Contrastive 3D Protein Prediction

Angelo Nazzaro[1], Luigina Costante[2]

5 marzo 2024

Sommario

Prevedere la struttura tridimensionale (3D) di una proteina a partire dalla sua sequenza primaria è una sfida importante per la biologia, la medicina e la farmacologia. Le reti neurali artificiali sono strumenti potenti per affrontare questo problema, in quanto possono integrare informazioni provenienti da diverse fonti e apprendere le relazioni tra le sequenze e le strutture delle proteine. In questa ricerca, proponiamo un approccio contrastivo chiamato C3DPNet che combina modelli Graph Neural Networks (GNNs) per analizzare le strutture 3D e modelli di linguaggio naturale, come DNABERT-2, per elaborare le sequenze primarie. L'obiettivo è identificare le caratteristiche rilevanti che correlano le sequenze e le strutture delle proteine.

***Keywords*—** Bioinformatica Strutturale, Predizione delle Strutture Terziarie delle Proteine, GNNs, DNABERT, Genomica Computazionale, Apprendimento Contrastivo, Bioinformatica Computazionale.

1. Introduzione

Nel contesto della bioinformatica, la predizione accurata delle strutture proteiche riveste un ruolo cruciale per la comprensione dei processi biologici sottostanti, nonché per lo sviluppo di terapie mediche mirate e la progettazione di farmaci innovativi. L'avanzamento tecnologico ha aperto la strada a metodologie sempre più sofisticate per affrontare questa sfida, tra cui l'utilizzo di reti neurali artificiali e l'integrazione di informazioni provenienti da diverse fonti.

Le reti neurali artificiali, grazie alla loro capacità di apprendere da dati complessi e di identificare pattern intricati, si sono dimostrate strumenti potenti nell'analisi e nella predizione delle strutture proteiche. In particolare, l'utilizzo di Graph Neural Networks (GNNs),[28] come GCN (Graph Convolutional Network) [19], GraphSAGE (Graph Sample and Aggregation) [12], GAT (Graph Attention Network) [35], GIN (Graph Isomorphism Network) [38] e Graph U-Nets [11], ha consentito di modellare efficacemente le relazioni strutturali tra gli atomi all'interno delle proteine.

Parallelamente, l'elaborazione delle sequenze di DNA ha visto l'avvento di modelli avanzati come DNABERT [15] in grado di catturare relazioni semantiche nelle sequenze genetiche, tramite l'ausilio di language model come BERT [8]. Inoltre, l'approccio contrastivo, che mira a massimizzare la similarità tra campioni simili e minimizzare quella tra campioni diversi, si è dimostrato efficace nell'apprendimento di rappresentazioni informative nei dati molecolari.

Il presente lavoro si propone di esplorare un approccio che sfrutta la combinazione di informazioni semantiche e strutturali provenienti dai grafi molecolari e dalle sequenze di DNA per migliorare la predizione delle strutture proteiche. Utilizzando metodologie basate su reti neurali e approcci contrastivi, ci proponiamo di ottenere rappresentazioni complesse e informative dei dati molecolari, al fine di migliorare la nostra comprensione delle relazioni strutturali e funzionali all'interno delle proteine.

Le sezioni seguenti coprono il background teorico (Sec. 2), i lavori correlati (Sec. 3), la metodologia (Sec. 4), gli esperimenti e l'analisi dei risultati (Sec. 5), e le conclusioni con proposte per lavori futuri (Sec. 6).

[1]Angelo Nazzaro, Computer Science Department, Università degli Studi di Salerno

[2]Luigina Costante, Computer Science Department, Università degli Studi di Salerno

2. Background

Nella presente sezione verrà fornita una panoramica dei principi teorici e metodologici della bioinformatica e della biologia computazionale, con particolare attenzione al processo di protein folding, all'approccio contrastivo e al ruolo delle Graph Neural Networks (GNNs) nell'analisi dei dati non-lineari.

2.1. Protein Folding

Le proteine, sono composte da lunghe catene di amminoacidi, si estendono in media per diverse centinaia di elementi. La sequenza specifica di amminoacidi che compone una proteina costituisce la sua struttura primaria. Al momento della formazione, la proteina assume una configurazione tridimensionale, identificata come la sua struttura terziaria, che riveste un ruolo fondamentale in quanto esercita un'influenza significativa sulla funzione globale della proteina [21].

Il folding delle proteine rappresenta il processo tridimensionale mediante il quale una catena polipeptidica, costituente la struttura primaria di una proteina, assume la sua configurazione tridimensionale. Nonostante i notevoli progressi ottenuti nel campo della tecnologia e della ricerca, la capacità di prevedere con precisione la conformazione finale di una proteina a partire dalla sua sequenza primaria rimane un enigma scientifico.

Anfinsen ha dimostrato che la sequenza di amminoacidi di una proteina racchiude tutte le informazioni essenziali per determinare la sua corretta struttura tridimensionale. Tuttavia, persiste ancora una certa incomprensione riguardo al processo attraverso il quale l'informazione intrinseca nella catena polipeptidica si traduce in una struttura tridimensionale distintiva [7].

Attualmente, la determinazione della struttura terziaria di una proteina si configura come un processo oneroso e temporalmente impegnativo. Per affrontare questa complessa sfida scientifica, sono stati sviluppati diversi approcci sia in laboratorio che nell'ambito computazionale risolutivi del problema del protein folding; tra questi, emergono tre metodi principali di predizione: Homology Model, Fold Recognition e Ab initio.

Homology Model L'approccio Homlogy Model si basa sull'assunzione che proteine con sequenze simili presenteranno strutture altrettanto simili. Questo approccio si avvale dell'allineamento delle sequenze, ma è limitato significativamente dalla complessità computazionale e alla necessità di un allineamento preciso delle sequenze.

Fold Recognition Il Fold Recognition (o Threading), analizza la sequenza di amminoacidi di una struttura sconosciuta confrontandola con tutte le strutture note presenti in un database. Durante ogni scansione, viene assegnato un punteggio per valutare la compatibilità della sequenza con la struttura nota, generando così un insieme di possibili modelli tridimensionali.

Ab Initio L'approccio Ab Inito mira a costruire modelli tridimensionali partendo da zero, utilizzando unicamente la sequenza aminoacidica e le conoscenze teoriche chimico-fisiche. Nonostante richieda notevoli risorse computazionali, i metodi ab initio giocano un ruolo significativo nella ricerca sul protein folding, contribuendo alla comprensione di questo intricato fenomeno biologico [10].

L'attuale sfida scientifica di prevedere con precisione la conformazione finale delle proteine ha portato allo sviluppo di approcci computazionali innovativi, i quali costituiscono strumenti essenziali nella ricerca sul protein folding.

2.2. CASP

CASP [24] (Critical Assessment of Methods for Protein Structure Prediction) è un competizione biennale che valuta la ricerca nel campo del protein folding che coinvolge laboratori e gruppi di ricerca impegnati nella previsione della struttura proteica tridimensionale. I partecipanti sono divisi in due categorie, sperimentale e teorico-computazionale, a seconda del grado di intervento umano nelle previsioni. L'obiettivo principale è quello di individuare i metodi più efficaci, orientando la ricerca bioinformatica per le successive edizioni [32].

Inizialmente caratterizzato da tre approcci di previsione (Homology Model, Fold Recognition e Ab initio), CASP ha ampliato le sue categorie, con una partecipazione in crescita da gruppi da 34 in CASP1 a 216 in CASP5 [43]. L'introduzione di AlphaFold [17] ha segnato un punto di svolta significativo nella competizione e nella previsione della struttura delle proteine, dimostrando una straordinaria capacità di prevedere con precisione le strutture tridimensionali delle proteine. La sua introduzione ha suscitato un crescente interesse, sottolineando l'importanza dello sfruttare tecnologie avanzate per migliorare la precisione delle previsioni strutturali delle proteine.

2.3. Grafi

Un grafo è un oggetto astratto costituito da nodi (anche chiamati vertici) ed archi che connettono coppie di

nodi. Ogni nodo nel grafo rappresenta un'entità, mentre ogni arco rappresenta la relazione esistente tra i nodi coinvolti. Per esempio, in una struttura molecolare, ai nodi corrisponderebbero gli atomi e agli archi i legami chimici.

Più formalmente, un grafo G è una coppia (V,E), dove V è l'insieme dei vertici ed E è l'insieme degli archi. Ogni arco in E è rappresentato da una coppia (u,v) dove $u,v \in V$.

2.4. Graph Neural Networks

Le Graph Neutral Networks (GNNs), introdotte da Scarselli et al. [28], sono una classe di modelli di deep learning progettati specificamente per interpretare le relazioni non-lineari presenti all'interno dei grafi. Negli ultimi anni hanno acquisito notorietà per la loro efficacia nell'affrontare diverse sfide, che vanno dal rilevamento di oggetti alla scoperta di farmaci e alla classificazione delle molecole.

Message passing L'idea fondamentale alla base delle GNNs è quella di apprendere una rappresentazione per ciascun nodo nel grafo aggregando le informazioni dai suoi vicini. Questo processo prevede la definizione di un meccanismo di "message passing" in cui i messaggi contenenti caratteristiche rilevanti vengono scambiati lungo gli archi e poi combinati da ciascun nodo per aggiornare la propria rappresentazione delle caratteristiche.

