Contrastive 3D Protein Prediction

Angelo Nazzaro[1], Luigina Costante[2]

March 5, 2024

Abstract

Predicting the three-dimensional (3D) structure of a protein from its primary sequence is a major challenge for biology, medicine and pharmacology. Artificial neural networks are powerful tools for tackling this problem, as they can integrate information from different sources and learn the relationships between protein sequences and structures. In this research, we explore a contrastive approach called C3DPNet, that combines Graph Neural Networks (GNNs) models to analyze 3D structures and natural language models, such as DNABERT-2, to process primary sequences. The goal is to identify relevant features that correlate protein sequences and structures.

***Keywords*—** Structural Bioinformatics, Prediction of Protein Tertiary Structures, GNNs, DNABERT, Computational Genomics, Contrastive Learning, Computational Bioinformatics.

1. Introduction

In the context of bioinformatics, the accurate prediction of protein structures plays a crucial role in understanding underlying biological processes, as well as in developing targeted medical therapies and designing innovative drugs. Technological advancement has paved the way for increasingly sophisticated methodologies to address this challenge, including the use of artificial neural networks and the integration of information from different sources.

Artificial neural networks, thanks to their ability to learn from complex data and identify intricate patterns, have proven to be powerful tools in the analysis and prediction of protein structures. In particular, the use of Graph Neural Networks (GNNs),[28] such as GCN (Graph Convolutional Network) [19], GraphSAGE (Graph Sample and Aggregation) [12] , GAT (Graph Attention Network) [35], GIN (Graph Isomorphism Network) [38] and Graph U-Nets [11], allowed us to effectively model the structural relationships between the atoms inside proteins.

At the same time, the processing of DNA sequences has seen the advent of advanced models such as DNABERT [15] capable of capturing semantic relationships in genetic sequences, through the help of language models such as BERT [8]. Furthermore, the contrastive approach, which aims to maximize the similarity between similar samples and minimize that between different samples, has proven effective in learning informative representations in molecular data.

The present work aims to explore an approach that exploits the combination of semantic and structural information coming from molecular graphs and DNA sequences to improve the prediction of protein structures. Using neural network-based methodologies and contrastive approaches, we aim to obtain complex and informative representations of molecular data, in order to improve our understanding of structural and functional relationships within proteins.

The following sections cover the theoretical background (Sec. 2), related works (Sec. 3), methodology (Sec. 4), experiments and the analysis of the results (Sec. 5), and the conclusions with proposals for future work (Sec. 6).

2. Background

In this section we will provide an overview of the theoretical and methodological principles of bioinformatics and computational biology, with particular attention to the protein folding process, the contrastive approach

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

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

and the role of Graph Neural Networks (GNNs) in the analysis of non-linear data.

2.1. Protein Folding

Proteins are composed of long chains of amino acids, extending on average for several hundred elements. The specific sequence of amino acids that make up a protein constitutes its primary structure. Upon formation, the protein assumes a three-dimensional configuration, identified as its tertiary structure, which plays a fundamental role as it exerts a significant influence on the global function of the protein [21].

Protein folding represents the three-dimensional process by which a polypeptide chain, constituting the primary structure of a protein, assumes its three-dimensional configuration. Despite notable advances in technology and research, the ability to accurately predict the final conformation of a protein from its primary sequence remains a scientific enigma.

Anfinsen demonstrated that the amino acid sequence of a protein contains all the essential information to determine its correct three-dimensional structure. However, some misunderstanding still persists regarding the process by which intrinsic information in the polypeptide chain translates into a distinctive three-dimensional structure [7].

Currently, the determination of the tertiary structure of a protein is an expensive and time-consuming process. To address this complex scientific challenge, various approaches have been developed both in the laboratory and in the computational field to solve the protein folding problem; among these, three main prediction methods emerge: Homology Model, Fold Recognition and Ab initio.

Homology Model The Homlogy Model approach is based on the assumption that proteins with similar sequences will have similarly similar structures. This approach takes advantage of sequence alignment, but is significantly limited by computational complexity and the need for precise sequence alignment.

Fold Recognition Fold Recognition (or Threading) analyzes the sequence of amino acids of an unknown structure by comparing it with all the known structures present in a database. During each scan, a score is assigned to evaluate the compatibility of the sequence with the known structure, thus generating a set of possible three-dimensional models.

Ab Initio The Ab Inito approach aims to build three-dimensional models starting from scratch, using only the amino acid sequence and theoretical chemical-physical knowledge. Despite requiring considerable computational resources, ab initio methods play a significant role in protein folding research, contributing to the understanding of this intricate biological phenomenon [10].

The current scientific challenge of accurately predicting the final conformation of proteins has led to the development of innovative computational approaches, which constitute essential tools in protein folding research.

2.2. CASP

CASP [24] (Critical Assessment of Methods for Protein Structure Prediction) is a biennial competition that evaluates research in the field of protein folding involving laboratories and research groups engaged in three-dimensional protein structure prediction. Participants are divided into two categories, experimental and theoretical-computational, depending on the degree of human intervention in the predictions. The main objective is to identify the most effective methods, guiding bioinformatics research for subsequent editions [32].

Initially featuring three prediction approaches (Homology Model, Fold Recognition, and Ab initio), CASP has expanded its categories, with participation growing from 34 groups in CASP1 to 216 in CASP5 [43].

The introduction of AlphaFold [17] marked a significant turning point within the competition and in protein structure prediction, demonstrating an extraordinary ability to accurately predict the three-dimensional structures of proteins. Its introduction sparked heightened interest, emphasizing the importance of leveraging advanced technologies to enhance the precision of protein structural predictions.

2.3. Graphs

A graph is an abstract object consisting of nodes (also called vertices) and edges connecting pairs of nodes. Each node in the graph represents an entity, while each edge represents the relantionship existing between connected nodes. For instance, in a molecular structure, nodes would correspond to atoms and edges to chemical bonds.

More formally, a graph G is a pair (V,E) where V is the set of vertices and E is the set of edges. Each edge in E is represented by a pair (u,v) where $u,v \in V$.

2.4. Graph Neural Networks

Graph Neural Networks (GNNs), first introduced by Scarselli et al. [28], are a class of deep learning methods specifically designed for handling graph-structured data. Over recent years, they have gained notoriety for their effectiveness in addressing diverse challenges, ranging from object detection to drug discovery and molecule classification.

