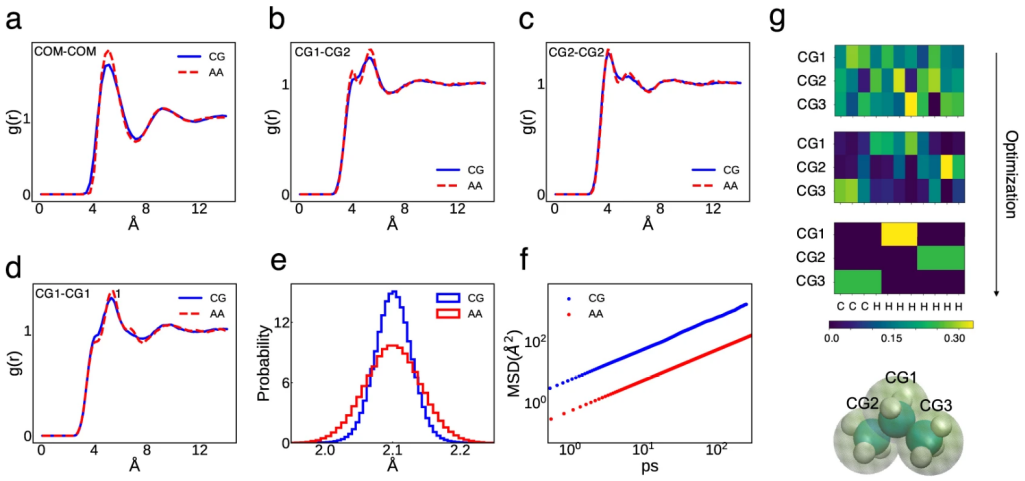

Figura 3.4: **a-e** Pair correlation function e distribuzione delle lunghezze di legame per il propano, confronto tra traiettorie atomistiche (AA) e ricostruite a partire dalla rappresentazione CG. **f** Confronto tra il mean-squared displacement (MSD) delle simulazioni atomistiche e CG. **g** Visualizzazione dell'apprendimento della matrice di encoding.

3.3.1 Discussione

Il framework presentato è dunque in grado di trovare la rappresentazione CG di una data molecola all'equilibrio rispettandone determinate caratteristiche originali.
Tuttavia l'utilizzo di un auto-encoder deterministico come quello in questione implica necessariamente una perdita di informazione, ovvero non si è in grado di ricostruire le configurazioni

[1] `https://github.com/learningmatter-mit/NeuralForceField`

istantanee originali bensì solamente delle loro medie. Un auto-encoder probabilistico potrebbe invece imparare una *distribuzione di probabilità di ricostruzione* che rifletta la termodinamica dei gradi di libertà persi in quanto mediati dalla procedura di coarse-graining.
Infine gli approcci basati sul force-matching non garantiscono la ricostruzione esatta delle pair-correlation function individuali delle traiettorie atomistiche; inoltre essendo ricostruite in un punto dello spazio termodinamico (in questo caso all'equilibrio) non è assicurata la possibilità di catturare le proprietà di trasporto al non-equilibrio.

Capitolo 4

Applicazione a polimeri

L'architettura descritta nel capitolo precedente è stata implementata in Python utilizzando il pacchetto PyTorch e pubblicata su Github[1] dagli autori dell'articolo descritto nel capitolo precedente.
Nella sua tesi *Coarse-graining auto-encoders per la dinamica molecolare*[2], G. Di Prima applica tale implementazione a dei polimeri ad anello composti da 70 monomeri. Le rappresentazioni CG di tali polimeri utilizzate in questo capitolo sono state ottenute a partire da un dataset composto da 10100 frame simulati utilizzando il codice di simulazione open-source LAMMPS con il termostato di Nosé-Hoover. L'allenamento dell'auto-encoder è stato effettuato su 7000 frame per 600 epoche. Il coarse-graining è stato realizzato imponendo 6, 12, 18 e 24 siti CG e i valori dell'iperparametro ρ della loss function utilizzati sono quelli riportati da G. Di Prima nella sua tesi.
A partire da questo lavoro si è voluto verificare se la topologia della molecola fosse mantenuta dal coarse-graining automatico oppure si venissero a formare dei nodi. La loro eventuale presenza permette di stabilire se la rappresentazione CG effettuata dall'auto-encoder possa essere ritenuta valida quando le molecole in esame (in questo caso una catena polimerica lunga) sono ben più complesse rispetto a quelle discusse nel capitolo 3.