In termini più formali, sia $G = (V,E)$ un grafo con matrice di caratteristiche dei nodi $\mathbf{X} \in \mathbb{R}^{|V| \times F}$, dove F è il numero di caratteristiche per ciascun nodo, e sia F_k la dimensione dimensionale di rappresentanza del nodo nello strato k-esimo, con $F_0 = F$; quindi il k-esimo livello di passaggio dei messaggi è definito come segue:

$$\mathbf{x}_u^{(k)} = \phi^{(k)} \left(\mathbf{x}_u^{(k-1)}, \bigoplus_{v \in N_u} \psi^{(k)}(\mathbf{x}_u^{(k-1)}, \mathbf{x}_v^{(k-1)}, \mathbf{e}_{uv}) \right)$$

Dove:

- $\mathbf{x}_u^{(k)}$ denota la rappresentazione del nodo u al k-esimo layer, con $\mathbf{x}_u^{(0)} = \mathbf{x}_u$ essendo il vettore delle caratteristiche corrispondente a u;
- $\psi^{(k)} \colon \mathbb{R}^{F_{k-1}} \times \mathbb{R}^{F_{k-1}} \times \mathbb{R}^{F'} \to \mathbb{R}^{F_{k-1}}$ è una funzione di trasformazione apprendibile che riceve le caratteristiche del nodo corrente $\mathbf{x}_u^{(k-1)}$, le caratteristiche vicine $\mathbf{x}_v^{(k-1)}$ e (facoltativamente) un vettore delle caratteristiche dell'arco $\mathbf{e}_{uv} \in \mathbb{R}^{F'}$ (ad esempio il peso dell'arco), producendo un nuovo vettore;
- $\psi^{(k)}(\mathbf{x}_u^{(k-1)}, \mathbf{x}_v^{(k-1)}, \mathbf{e}_{uv})$, rappresenta il messaggio inviato dal nodo v al nodo u attraverso l'arco (u,v);
- $\bigoplus$, denota una funzione di aggregazione invariante per permutazione (ad esempio somma, media, massimo) che combina i vari vettori di messaggi prodotti da $\psi^{(k)}$ e restituisce un vettore di caratteristiche aggregato $\bigoplus_{v \in N_u} \psi^{(k)}(\mathbf{x}_u^{(k-1)}, \mathbf{x}_v^{(k-1)}, \mathbf{e}_{uv})$;
- $\phi^{(k)} \colon \mathbb{R}^{F_{k-1}} \times \mathbb{R}^{F_{k-1}} \to \mathbb{R}^{F_k}$ è una funzione di trasformazione apprendibile che riceve le funzionalità del nodo corrente $\mathbf{x}_u^{(k-1)}$ e le funzionalità dei vicini aggregati $\bigoplus_{v \in N_u} \psi^{(k)}(\mathbf{x}_u^{(k-1)}, \mathbf{x}_v^{(k-1)}, \mathbf{e}_{uv})$ e producendo il vettore di caratteristiche del nodo aggiornato $\mathbf{x}_u^{(k-1)}$;

Il meccanismo di passaggio dei messaggi viene applicato in modo ricorsivo su più livelli, attraverso i quali le GNN combinano informazioni sia locali che globali, ottenendo una rappresentazione globale del grafo.

Per ottenere una comprensione completa del grafo, viene utilizzata una funzione di aggregazione globale, denominata READOUT o POOLING:

$$\mathbf{x}_G = READOUT\left(\{ \mathbf{x}_u^{(k)} \mid u \in V \} \right)$$

Qui, $\mathbf{x}_G$ indica la rappresentazione globale del grafo, ottenuta aggregando informazioni dai singoli nodi $\mathbf{x}_u^{(k)}$ tramite la funzione READOUT.

2.5. Graph Convolutional Neural Networks

Graph Convolutional Neural Networks (GCN), introdotte da Kipf et al. [19], traggono ispirazione dalle convolutional neural networks (CNN) e utilizzano concetti della teoria dei grafi spettrali per definire le operazioni di convoluzione sui grafi.

Sia $G = (V,E)$ un grafo non orientato con N nodi, una matrice di caratteristiche dei nodi $\mathbf{X}$ e una matrice di adiacenza $\mathbf{A} \in \mathbb{R}^{N \times N}$; il layer GCN è definito come segue:

$$\mathbf{X}^{(l)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{H}^{(l-1)} \mathbf{W}^{(l-1)} \right)$$

dove:

- σ denota una funzione di attivazione, come la funzione ReLU;
- $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ è la matrice delle adiacenze del grafo non orientato G con self-connections aggiunte, dove $\mathbf{I}_N$ è la matrice identità;
- $\mathbf{H}^{(l-1)} \in \mathbb{R}^{N \times D}$ è la matrice delle attivazioni nel layer l-esimo, con $\mathbf{H}^{(0)} = \mathbf{X}$;
- $\tilde{\mathbf{D}}$ è la matrice diagonale dei gradi rispetto a $\tilde{\mathbf{A}}$;

L'operazione di convoluzione spettrale sui grafi è definita come segue:

$$\mathbf{x} * g(\mathbf{w}) := \mathbf{U}\, g(\mathbf{w})\, \mathbf{U}^T \mathbf{x}$$

dove $\mathbf{U}$ è la matrice degli autovettori del grafo normalizzato Laplaciano $\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-1/2}\mathbf{A}\,\mathbf{D}^{-1/2}$, con una matrice diagonale di Λ e $\mathbf{U}^T\mathbf{x}$ essendo il grafo della trasformata di Fourier di $\mathbf{x}$, dove $\mathbf{x}$ è un segnale mono-dimensionale $\mathbf{x} \in \mathbb{R}^N$.

Come affermato da [19], calcolare l'equazione precedente è computazionalmente costoso, soprattutto per grafici di grandi dimensioni, poiché la sua complessità è $O(N^2)$. Per superare questa sfida, l'equazione può essere riformulata come:

$$\mathbf{x} * g(\mathbf{w}) \approx \mathbf{w}^T \left(\mathbf{I}_N + \tilde{\mathbf{D}}^{-1/2}\,\tilde{\mathbf{A}}\,\tilde{\mathbf{D}}^{-1/2}\right) \mathbf{x}$$

riducendo significativamente la sua complessità temporale a $O(|E|)$, ovvero lineare al numero di archi.

2.6. GraphSAGE

Proposto da Hamilton et al. [12], GraphSAGE (SAmple e aggreGatE) è un framework per l'embedding induttivo di nodi. A differenza degli approcci trasduttivi tradizionali che si basano sulla fattorizzazione matriciale dell'intero grafo, GraphSAGE adotta una strategia di campionamento e sfrutta le caratteristiche dei nodi per apprendere una funzione di embeddings che generalizza ai nodi non visti, sfruttando sia le caratteristiche strutturali che quelle di distribuzione.

Invece di addestrare vettori di embeddings distinti per ciascun nodo, GraphSAGE apprende un insieme di funzioni aggregatrici K, denotate come $\bigoplus_k$, $\forall k \in \{1,\ldots,K\}$, che aggregano informazioni dai nodi vicini, nonché un insieme di matrici di peso $\mathbf{W}_k$, $\forall k \in \{1,\ldots,K\}$ che vengono utilizzate per propagare le informazioni tra diversi livelli del modello. In questo contesto, K rappresenta il numero di salti, o profondità di ricerca, con cui ciascuna funzione aggregatrice aggrega le informazioni da un dato nodo.

Formalmente, la regola di aggiornamento dell'operatore GraphSAGE è definita come segue:

$$\mathbf{x}_u^{(l)} := \sigma\left(\mathbf{W_k} \cdot \mathrm{CONCAT}(\mathbf{x}_u^{(l-1)}, \bigoplus_{v \in N_u} \left(\mathbf{x}_v^{(l-1)} + \mathbf{b}\right)\right)$$

Dove:

- $\mathbf{x}_u^{(l)}$ denota l'embedding del nodo u aggiornato allo strato l-esimo;
- $\mathbf{b}$ è un vettore di bias apprendibile;
- σ è una funzione di attivazione non lineare;

L'intuizione alla base della regola di aggiornamento di GraphSAGE è che ad ogni iterazione, o profondità di ricerca, i nodi aggregano le informazioni dai loro vicini locali e, man mano che questo processo si ripete, i nodi ottengono in modo incrementale sempre più informazioni da ulteriori nodi del grafo.

2.7. Graph Attention Networks

Le Graph Attention Networks (GAT), introdotte da Veličković et al. [35], sono una potente famiglia di GNN che si basa sul meccanismo di self-attention.

Attenzione Il **meccanismo di attenzione**, introdotto per la prima volta da Bahdanau et al. [2] e successivamente migliorato da Vaswani et al. [34], abilita il modello a *prestare attenzione* a porzioni specifiche dei dati e ad assegnare loro maggiore importanza quando si effettuano previsioni. Ad esempio, nelle attività di natural language processing (NLP) come la traduzione automatica, il meccanismo di attenzione consente al modello di comprendere i significati delle parole nel loro contesto appropriato.

L'equazione dell'attenzione è composta da tre componenti principali:

1. **Vettore di query - Q**: Questo vettore viene utilizzato per confrontare un insieme di chiavi, producendo un punteggio;
2. **Vettore chiave - K**: Il vettore chiave è costituito da pesi che rappresentano l'importanza di ciascun elemento della query;
3. **Vettore valori - V**: Il vettore valori contiene i valori effettivi associati agli elementi di interesse;

L'operazione di attenzione può essere pensata come un processo di recupero, tracciando un'analogia tra il concetto di **chiave/valore/query** e i **sistemi di recupero**. Ad esempio, durante la ricerca di video su Youtube, il motore di ricerca mapperà la query (testo nella barra di ricerca) rispetto a un insieme di chiavi (titolo del video, descrizione, ecc.) associate ai video candidati nel proprio database, quindi presenterà i migliori -video corrispondenti (valori) [22].