Message passing The fundamental idea behind GNNs is to learn a representation for each node in the graph by aggregating information from its local neighborhoods. This process involves defining a message-passing mechanism where messages containing relevant features are exchanged along the edges and then combined at each node to update its own feature representation.

In more formal terms, let $G = (V, E)$ be a graph with node feature matrix $\mathbf{X} \in \mathbb{R}^{|V| \times F}$, where F is the number of features for each node, and let F_k be the node repesentation dimensionality in the k-th layer, with $F_0 = F$; then the k-th message passing layer is defined as follows:

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

where:

- $\mathbf{x}_u^{(k)}$ denotes the representation of the u node at the k-th layer, with $\mathbf{x}_u^{(0)} = \mathbf{x}_u$ being the feature vector corresponding to u;
- $\psi^{(k)} : \mathbb{R}^{F_{k-1}} \times \mathbb{R}^{F_{k-1}} \times \mathbb{R}^{F'} \to \mathbb{R}^{F_{k-1}}$ is a learnable transformation function receiving the current node features $\mathbf{x}_u^{(k-1)}$, the neighbour features $\mathbf{x}_v^{(k-1)}$ and (optionally) an edge feature vector $\mathbf{e}_{uv} \in \mathbb{R}^{F'}$ (e.g. edge weight), producing a new vector;
- $\psi^{(k)}(\mathbf{x}_u^{(k-1)}, \mathbf{x}_v^{(k-1)}, \mathbf{e}_{uv})$, represents the message passed from the node v to the node u through the (u, v) edge;
- $\bigoplus$, denotes a permutation-invariant aggregation function (e.g. sum, mean, max) which combines the various message vectors produced by $\psi^{(k)}$ and outputs an aggregated feature vector $\bigoplus_{v \in N_u} \psi^{(k)}(\mathbf{x}_u^{(k-1)}, \mathbf{x}_v^{(k-1)}, \mathbf{e}_{uv})$;
- $\phi^{(k)} : \mathbb{R}^{F_{k-1}} \times \mathbb{R}^{F_{k-1}} \to \mathbb{R}^{F_k}$ is a learnable transformation function receiving the current node features $\mathbf{x}_u^{(k-1)}$ and the aggregated neighbours features $\bigoplus_{v \in N_u} \psi^{(k)}(\mathbf{x}_u^{(k-1)}, \mathbf{x}_v^{(k-1)}, \mathbf{e}_{uv})$ and producing the updated node feature vector $\mathbf{x}_u^{(k-1)}$;

The message passing mechanism is recursively applied across multiple layers, through which GNNs combine both local and global information, effectively capturing higher-order dependencies within the graph. To obtain a comprenshive understanding of the graph, a global aggregation function, denoted as READOUT or POOLING, is employed:

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

Here, $\mathbf{x}_G$ signifies the global representation of the graph, obtained by aggregating information from individual nodes $\mathbf{x}_u^{(k)}$ through the READOUT function.

2.5. Graph Convolutional Neural Networks

Graph Convolutional Neural Networks (GCNs), introduced by Kipf et al. [19], draw inspiration from convolutional neural networks (CNNs) and employ concepts from spectral graph theory to define convolution operations on graphs.

Let $G = (V, E)$ be an undirected graph with N nodes, a node feature matrix $\mathbf{X}$ and an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$; the GCN layer is defined as follows:

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

where:

- σ denotes an activation function, such as the ReLU function;
- $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ is the adjancey matrix of the undirected graph G with added self-connections, with $\mathbf{I}_N$ being the identity matrix;
- $\mathbf{H}^{(l-1)} \in \mathbb{R}^{N \times D}$ is the matrix of activations in the l-th layer, with $\mathbf{H}^{(0)} = \mathbf{X}$;
- $\tilde{\mathbf{D}}$ is the diagonal degree matrix with respect to $\tilde{\mathbf{A}}$;

The spectral graph convolution is defined as follows:

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

where $\mathbf{U}$ is the matrix of eigenvectors of the normalized graph Laplacian $\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$, with a diagonal matrix of Λ and $\mathbf{U}^T \mathbf{x}$ being the graph Fourier transform of $\mathbf{x}$, where $\mathbf{x}$ is a mono-dimensional signal $\mathbf{x} \in \mathbb{R}^N$.

As stated by [19], computing the previous equation is computationally expensive, especially for large graphs, since its complexity is $O(N^2)$. To overcome this challenge, the equation can be reformulated as:

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

significantly reducing its time complexity to $O(|E|)$, i.e. linear in the number of edges.

2.6. GraphSAGE

Proposed by Hamilton et al. [12], GraphSAGE (SAmple and aggreGatE) is a general framework for inductive node embeddings. Unlike traditional transductive approchaes which rely on matrix factorization of the entire graph, GraphSAGE adopts a *fixed-size* sampling strategy and leverages node features to learn an embedding function that generalizes to unseen nodes, harnessing both structural and distribution features.

Rather than training distinct embedding vectors for each node, GraphSAGE learns a set of K aggregator functions, denoted as $\bigoplus_k$, $\forall k \in \{1, \ldots, K\}$, which aggregate information from node neighbors, as well as a set of weight matrices $\mathbf{W}_k$, $\forall k \in \{1, \ldots, K\}$ which are used to propagate information between different layers of the model. In this context, K represents the number of hops, or search depth, at which each aggregator function aggregates information from a given node.

Formally, the GraphSAGE operator update rule is defined as follows:

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

where:

- $\mathbf{x}_u^{(l)}$ denotes the embedding of the u node updated at the l-th layer;
- $\mathbf{b}$ is a learnable bias vector;
- σ is a non-linear activation function;

The intuition behind GraphSAGE update rule is that at each iteration, or search depth, nodes aggregate information from their local neighbors, and as this process iterates, nodes incrementally gain more and more information from further reaches of the graph.

2.7. Graph Attention Networks

Graph Attention Networks (GATs), introduced by Veličković et al [35], are a powerful family of GNNs relying on the self-attention mechanism.

Attention The **attention mechanism**, first introduced by Bahdanau et al. [2] and subsequently improved by Vaswani et al. [34], enables the model to *pay attention* to specific portions of the data and assign them greater importance when making predictions. For instance, in natural language processing (NLP) tasks like machine translation, the attention mechanism empowers the model to comprehend the meanings of words in their proper context.