4.1 Polimeri e topologia

I nodi sono caratteristiche topologiche comuni sia a livello macroscopico che a livello molecolare[11], in particolare in tre principali categorie di biopolimeri: DNA, RNA e proteine. Nonostante non sia ancora chiaro se queste topologie complesse siano vantaggiose dal punto di vista evolutivo, si ritiene che la maggior parte dei nodi naturali svolgano un ruolo importante nelle proprietà strutturali, dinamiche e funzionali dei sistemi biologici a cui sono associati.

[1] `https://github.com/learningmatter-mit/Coarse-Graining-Auto-encoders`
[2] `https://hdl.handle.net/20.500.12608/35067`

4.1.1 Nodi

In topologia si definisce un nodo[12] una curva chiusa in $\mathbb{R}^3$. Viene descritto attraverso un diagramma detto *rappresentazione* che ne raffigura la proiezione su un piano e specifica per ogni incrocio che viene a formarsi quale delle due parti della curva passa sopra o sotto.
Una rappresentazione con il numero minimo di incroci N_{min} è detta *ridotta* e si possono ottenere infinite rappresentazioni di un nodo attorcigliandolo su se stesso, ottenendo così degli incroci detti *insignificanti* in quanto sono sempre riconducibili al nodo originale. In particolare N_{min} è un'*invariante del nodo*, ovvero una sua proprietà intrinseca.
I nodi non riconducibili a un cerchio (detti *non banali*) hanno almeno 3 incroci e se non possono essere costruiti a partire da nodi più semplici sono detti *primi*, altrimenti *composti*.
Per classificare i nodi primi a meno di 10 incroci è di uso comune la *notazione di Alexander-Briggs*: ogni nodo è denotato dalla sigla X_Y, dove X rappresenta il numero di incroci e Y è un indice per distinguere nodi diversi ma con lo stesso numero di incroci. Ad esempio, il nodo più semplice dopo quello banale è il nodo a trifoglio che ha 3 incroci e sarà denotato con 3_1.
Inoltre i nodi possono avere diverse proprietà: ad esempio, se un nodo non è sovrapponibile alla sua immagine speculare è detto *chirale* (altrimenti è detto *achirale*).
Infine i possibili nodi primi vengono riportati in tabelle come quella riportata in figura 4.1, detta tabella di Rolfsen.

