Integrating Structure and Sequence: Protein Graph Embeddings via GNNs and LLMs

Francesco Ceccarelli[a, *], **Lorenzo Giusti**[b], **Sean B. Holden**[a], **Pietro Liò**[a]
[a]Department of Computer Science and Technology , University of Cambridge, Cambridge, United Kingdom
[b]Department of Computer, Control and Management Engineering, Sapienza University, Rome, Italy

Abstract. Proteins perform much of the work in living organisms, and consequently the development of efficient computational methods for protein representation is essential for advancing large-scale biological research. Most current approaches struggle to efficiently integrate the wealth of information contained in the protein sequence and structure. In this paper, we propose a novel framework for embedding protein graphs in geometric vector spaces, by learning an encoder function that preserves the structural distance between protein graphs. Utilizing Graph Neural Networks (GNNs) and Large Language Models (LLMs), the proposed framework generates structure- and sequence-aware protein representations. We demonstrate that our embeddings are successful in the task of comparing protein structures, while providing a significant speed-up compared to traditional approaches based on structural alignment. Our framework achieves remarkable results in the task of protein structure classification; in particular, when compared to other work, the proposed method shows an average F1-Score improvement of 26% on out-of-distribution (OOD) samples and of 32% when tested on samples coming from the same distribution as the training data. Our approach finds applications in areas such as drug prioritization, drug re-purposing, disease sub-type analysis and elsewhere.

Keywords: Graph Neural Networks, Large Language Models, Protein Representation Learning.

*Corresponding author: fc485@cam.ac.uk

1 INTRODUCTION

Proteins are organic macro-molecules made up of twenty types of natural amino acids. Almost all interactions and reactions which occur in living organisms, from signal transduction, gene transcription and immune function to catalysis of chemical reactions, involve proteins (1). The comparison of proteins and their structures is an essential task in bioinformatics, providing support for protein structure prediction (2), the study of protein-protein docking (3), structure-based protein function prediction (4) and many further tasks. Considering the large quantity of protein data stored in the Protein Data Bank (PDB) (5) and the rapid development of methods for performing protein structure prediction (for example, AlphaFold2 (6)), it is desirable to develop methods capable of efficiently comparing the tertiary structures of proteins.

Generally, protein comparison methods can be divided into two classes: alignment-based methods (7; 8; 9) and alignment-free methods (10; 11; 12; 13). The former aim at finding the optimal structural superposition of two proteins. A scoring function is then used to measure the distance between each pair of superimposed residues. For such methods (for example (14; 15)) the superposition of the atomic structures is the main bottleneck as it has been proven to be an NP-hard problem (16). On the other hand, alignment-free methods try to represent each protein in the form of a descriptor, and then to measure the distance between pairs of descriptors (10). Descriptors need to satisfy two requirements: (1) their size should be fixed and independent of the length of proteins; (2) they should be invariant to rotation and translation of proteins.

The template modeling score (TM-score) (17) is a widely used metric for assessing the structural similarity between two proteins. It is based on the root-mean-square deviation (RMSD) of the atomic positions in the proteins, but considers the lengths of the proteins and the number of residues that can be superimposed. TM-score has been shown to be highly correlated with the similarity of protein structures and can be used to identify structurally similar proteins, even when they have low sequence similarity. Unfortunately, computing TM-scores is computationally intractable even for relatively small numbers of proteins. TM-align (15), one of the popular alignment-based methods, takes about 0.5 seconds for one structural alignment on a 1.26 GHz PIII processor. As such, computing TM-scores for existing databases, containing data for millions of proteins, is unaffordable. While several deep learning methods for protein comparison have been developed (for example, DeepFold (18) and GraSR (10)) they suffer from major drawbacks: (1) they are trained by framing the protein comparison task as a classification problem—that is, predicting if two proteins are structurally similar—and hence fail to directly incorporate TM-scores in the loss function formulation; (2) they produce latent representations (embeddings) which do not integrate the information contained in the protein sequences and structures; (3) they usually do not exploit the inductive bias induced by the topology of graph-structured proteins, and they fail to consider different geometries of the latent space to match well the underlying data distribution.

In this paper, we address the aforementioned limitations of current protein embedding methods by proposing an efficient and accurate technique that integrates both protein sequence and structure information. In detail, we first construct protein graphs where each node represents an amino acid in the protein sequence. We then generate features for each amino acid (node in the graph) using Large Language Models (LLMs) before applying Graph Neural Networks (GNNs) to embed the protein graphs in geometric vector spaces while combining structural and sequence information. By incorporating TM-scores in the formulation of the loss function, the trained graph models are able to learn a mapping that preserves the distance between the input protein graphs, providing a way to quickly compute similarities for every pair of unseen proteins. We evaluated the proposed approach and its ability to generate meaningful embeddings for downstream tasks on two protein datasets. On both, the proposed approach reached good results, outperforming other current state-of-the-art methods on the task of structural classification of proteins on the SCOPe dataset (19).

Contribution The main contributions of this paper can be summarised as follow: *(i)* A novel learning framework for generating protein representations in geometric vector spaces by merging structural and sequence information using GNNs and LLMs. *(ii)* A quick and efficient method for similarity computation between any pair of proteins. *(iii)* An evaluation of the ability of our embeddings, in both supervised and unsupervised settings, to solve downstream protein classification tasks, and a demonstration of their superior performance when compared to current state-of-the-art methods. Our approach finds a plethora of applications in the fields of bioinformatics and drug discovery.