Da un punto di vista matematico, il meccanismo di attenzione è una funzione a che prende come input una **matrice di query** $\mathbf{Q} \in \mathbb{R}^{q\times m}$, una raccolta di **coppie chiave-valore** $\mathbf{K} \in \mathbb{R}^{v\times m}$ e $\mathbf{V} \in \mathbb{R}^{v\times m}$, e genera una sequenza di valori ponderati q basati sui **punteggi di attenzione** $\mathbf{A} \in \mathbb{R}^{v\times m}$:

$$a : \mathbb{R}^{q\times m} \times \mathbb{R}^{v\times m} \times \mathbb{R}^{v\times m} \to \mathbb{R}^{q\times m}$$
$$\text{dove } (\mathbf{Q},\mathbf{K},\mathbf{V}) \to \mathbf{V}' = \mathbf{A}\cdot\mathbf{V}$$

Ogni α_{ij} viene calcolato tramite una funzione di compatibilità che valuta la correlazione tra la query q_i e la chiave corrispondente k_j.

Self-Attention La self-attention è un tipo specifico di meccanismo di attenzione in cui le chiavi, le query e i valori si riferiscono tutti alla stessa sequenza di input X. In questo scenario, la funzione di attenzione calcola le correlazioni tra ciascun elemento della sequenza e tutti gli altri elementi.

Multi-head Attention La Multi-head Attention (**MHA**) introduce il **parallelismo**. Consiste in h livelli di self-attentions o **heads**, che funzionano in parallelo. Ogni testa ha il suo insieme di trasformazioni lineari, rappresentate come $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$, con $i = 1,\ldots,h$.

Una volta calcolati gli output $\mathbf{h}_i = A_i(\mathbf{Q},\mathbf{K},\mathbf{V})$, vengono concatenati e ulteriormente moltiplicati per un'altra trasformazione di output $\mathbf{W}^O \in \mathbb{R}^{d_v\times d_{model}}$ proiettandoli in uno spazio di dimensione d_{model}:

$$\text{MHA}(\mathbf{Q},\mathbf{K},\mathbf{V}) := \text{CONCAT}(\mathbf{h}_1,\ldots,\mathbf{h}_h)\mathbf{W}^o$$

Questo meccanismo multi-testa consente a **ciascuna testa di concentrarsi su aspetti semantici specifici** della sequenza di input simultaneamente considerando il contesto globale senza fare affidamento sulla ricorrenza.

Graph Self-Attention Il layer di Graph Self-Attention esegue una somma ponderata dei vettori delle caratteristiche di un nodo e di *alcuni* dei suoi vicini:

$$e_{ij} = a\left(\mathbf{W}\vec{\mathbf{h}}_i, \mathbf{W}\vec{\mathbf{h}}_j\right)$$

dove $\mathbf{W} \in \mathbb{R}^{F'\times F}$ è una matrice dei pesi. Il coefficiente di attenzione viene calcolato solo per alcuni vicini del nodo al fine di iniettare la struttura del grafo nel meccanismo eseguendo masked attention. Per rendere i coefficienti facilmente confrontabili tra nodi diversi, vengono normalizzati su tutte le scelte di j utilizzando la funzione softmax:

$$\alpha_{ij} = \text{Softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k\in N_i}\exp(e_{ik})}$$

La graph self-attention può essere estesa per utilizzare il meccanismo multi-head attention:

$$\mathbf{x}_u^{(l)} := \Big\|_{k=1}^{K} \sigma\left(\sum_{v\in N_u} \alpha_{u,v}^k \mathbf{W}^{(l)k}\mathbf{x}_v^{(l-1)}\right)$$

Infine, nell'ultimo livello, gli output delle varie teste di attention vengono aggregati facendo la media:

$$\mathbf{x}_u^{(l)} := \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{v\in N_u} \alpha_{u,v}^k \mathbf{W}^{(l)k}\mathbf{x}_v^{(l-1)}\right)$$

Attraverso l'applicazione della self-attentions, le GATs sono altamente efficienti grazie alla possibilità di parallalizzare le operazioni su tutti gli archi e al calcolo delle caratteristiche di output su tutti i nodi, più interpretabili e possono essere applicate direttamente all'apprendimento induttivo.

2.8. Graph Isomorphism Networks

Le Graph Isomorphism Networks (GIN), introdotte da Xu et al. [38], rappresentano una categoria di modelli neurali appositamente concepiti per esaminare l'isomorfismo tra grafi. Tale concetto si definisce come una relazione tra due grafi che preserva la loro struttura e connettività, indipendentemente dall'etichettatura dei nodi. In altre parole, due grafi sono considerati isomorfi se esiste una biezione tra i loro insiemi di nodi che mantiene gli archi tra i nodi.

Il framework considerato trae ispirazione dalla stretta correlazione tra le GNN e il test di isomorfismo dei grafi di Weisfeiler-Lehman (WL), rinomato per la sua abilità nel distinguere una vasta gamma di grafi.

Test di Weisfeiler-Lehman Il problema dell'isomorfismo, caratterizzato dalla sua complessità, rimane irrisolto da un algoritmo polinomiale. Tuttavia, il Test di Weisfeiler-Lehman (WL) si è dimostrato uno strumento efficace e computazionalmente efficiente per distinguere una vasta classe di grafi, fatta eccezione per alcuni casi particolari come ad esempio i grafi regolari [5]. Questo test, nella sua forma unidimensionale conosciuta come "*naïve vertex refinement*", opera attraverso l'aggregazione delle etichette dei nodi e dei loro vicini, seguita dall'hashing delle etichette aggregate in nuove etichette univoche. Se, durante l'iterazione del test, le etichette dei nodi differiscono tra i due grafi, il test WL conclude che i grafi non sono isomorfi.

Sulla base del test WL, Shervashidze et al. [29] hanno introdotto il kernel di sottoalberi WL, che valuta la similarità tra i grafi utilizzando i conteggi delle etichette dei nodi in diverse iterazioni del test WL come

vettore di caratteristiche. L'etichetta di un nodo durante l'iterazione k del test WL rappresenta una struttura di sottoalbero di altezza k radicata nel nodo stesso; quindi, le caratteristiche del grafo considerate dal kernel di sottoalberi WL consistono principalmente nei conteggi di sottoalberi con differenti radici all'interno del grafo.

Il modello GIN è progettato per ampliare il concetto del test di Weisfeiler-Lehman (WL) e, come dimostrato in [38], rivendica una notevole capacità discriminativa tra le GNN. L'idea alla base prevede che una GNN aggiorni i vettori delle caratteristiche dei nodi per riflettere la struttura della rete e le caratteristiche dei nodi vicini. Questi vettori formano un multiset, consentendo la presenza di più istanze degli stessi elementi. Una GNN altamente efficiente mappa due nodi nella stessa posizione solo se hanno strutture di sottoalberi e caratteristiche identiche. Poiché le strutture sono definite ricorsivamente dai vicini dei nodi, l'attenzione si concentra sulla capacità di una GNN di associare due multisets alla stessa rappresentazione. Ciò richiede uno schema di aggregazione iniettivo, il quale viene astratto come una classe di funzioni sui multisets, per valutare la capacità di rappresentare funzioni iniettive.

Per gestire l'aggregazione dei vicini e la modellazione delle funzioni multiset, è stata sviluppata una teoria dei "*deep multisets*", che consiste nell'uso di reti neurali per parametrizzare funzioni multiset universali. Gli aggregatori di somma, in particolare, sono in grado di rappresentare funzioni iniettive e universali sui multisets. Utilizzando i multi-layer perceptrons (MLPs), è possibile modellare e apprendere le funzioni f e φ.

Nella pratica, si adotta un approccio che consiste nel combinare $f^{(k+1)}$ e $\varphi^{(k)}$ mediante un MLP, in quanto gli MLP sono capaci di rappresentare la composizione di funzioni. Introducendo il parametro ε, che può essere appreso o predefinito, GIN aggiorna le rappresentazioni dei nodi come segue:

$$h_v^{(k)} = MLP^{(k)}\left((1+\varepsilon^{(k)})\cdot h_v^{(k-1)} + \sum_{u\in N(v)} h_u^{(k-1)}\right).$$

Le rappresentazioni dei nodi apprese dalla rete sono utilizzabili direttamente per compiti quali la classificazione dei nodi e la previsione degli archi. Per la classificazione dei grafi, si propone una funzione "READOUT" che, a partire dagli embedding dei singoli nodi, genera l'embedding dell'intero grafo.

Un aspetto chiave del graph-level readout è che le rappresentazioni dei nodi diventano sempre più raffinate e globali all'aumentare del numero di iterazioni. È fondamentale un numero sufficiente di iterazioni per ottenere una buona capacità discriminativa, anche se in alcuni casi le caratteristiche delle prime iterazioni possono generalizzare meglio. Per catturare tutte le informazioni strutturali, vengono considerate le informazioni di tutte le profondità del modello. Questo approccio è implementato mediante un'architettura simile alle Jumping Knowledge Networks di Xu et al. [39], dove le rappresentazioni del grafo sono concatenate attraverso tutte le iterazioni di GIN come segue. Sia

$$r = \text{READOUT}\left(\left\{h_v^{(k)} \mid v \in G\right\}\right)$$

allora,

$$h_G = \text{CONCAT}\left(r \mid k = 0, 1, ..., K\right). \tag{1}$$

GIN sostituisce READOUT in Eq. (1) con la somma di tutte le caratteristiche dei nodi dalle stesse iterazioni, ottenendo una generalizzazione probabile del test WL e del kernel del sottoalbero WL.

2.9. Graph U-Nets

Graph U-Nets, introdotto da Gao et al. [11] e ispirata a U-Nets [27], è un'architettura encoder-decoder che cerca di replicare le operazioni di pooling e unpooling tipiche della CNN dal dominio delle immagini al dominio dei grafi. Nel loro lavoro, gli autori introducono nuove operazioni di graph pooling (gPool) e unpooling (gUnpool).

gPool Il layer gPool campiona in modo adattivo un sottoinsieme di nodi per formare un grafo nuovo ma più piccolo, utilizzando un vettore di proiezione addestrabile $\mathbf{p}$. La selezione dei nodi viene effettuata proiettando tutte le caratteristiche dei nodi su 1D, in particolare, dato un nodo i con il suo vettore di caratteristiche $\mathbf{x}_i$, la proiezione scalare di $\mathbf{x}_i$ su $\mathbf{p}$ è $y_i = \mathbf{x}_i\mathbf{p}/||\mathbf{p}||$, che misura il quantitativo di informazioni del nodo che possono essere conservate quando proiettate. Per preservare quante più informazioni possibili dal grafo originale, vengono selezionati solo i nodi con i valori di proiezione scalare più grandi.