The attention equation is made up of three primary components:

1. **Query vector - Q**: This vector is employed to match against a set of keys, yielding a score;
2. **Key vector - K**: The key vector consists of weights that represent the importance of each query element;
3. **Value vector - V**: The value vector contains the actual values associated with the elements of interest;

The attention operation can be thought of as a retrieval process, drawing an analogy between the **key/value/query** concept and **retrieval systems**. For example, when searching for videos on Youtube, the search engine will map the query (text in the search bar) against a set of keys (video title, description, etc.) associated with candidate videos in their database, then present the best-matched videos (values) [22].

From a mathematical perspective, the attention mechanism is a function a that takes as input a **query matrix** $\mathbf{Q} \in \mathbb{R}^{q \times m}$, a collection of **key-value pairs** $\mathbf{K} \in \mathbb{R}^{v \times m}$ and $\mathbf{V} \in \mathbb{R}^{v \times m}$, and generates a sequence of q weighted values based on the **attention scores** $\mathbf{A} \in \mathbb{R}^{v \times m}$:

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

Each α_{ij} is calculated through a compatibility function that assesses the correlation between the query q_i and the corresponding key k_j.

Self-Attention Self-attention is a specific type of attention mechanism where the keys, queries and values all refer to the same input sequence X. In this scenario, the attention function calculates correlations between each element in the sequence and all other elements.

Multi-Head Attention The multi-head attention (**MHA**) mechanism introduces **parallelism**. It consists of h self-attention layers or **heads**, that work in parallel.

Each head has its set of linear transformations, represented as $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$, with $i = 1, \ldots, h$.

Once the outputs $\mathbf{h}_i = A_i(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ are computed, they are concatenated and further multiplied by another output transformation $\mathbf{W}^O \in \mathbb{R}^{d_v \times d_{model}}$ projecting them into a space of dimension d_{model}:

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

This multi-head mechanism allows **each head to focus on specific semantic aspects** of the input sequence simultaneously while considering the global context without relying on recurrence.

Graph Self-Attention The Graph Self-Attention layer performs a weighted sum of the feature vectors of a node and of *some* of its neighboring nodes:

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

where $\mathbf{W} \in \mathbb{R}^{F' \times F}$ is a weight matrix. The attention coefficient is calculated only for some of the node's neighboorhoods in order to inject graph structure into the mechanism by performing masked attention. To make coefficients easily comparable across different nodes, they are normalized across all choices of j using the softmax function:

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

The graph self-attention can be extended to employ the multi-head attention mechanism:

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

Finally, in the last prediction layer, the outputs of the various attention heads are aggregated by averaging:

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

Through the application of self-attention, GATs are highly efficient due possibility to parallalize the operations across all edges and the computation of the output features across all nodes, more interpretable and can be directly applied to inductive learning.

2.8. Graph Isomorphism Networks

Graph Isomorphism Networks (Gin), introduced by Xu et al. [38], represent a category of neural models specially designed to examine isomorphism between graphs. This concept is defined as a relationship between two graphs that preserves their structure and connectivity, regardless of the tire of the nodes. In other words, two graphs are considered isomorphic if there is a bifier among their sets of nodes that maintains the edges between the nodes.

The framework considered draws inspiration from the close correlation between the GNN and the isomorphism test of the graphs of Weisfeiler-Lehman (WL), renowned for its ability to distinguish a wide range of graphs.

Test of Weisfeiler-Lehman The problem of isomorphism, characterized by its complexity, remains unsolved by a polynomial algorithm. However, the Weisfeiler-Lehman (WL) test has proven to be an effective and computationally efficient tool to distinguish a vast class of graphs, except for some particular cases such as the regular graphs [5]. This test, in its unidimensional form known as " Textit Naïve Vertex Refinement", works through the aggregation of the labels of the knots and their neighbors, followed by the hashing of the labels aggregated in new univocal labels. If, during the iteration of the test, the nodes labels differ between the two graphs, the WL test concludes that graphs are not isomorphs.

Based on the WL test, Shervashidze et al. [29] have introduced the sub-embezzling kernel WL, which evaluates the similarity between the graphs using the counts of the knots labels in different iterations of the WL test as a vector of characteristics. The label of a node during the k of the WL testing represents a underground structure K rooted in the node itself; Therefore, the characteristics of the graph considered by the subalborne kernel WL mainly consist of the counts of subalberings with different roots within the graph.

The GIN model is designed to expand the concept of the Weisfeiler-Lehman (WL) test and, as demonstrated in [38], claims a significant discriminative capacity between the GNN. The idea at the base provides that a GNN updates the vectors of the characteristics of the nodes to reflect the structure of the network and the characteristics of the nearby nodes. These vectors form a multiset, allowing the presence of multiple instances of the same elements. A highly efficient GNN map two nodes in the same position only if they have underground

structures and identical characteristics. Since the structures are defined as recursively by the neighbors of the nodes, the attention focuses on the ability of a GNN to associate two multisets with the same representation. This requires an injective aggregation scheme, which is abstract as a class of functions on multi-meshes, to evaluate the ability to represent injective functions.

To manage the aggregation of neighbors and the modeling of the multiset functions, a theory of "*deep multisets*" has been developed, which consists in the use of neural networks to parameterize universal multiset functions. The sum aggregators, in particular, are able to represent injective and universal functions on multisets. Using the multi-layer perceptrons (MLPs), it is possible to model and learn the functions f and φ.

In practice, an approach is adopted that consists in combining $f(k+1)$ and $\varphi^{(k)}$ by means of an MLP, as the MLP are capable of representing the Composition of functions. By introducing the parameter ε, which can be learned or predefined, Gin updates the representations of the nodes as follows:

$$h_v^{(k)} = \text{MLP}^{(k)}\left((1+\varepsilon^{(k)})\cdot h_v^{(k-1)} + \sum_{u\ inn(V)} h_u^{(k-1)}\right)$$

The representations of the nodes learned from the network can be used directly for tasks such as the classification of the nodes and the forecast of the arches. For the classification of graphs, a "readout" function is proposed which, starting from the embedding of individual nodes, generates the embedding of the entire graph.