2 BACKGROUND AND RELATED WORK

Several alignment-based methods have been proposed over the years, each exploiting different heuristics to speed up the alignment process. For example, in DALI (20), Monte Carlo optimization is used to search for the best structural alignment. In (8), the authors proposed combinatorial extension (CE) for similarity evaluation and path extension. An iterative heuristic based on the Needleman–Wunsch dynamic programming algorithm (21) is employed in TM-align (15), SAL (22) and STRUCTAL (23). Examples of alignment-free approaches are Scaled Gauss Metric (SGM) (11) and the Secondary Structure Element Footprint (SSEF) (13). SGM treats the protein backbone as a space curve to construct a geometric measure of the conformation of a protein, and then uses this measure to provide a distance between protein shapes. SSEF splits the protein into short consecutive fragments and then uses these fragments to produce a vector representation of the protein structure as a whole. More recently, methods based on deep learning have been developed for the task of protein structure comparison. For instance, DeepFold (18) used a deep convolutional neural network model trained with the max-margin ranking loss function (24) to extract structural motif features of a protein, and learn a fingerprint representation for each protein. Cosine similarity was then used to measure the similiarity scores between proteins. DeepFold has a large number of parameters, and fails to exploit the sequence information and the topology of graph-structured data. GraSR (10) employs a contrastive learning framework, GNNs and a raw node feature extraction method to perform protein comparison. Compared to GraSR, we present a general framework to produce representations of protein graphs where the distance in the embedding space is correlated with the structural distance measured by TM-scores between graphs. Finally, our approach extends the work presented in (25), which was limited to biological sequence embeddings, to the realm of graph-structured data.

3 MATERIAL AND METHODS

The core approach, shown in Figure 1, is to map graphs into a continuous space so that the distance between embedded points reflects the distance between the original graphs measured by the TM-scores. The main components of the proposed framework are the geometry of the latent space, a graph encoder model, a sequence encoder model, and a loss function. Details for each are as follows.

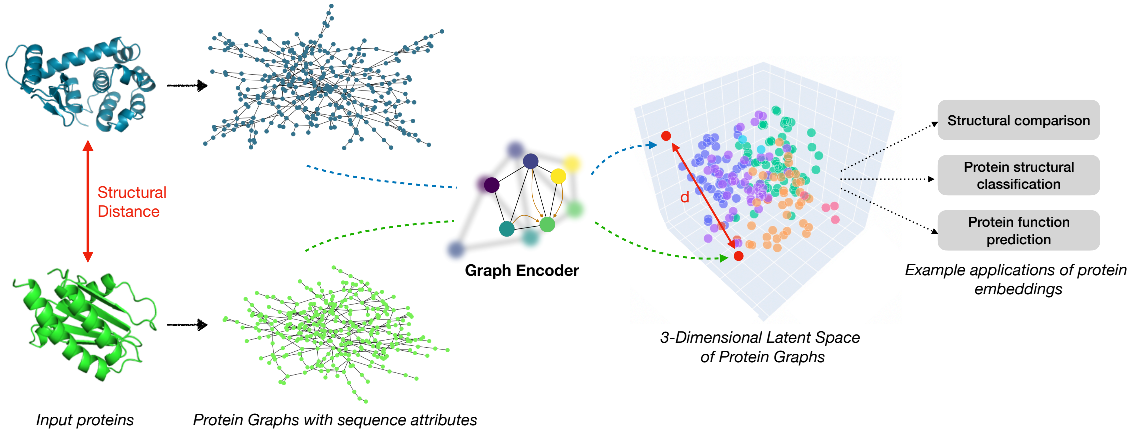
Structural Distance
Graph Encoder
d
Structural comparison
Protein structural classification
Protein function prediction
Example applications of protein embeddings
3-Dimensional Latent Space of Protein Graphs
Input proteins
Protein Graphs with sequence attributes

Fig 1 We learn an encoder function that preserves the structural distance, measured by the TM-score, between two input proteins. We construct protein graphs by combining sequence and structure information as shown in Figure 2. A distance function d defines the shape of the latent space. The generated embeddings can be used for a variety of applications in bioinformatics and drug discovery. (For simplicity, this Figure depicts a 3-dimensional latent space.)

3.1 *Latent Space Geometry*

The distance function used (d in Figure 1) defines the geometry of the latent space into which embeddings are projected. In this work we provide a comparison between Euclidean, Manhattan, Cosine and squared Euclidean (referred to as Square) distances (details in Appendix B).

3.2 Graph Encoder Model

The encoder performs the task of mapping the input graphs to the embedding space. A variety of models exist for this task, including linear, Multi-layer Perceptron (MLP), LSTM (26), CNN (27) and Transformers (28). Given the natural representation of proteins as graphs, we chose GNNs as encoder models. We have constructed the molecular graphs of proteins starting from PDB files. A PDB file contains structural information such as 3D atomic coordinates. Let $G = (V, E)$ be a graph representing a protein, where each node $v \in V$ is a residue and interaction between the residues is described by an edge $e \in E$. Two residues are connected if they have any pair of atoms (one from each residue) separated by a Euclidean distance less than a threshold distance. The typical cut-off, which we adopt in this work, is 6 angstroms (Å) (29).