La regola di propagazione del layer di pooling l è definita come:

$$\begin{aligned}\mathbf{y} &= \mathbf{X}^{(l)}\mathbf{p}^{(l)}/\|\mathbf{p}^{(\mathbf{l})}\|,\\ \text{idx} &= \text{rank}(\mathbf{y}, k),\\ \tilde{\mathbf{X}}^{(l)} &= \mathbf{X}^{(l)}(\text{idx})\end{aligned}$$

dove k è il numero di nodi selezionati nel grafo. Con gli indici selezionati idx, otteniamo il gate vectir $\tilde{\mathbf{y}}$

applicando la funzione sigmoidiana a ciascun elemento nel vettore di proiezione scalare estratto: $\tilde{\mathbf{y}} = \sigma(\mathbf{y}(idx))$ successivamente $\tilde{\mathbf{y}}$ viene utilizzato per controllare le informazioni dei nodi selezionati nel sottografo risultante, che viene restituito come:

$$\mathbf{A}^{(l+1)} = \mathbf{A}^{(l)}(idx, idx),$$
$$\mathbf{X}^{(l+1)} = \tilde{\mathbf{X}}^{(l)} \odot (\tilde{\mathbf{y}} 1_F^T)$$

dove $1_F \in \mathbb{R}^F$ e i suoi componenti sono impostati su 1.

gUnpool Il layer gUnpool esegue l'operazione inversa rispetto al livello gPool e ripristina il grafo alla sua struttura originale. Per raggiungere questo obiettivo, le posizioni dei nodi selezionati nel corrispondente layer gPool vengono registrate e utilizzate per ripristinare i nodi nelle loro posizioni originali. Formalmente, la regola di aggiornamento del layer gUnpool è definita come:

$$\mathbf{X}^{(l+1)} = \text{distribute}(0_{N \times C}, \mathbf{X}^l, \text{idx})$$

dove idx $\in \mathbb{Z}^{*k}$ contiene gli indici dei nodi selezionati nel corrispondente layer gPool. $\mathbf{X}^{(l)} \in R^{k \times F}$ sono la matrice delle caratteristiche del grafo corrente e $0_{N \times F}$ sono la matrice delle caratteristiche inizialmente vuota per il grafo corrente nuovo grafo. distribute$(0_{N \times F}, \mathbf{X}^{(l)}, idx)$ è l'operazione che distribuisce i vettori di riga in $\mathbf{X}^{(l)}$ nella matrice di caratteristiche $0_{N \times F}$ in base agli indici corrispondenti memorizzati in idx. In $\mathbf{X}^{(l+1)}$, i vettori riga con indici in *idx* vengono aggiornati dai vettori riga in $\mathbf{X}^{(l)}$, mentre gli altri vettori riga rimangono zero.

2.10. Differential Pooling

Differentiable Pooling (DIFFPOOL), proposto da Ying et al. [40], è un modulo differenziabile di pooling per grafi sviluppato al fine di superare i limiti delle GNN nel dedurre e aggregare informazioni in maniera gerarchica. Esso mira a creare un operatore di pooling analogo a quello trovato nelle CNN classiche, consentendo alle GNNs di conservare strutture gerarchiche e informazioni presenti all'interno di un grafo.

Per raggiungere questo obiettivo, DIFFPOOL apprende una matrice di assegnazione cluster $\mathbf{S}^{(l)}$ sui nodi del grafo utilizzando l'output di L moduli GNN. Tale matrice è cruciale per mappare i nodi del layer l ai cluster dello prossimo layer $l+1$. L'informazione gerarchica viene preservata attraverso delle matrici di embeddings $\mathbf{Z}^{(l)}$. Queste due matrici vengono calcolate utilizzando due GNN diverse, definite come segue:

$$\mathbf{Z}^{(l)} = \text{GNN}_{l,\text{embed}}(\mathbf{A}^{(l)}, \mathbf{X}^{(l)})$$
$$\mathbf{S}^{(l)} = \text{Softmax}\left(\text{GNN}_{l,\text{pool}}(\mathbf{A}^{(l)}, \mathbf{X}^{(l)})\right)$$

Dove:

- $\mathbf{A}^{(l)}$ è la matrice di adiacenza del grafo al layer l-esimo;
- $\mathbf{X}^{(l)}$ rappresenta la matrice di embeddings al layer l-esimo;

Dati questi input, il layer DIFFPOOL $(\mathbf{A}^{(l+1)}, \mathbf{X}^{(l+1)}) = \text{DIFFPOOL}(\mathbf{A}^{(l)}, \mathbf{X}^{(l)})$ semplifica il grafo di input, generando una nuova matrice di adiacenza semplificata $A^{(l+1)}$ e una nuova matrice di embeddings $X^{(l+1)}$ per ciascuno dei nodi/cluster nel nuovo grafo. $\mathbf{X}^{(l+1)}$ e $\mathbf{A}^{(l+1)}$ sono formalmente definiti come segue:

$$\mathbf{X}^{(l+1)} = {\mathbf{S}^{(l)}}^T \mathbf{Z}^{(l)}$$
$$\mathbf{A}^{(l+1)} = {\mathbf{S}^{(l)}}^T \mathbf{A}^{(l)} \mathbf{S}^{(l)}$$

2.11. DNABERT

DNABERT, proposto da Ji et al. [15], è un encoder bidirezionale pre-addestrato basato su BERT [8] in grado di acquisire una comprensione globale e trasferibile delle sequenze genomiche di DNA.

BERT BERT (Bidirection Encoder Representations from Transformers) è un modello di rappresentazione del linguaggio basato sull'architettura Transformer [34], sviluppato da Devlin et al. [8], progettato per pre-addestrare rappresentazioni bidirezionali profonde da testo non etichettato, condizionando congiuntamente sia il contesto sinistro che quello destro. Le caratteristiche chiave di BERT includono:

- **Contesto bidirezionale**: BERT è progettato per catturare il contesto bidirezionale mediante pre-addestramento su un ampio corpus di testo. A differenza dei modelli precedenti che elaboravano il testo in modo unidirezionale, BERT può considerare contemporaneamente sia il contesto sinistro che quello destro di una parola. Più specificamente, BERT è pre-addestrato su due compiti non supervisionati:

    1. **Masked LM**: per addestrare una rappresentazione bidirezionale profonda, una certa percentuale dei token di input in modo casuale

viene mascherata e quindi prevista. Ciò è necessario perché i modelli bidirezionali potrebbero altrimenti prevedere la parola target troppo facilmente, poiché ogni parola effettivamente "vedrebbe se stessa" in un contesto a più livelli.

2. **Next Sentence Prediction (NSP)**: alcuni compiti a valle, come la risposta alle domande e l'inferenza del linguaggio naturale, dipendono dalla comprensione della relazione tra due frasi, un concetto non direttamente affrontato dalla modellazione linguistica tradizionale. Per affrontare questo problema, BERT prevede una formazione preliminare per un compito binario chiamato previsione della frase successiva. Questo compito si concentra nel determinare se una coppia di frasi si susseguono naturalmente e possono essere derivate da qualsiasi corpus testuale monolingue.

- **Fine-tuned**: Dopo il pre-addestramento, BERT può essere "**fine-tunato**"[1] su attività NLP specifiche, come la classificazione del testo, la risposta alle domande e il riconoscimento delle entità denominate, mantenendo un'architettura unificata.

DNABERT segue lo stesso processo di addestramento di BERT. Richiede come input un insieme di sequenze rappresentate come token k-mer. Ogni sequenza è rappresentata come una matrice $\mathbf{M}$ incorporando ciascun token in un vettore numerico e le informazioni contestuali vengono catturate applicando la multi-head self-attention su $\mathbf{M}$. In maniera simile a BERT, anche DNABERT adotta il pre-training, tralasciando la fase di next sentence prediction. La lunghezza della sequenza viene modificata e il modello è costretto a prevedere k token contigui adattandolo allo scenario del DNA.

DNABERT-2 Nonostante DNABERT abbia dimostrato notevoli promesse e abbia ottenuto risultati all'avanguardia in vari compiti, l'utilizzo della strategia di tokenizzazione k-mer ha posto sfide significative nello sviluppo di modelli fondamentali di genomi di grandi dimensioni, principalmente a causa di inefficienze computazionali e campionarie. Per superare queste sfide, DNABERT-2 è stato introdotto da Zhou et al. [46]. Si tratta di una variante di DNABERT che adotta la metodologia Byte Pair Encoding (BPE), un metodo statistico per la costruzione dei token, invece della tokenizzazione k-mer. DNABERT-2 raggiunge prestazioni paragonabili ai modelli all'avanguardia pur essendo notevolmente più piccolo.

[1] Il fine-tuning adatta il modello per ottenere buone prestazioni su compiti specifici con set di dati più piccoli e specifici per attività.

2.12. Apprendimento contrastivo

L'apprendimento contrastivo è uno degli approcci più utilizzati per apprendere un rappresentazioni unificate. Il suo scopo è quello di apprendere gli attributi comuni condivisi all'interno delle classi di dati e gli attributi che differenziano una classe di dati da un'altra confrontando e contrastando dei campioni di dati.