A key aspect of the Graph-Level Repodout is that the representations of the knots become increasingly refined and global as the number of iterations increases. A sufficient number of iterations is essential to obtain a good discriminatory capacity, even if in some cases the characteristics of the first iterations can generalize better. To capture all the structural information, the information of all the depths of the model is considered. This approach is implemented through an architecture similar to the Jumping Knowledge Networks of Xu et al. [39], where the representations of the graph are concatenated through all the gin iterations as follows. Is

$$R = \text{READOUT}\left(\left\{h_v^{(k)} \mid v\ ing\right\}\right)$$

At that time,

$$H_g = \text{Concat}\,(r|\,k = 0.1,...,k) \tag{1}$$

GIN replaces Readout in EQ. (1) with the sum of all the characteristics of the nodes from the same iterations, obtaining a probable generalization of the WL test and the sub-tree kernel WL.

2.9. Graph U-Nets

Graph U-Nets, introduced by Gao et al. [11] and inspired by U-Nets [27], is an encoder-decoder architecture that seeks to replicate the typical CNN's pooling and unpooling operations from the image domain to the graph domain. In their work, the authors introduce novel graph pooling (gPool) and unpooling (gUnpool) operations.

gPool The gPool layer adaptively samples a subset of nodes to form a new but smaller graph, employing a trainable projection vector $\mathbf{p}$. The node selection is done by projecting all node features to 1D, specifically, given a node i with its feature vector $\mathbf{x}_i$, the scalar projection of $\mathbf{x}_i$ on $\mathbf{p}$ is $y_i = \mathbf{x}_i\mathbf{p}/||\mathbf{p}||$, that measures how much information of the node can be retained when projected. To preserve as much information as possible from the original graph, only the nodes with the largest scalar projection values are selected.

The layer-wise propagation rule of the graph pooling layer l is defined as:

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

where k is the number of nodes selected in the graph. With selected indices idx, we obtain the gate vector $\tilde{\mathbf{y}}$ by applying sigmoid to each element in the extracted scalar projection vector: $\tilde{\mathbf{y}} = \sigma(\mathbf{y}(idx))$ afterward $\tilde{\mathbf{y}}$ is employed to control the information of selected nodes in the resulting sub-graph, which is outputted as:

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

where $1_F \in \mathbb{R}^F$ and its components are set to 1.

gUnpool The gUnpool layer performs the inverse operation to the gPool layer and restores the graph to its original structure. To achieve this, the locations of the nodes selected in the corresponding gPool layer are recorded and used to restore the nodes back to their original positions. Formally, the gUnpool layer update rule is defined as:

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

where idx $\in \mathbb{Z}^{*k}$ contains indices of selected nodes in the corresponding gPool layer. $\mathbf{X}^{(l)} \in R^{k\times F}$ are the feature

matrix of the current graph, and $0_{N\times F}$ are the initially empty feature matrix for the new graph. distribute($0_{N\times F}$, $\mathbf{X}^{(l)}, idx$) is the operation that distributes row vectors in $\mathbf{X}^{(l)}$ into $0_{N\times F}$ feature matrix according to their corresponding indices stored in idx. In $\mathbf{X}^{(l+1)}$, row vectors with indices in idx are updated by row vectors in $\mathbf{X}^{(l)}$, while other row vectors remain zero.

2.10. Differential Pooling

Differentiable Pooling (DIFFPOOL), proposed by Ying et al. [40], is a differentiable graph pooling module developed to address the limitations of GNNs in inferring and aggregating information hierarchically. It aims to create an analogous pooling operator to that found in classical CNNs, allowing GNNs to retain hierarchical structures and information within a graph.

To achieve this, DIFFPOOL learns a cluster assignment matrix $\mathbf{S}^{(l)}$ over the nodes of the graph using the output of L stacked GNN modules. This assignment is crucial for mapping nodes at layer l to clusters in the coarsened layer $l+1$. The hierarchically information is preserved through embedding matrices $\mathbf{Z}^{(l)}$. These two matrices are computed using two differenet GNNs, defined as follows:

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

where:

- $\mathbf{A}^{(l)}$ is the adjancey matrix of the graph at the l-th layer;
- $\mathbf{X}^{(l)}$ represents the embedding matrix at the l-th layer;

Given these inputs, the DIFFPOOL layer $(\mathbf{A}^{(l+1)}, \mathbf{X}^{(l+1)}) = \text{DIFFPOOL}(\mathbf{A}^{(l)}, \mathbf{X}^{(l)})$ coarsens the input graph, generating a new coarsened adjacency matrix $A^{(l+1)}$ and a new matrix of embeddings $X^{(l+1)}$ for each of the nodes/clusters in this coarsened graph. $\mathbf{X}^{(l+1)}$ and $\mathbf{A}^{(l+1)}$ are formally defined as follows:

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

2.11. DNABERT

DNABERT, proposed by Ji et al. [15], is a pre-trained bidirectional encoder based on BERT [8] able to capture global and transferrable understanding of genomic DNA sequences.

BERT BERT, short for Bidirectional Encoder Representations from Transformers, is a language representation Transformer [34] model, developed by Devlin et al. [8], designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. The key charachteristics of BERT include:

- **Bidirectional Context**: BERT is designed to capture bidirectional context by pre-training on a large corpus of text. Unlike earlier models that processed text in a unidirectional manner, BERT can consider both the left and right context of a word simultaneously. More specifically, BERT is pre-trained on two unsupervised tasks:

  1. **Masked LM**: In order to train a deep bidirectional representation, some percentage of the input tokens at random is masked and then predicted. This is necessary because bidirectional models could otherwise predict the target word too easily, as each word would effectively "see itself" in a multi-layered context.
  2. **Next Sentence Prediction (NSP)**: Certain downstream tasks, such as Question Answering and Natural Language Inference, depend on understanding the relationship between two sentences, a concept not directly addressed by traditional language modeling. To tackle this, BERT includes pre-training for a binary task called next sentence prediction. This task focuses on determining if a pair of sentences naturally follows each other and can be derived from any monolingual text corpus.

- **Fine-tuning**: After pre-training, BERT can be **fine-tuned**[1] on specific NLP tasks, such as text classification, question answering, and named entity recognition, keeping a unified architecture across different tasks.

DNABERT follows the same training process as BERT. It first takes a set of sequences represented as k-mer tokens as input. Each sequence is represented as a matrix $\mathbf{M}$ by embedding each token into a numerical vector and contextual information is captured by performing multi-head self-attention on $\mathbf{M}$. Similar to BERT, DNABERT also adopts pre-training, omitting the next sentence prediction phase. The sequence length is

[1] Fine-tuning adapts the model to perform well on specific tasks with smaller, task-specific datasets.

adjusted and the model is forced to predict contiguous k tokens adapting it to the DNA scenario.