3.3 Sequence Encoder Model

Given the graph representation of a protein, each node v of the graph (each residue) must be associated with a feature vector. Typically, features extracted from protein sequences by means of LLMs have exhibited superior performances compared to handcrafted features. We experimented with five different sequence encoding methods: (1) a simple one-hot encoding of each residue in the graph, (2) seven physicochemical properties of residues as extracted by (30), which are assumed to influence the interactions between proteins by creating hydrophobic forces or hydrogen bonds between them, (3) the BLOcks SUbstitution Matrix (BLOSUM) (31), which counts the relative frequencies of amino acids and their substitution probabilities, (4) features extracted from protein sequences employing a pre-trained BERT-based transformer model (ProBert (32)), and (5) node features extracted using a pre-trained LSTM-based language model (SeqVec (33)). Table 1 summarizes the node features and their dimensions, while Figure 2 depicts the process of constructing

Table 1 Investigated node attributes and their dimensions. BERT and LSTM features are extracted using LLMs pre-trained on protein sequences (ProBert (32) and SeqVec (33)).

Feature	Dimension
One hot encoding of amino acids	20
Physicochemical properties	7
BLOcks SUbstitution Matrix	25
BERT-based language model	1024
LSTM-based language model	1024

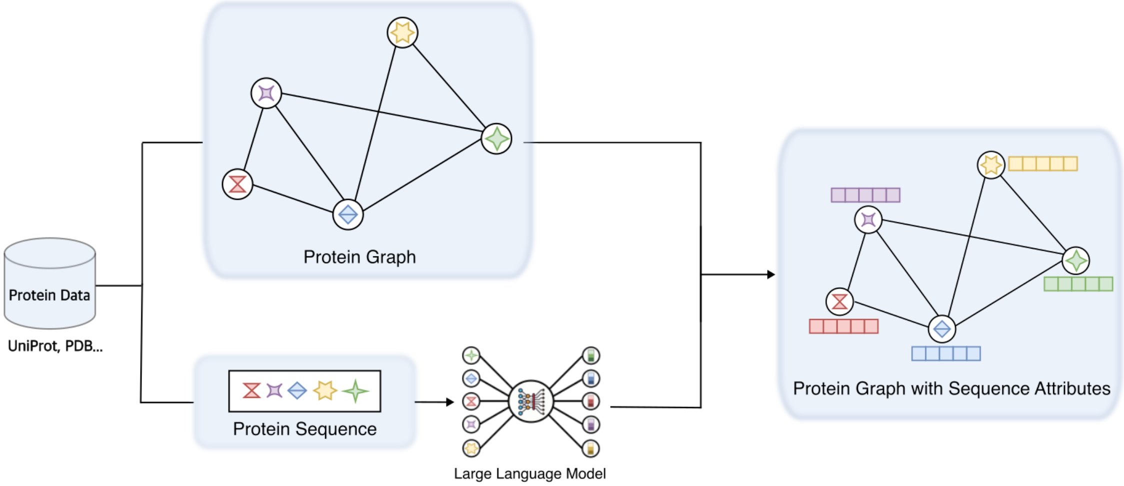

Fig 2 Graph representation of a protein, which combines sequence and structure. Starting from protein data (a PDB file from, for example, UniProt or PDB), we extract protein sequence and structure information. We construct graphs where each node represents an amino acid in the protein sequence. We then generate features for each node in the graph using Large Language Models pre-trained on protein sequences.

a protein graph with node features, starting from the corresponding protein data.

3.4 Loss Function

The loss function used, which minimises the MSE between the graph distance and its approximation as the distance between the embeddings, is

$$L = \sum_{g_1,g_2 \in G} \left(\mathrm{TM}\left(g_1, g_2\right) - d\left(\mathrm{GNN}_\theta\left(g_1\right), \mathrm{GNN}_\theta\left(g_2\right)\right)\right)^2 \tag{1}$$

where G is the training set of protein graphs, GNN_θ is the graph encoder and θ represents the parameters of the model. The TM-score is a similarity metric in the range (0,1], where 1 indicates a perfect match between two structures. Since the formulation of the loss is expressed in terms of distances, we reformulate the TM-scores as a distance metric by simply computing $\text{TM}(g_1, g_2) = 1 - \text{TM}_{\text{score}}(g_1, g_2)$. By training neural networks to minimize the loss in Equation 1, we encourage the networks to produce latent representations such that the distance between these representations is proportional to the structural distance between the input graphs.

4 PROTEIN DATASETS

We evaluated the proposed approach on two protein datasets. First, we downloaded the human proteome from UniProt[1] and sub-selected 512 protein kinases. To obtain the TM-scores to train the graph models, we evaluated the structural similarity using TM-align (15). All-against-all alignment yielded a dataset composed of 130,816 total comparisons. Every kinase in the dataset is categorized in one of seven family groups: (a) AGC (63 proteins), (b) CAMK (82 proteins), (c) CK1 (12 proteins), (d) CMGC (63 proteins), (e) STE (48 proteins), (f) TK (94 proteins), and (g) TKL (43 proteins). The number of nodes in the graphs ranges from 253 to 2644, with an average size of approximately 780 nodes. The average degree in the graphs is approximately 204, the average diameter of the graphs is approximately 53 nodes and the maximum diameter is 227 nodes. We further used the 40% identity filtered subset of SCOPe v2.07 (March 2018) as a benchmark dataset (19). This dataset contains 13,265 protein domains classified in one of seven classes: (a) all alpha proteins (2286 domains), (b) all beta proteins (2757 domains), (c) alpha and beta proteins (a/b) (4148 domains), (d) alpha and beta proteins (a+b) (3378 domains), (e) multi-domain proteins (alpha and beta) (279 domains), (f) membrane and cell surface proteins and peptides (213 domains), and g) small proteins (204 domains). We again used TM-align with all-against-all settings

---

[1] https://www.uniprot.org

to construct a dataset of approximately 170 millions comparisons. To reduce the computational time and cost during training, we randomly sub-sampled 100 comparisons for each protein to create a final dataset of 1,326,500 comparisons. For this dataset, the number of nodes in the graphs ranges from 30 to 9800, with an average size of approximately 1978 nodes. The average degree is approximately 90, the average diameter of the graphs is approximately 9 nodes and the maximum diameter is 53 nodes. Compared to benchmark graph datasets (for example (34) and (35)) we evaluated our approach on graphs of significantly larger size (84 and 13 times more nodes than the molecular graphs in (34) and in (35), respectively).