L'essenza dell'apprendimento contrastivo prevede l'apprendimento dei parametri $\boldsymbol{\theta}$ di una funzione di embedding $f_{\boldsymbol{\theta}}$, in modo tale che le **coppie positive** della stessa classe mostrino una stretta vicinanza tra loro nello spazio di embedding, mentre quelli appartenenti a classi diverse siano posizionati più distanti tra loro. Di conseguenza, un modello di apprendimento contrastivo, dato un esempio di riferimento (o **anchor**) indicato con $\mathbf{I}^a$, mira a minimizzare la distanza d^+ con un esempio ***positivo*** $\mathbf{I}^+$, e massimizza la distanza d^- con un esempio ***negativo*** $\mathbf{I}^-$.

Il Graph Constrastive Learning (GCL) è un'estensione di questo framework nel dominio dei grafi. In questo contesto, l'obiettivo rimane lo stesso: vogliamo identificare e catturare relazioni semantiche significative tra istanze di grafi per ottenere rappresentazioni che riflettano la loro struttura e connessioni. Tuttavia, a differenza delle immagini o dei testi, i grafi presentano una struttura molto più complessa che richiede approcci specifici per estrarre caratteristiche e ottenere rappresentazioni [41].

3. Lavori Correlati

Nel contesto della crescente importanza della comprensione delle strutture proteiche, questa sezione esamina gli sforzi della comunità scientifica nel risolvere il problema del protein folding.

3.1. Metodi di predizione della struttura terziaria

I metodi di previsione della struttura terziaria sono numerosi e negli ultimi anni hanno subito una significativa evoluzione. L'attuale panorama scientifico è passato dai metodi tradizionali, come i raggi X e la Risonanza Magnetica Nucleare, ad algoritmi innovativi basati sull'intelligenza artificiale, riflettendo il progresso tecnologico e, soprattutto, la crescente comprensione delle complessità intrinsiche della previsione del struttura terziaria delle proteine [43].

AlphaFold Il lavoro presentato in [17] ha avuto un impatto significativo sul protein folding introducendo AlphaFold, una rete di deep learning in grado di prevedere direttamente le coordinate 3D di tutti gli atomi pesanti per una determinata proteina utilizzando la sequenza di aminoacidi primari e sequenze allineate di omologhi.

La rete si compone di due fasi principali. Innanzitutto, il *trunk* della rete elabora gli input attraverso strati ripetuti di un nuovo blocco di rete neurale basato sull'attenzione chiamato *Evoformer* per produrre due array: uno che rappresenta un MSA elaborato e l'altro che rappresenta le coppie di residui.

Il tronco della rete è seguito da un *modulo struttura* che introduce un'esplicita struttura 3D attraverso la rotazione e la traslazione di ciascun residuo della proteina. Inizializzate in uno stato banale, queste rappresentazioni subiscono un rapido processo evolutivo sviluppando e perfezionando una struttura proteica altamente accurata con dettagli atomici precisi. Un perfezionamento iterativo, chiamato *recycling*, viene applicato a tutta la rete applicando la loss finale agli output e ripassandoli ricorsivamente come input negli stessi moduli. Questo perfezionamento iterativo contribuisce notevolmente all'accuratezza con un minor tempo di addestramento aggiuntivo.

AlphaFold ha fatto passi da gigante nel miglioramento della precisione della previsione della struttura delle proteine, generando oltre 200 milioni di strutture nel corso degli anni; consolidandosi come una nuova pietra miliare nel campo della previsione della struttura 3D delle proteine.

RoseTTAFold RoseTTAFold, proposto da Baek et al. [1], costituisce un significativo progresso nel campo della predizione della struttura proteica, raggiungendo l'ambizioso obiettivo di prevedere con precisione le strutture proteiche esclusivamente sulla base della sequenza degli amminoacidi.

Guidati dai successi ottenuti da AlphaFold2 [17], i ricercatori hanno sviluppato RoseTTAFold con l'intento di migliorare l'accuratezza delle previsioni delle strutture proteiche e di promuovere gli avanzamenti nel design delle proteine. Attraverso iterazioni ed esperimenti, hanno adottato una "rete a due tracce", superando trRosetta [9] nell'ambito del CASP14. Inoltre, ampliando questa architettura con una terza traccia in 3D, hanno cercato di stabilire connessioni più strette tra sequenza, distanze, orientamenti residuo-residuo e coordinate atomiche.

L'architettura di RoseTTAFold prevede molteplici connessioni tra le tracce, consentendo alla rete di apprendere contemporaneamente le relazioni interne e tra sequenze, distanze e coordinate. Questo approccio si discosta da AlphaFold2, contribuendo a migliorare l'accuratezza delle previsioni della struttura proteica.

Il modello ha mostrato risultati promettenti nei test di benchmark su CAMEO, mostrando una maggiore precisione rispetto ad altri server di previsione della struttura. Inoltre, si è rivelato prezioso nella risoluzione di problemi di sostituzione molecolare in cristallografia a raggi X e nella generazione di modelli per proteine coinvolte in malattie.

RoseTTAFold consente inoltre la predizione diretta delle strutture dei complessi proteina-proteina basandosi solo sulla sequenza, riducendo notevolmente i tempi di calcolo e offrendo opportunità di modellazione per complessi proteici ancora sconosciuti.

OmegaFold Un ulteriore contributo chiave è quello di OmegaFold [36], un metodo di deep learning in grado di prevedere la struttura proteica a partire dalla sola sequenza primaria. OmegaFold sfrutta il meccanismo dell'attention per estrarre correlazioni evolutive dalle relazioni presenti nelle sequenze proteiche e incorpora intuizioni geometriche nell'architettura del Transformer [34] per modellare strutture proteiche 3D.

Il blocco base dell'architettura è OmegaPLM [36], un modello di linguaggio proteico (PLM) basato su deep transformer, addestrato su un ampia raccolta di sequenze proteiche non allineate e non etichettate. È in grado di catturare informazioni strutturali e funzionali codificate nell'amminoacido sequenze attraverso gli incorporamenti, che vengono poi immesse in Geoformer [45], una nuova rete neurale trasformatrice ispirata alla geometria, per distillare ulteriormente le relazioni strutturali e fisiche a coppie tra gli amminoacidi. Infine, un modulo strutturale prevede le coordinate 3D di tutti gli atomi pesanti.

OmegaFold ha dimostrato una notevole capacità nel prevedere con precisione le strutture proteiche su set di dati di riferimento come CASP e CAMEO. I risultati, ottenuti senza l'uso di MSA e basati su una singola sequenza di input, hanno mostrato un livello di accuratezza paragonabile o superiore ai metodi avanzati basati su MSA, come AlphaFold [17] e RoseTTAFold [1]. Inoltre, va sottolineato che OmegaFold ha migliorato significativamente i tempi di esecuzione rispetto ad approcci simili basati su MSA, rappresentando un progresso significativo nell'efficienza computazionale delle previsioni della struttura delle proteine.

3.2. Metodi contrastivi nella predizione della struttura proteica terziaria

Per quanto ne sappiamo, attualmente non esistono lavori che si concentrino specificamente sul GCL per

prevedere la struttura 3D di una proteina direttamente dalla sua sequenza primaria. Sebbene esistano lavori correlati con obiettivi simili, questi spesso enfatizzano aspetti come le proprietà geometriche della struttura 3D o i miglioramenti nelle relazioni espresse dai bordi all'interno del grafico delle proteine. Nella sezione seguente offriamo una panoramica dei lavori più importanti.

Hermosilla & Ropinski A causa del basso numero di strutture proteiche annotate che rende difficile l'addestramento dei modelli esistenti e inclini all'over-fitting, Hermosilla e Ropinski [13] hanno introdotto un nuovo framework di apprendimento contrastivo per la rappresentazione della struttura proteica 3D. In questo framework, il modello viene addestrato con l'obiettivo di massimizzare la somiglianza tra rappresentazioni derivate da sottostrutture all'interno della stessa proteina, minimizzando contemporaneamente la somiglianza tra sottostrutture originate da proteine diverse.

Lo studio dimostra l'efficacia del metodo proposto su una varietà di compiti e dataset, dimostrando che i modelli pre-addestrati utilizzano questo nuovo algoritmo mostrano un miglioramento significativo delle prestazioni in svariate attività, portando al raggiungimento di nuovi risultati all'avanguardia.

SimGRACE Simple framework for Graph Contrastive Learning (SimGRACE) [37] è un framework per il learning di rappresentazioni grafiche che si concentra sull'apprendimento contrastivo senza l'ausilio di tecniche di data augmentation. Esso è costituito da tre componenti principali: la perturbazione dell'encoder, la projection head e la loss contrastiva.

Nella fase di *perturbazione dell'encoder*, SimGRACE utilizza una tecnica basata su rumore gaussiano per aggiornare l'encoder della GNN. Questo approccio si differenzia da altre metodologie come BGRL [31] e MERIT [16], che utilizzano aggiornamenti basati su momentum e richiedono data augmentation.

Successivamente, SimGRACE adotta una *projection head* non lineare per trasformare le rappresentazioni del grafo in uno spazio latente, facilitando ulteriormente l'apprendimento delle relazioni tra le rappresentazioni.

Infine, SimGRACE utilizza una *loss contrastiva* normalizzata per massimizzare l'accordo tra coppie di rappresentazioni positive rispetto alle coppie negative. Questo approccio consente di migliorare la coerenza delle rappresentazioni apprese, facilitando il processo di classificazione e predizione.

SimGRACE si distingue per la sua semplicità ed efficacia, ottenendo risultati promettenti su diversi compiti di apprendimento supervisionato e semi-supervisionato, dimostrando la sua robustezza e flessibilità anche con quantità ridotte di dati di addestramento.

GearNet GeomEtry-Aware Relational Graph Neural Network (GearNet) [44] è uno structure-based encoder progettato per codificare informazioni spaziali integrando vari tipi di archi sequenziali o strutturali attraverso uno *sparse edge message* mechanism. Questo approccio facilita il passaggio di messaggi a livello degli archi, consentendo la modellazione esplicita delle interazioni tra gli archi.