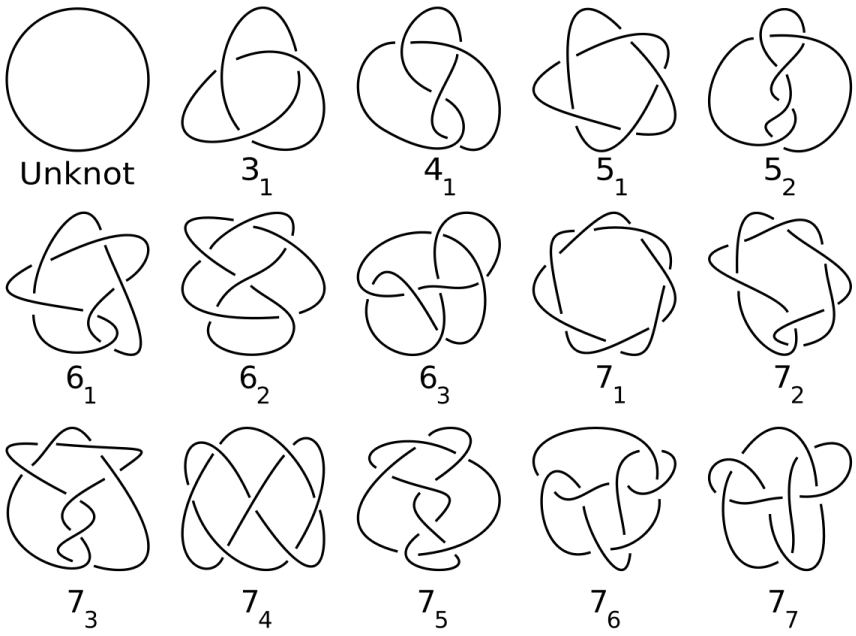

Figura 4.1: Tabella di Rolfsen per i nodi primi possibili con 7 incroci o meno.

4.1.2 Individuare i nodi

Per verificare la presenza di nodi nei polimeri ad anello è stato utilizzato KymoKnot[13], un pacchetto disponibile su Github[3]. Esso trova la regione annodata di un polimero, ovvero la parte più piccola della catena che abbia la stessa topologia del polimero, attraverso un algoritmo detto *minimally interfering closure scheme* e successivamente ne determina la tipologia di nodo in notazione di Alexander-Briggs.
Gli schemi di ricerca possibili sono detti bottom-up e top-down: nel primo si considera una porzione di catena molto breve (quindi senza nodi) e gradualmente si allunga finché non si trova un nodo dalla stessa topologia del polimero oppure il resto della catena è senza nodi; nel secondo si parte dal polimero intero e si considerano porzioni di catena sempre più corte finché non si ottiene una catena senza nodi.

4.1.3 Applicazione alle conformazioni coarse-grained

Sui polimeri ad anello composti da 70 monomeri sono state effettuate sia ricerche bottom-up (BU) che top-down (TU) per le diverse rappresentazioni CG.
Essendo i risultati della ricerca BU molto simili a quelli della ricerca TD, in figura 4.3 sono riportati solamente i primi. Ogni grafico riporta 6 colonne: la prima e l'ultima rappresentano rispettivamente la percentuale di frame senza nodi e quella dei nodi più complessi; quelle centrali invece riportano la probabilità di trovare nodi più semplici quali 3_1, 4_1, 5_1 e 5_2.
Come si può osservare all'aumentare del numero di siti CG aumenta anche il numero di frame annodati e di nodi più complessi. Nella ricostruzione dei polimeri a partire dalla rappresentazione coarse-grained invece non si riscontrano nodi (figura 4.2).

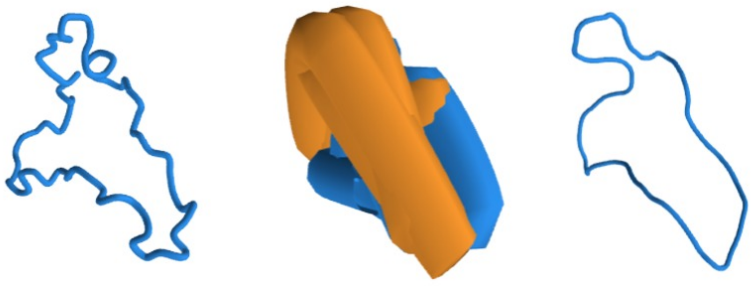

Figura 4.2: Rappresentazione in KymoKnot webserver del coarse-graining. **Sinistra:** polimero ad anello di 70 monomeri; **Centro:** rappresentazione a 18 siti CG del polimero, in arancione le parti di catena annodate; **Destra:** ricostruzione del polimero a partire dalle coordinate CG.