DNABERT-2 Despite DNABERT demonstrating remarkable promise and achieving state-of-the-art results across various tasks, the utilization of the k-mer tokenization strategy posed significant challenges in developing large genome foundational models, primarily due to computational and sample inefficiencies. To overcome this challenges, DNABERT-2 was introduced by Zhou et al. [46]. It is a variant of DNABERT adopting Byte Pair Encoding (BPE), a statistics-based methodology for the token construction, instead of k-mer tokenization. DNABERT-2 achieves comparable performance to the state-of-the-art models while being considerably smaller.

2.12. Contrastive Learning

Contrastive learning is one of the most commonly used approaches to learn unified representation. It aims to learn the common attributes shared within data classes and the attributes that differentiate one data class from another by *comparing* and *contrasting* data samples.

The fundamental contrastive learning framework involves learning the parameters $\boldsymbol{\theta}$ of an embedding function $f_{\boldsymbol{\theta}}$, such that **positive couples** from the same class exhibit close proximity to each other in the embedding space, while those belonging to different classes are positioned farther apart from each other. Consequently, a contrastive learning model, given a reference (or **anchor**) example denoted by $\mathbf{I}^a$, aims to minimize the distance d^+ with a ***positive*** example $\mathbf{I}^+$, and maximize the distance d^- with a ***negative*** example $\mathbf{I}^-$.

Graph Contrastive Learning (GCL) is an extension of this framework to the graph domain. In this context, the objective stays the same: we want to identify and capture meaningful semantic relantionship between graph instances to obtain rappresentations that reflect their structure and connections. However, unlike images or texts, graphs exhibit a much more complex structure that require specific approaches for extracting features and obtaining representations [41].

3. Related Works

In the context of the growing importance of understanding protein structures, this section reviews the efforts of the scientific community to solve the problem of protein folding.

3.1. Tertiary Structure Prediction

Tertiary structure prediction methods are numerous, and in recent years, they have undergone significant evolution. The current scientific panoram has transitioned from traditional methods, such as X-rays and Nuclear Magnetic Resonance, to innovative algorithms based on artificial intelligence, reflecting the technological progress and, more importantly, the growing understanding of the intrinsic complexities of the prediction of the tertiary structure of proteins [43].

AlphaFold The work presented in [17] has significantly impacted the protein folding by introducing AlphaFold, a deep learning network able to directly predict the 3D coordinates of all heavy atoms for a given protein using the primary amino acid sequence and aligned sequences of homologues.

The network comprises two main stages. First, the *trunk* of the network processes the inputs through repeated layers of a nove attention-based neural network block called *Evoformer* to produce two arrays: one representing a processed MSA and the other representing residue pairs.

The trunk of the network is followed by a *structure module* that introduces an explicit 3D structure through rotation and translation of each residue of the protein. Initialized in a trivial state, these representation undergone a rapid evolutionary process developing and refining a highly accurate protein structure with precise atomic details. An iterative refinement, called *recycling*, is applied throughout the whole networkby by applying the final loss to outputs and then feeding the outputs recursively into the same modules. This iterative refinement contributes markedly to accuracy with minor extra training time.

AlphaFold has made substantial strides in improving the accuracy of protein structure prediction, generating over 200 million predicted structures over the years. It has become a fundamental cornerstone in the field of protein structure prediction.

RoseTTAFold Rosettafold, proposed by Baek et al. [1], constitutes a significant progress in the field of prediction of the protein structure, achieving the ambitious objective of precisely providing the protein structures exclusively only on the basis of the sequence of amino acids.

Guided by the successes achieved by Alphafold2 [17], the researchers developed Rosettafold with the intention of improving the accuracy of the provisions of protein structures and promoting advancements in the design of proteins. Through iterations and experiments,

they have adopted a "two -track network", overcoming Trorosetta [9] as part of the CASP14. Furthermore, by expanding this architecture with a third trace in 3D, they tried to establish closer connections between sequence, distances, residual-residuous guidelines and atomic coordinates.

Rosettafold's architecture provides multiple connections between the tracks, allowing the network to at the same time learn internal relationships and between sequences, distances and coordinates. This approach differs from Alphafold2, helping to improve the accuracy of the provisions of the protein structure.

The model showed promising results in the benchmark tests on Cameo, showing greater precision than other forecast servers of the structure. In addition, it turned out to be precious in the resolution of molecular replacement problems in X-ray crystallography and in the generation of protein models involved in diseases.

Rosettafold also allows the direct prediction of the structures of protein-protectin complexes based only on the sequence, significantly reducing calculation times and offering modeling opportunities for still unknown protein complexes.

OmegaFold Another key contribution is that of OmegaFold [36], a deep learning method able to predict high-resolution protein structure from a single primary sequence alone. OmegaFold levarages the attention mechanism in order to extract correlations from evolutionary relationships present in protein sequences and incorporates geometric intuition into the Transformer architecture [34] to model 3D protein structures.

The base block of the architecture is OmegaPLM [36], a deep transformer-based protein language model (PLM), trained on a large collection of unaligned and unlabeled protein sequences. It is able to capture structural and functional information encoded in the amino-acid sequences through the embeddings, which are then fed into Geoformer [45], a new geometry-inspired transformer neural network, to further distill the structural and physical pairwise relationships between amino acids. Lastly, a structural module predicts the 3D coordinates of all heavy atoms.

OmegaFold has demonstrated remarkable ability in accurately predicting protein structures on benchmark datasets such as CASP and CAMEO. The results, obtained without the use of MSA and based on a single input sequence, showed a level of accuracy comparable or superior to advanced MSA-based methods, such as AlphaFold [17] and RoseTTAFold [1]. Furthermore, it should be emphasized that OmegaFold significantly improved execution times compared to similar MSA-based approaches, representing a significant advance in the computational efficiency of protein structure predictions.

3.2. Constrastive Tertiary Structure Prediction

To the best of our knowledge, there are currently no works that specifically concentrate on GCL for predicting the 3D structure of a protein directly from its primary sequence. While there are related works with similar objectives, these often emphasize aspects such as the geometric properties of the 3D structure or enhancements in the relationships expressed by the edges within the protein graph. In the following section, we offer an overview over the most prominent works.