Inoltre, GearNet impiega un framework di apprendimento contrastivo multi-vista con nuove funzioni di agumentation per scoprire la co-occorrenza correlata di sottostrutture proteiche e allineare le loro rappresentazioni nello spazio latente. Nello specifico, GearNet utilizza due schemi di mappatura, chiamati subsequence cropping e subspace cropping. Il subsequence cropping prevede il campionamento di residui casuali tra un residuo sinistro l e un residuo destro r per catturare domini proteici, mentre il subspace cropping viene utilizzato per identificare modelli strutturali spazialmente correlati. Seguendo SimCLR [6], GearNet utilizza una funzione di perdita contrastiva per massimizzare le informazioni reciproche tra queste visualizzazioni biologicamente correlate. Questo approccio mira a incorporare sottostrutture biologicamente correlate vicine l'una all'altra nello spazio latente, mantenendo distanti quelle non correlate.

GearNet si è dimostrato davvero promettente, ottenendo risultati comparabili sia sui compiti di previsione delle funzioni che su quelli di classificazione delle pieghe con metodi all'avanguardia, pur essendo addestrato su una quantità di dati molto inferiore.

4. Metodologia

La presente sezione espone la metodologia adottata nel contesto del presente lavoro, presentando il dataset impiegato, l'architettura del modello proposto e le relative strategie di addestramento e valutazione.

4.1. Dataset

Il dataset utilizzato in questa ricerca costituisce un sottoinsieme AlphaFold Protein Structure Database [33], riguardante specificamente i proteomi umani e comprendendo previsioni di circa 24.000 proteine.

Pre-processing I dati grezzi del dataset erano disponibili sia nei formati di file PDB che CIF. Abbiamo optato esclusivamente per i file PDB e successivamente abbiamo costruito i grafi corrispondenti. Tutta l'analisi e

la costruzione dei grafici dai file PDB sono state condotte utilizzando la libreria Graphein [14].

Le nostre decisioni riguardanti la costruzione del grafico sono ruotate attorno alla granularità dei nodi (cioè a livello di atomo o di residuo), alle caratteristiche da considerare per ciascun nodo e ai criteri per l'aggiunta degli archi. Queste considerazioni vengono approfondite nei paragrafi successivi.

Granularità Nel contesto della predizione delle strutture tridimensionali delle proteine tramite GNNs, la selezione della granularità del grafo costituisce una decisione di fondamentale importanza, poiché incide in modo significativo sulle performance del modello. Tale granularità si riferisce al livello di dettaglio con cui viene rappresentata la struttura della proteina nel grafo, che può essere a livello di residuo o a livello di atomo.

Un grafo a **livello di residuo** modella ogni residuo proteico come un nodo nel grafo, mentre i collegamenti tra tali nodi riflettono le interazioni tra i residui. Questo approccio semplifica la rappresentazione del grafo, riducendo la complessità computazionale, ma potrebbe comportare la perdita di informazioni cruciali relative alle interazioni atomiche all'interno di ciascun residuo.

Diversamente, un grafo a **livello di atomo** considera ogni atomo della proteina come un nodo del grafo, consentendo una rappresentazione più dettagliata delle interazioni atomiche presenti nella struttura proteica. Questo approccio mantiene un alto livello di dettaglio strutturale, permettendo al modello di cogliere relazioni più sottili tra gli atomi e di migliorare la precisione nella predizione della struttura tridimensionale della proteina.

Nel corso della nostra ricerca, abbiamo scelto di impiegare un grafo a livello di atomo per una serie di ragioni. Innanzitutto, tale approccio consente di acquisire dettagli strutturali cruciali che esercitano un'influenza significativa sulla conformazione tridimensionale della proteina e che potrebbe portare il modello ad avere una maggiore capacità di generalizzazione.

Caratteristiche dei nodi Per ogni nodo a nel grafo atom-level G, associamo un vettore di caratteristiche $\mathbf{x}_a$:

$$\mathbf{x_a} = [\mathbf{coords}_a \,||\, \mathbf{meiler}_a \,||\, \mathbf{expasy}_a \,||\, \mathbf{amino}_a]$$

Dove:

- $\mathbf{coords}_a := [x_a, y_a, z_a]$ è il vettore che descrive le coordinate spaziali dell'atomo a;
- $\mathbf{meiler}_a := \left[m_1^a, m_2^a, \ldots, m_7^a\right]$ denota il vettore di embeddings di Meiler [23] precedentemente calcolati per l'atomo a;
- $\mathbf{expasy}_a$ contiene le caratteristiche aminoacidiche che provengono dalla scala proteica EXPASY;
- $\mathbf{amino}_a$ è l'encoding one-hot dei tipi di amminoacidi associati a a;

La nostra selezione di attributi per la rappresentazione dei nodi è stata guidata da una serie di considerazioni chiave. Innanzitutto, l'inclusione delle coordinate tridimensionali è stata una scelta cruciale in quanto esse forniscono al modello una visione dettagliata della disposizione spaziale degli atomi nella molecola. Poiché la struttura tridimensionale è legata alle interazioni molecolari e alle proprietà funzionali delle proteine, la loro inclusione consente al modello di acquisire una comprensione più profonda e accurata dei processi biologici in gioco.

Inoltre, gli embeddings di Meiler sono stati integrati nella rappresentazione dei nodi per fornire al modello informazioni sulle caratteristiche psico-chimiche degli atomi. Questi embeddings catturano le sfumature delle interazioni atomiche e dei legami molecolari, consentendo al modello di esplorare in modo più dettagliato la struttura e le proprietà chimiche della proteina.

Le informazioni aminoacidiche provenienti dalla scala proteica EXPASY e il one-hot encoding dei tipi di amminoacidi sono stati inclusi per consentire al modello di discriminare tra diversi amminoacidi e di catturare le peculiarità della sequenza proteica. Poiché la composizione degli amminoacidi influenza direttamente la struttura e la funzione delle proteine, l'aggiunta di queste informazioni consente al modello di comprendere meglio le relazioni tra sequenza, struttura e funzione proteica.

La selezione e l'integrazione di queste diverse caratteristiche nei vettori dei nodi consentono al modello di ottenere una visione completa e dettagliata della struttura e della composizione della molecola proteica, fornendo una solida base per l'analisi e la previsione delle proprietà biologiche e funzionali.

Costruzione degli archi Nella fase di costruzione degli archi del grafo proteico, è stato impiegato il criterio dei k-nearest neighbors (KNN) [25], il quale stabilisce la vicinanza tra gli atomi al fine di creare connessioni tra quelli che sono prossimi nello spazio tridimensionale della molecola. Tale approccio tiene in considerazione sia le interazioni a lungo raggio sia quelle a corto raggio tra gli atomi, contribuendo così a una rappresentazione più completa della struttura proteica.

Inoltre, sono stati integrate diverse caratteristiche al grafo per arricchirlo di informazioni chimiche e strutturali dettagliate. Tra questi attributi sono inclusi i legami ad idrogeno, peptidici, aromatici e covalenti tra gli atomi,

al fine di riflettere le interazioni molecolari essenziali per la struttura e la funzione delle proteine.

I **legami ad idrogeno**, fondamentali per stabilizzare la struttura tridimensionale delle proteine e per numerose interazioni molecolari intracellulari, sono stati introdotti al fine di catturare tali interazioni e considerarle durante l'analisi e la predizione della struttura proteica.

I **legami peptidici**, che connettono gli amminoacidi all'interno delle catene polipeptidiche delle proteine, sono stati inclusi nel grafo per consentire al modello di comprendere la struttura primaria delle proteine e di valutare le relazioni tra gli amminoacidi lungo la catena polipeptidica.

Infine, i **legami aromatici**, caratterizzati dalla presenza di anelli aromatici nelle catene laterali degli amminoacidi, sono stati aggiunti alla struttura per fornire al modello una visione completa delle interazioni tra gli amminoacidi, le quali sono cruciali per la stabilità strutturale e le proprietà chimiche delle proteine.

L'utilizzo del criterio dei KNN per la costruzione degli archi, insieme alle informazioni chimiche e strutturali dettagliate, garantisce una rappresentazione esaustiva delle interazioni molecolari fondamentali per la comprensione della struttura proteica.

Suddivisione del dataset Per la divisione dei dati nei set di training, validazione e test, abbiamo scelto rispettivamente una suddivisione di 80%-10%-10%.

4.2. Architettura del modello

In questo lavoro, proponiamo C3DPNet (Contrastive 3D Protein Prediction Network) che combina diversi approcci per l'apprendimento delle rappresentazioni nei domini dei grafi e dei dati genetici, con l'obiettivo principale di integrare le informazioni semantiche e strutturali dei grafi molecolari e delle sequenze di DNA, facilitando così la realizzazione di compiti di classificazione e analisi predittiva.

C3DPNet si avvale di un graph-encoder basato su GNN per estrarre informazioni strutturali dai grafi e di un encoder testuale basato su DNABERT-2 [46] per acquisire informazioni semantiche e relazioni contestuali dalla sequenza di DNA. Per il graph-encoder, abbiamo esplorato diverse architetture: GraphSAGE, GCN, GIN, e GAT.

Le rappresentazioni estratte dagli encoder vengono poi avvicinate tramite l'impiego della loss contrastiva descritta nella sezione successiva. In Figura 1 è raffigurata l'architettura del modello proposto.