[3] `https://github.com/luca-tubiana/KymoKnot`

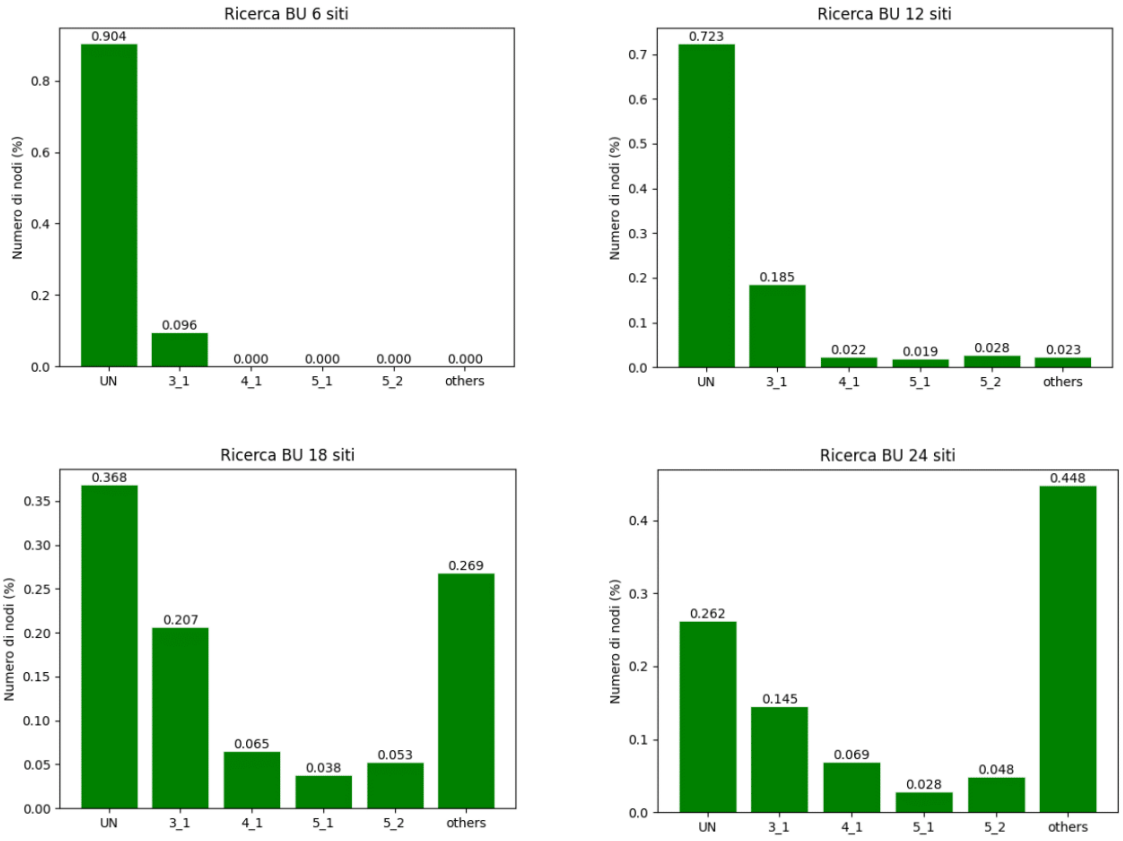

Figura 4.3: Grafici riportanti le percentuali di nodi che si formano nelle rappresentazioni coarse-grained a 6, 12, 18 e 24 siti. Nella prima colonna di ognuno si ha la percentuale di frame non annodati mentre nell'ultima si ha quella di nodi più complicati.

4.2 Neural Force Field

Successivamente si è voluto verificare se fosse possibile imparare le forze agenti sui singoli atomi con la rete descritta nel capitolo 2. In particolare è stato utilizzato il pacchetto Neural Force Field (NFF) messo a disposizione dal Learning Matter Lab del MIT[4], che implementa SchNet. La rete riceve in input le posizioni e le forze dei polimeri ad anello assieme alle energie (calcolate con il potenziale *FENE*[5] e Lennard-Jones) e gli smiles, ovvero una sigla rappresentante gli atomi delle molecole e i loro legami (gli atomi sono stati tutti considerati di carbonio). L'ottimizzatore utilizzato è Adam.