Hermosilla & Ropinski Due to the low number of annotated protein structures making the training of existing models difficult and prone to over-fitting, Hermosilla and Ropinski [13] introduced a new representation learning framework for 3D protein structure. In this innovative framework, the model is trained with the objective of maximizing the similarity between representations derived from substructures within the same protein, while concurrently minimizing the similarity between substructures originating from different proteins.

The study demonstrates the efficacy of proposed method on a variety of tasks and datasets, showing that pre-trained models using this novel algorithm exhibit significantly improved task performance, leading to the attainment of new state-of-the-art results across various tasks

SimGRACE Simple framework for Graph Contrastive Learning (SimGRACE) [37] is a framework for the learning of graphic representations that focuses on contrastive learning without the help of data augmentation techniques. It consists of three main components: the disturbance of the encoder, the projection head and the contrastive loss.

In the *perturbation of the encoder*, Simgrace uses a Gaussian noise-based technique to update the GNN encoder. This approach differs from other methodologies such as BGRL [31] and Merit [16], which use Momentum -based updates and require data augmentation.

Subsequently, Simgrace adopts a non-linear *projection head* to transform the representations of the graph into a latent space, further facilitating the learning of the relationships between the representations.

Finally, Simgrace uses a normalized *contrastive loss* to maximize the agreement between couples of positive representations compared to negative couples. This approach allows to improve the consistency of the rep-

resentations learned, facilitating the classification and prediction process.

Simgrace stands out for its simplicity and effectiveness, obtaining promising results on different supervised and semi-supreme learning learning tasks, demonstrating its robustness and flexibility even with reduced quantities of training data.

GearNet GeomEtry-Aware Relational Graph Neural Network (GearNet) [44] is a structure-based encoder designed to encode spatial information by integrating various types of sequential or structural edges through a *sparse edge message passing* mechanism. This approach facilitates edge-level message passing, enabling the explicit modeling of interactions between edges.

In addition, GearNet employs a multi-view contrastive learning framework with novel augmentation functions to discover correlated co-occurrence of protein substructures and align their representations in the latent space. Specifically, GearNet employs two mapping schemes, termed subsequence cropping and subspace cropping. Subsequence cropping involves sampling random residues between a left residue l and a right residue r to capture protein domains, while subspace cropping is utilized to identify spatially correlated structural patterns. Following SimCLR [6], GearNet employs a contrastive loss function to maximize the mutual information between these biologically correlated views. This approach aims to embed biologically related substructures close to each other in the latent space, while keeping unrelated ones distant.

GearNet showed remarkable promise, achieving comparable results on both function prediction and fold classification tasks with state-of-the-art methods, while being trained on much less data.

4. Methods

This section explains the methodology adopted in the context of this work, presenting the dataset used, the architecture of the proposed model and the related training and evaluation strategies.

4.1. Dataset

The dataset utilized in this research constitutes a subset of the AlphaFold Protein Structure Database [33], regarding human proteomes and encompassing around $24,000$ predicted human proteins.

Pre-processing The dataset's raw data was available in both PDB and CIF file formats. We exclusively opted for the PDB files and subsequently constructed the corresponding graph. All parsing and graph construction from the PDB files were conducted using the Graphein library [14].

Our decisions regarding graph construction revolved around the granularity of the nodes (i.e., atom-level or residue-level), the features to be considered for each node and the criteria for adding edges. These considerations are elaorated in the subsequent paragraphs.

Granularity In the context of predicting the three-dimensional structures of proteins using GNNs, the selection of the granularity of the graph constitutes a decision of fundamental importance, since it significantly affects the performance of the model. This granularity refers to the level of detail with which the structure of the protein is represented in the graph, which can be at the residue level or at the atom level.

A **residue-level** graph models each protein residue as a node in the graph, while the links between those nodes reflect the interactions between the residues. This approach simplifies the graph representation, reducing computational complexity, but may result in the loss of crucial information regarding the atomic interactions within each residue.

Otherwise, an **atom-level** graph considers each atom of the protein as a node of the graph, allowing a more detailed representation of the atomic interactions present in the protein structure. This approach maintains a high level of structural detail, allowing the model to capture more subtle relationships between atoms and improve accuracy in predicting the three-dimensional structure of the protein.

Throughout our research, we chose to employ an atom-level graph for a number of reasons. First of all, this approach allows to acquire crucial structural details that exert a significant influence on the three-dimensional conformation of the protein and which could lead the model to have a greater generalization capacity.

Node Features For each node a in the atom-level graph G, we associate a feature vector $\mathbf{x}_a$:

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

where:

- $\mathbf{coords}_a := [x_a, y_a, z_a]$ is the vector describing the spacial coordinates of the atom a;
- $\mathbf{meiler}_a := \left[m_1^a, m_2^a, \ldots, m_7^a\right]$ is the pre-computed Meiler's embedding vector [23] for the atom a;

- **expasy**$_a$ contains the amino acid features that come from the EXPASY protein scale;
- **amino**$_a$ is the one-encoding of amino acid types;

Our selection of attributes for the representation of the nodes has been led by a series of key considerations. First of all, the inclusion of the 3D coordinates was a crucial choice as they provide the model a detailed vision of the spatial arrangement of the atoms in the molecule. Since the three -dimensional structure is linked to the molecular interactions and functional properties of proteins, their inclusion allows the model to acquire a deeper and more accurate understanding of the biological processes involved.

In addition, Meiler's embeddings have been integrated into the representation of the nodes to provide the model with information on the psychimic characteristics of the atoms. These embeddings capture the nuances of atomic interactions and molecular ties, allowing the model to explore the structure and chemical properties of the protein in more detail.

The amino acid information from the Expasy protein scale and the one-hot encoding of the types of amino acids has been included to allow the model to discriminate between different amino acids and to capture the peculiarities of the protein sequence. Since the composition of amino acid directly influences the structure and function of proteins, the addition of this information allows the model to better understand the relationships between sequence, structure and protein function.

The selection and integration of these different characteristics in the vectors of the nodes allow the model to obtain a complete and detailed vision of the structure and composition of the protein molecule, providing a solid base for the analysis and forecast of biological and functional properties.

Edge Construction In the construction phase of the edges we employed the the K-Nearest Neighbors (KNN) [25] criterion, which establishes the proximity between the atoms in order to create connections between those who are close to the three-dimensional space of the molecule. This approach takes into consideration both long-range interactions and short-range those between atoms, thus contributing to a more complete representation of the protein structure.