4.3. Loss

Nell'ambito dell'apprendimento delle rappresentazioni, una sfida fondamentale è massimizzare la capacità del modello di catturare informazioni rilevanti e discriminative. Ispirati dal successo di CLIP [26] nell'ambito della computer vision, abbiamo deciso di utilizzare la loss InfoNCE, data la sua efficacia nel promuovere l'apprendimento di rappresentazioni contestualmente informative e significative, che si sono dimostrate cruciali in molte applicazioni di computer vision e NLP.

$$L_{InfoNCE} = -\mathbb{E}_S \left[\log \frac{f_\theta(s_{t+k}, c_t)}{\sum_{s_j \in S} f_\theta(s_j, c_t)} \right] \tag{2}$$

Qui, f_θ è una funzione che mappa la rappresentazione del contesto c_t in uno spazio di caratteristiche, in particolare in un passo temporale futuro k, S è l'insieme di campioni, che include un campione positivo (solitamente l'ancoraggio) e campioni negativi $N-1$. La funzione InfoNCE misura essenzialmente quanto bene la rappresentazione del contesto c_t si allinea con la futura incorporazione di funzionalità $f_\theta(c_t)$ rispetto agli incorporamenti di funzionalità di campioni negativi. Incoraggia il modello a massimizzare la somiglianza tra il contesto e le rappresentazioni future riducendo al minimo la somiglianza con i campioni negativi. Ciò aiuta ad apprendere rappresentazioni informative e significative per le attività specifiche.

5. Esperimenti e Risultati

Configurazione sperimentale Gli esperimenti sono stati eseguiti su una workstation comprendente 1 NVIDIA RTX 4000, 256 GB di RAM e una CPU a 16 core da 2.3 GHz. Il sistema operativo e le piattaforme di deep learning utilizzate sono state: Ubuntu 20.0.4 e PyTorch 2.1.0.

Iper-parametri Gli iper-parametri sono stati impostati utilizzando l'ottimizzazione bayesiana [30]. La Tabella 1 mostra i valori e gli intervalli considerati per gli iper-parametri in tutte le famiglie di modelli. Per il learning rate scheduler, abbiamo adottato una strategia lineare. Per l'ottimizzatore abbiamo esplorato le opzioni tra Adam [18], AdamW [20] e SGD. La batch size è stata fissata a 8 per tutti gli esperimenti.

Varianti dei Modelli Di seguito sono riportate le migliori combinazioni di iper-parametri trovate per ogni tipologia di GNN considerata.

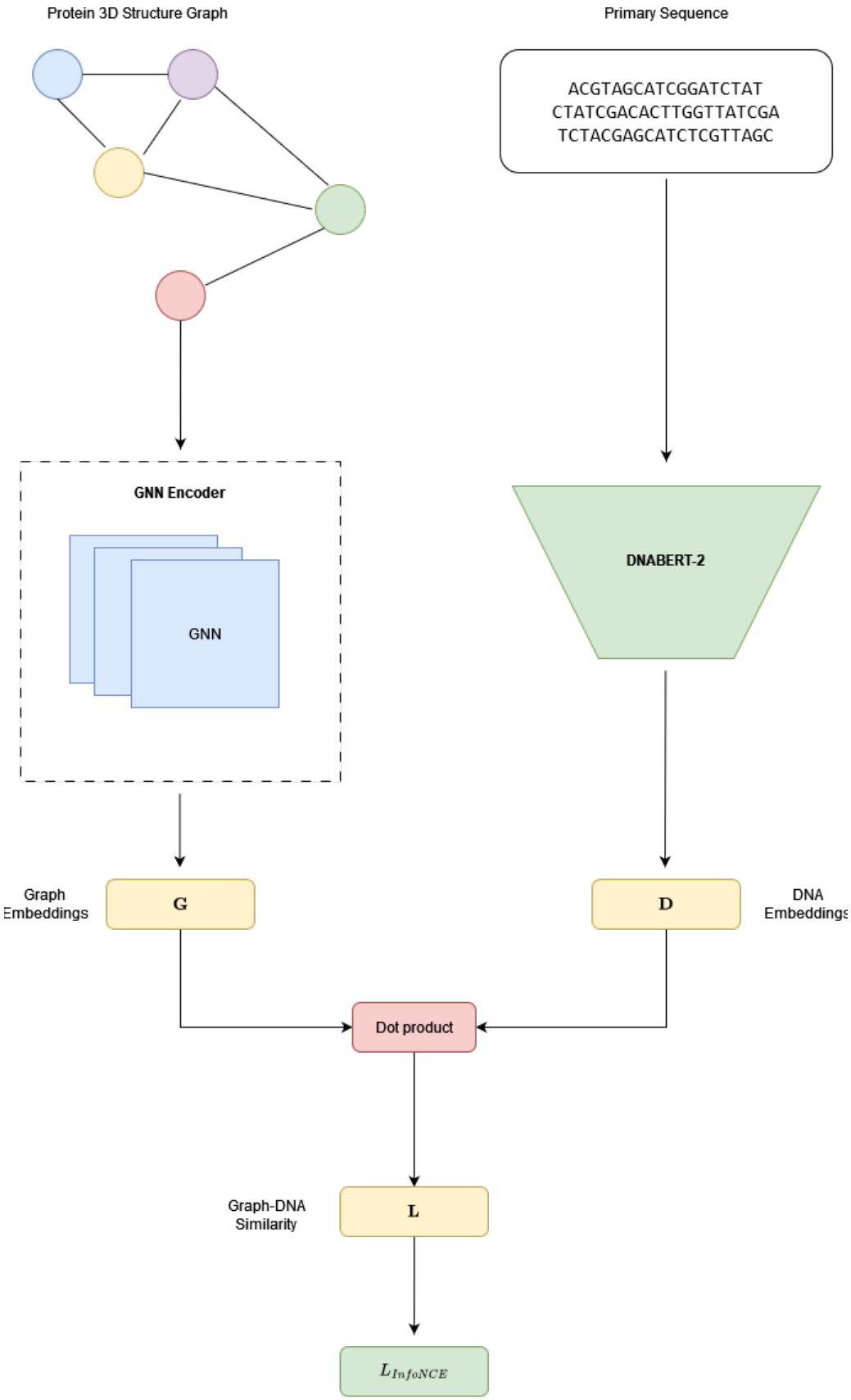

Figura 1: Overview di C3DPNet.

Tabella 1: Valori e intervalli degli iper-parametri considerati durante l'allenamento.

Learning Rate		Weight Decay		Layers	
Min	**Max**	**Min**	**Max**	**Min**	**Max**
1e−8	1e−5	0.01	0.1	3	6

- GraphSAGE: learning rate impostato a 6.25e−6, weight decay pari a 0.038, 6 layers e l'ottimizzatore AdamW.
- GCN: learning rate impostato a 3.34e−6, weight decay pari a 0.098, 6 layers e l'ottimizzatore AdamW.
- GIN: learning rate impostato a 9.28e−6, weight decay pari a 0.083, 6 layers e l'ottimizzatore AdamW.
- GAT: learning rate impostato a 5.12e−6, weight decay pari a 0.062, 3 layers e l'ottimizzatore AdamW.

Tabella 2: Risultati delle valutazione dei modelli sul dataset impiegato durante il training. I valori riportati sono i valori intra-batch. GSAGE denota il modello GraphSAGE. I risultati migliori sono riportati in grasseto.

Model	Accuracy	Precision	F1-Score	Recall
GSAGE	**0.778**	**0.685**	**0.715**	**0.778**
GIN	0.373	0.211	0.253	0.373
GAT	0.731	0.617	0.653	0.731
GCN	0.580	0.450	0.489	0.580

Risultati In Tabella 2, sono riportati i risultati della valutazione delle varianti di GNNs considerate sul dataset impiegato per l'addestramento. Dai risultati, emerge che il modello migliore si ottiene utilizzando un GNN encoder basato su GraphSAGE.

Particolarmente interessanti sono le capacità generalizzanti dimostrate da ogni modello, specialmente nelle prime epoche di addestramento, come evidenziato nelle Figure 2, 3, 4, 5, dove è possibile notare una performance migliore nella fase di validazione rispetto a quella di addestramento. Si suppone che ciò sia dovuto all'integrazione di informazioni tra l'avvicinamento della sequenza primaria e terziaria, che elevano le capacità generalizzanti dei modelli.

Tabella 3: Confronto tra GraphSAGE e C3DPNet su ENZYMES [3]. GSAGE denota il modello GraphSAGE. I risultati migliori sono riportati in grasseto.

Model	Accuracy	Precision	F1-Score	Recall
C3DPNet	**0.542**	**0.613**	**0.572**	**0.667**
GSAGE	0.508	0.376	0.402	0.499

Al fine di constatare l'efficacia dell'integrazione dell'approccio contrastivo, abbiamo effettuato un confronto tra il nostro modello migliore e la corrispettiva baseline GNN. Nel caso specifico, abbiamo confrontato l'encoder GraphSAGE di C3DPNet con la sua corrispettiva baseline. I due modelli sono stati addestrati e valutati sul dataset ENZYMES [3]. In Tabella 3 sono riportati i risultati del confronto; dove GraphSAGE baseline evidenzia performance quasi casuali, il nostro encoder dimostra capacità discriminanti superiori, evidenziando l'efficacia dell'approccio contrastivo impiegato.

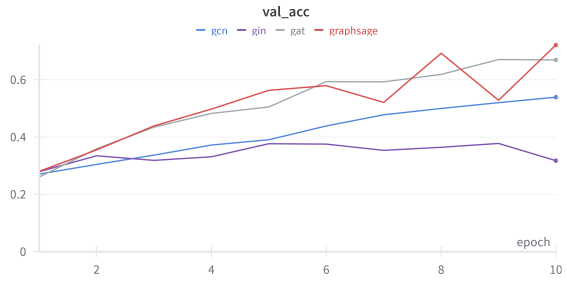

Figura 2: Validation accuracy.