5 EXPERIMENTAL RESULTS

5.1 Experimental Settings

We evaluate the proposed framework using Graph Convolutional Networks (GCNs) (36), Graph Attention Networks (GATs) (37), and GraphSAGE (38) (Appendix A). All the models were implemented with two graph layers in PyTorch geometric (39) to learn protein embeddings of size 256. Adam optimizer (40) with a learning rate of 0.001 was used to train the models for 100 epochs with a patience of 10 epochs. The batch size was set to 100. We used 4 attention heads in the GAT architecture. For each model, Rectified Linear Units (ReLUs) (41) and Dropout (42) were applied after each layer, and mean pooling was employed as readout function to obtain graph-level embeddings from the learned node-level representations. Finally, each experiment was run with 3 different seeds to provide uncertainty estimates.

5.2 Kinase Embeddings

For the generation of the embeddings, we used 80% of the kinase proteins for training and the remaining 20% for testing. Table 2 shows the MSE values for the graph encoders, using different choices of distance functions and node features. For each model, the best scores are consistently reached with LSTM-extracted features and Euclidean geometry of the embedding space. Across all models, GAT embeddings exhibit the lowest MSE, followed by GarphSAGE and GCN. From Table 2, it is clear that using pre-trained language models to extract node features from protein sequences leads to better results. MSE scores for all distances across all encoder models are lower when using BERT and LSTM features. Furthermore, the LSTM-extracted features perform consistently better compared to the BERT ones. BLOSUM and Physicochemical features are also

Table 2 MSE results for different feature types, distance functions and graph encoder models on the kinase dataset. We use gold ●, silver ●, and bronze ● colors to indicate the first, second and third best performances, respectively. For each model, the best scores are consistently reached with LSTM-extracted features and Euclidean geometry of the embedding space. Across all models, GAT embeddings exhibit the best performance. For all the models, MSE scores are lower for features extracted by means of LLMs (BERT and LSTM) compared to handcrafted feature extraction methods (one-hot, biochemical and BLOSUM).

Model	**Feature**	**Distance**			
		`Cosine`	`Euclidean`	`Manhattan`	`Square`
GCN	One hot	0.0194 ± 0.002	0.0380 ± 0.003	0.0192 ± 0.001	0.0729 ± 0.004
	Physicochemical	0.0343 ± 0.012	0.0483 ± 0.009	0.0397 ± 0.003	0.1109 ± 0.007
	BLOSUM	0.0327 ± 0.071	0.0271 ± 0.043	0.0450 ± 0.013	0.0697 ± 0.023
	BERT	0.0110 ± 0.003	0.0103 ± 0.001	0.0131 ± 0.006	0.0138 ± 0.009
	LSTM	0.0105 ± 0.002	0.0088 ± 0.004	0.0156 ± 0.001	0.0107 ± 0.004
GAT	One hot	0.0171 ± 0.001	0.0320 ± 0.012	0.0171 ± 0.011	0.0758 ± 0.009
	Physicochemical	0.0295 ± 0.007	0.0328 ± 0.006	0.0220 ± 0.004	0.0856 ± 0.023
	BLOSUM	0.0245 ± 0.012	0.0163 ± 0.009	0.0124 ± 0.011	0.0307 ± 0.009
	BERT	0.0091 ± 0.018	0.0095 ± 0.008	0.0078 ± 0.009	0.0133 ± 0.011
	LSTM	0.0088 ± 0.009	0.0073 ± 0.004	0.0086 ± 0.006	0.0101 ± 0.009
Graph SAGE	One hot	0.0243 ± 0.002	0.0227 ± 0.011	0.0156 ± 0.009	0.0424 ± 0.010
	Physicochemical	0.0301 ± 0.004	0.0266 ± 0.008	0.0310 ± 0.011	0.0578 ± 0.009
	BLOSUM	0.0285 ± 0.007	0.0172 ± 0.008	0.0342 ± 0.002	0.0368 ± 0.007
	BERT	0.0097 ± 0.011	0.0089 ± 0.007	0.0101 ± 0.007	0.0107 ± 0.009
	LSTM	0.0093 ± 0.003	0.0084 ± 0.005	0.0143 ± 0.007	0.0094 ± 0.008

usually associated with higher MSE for all distances and models, indicating that they are poorly correlated to TM-scores.