Gli iperparametri controllabili sono quelli riportati nel capitolo 3.3 oltre al parametro di trade-off ρ introdotto nella loss function di SchNet (distinto da quello dell'auto-encoder), al numero di frame utilizzati per il train (*train*), il validation (*val*) e il test (*test*) e al valore della learning rate (*lr*) dell'ottimizzatore. In figura 4.4 è presentato un esempio di risultato di training della rete effettuato con gli iperparametri riportati in tabella 4.1 per 50 epoche. Il grafico mostra

[4] `https://github.com/learningmatter-mit`

[5] `https://docs.lammps.org/bond_fene.html`

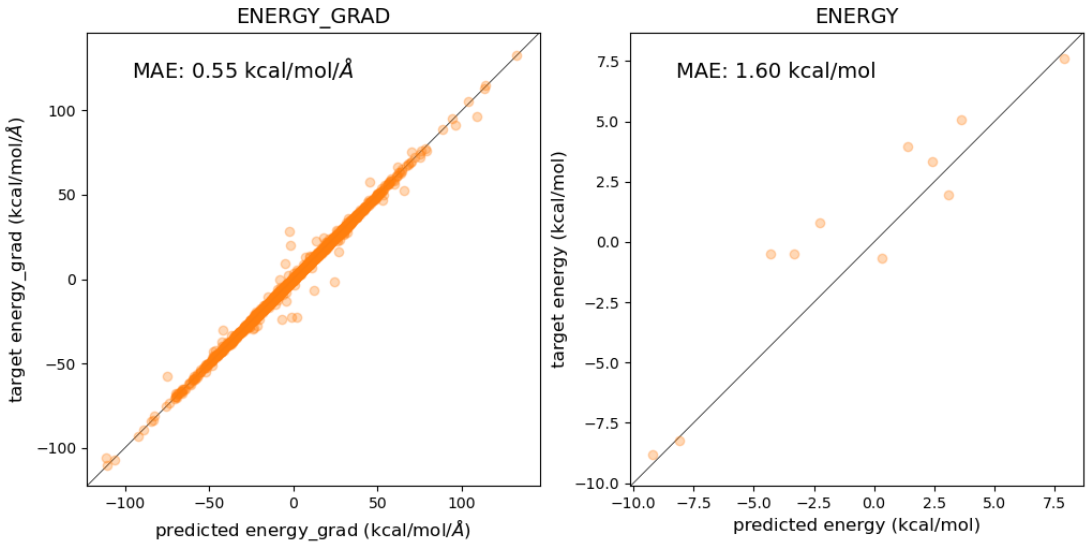

Figura 4.4: Previsioni di SchNet di forze e energie dei polimeri nel dataset di test, immagine prodotta tramite NFF. **Sinistra:** forze originali contro forze calcolate dalla rete sui dati di test; **Destra:** energie originali contro energie calcolate dalla rete sui dati di test.

un buon accordo tra le forze previste dalla rete e quelle dei dati di test, nonostante ci sia un leggero discostamento all'avvicinarsi del valore delle forze allo zero. Per quanto riguarda l'energia potenziale invece si nota una maggior differenza tra i dati e le previsioni. Si presenta inoltre un piccolo offset, in accordo con la definizione a meno di una costante dell'energia potenziale.

Tabella 4.1: Valori degli iperparametri utilizzati nel training della rete.