In addition, several graph features have been integrated to enrich it with detailed chemical and structural information. These attributes include hydrogen, peptide, aromatic and covalent links between atoms, in order to reflect the essential molecular interactions for the structure and function of proteins.

The **hydrogen ties**, essential to stabilize the three-dimensional structure of proteins and for numerous intracellular molecular interactions, have been introduced in order to capture these interactions and consider them during the analysis and prediction of the protein structure.

The **peptide ties**, which connect the amino acids within the polypeptide chains of the proteins, have been included in the graph to allow the model to understand the primary structure of proteins and to evaluate the relationships between the amino acids along the polypeptide chain.

Finally, the **aromatic ties**, characterized by the presence of aromatic rings in the lateral chains of the amino acids, have been added to the structure to provide the model a complete view of the interactions between the amino acids, which are crucial for structural stability and chemical properties proteins.

The use of the KNN criterion for the construction of the arches, together with the detailed chemical and structural information, guarantees an exhaustive representation of the fundamental molecular interactions for understanding the protein structure.

Dataset Splits For the division of the dataset into training, validation, and test sets, we chose an 80%-10%-10% split, respectively.

4.2. Architecture

In this work, we introduce C3DPNet (Contrastive 3D Protein Prediction Network) which combines different approaches for learning representations in the graph and genetic data domains, with the main objective of integrating semantic and structural information from molecular graphs and DNA sequences, thus facilitating the realization of classification and predictive analysis tasks.

The model uses a GNN-based graph-encoder to extract structural information from graphs and a DNABERT-2 [46]-based text encoder to acquire semantic information and contextual relationships from the DNA sequence. For the graph-encoder, we explored several architectures: GraphSAGE, GCN, GIN, and GAT.

The representations extracted from the encoders are then approached through the use of contrastive loss described in the next section. Figure 1 depicts the architecture of the proposed model.

4.3. Loss

In the context of representation learning, a fundamental challenge is to maximize the model's ability to

capture relevant and discriminative information. Inspired by the success of CLIP [26] in computer vision, we decided to use loss InfoNCE, given its effectiveness in promoting the learning of contextually informative and meaningful representations, which have proven crucial in many computer vision and NLP applications.

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

Here, f_θ is a function that maps the context representation c_t to a feature space, specifically at a future time step k, S is the set of samples, which includes one positive sample (usually the anchor) and $N-1$ negative samples. The InfoNCE function essentially measures how well the context representation c_t aligns with the future feature embedding $f_\theta(c_t)$ compared to the feature embeddings of negative samples. It encourages the model to maximize the similarity between the context and future representations while minimizing the similarity with negative samples. This helps in learning informative and meaningful representations for downstream tasks.

5. Experiments and Results

Experimental setup Experiments were performed on a workstation comprising 1 NVIDIA RTX 4000, 256GB RAM and a 2.3GHz 16-Core CPU. The operating system and deep learning platforms used were: Ubuntu 20.0.4, PyTorch 2.1.0 and PyTorchGeometric 2.4.0.

Hyper-parameters Hyper-parameters were set using the bayesian optimization technique [30]. Table 1 shows the considered values and ranges for hyper-parameters in all model families. For the learning rate scheduler, we employed a linear strategy. For the optimizer we explored options between Adam [18], AdamW [20] and SGD. The batch size was set to 8 for all experiments.

Table 1: Hyper-parameters values and ranges considered during training.

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

Model Variants Below are the best combinations of hyper-parameters found for each type of GNN considered.

- GraphSAGE: learning rate set to 6.25e−6, weight decay equal to 0.038, 6 layers and the AdamW optimizer.
- GCN: learning rate set to 3.34e−6, weight decay equal to 0.098, 6 layers and the AdamW optimizer.
- GIN: learning rate set to 9.28e−6, weight decay equal to 0.083, 6 layers and the AdamW optimizer.
- GAT: learning rate set to 5.12e−6, weight decay equal to 0.062, 3 layers and the AdamW optimizer.

Table 2: Results of model evaluations on the dataset used during training. The values reported are intra-batch values. GSAGE denotes the GraphSAGE model. Best results are in bolds.

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

Results Table 2 shows the results of the evaluation of the GNNs variants considered on the dataset used for training. From the results, it emerges that the best model is obtained using a GNN encoder based on GraphSAGE.

Particularly interesting are the generalizing capabilities demonstrated by each model, especially in the first training epochs, as highlighted in Figures 2, 3, 4, 5, where it is possible to notice a better performance in the validation phase compared to the training phase. This is assumed to be due to the integration of information between the approaching primary and tertiary sequences, which elevate the generalizing capabilities of the models.

Table 3: Comparison between GraphSAGE and C3DPNet on the ENZYMES [3] dataset. GSAGE dentoes the GraphSAGE model. Best results are in bold.

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

In order to ascertain the effectiveness of the integration of the contrastive approach, we carried out a