5.3 *Fast Inference of TM-scores*

We employed the trained GAT architectures from Table 2 to predict the TM-scores for the kinase pairs in the test set. In Figure 3, we show the predicted versus actual TM-scores for two combinations of features and embedding geometries. The left plot in Figure 3 uses LSTM-extracted features and Euclidean space, while the right one shows predictions for BLOSUM features and Manhattan space. The complete quantitative evaluations, measured by Pearson correlation between model predictions and true TM-scores for all distances and features, are reported in Appendix D. As in Table 2, the best performances are reached when employing LSTM and BERT features while BLOSUM and Physicochemical features lead to the poorest performances (Appendix D). The highest correlation score, reflecting the results reported in Table 2, is reached when employing LSTM features and Euclidean distance (Figure 3). It is worth noticing that, for the 26,164 comparisons in

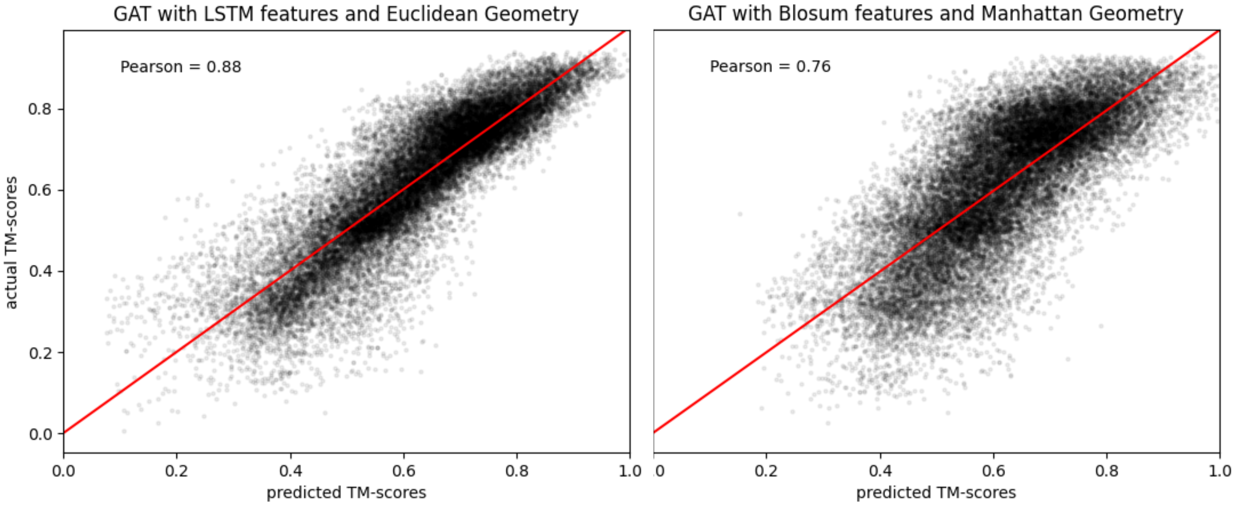

Fig 3 Actual versus predicted TM-scores. Using LSTM features and Euclidean geometry (*left*) results in predictions which follow more tightly the red line of the oracle compared to BLOSUM features in the Manhattan space (*right*).

the test set, the proposed approach took roughly 120 seconds to compute TM-scores. Executing TM-align with the same number of comparisons took 57,659 seconds ($\approx$ 16 hours). Details of the TM-score inference times for all the models are given in Appendix D. *The major speed-up provided by performing inference using machine learning models makes the proposed approach applicable to datasets comprising millions of proteins.*

5.4 Ablation Study: Structure Removal

Coupling GNNs with LLMs provides a means of integrating the information coming from the structure and sequence of proteins. To analyse the benefits of exploiting the topology induced by the graph structures, we performed an ablation study which disregards such information. DeepSet (43) considers objective functions defined on sets, that are invariant to permutations. Using a DeepSet formulation, we constructed protein graphs with features where each node is only connected to itself. As for the graph models, we trained DeepSet to minimize the loss function in Equation 1 and report the results in Table 3. Similarly to Table 2, the best MSE scores are reached when using LSTM features and Euclidean geometry. The scores in Table 3, computed by disregarding the graph connectivity and neighborhood information, are significantly higher than those reported in Table 2 (p-value of t-test < 0.05 compared to GCN, GAT and GraphSAGE). By considering patterns of local connectivity and structural topology, GNNs are able to learn better protein graph representations compared to models which only exploit sequence-derived features.

5.5 Downstream Task of Kinase Classification

To prove the usefulness of the learned embeddings for downstream tasks, we set out to classify each kinase into one of the seven family groups (AGC, CAMK, CK1, CMGC, STE, TK, TKL).

Table 3 MSE values for an ablation study which disregards the topological information induced by the structure of the protein graphs. We use gold ●, silver ●, and bronze ● colors to indicate the first, second and third best performances, respectively. By ignoring the neighborhood and the structural information, the MSEs are significantly higher (p-value of t-test < 0.05) compared to GNNs.

Model	**Feature**	**Distance**			
		`Cosine`	`Euclidean`	`Manhattan`	`Square`
DeepSet	One Hot	0.1742 ± 0.003	0.0421 ± 0.002	0.0358 ± 0.001	0.0714 ± 0.003
	Physicochemical	0.1766 ± 0.010	0.0437 ± 0.006	0.0464 ± 0.004	0.0900 ± 0.006
	BLOSUM	0.1553 ± 0.003	0.0381 ± 0.009	0.0558 ± 0.008	0.0914 ± 0.008
	BERT features	0.0132 ± 0.004	0.0129 ± 0.005	0.0192 ± 0.005	0.0220 ± 0.004
	LSTM features	0.0141 ± 0.003	0.0116 ± 0.010	0.0348 ± 0.006	0.0200 ± 0.007