Iperparametro	Valore	Iperparametro	Valore
T	2	ρ	0.01
$n_{filters}$	256	$train$	8069
$n_{atombasis}$	128	val	2020
$n_{gaussians}$	32	$test$	11
d_{cutoff}	3.0	lr	10^{-4}

4.3 Discussione

Le figure 4.2 e 4.4 mostrano come in situazioni di statica le tecniche illustrate in questa tesi permettano di apprendere in maniera automatica le forze agenti sui singoli atomi di una molecola e la rappresentazione CG di un polimero ad anello ricostruendone la struttura originale

approssimativamente (si può notare infatti come in figura 4.2 la ricostruzione del polimero risulta molto più "liscia" rispetto all'originale).
Questo non garantisce l'equivalenza tra i risultati delle simulazioni di dinamica molecolare effettuate sul sistema originale e sulla rappresentazione CG. Infatti la modifica della topologia dei polimeri data dalla formazione di nodi nello spazio latente implica una differenza nell'evoluzione temporale dei due sistemi. Per ovviare a ciò potrebbe essere sufficiente introdurre nella loss function dell'auto-encoder un termine che penalizzi la formazione di nodi qualora questi non siano presenti nella configurazione di partenza.
Infine effettuando diverse simulazioni di dinamica molecolare con l'*Atomic Simulation Environment*[14] (ASE) utilizzando il potenziale ottenuto da SchNet si osserva che il polimero non rimane intatto per più di poche migliaia di frame. Questo risultato potrebbe avere origine da motivi computazionali, come la mancata implementazione delle *periodic boundary conditions* o una scelta non accurata degli iperparametri, o anche dalla possibile incompatibilità tra NFF e le caratteristiche del sistema in esame come la lunghezza delle catene e la loro topologia ad anello.
In conclusione i risultati ottenuti mostrano come le reti neurali descritte nei capitoli precedenti possano essere utilizzate per effettuare un coarse-graining di polimeri ad anello in condizioni di statica ma debbano essere modificate per effettuare delle simulazioni di dinamica molecolare.

Bibliografia

[1] K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, K.-R. Müller, Schnet: A continuous-filter convolutional neural network for modeling quantum interactions, Advances in neural information processing systems 30 (2017).

[2] W. Wang, R. Gómez-Bombarelli, Coarse-graining auto-encoders for molecular dynamics, npj Computational Materials 5 (1) (2019) 125.

[3] W. G. Noid, J.-W. Chu, G. S. Ayton, V. Krishna, S. Izvekov, G. A. Voth, A. Das, H. C. Andersen, The multiscale coarse-graining method. i. a rigorous bridge between atomistic and coarse-grained models, The Journal of chemical physics 128 (24) (2008) 244114.

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

[5] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, K.-R. Müller, Schnet–a deep learning architecture for molecules and materials, The Journal of Chemical Physics 148 (24) (2018) 241722.

[6] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[7] L. Zhang, J. Han, H. Wang, R. Car, et al., Deepcg: Constructing coarse-grained models via deep neural networks, The Journal of chemical physics 149 (3) (2018).

[8] S. Izvekov, G. A. Voth, A multiscale coarse-graining method for biomolecular systems, The Journal of Physical Chemistry B 109 (7) (2005) 2469–2473.

[9] S. Izvekov, G. A. Voth, Multiscale coarse-graining of mixed phospholipid/cholesterol bilayers, Journal of Chemical Theory and Computation 2 (3) (2006) 637–648.

[10] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization. iclr. 2015, arXiv preprint arXiv:1412.6980 9 (2015).

[11] N. C. Lim, S. E. Jackson, Molecular knots in biology and chemistry, Journal of Physics: Condensed Matter 27 (35) (2015) 354101.

[12] S. D. P. Fielden, D. A. Leigh, S. L. Woltering, Molecular knots, Angewandte Chemie International Edition 56 (37) (2017) 11166–11194.

[13] L. Tubiana, G. Polles, E. Orlandini, C. Micheletti, Kymoknot: A web server and software package to identify and locate knots in trajectories of linear or circular polymers, The European Physical Journal E 41 (2018) 1–7.

[14] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, et al., The atomic simulation environment—a python library for working with atoms, Journal of Physics: Condensed Matter 29 (27) (2017) 273002.