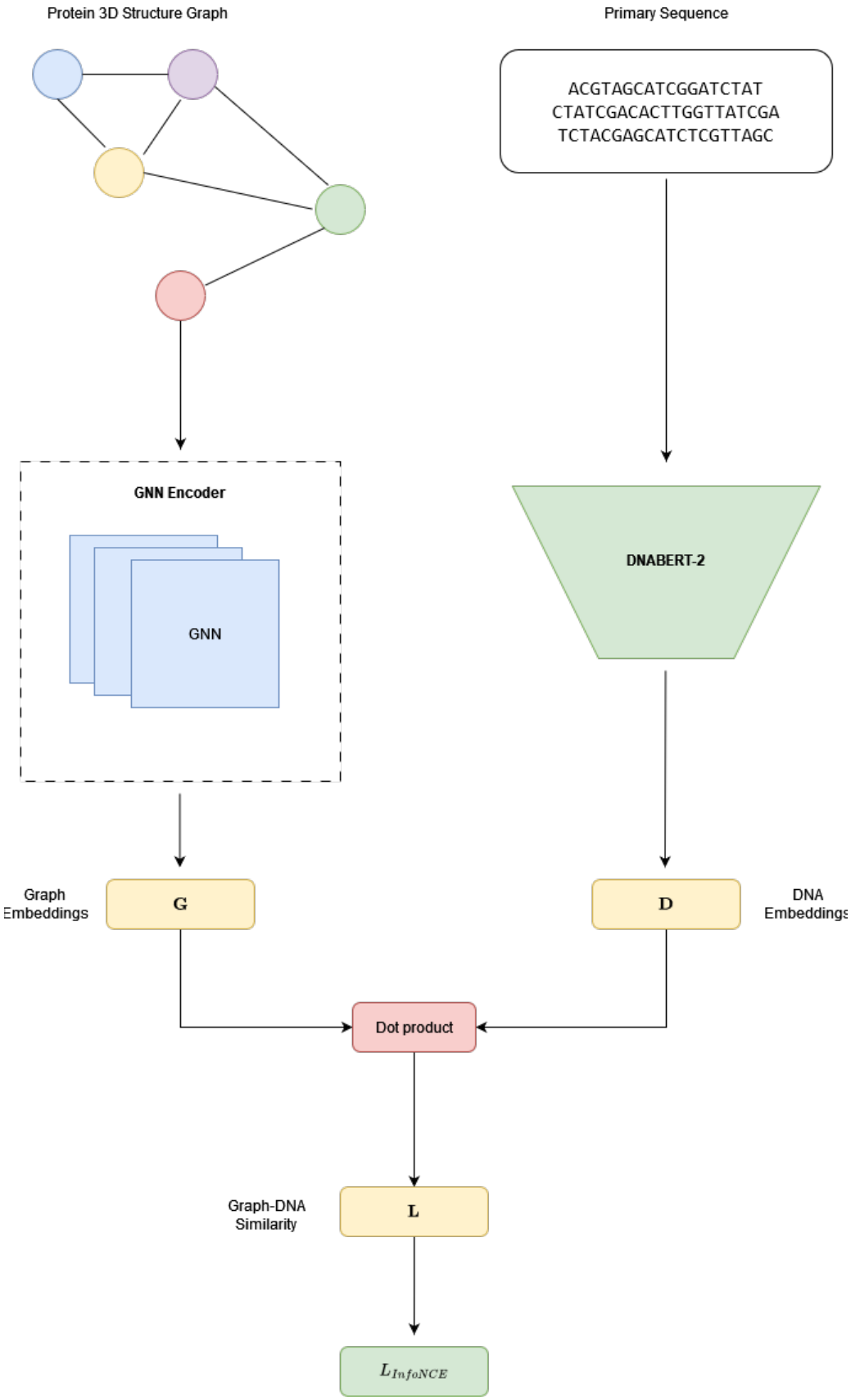

Figure 1: Overview di C3DPNet.

comparison between our best model and the corresponding GNN baseline. In this specific case, we compared C3DPNet's GraphSAGE encoder its correspondive baseline. The two models were trained and evaluated on the ENZYMES [3] dataset. Table 3 shows the results of the comparison; where the GraphSAGE baseline highlights almost random performance, our model demonstrates superior discriminative capabilities, highlighting the effectiveness of the contrastive approach employed.

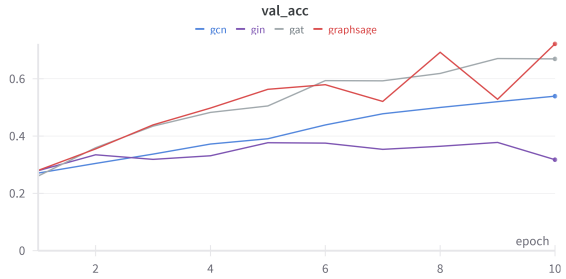

Figure 2: Validation accuracy.

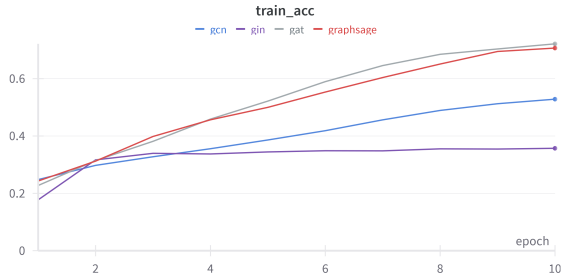

Figure 3: Training accuracy.

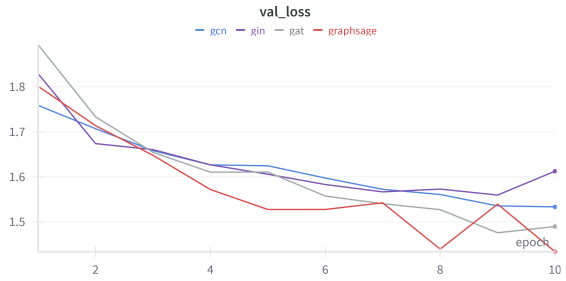

Figure 4: Validation loss.

6. Conclusions

The present research introduced the C3DPNet architecture, an approach that examined the effectiveness of the contrastive approach in predicting the three-dimensional structure of proteins. This model integrates GNN models for three-dimensional structure modeling and natural language models, such as DNABERT-2 [46], to extract information from the primary sequence. Several GNN models have been explored, highlighting a notable generalization capacity of the model, underlined by greater accuracy in the validation phase, increasing confidence in the model's ability to provide precise and reliable predictions on the three-dimensional structure of proteins. Finally, a comparison was conducted between C3DPNet's graph-encoder and its corresponding baseline on a classification task. Our method demonstrated superior discriminative capabilities, while the counterpart highlighted pseudo-random tendencies, confirming the effectiveness of the contrastive approach.

Figure 5: Training loss.

Despite the promising results obtained, we recognize the need for further efforts aimed at optimizing the model's performance. For future prospects, it is planned to:

- Explore the use of other GNN architectures, such as DiffPool[40] and Graph-UNets[11], to evaluate their impact on predictive performance;
- Employing data augmentation techniques to increase the number of positive pairs per graph;
- Using other contrastive losses, such as the Sigmoid loss [42], could help to further optimize the training phase, improving its accuracy and stability;
- Consider using protein-specific pre-trained models, such as ProteinBERT [4], to take full advantage of the information contained in protein data;
- Extend the evaluation of the model on a wide range of protein datasets, in order to evaluate its generalization capacity and robustness in different contexts.

References

[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, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL].

[3] Karsten M Borgwardt and 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, and 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 and Shuiwang Ji. *Graph U-Nets*. 2019. arXiv: 1905.05178 [cs.LG].

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

[13] Pedro Hermosilla and 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 and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

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

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

[21] Richard Maclin and 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 (Sept. 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, and 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, and 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*. 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, and 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].