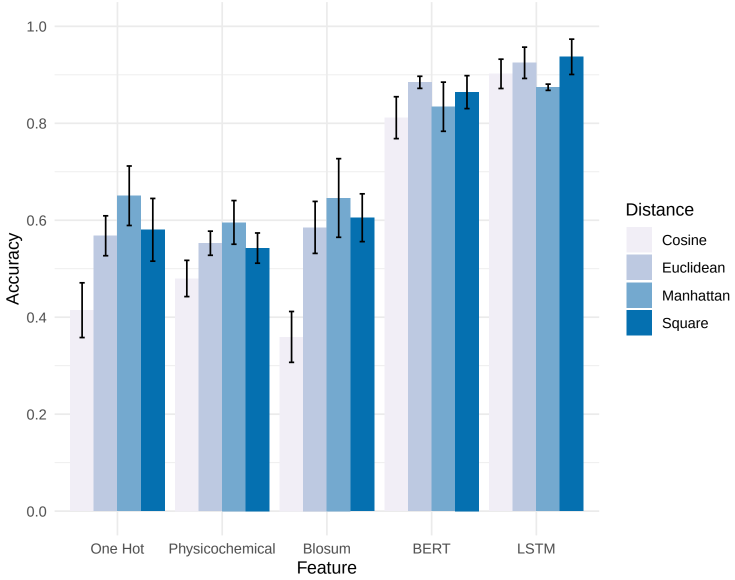

Fig 4 Accuracy of classification for kinase family prediction using the embeddings generated by the GAT models. The highest accuracy value of 93.7% is reached with LSTM features and the Square distance function.

Using the embeddings generated by the GAT models, we trained an MLP, composed of 3 layers of size 128, 64 and 32 respectively, and a SoftMax classification head. The accuracy of classification, computed as the average result of 5-fold cross-validation, for each feature type and distance function is reported in Figure 4. The results are consistent with Table 2: the best accuracies are obtained when using LSTM- and BERT-extracted sequence features, while handcrafted feature extraction methods (one hot, BLOSUM and physicochemical) provide the poorest performance. The highest accuracy values of 93.7% and 92.48% are reached with LSTM features and Square and Euclidean

Table 4 Out of distribution (OOD) classification results on SCOPe proteins (**F1-Score (OOD)**). We use gold ●, silver ●, and bronze ● colors to indicate the first, second and third best performances, respectively. Despite the different training data, the GAT model with Euclidean and Square geometry outperforms all other approaches trained on SCOPe proteins. Classification results for embeddings generated after training on SCOPe proteins are also shown (**F1-Score**); in this case, the proposed approach outperforms the others by a larger margin for all choices of latent geometries.

Model	**Distance**	**F1-Score (OOD)**	**F1-Score**
GAT	Cosine	0.6906 ± 0.0044	0.8290 ± 0.008
	Euclidean	0.8204 ± 0.006	0.8557 ± 0.002
	Manhattan	0.7055 ± 0.006	0.8481 ± 0.007
	Square	0.8185 ± 0.004	0.8406 ± 0.006
SGM (11)	-	-	0.6289
SSEF (13)	-	-	0.4920
DeepFold (18)	-	-	0.7615
GraSR (10)	-	-	0.8124

distance functions, respectively.

5.6 *Embedding Out of Distribution Samples*

Being able to use pre-trained models for related or similar tasks is essential in machine learning. We tested the ability of the proposed graph models to generalize to new tasks by generating embeddings for the 13,265 proteins in the SCOPe dataset after being trained only on kinase proteins. Given the better performance provided by the use of LSTM features, in this section we constructed protein graphs with LSTM attributes and used a 3-Layer MLP as before to assign the GAT-generated protein embeddings from the SCOPe dataset to the correct class. Results of this evaluation, measured as average F1-score across 5 folds for each distance function, are shown in Table 4 (F1-Score out of distribution (OOD)). Euclidean and Square geometry of the embedding space exhibited the best classification performances. Despite being trained on OOD samples, the proposed framework with Euclidean and Square geometry still managed to outperform the current state-of-the-art results reported from models trained and tested on SCOPe proteins, as shown in

Table 4. The superior performance, despite the different training data, suggests the ability of the proposed approach to learn meaningful protein representations by (1) merging structural and sequence information into a single pipeline, and (2) capturing different and relevant properties of the geometries of the latent space into which embeddings are projected.

5.7 *Protein Structural Classification*

We constructed protein graphs with LSTM features and trained the proposed GAT architectures on the SCOPe dataset. The resulting MSE scores are reported in Appendix D. The lowest score was again reached when using Euclidean geometry for the latent space. Using this model, we projected the protein embeddings onto two dimensions using t-SNE (44) as shown in Figure 5. The high-level structural classes as defined in SCOPe were captured by the proposed embeddings. While not directly trained for this task, combining structural and sequence information allowed us to identify small, local clusters representing the different protein families in the SCOPe dataset. We