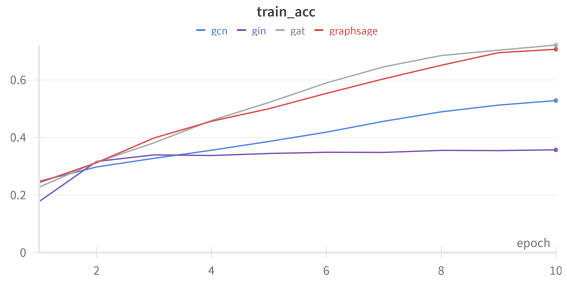

Figura 3: Training accuracy.

6. Conclusioni

La presente ricerca ha introdotto l'architettura C3DPNet, un approccio che ha esaminato l'efficacia

Figura 4: Validation loss.

Figura 5: Training loss.

dell'approccio contrastivo nella predizione della struttura tridimensionale delle proteine. Questo modello integra modelli GNN per la modellazione della struttura tridimensionale e modelli di linguaggio naturale, come DNABERT-2 [46], per estrapolare informazioni dalla sequenza primaria. Diversi modelli GNN sono stati esplorati, evidenziando una notevole capacità di generalizzazione del modello, sottolineata da una maggiore accuratezza nella fase di validazione, aumentando la fiducia nella capacità del modello di fornire previsioni precise e affidabili sulla struttura tridimensionale delle proteine. Infine, è stato condotto un confronto tra il graph-encoder di C3DPNet e la corrispettiva baseline su un compito di classificazione. Il nostro metodo ha dimostrato capacità discriminanti superiori, mentre la controparte ha evidenziato tendenze pseudo-casuali, confermando l'efficacia dell'approccio contrastivo.

Nonostante i risultati promettenti ottenuti, si riconosce la necessità di ulteriori sforzi volti a ottimizzare le prestazioni del modello. Per le prospettive future, si prevede di:

- Esplorare l'impiego di altre architetture di GNN, come DiffPool[40] e Graph-UNets[11], per valutarne l'impatto sulle prestazioni predittive;
- Utilizzare tecniche di data augmentation per aumentare il numero di coppie positive per grafo;
- Utilizzare altre loss contrastive, come la Sigmoid loss [42], potrebbe contribuire a ottimizzare ulteriormente la fase di addestramento, migliorandone l'accuratezza e la stabilità;
- Considerare l'utilizzo di modelli pre-addestrati specifici per le proteine, come ProteinBERT [4], per sfruttare appieno le informazioni contenute nei dati proteici;
- Estendere la valutazione del modello su una vasta gamma di dataset proteici, al fine di valutarne la capacità di generalizzazione e la robustezza in contesti diversi.

Riferimenti bibliografici

[1] Minkyung Baek et al. «Accurate prediction of protein structures and interactions using a three-track neural network». In: *Science* 373.6557 (2021), pp. 871–876. DOI: 10.1126/science.abj8754. eprint: https://www.science.org/doi/pdf/10.1126/science.abj8754. URL: https://www.science.org/doi/abs/10.1126/science.abj8754.

[2] Dzmitry Bahdanau, Kyunghyun Cho e Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL].

[3] Karsten M Borgwardt e et al. «Protein function prediction via graph kernels». In: *Bioinformatics* 21.Suppl 1 (2005), pp. i47–i56. DOI: 10.1093/bioinformatics/bti1007.

[4] Nadav Brandes et al. «ProteinBERT: a universal deep-learning model of protein sequence and function». In: *Bioinformatics* 38.8 (2022), pp. 2102–2110.

[5] Jin-Yi Cai, Martin Fürer e Neil Immerman. «An optimal lower bound on the number of variables for graph identification». In: *Combinatorica* 12.4 (1992), pp. 389–410.

[6] Ting Chen et al. «A simple framework for contrastive learning of visual representations». In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[7] Peter JT Dekker et al. «La grande scienza. Il folding delle proteine all'interno della cellula». In: ().

[8] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[9] Zongyang Du et al. «The trRosetta server for fast and accurate protein structure prediction». In: *Nature protocols* 16.12 (2021), pp. 5634–5651.

[10] Martino Fantato. «Un Algoritmo Genetico per la predizione della configurazione spaziale del nucleo idrofobico di proteine». In: ().

[11] Hongyang Gao e Shuiwang Ji. *Graph U-Nets*. 2019. arXiv: 1905.05178 [cs.LG].

[12] William L. Hamilton, Rex Ying e Jure Leskovec. *Inductive Representation Learning on Large Graphs*. 2018. arXiv: 1706.02216 [cs.SI].

[13] Pedro Hermosilla e Timo Ropinski. «Contrastive representation learning for 3d protein structures». In: *arXiv preprint arXiv:2205.15675* (2022).

[14] Arian R. Jamasb et al. «Graphein - a Python Library for Geometric Deep Learning and Network Analysis on Protein Structures and Interaction Networks». In: *bioRxiv* (2021). DOI: 10.1101/2020.07.15.204701. eprint: https://www.biorxiv.org/content/early/2021/10/12/2020.07.15.204701.full.pdf. URL: https://www.biorxiv.org/content/early/2021/10/12/2020.07.15.204701.

[15] Yanrong Ji et al. «DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome». In: *Bioinformatics* 37.15 (feb. 2021), pp. 2112–2120. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab083. eprint: https://academic.oup.com/bioinformatics/article-pdf/37/15/2112/53921029/btab083.pdf. URL: https://doi.org/10.1093/bioinformatics/btab083.

[16] Ming Jin et al. «Multi-scale contrastive siamese networks for self-supervised graph representation learning». In: *arXiv preprint arXiv:2105.05682* (2021).

[17] John Jumper et al. «Highly accurate protein structure prediction with AlphaFold». In: *Nature* 596.7873 (2021), pp. 583–589.

[18] Diederik P. Kingma e Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[19] Thomas N. Kipf e Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: 1609.02907 [cs.LG].

[20] Ilya Loshchilov e Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG].

[21] Richard Maclin e Jude W Shavlik. «Using knowledge-based neural networks to improve algorithms: Refining the Chou-Fasman algorithm for protein folding». In: *Machine Learning* 11 (1993), pp. 195–215.

[22] Navaneeth Malingan. *Attention Mechanism in Deep Learning*. 2023. URL: https://www.scaler.com/topics/deep-learning/attention-mechanism-deep-learning/.

[23] Jens Meiler et al. «Generation and Evaluation of Dimension-Reduced Amino Acid Parameter Representations by Artificial Neural Networks». In: *Molecular Modeling Annual* 7.9 (set. 2001), pp. 360–369. ISSN: 0948-5023. DOI: 10.1007/s008940100038.

URL: https://doi.org/10.1007/s008940100038.

[24] John Moult et al. «Critical assessment of methods of protein structure prediction (CASP)—round x». In: *Proteins: Structure, Function, and Bioinformatics* 82 (2014), pp. 1–6.

[25] Leif E Peterson. «K-nearest neighbor». In: *Scholarpedia* 4.2 (2009), p. 1883.

[26] Alec Radford et al. «Learning transferable visual models from natural language supervision». In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.

[27] Olaf Ronneberger, Philipp Fischer e Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[28] Franco Scarselli et al. «The Graph Neural Network Model». In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605.

[29] Nino Shervashidze et al. «Weisfeiler-lehman graph kernels.» In: *Journal of Machine Learning Research* 12.9 (2011).

[30] Jasper Snoek, Hugo Larochelle e Ryan P. Adams. *Practical Bayesian Optimization of Machine Learning Algorithms*. 2012. arXiv: 1206.2944 [stat.ML].

[31] Shantanu Thakoor et al. «Bootstrapped representation learning on graphs». In: *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*. 2021.

[32] Valeria Thiella. «Valutazione della predizione della struttura proteica: l'iniziativa CASP». In: (2010).

[33] Mihaly Varadi et al. «AlphaFold Protein Structure Database in 2024: providing structure coverage for over 214 million protein sequences». In: *Nucleic Acids Research* 52.D1 (2024), pp. D368–D375.

[34] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[35] Petar Veličković et al. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML].

[36] Ruidong Wu et al. «High-resolution de novo structure prediction from primary sequence». In: *BioRxiv* (2022), pp. 2022–07.

[37] Jun Xia et al. «SimGRACE: A Simple Framework for Graph Contrastive Learning without Data Augmentation». In: *Proceedings of the ACM Web Conference 2022*. WWW '22. ACM, apr. 2022. DOI: 10.1145/3485447.3512156. URL: http://dx.doi.org/10.1145/3485447.3512156.

[38] Keyulu Xu et al. *How Powerful are Graph Neural Networks?* 2019. arXiv: 1810.00826 [cs.LG].

[39] Keyulu Xu et al. «Representation learning on graphs with jumping knowledge networks». In: *International conference on machine learning*. PMLR. 2018, pp. 5453–5462.

[40] Rex Ying et al. *Hierarchical Graph Representation Learning with Differentiable Pooling*. 2019. arXiv: 1806.08804 [cs.LG].

[41] Yuning You et al. «Graph contrastive learning with augmentations». In: *Advances in neural information processing systems* 33 (2020), pp. 5812–5823.

[42] Xiaohua Zhai et al. *Sigmoid Loss for Language Image Pre-Training*. 2023. arXiv: 2303.15343 [cs.CV].

[43] Qing Zhang, Stella Veretnik e Philip E Bourne. «Overview of structural bioinformatics». In: *Bioinformatics Technologies*. Springer, 2005, pp. 15–44.

[44] Zuobai Zhang et al. «Protein representation learning by geometric structure pretraining». In: *arXiv preprint arXiv:2203.06125* (2022).

[45] Jiaxuan Zhao et al. «GeoFormer: A Geometric Representation Transformer for Change Detection». In: *IEEE Transactions on Geoscience and Remote Sensing* 61 (2023), pp. 1–17.

[46] Zhihan Zhou et al. *DNABERT-2: Efficient Foundation Model and Benchmark For Multi-Species Genome*. 2023. arXiv: 2306.15006 [q-bio.GN].