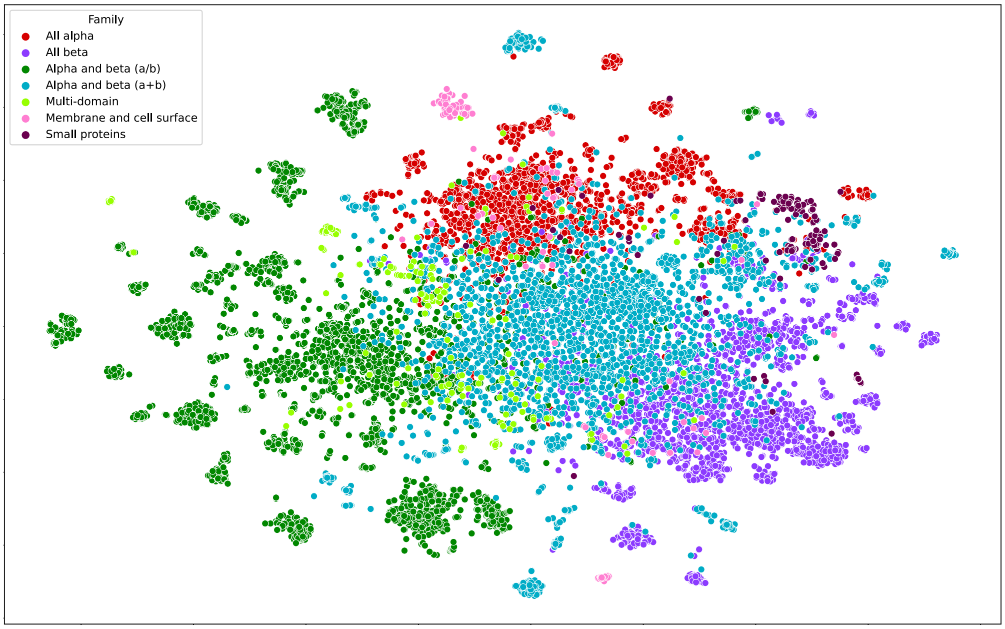

Fig 5 t-SNE visualization of the learned embeddings, coloured by protein structural family. The proposed approach generates protein embeddings which recapitulate the different families in the SCOPe dataset.

employed supervised learning and trained a 3-layer MLP classifier to label each protein embedding in the correct family. Results of this evaluation, measured as average F1-score across 5 folds, are shown in Table 4 (F1-Score). When directly trained on SCOPe proteins, the proposed approach outperforms the others by a large margin for all choices of geometries (Table 4).

6 CONCLUSION

In this paper, we presented a novel framework for generating both structure- and sequence-aware protein representations. We mapped protein graphs with sequence attributes into geometric vector spaces, and showed the importance of considering different geometries of the latent space to match the underlying data distributions. We showed that the generated embeddings are successful in the task of protein structure comparison, while providing an accurate and efficient way to compute similarity scores for large-scale datasets, compared to traditional approaches (Appendix D). The protein graph representations generated by our approach showed state-of-the-art results for the task of protein structural classification on the SCOPe dataset. This work opens opportunities for future research, with potential for significant contributions to the fields of bioinformatics, structural protein representation and drug discovery (Appendix E).

Acknowledgments

For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission. The protein structures in Figure 1 were downloaded from UniProt [2] [3] under the Creative Commons Attribution 4.0 International (CC BY 4.0) License [4] and used without modifications.

---

[2] https://www.uniprot.org

[3] https://www.uniprot.org/help/license

[4] https://creativecommons.org/licenses/by/4.0/

References

[1] R. Morris, K. A. Black, and E. J. Stollar, “Uncovering protein function: from classification to complexes,” *Essays in Biochemistry* **66**(3), 255–285 (2022).

[2] A. Kryshtafovych, T. Schwede, M. Topf, *et al.*, “Critical assessment of methods of protein structure prediction (CASP)—Round XIII,” *Proteins: Structure, Function, and Bioinformatics* **87**(12), 1011–1020 (2019).

[3] M. F. Lensink, S. Velankar, M. Baek, *et al.*, “The challenge of modeling protein assemblies: the CASP12-CAPRI experiment,” *Proteins: Structure, Function, and Bioinformatics* **86**, 257–273 (2018).

[4] P. F. Gherardini and M. Helmer-Citterich, “Structure-based function prediction: approaches and applications,” *Briefings in Functional Genomics and Proteomics* **7**(4), 291–302 (2008).

[5] H. Berman, K. Henrick, and H. Nakamura, “Announcing the worldwide protein data bank,” *Nature Structural & Molecular Biology* **10**(12), 980–980 (2003).

[6] J. Jumper, R. Evans, A. Pritzel, *et al.*, “Highly accurate protein structure prediction with AlphaFold,” *Nature* **596**(7873), 583–589 (2021).

[7] M. Akdel, J. Durairaj, D. de Ridder, *et al.*, “Caretta–a multiple protein structure alignment and feature extraction suite,” *Computational and structural biotechnology journal* **18**, 981–992 (2020).

[8] I. N. Shindyalov and P. E. Bourne, “Protein structure alignment by incremental combinatorial extension (CE) of the optimal path.,” *Protein engineering* **11**(9), 739–747 (1998).

[9] D. Kihara and J. Skolnick, “The PDB is a covering set of small protein structures,” *Journal of molecular biology* **334**(4), 793–802 (2003).

[10] C. Xia, S.-H. Feng, Y. Xia, *et al.*, “Fast protein structure comparison through effective representation learning with contrastive graph neural networks,” *PLoS computational biology* **18**(3), e1009986 (2022).

[11] P. Røgen and B. Fain, “Automatic classification of protein structure by using Gauss integrals,” *Proceedings of the National Academy of Sciences* **100**(1), 119–124 (2003).

[12] I. Budowski-Tal, Y. Nov, and R. Kolodny, “FragBag, an accurate representation of protein structure, retrieves structural neighbors from the entire PDB quickly and accurately,” *Proceedings of the National Academy of Sciences* **107**(8), 3481–3486 (2010).

[13] E. Zotenko, D. P. O’Leary, and T. M. Przytycka, “Secondary structure spatial conformation footprint: a novel method for fast protein structure comparison and classification,” *BMC Structural Biology* **6**, 1–12 (2006).

[14] L. Holm and C. Sander, “Protein structure comparison by alignment of distance matrices,” *Journal of molecular biology* **233**(1), 123–138 (1993).

[15] Y. Zhang and J. Skolnick, “TM-align: a protein structure alignment algorithm based on the TM-score,” *Nucleic acids research* **33**(7), 2302–2309 (2005).

[16] R. H. Lathrop, “The protein threading problem with sequence amino acid interaction preferences is NP-complete,” *Protein Engineering, Design and Selection* **7**(9), 1059–1068 (1994).

[17] Y. Zhang and J. Skolnick, “Scoring function for automated assessment of protein structure template quality,” *Proteins: Structure, Function, and Bioinformatics* **57**(4), 702–710 (2004).

[18] Y. Liu, Q. Ye, L. Wang, *et al.*, “Learning structural motif representations for efficient protein structure search,” *Bioinformatics* **34**(17), i773–i780 (2018).

[19] N. K. Fox, S. E. Brenner, and J.-M. Chandonia, “SCOPe: Structural Classification of Proteins—extended, integrating SCOP and ASTRAL data and classification of new structures,” *Nucleic acids research* **42**(D1), D304–D309 (2014).

[20] L. Holm and C. Sander, “Using dali for structural comparison of proteins,” *Current opinion in structural biology* **9**(3), 408–415 (1999).

[21] S. B. Needleman and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins,” *Journal of molecular biology* **48**(3), 443–453 (1970).

[22] S. S. Krishna, I. Majumdar, N. Grishin, *et al.*, “The PDB is a covering set of small protein structures,” *Journal of molecular biology* **267**(3), 638–657 (1997).

[23] C. Zhang and C. DeLisi, “A unified statistical framework for sequence comparison and structure comparison,” *Proceedings of the National Academy of Sciences* **94**(11), 5917–5922 (1997).

[24] L. Wang, Y. Li, and S. Lazebnik, “Learning deep structure-preserving image-text embeddings,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5005–5013 (2016).

[25] G. Corso, Z. Ying, M. Pándy, *et al.*, "Neural distance embeddings for biological sequences," *Advances in Neural Information Processing Systems* **34**, 18539–18551 (2021).

[26] K. Cho, B. Van Merriënboer, C. Gulcehre, *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078* (2014).

[27] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics* **36**(4), 193–202 (1980).

[28] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems* **30** (2017).

[29] J. Chen, S. Zheng, H. Zhao, *et al.*, "Structure-aware protein solubility prediction from sequence through graph convolutional network and predicted contact map," *Journal of cheminformatics* **13**(1), 1–10 (2021).

[30] J. Meiler, M. Müller, A. Zeidler, *et al.*, "Generation and evaluation of dimension-reduced amino acid parameter representations by artificial neural networks," *Molecular modeling annual* **7**(9), 360–369 (2001).

[31] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks.," *Proceedings of the National Academy of Sciences* **89**(22), 10915–10919 (1992).

[32] N. Brandes, D. Ofer, Y. Peleg, *et al.*, "ProteinBERT: a universal deep-learning model of protein sequence and function," *Bioinformatics* **38**(8), 2102–2110 (2022).

[33] M. Heinzinger, A. Elnaggar, Y. Wang, *et al.*, "Modeling aspects of the language of life through transfer-learning protein sequences," *BMC bioinformatics* **20**(1), 1–17 (2019).

[34] T. Sterling and J. J. Irwin, "ZINC 15 – ligand discovery for everyone," *Journal of Chemical Information and Modeling* **55**, 2324–2337 (2015).

[35] V. P. Dwivedi, L. Rampášek, M. Galkin, *et al.*, "Long range graph benchmark," *Advances in Neural Information Processing Systems* **35**, 22326–22340 (2022).

[36] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907* (2016).

[37] P. Veličković, G. Cucurull, A. Casanova, *et al.*, "Graph attention networks," *arXiv preprint arXiv:1710.10903* (2017).

[38] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems* **30** (2017).

[39] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, (2019).

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980* (2014).

[41] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International conference on machine learning*, 807–814 (2010).

[42] N. Srivastava, G. Hinton, A. Krizhevsky, *et al.*, "Dropout: a simple way to prevent neu-

ral networks from overfitting," *The journal of machine learning research* **15**(1), 1929–1958 (2014).

[43] M. Zaheer, S. Kottur, S. Ravanbakhsh, *et al.*, "Deep sets," *Advances in neural information processing systems* **30** (2017).

[44] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE.," *Journal of machine learning research* **9**(11) (2008).

[45] F. Scarselli, M. Gori, A. C. Tsoi, *et al.*, "The graph neural network model," *IEEE Trans. on neural networks* **20**(1), 61–80 (2008).

[46] J. Gilmer, S. S. Schoenholz, P. F. Riley, *et al.*, "Neural message passing for quantum chemistry," in *International conference on machine learning*, 1263–1272, PMLR (2017).

[47] F. Yang, K. Fan, D. Song, *et al.*, "Graph-based prediction of Protein-protein interactions with attributed signed graph embedding," *BMC bioinformatics* **21**(1), 1–16 (2020).

[48] Y. Moreau and L.-C. Tranchevent, "Computational tools for prioritizing candidate genes: boosting disease gene discovery," *Nature Reviews Genetics* **13**(8), 523–536 (2012).

[49] H. Zhou, J. F. Beltrán, and I. L. Brito, "Functions predict horizontal gene transfer and the emergence of antibiotic resistance," *Science Advances* **7**(43), eabj5056 (2021).

[50] V. Gligorijević, P. D. Renfrew, T. Kosciolek, *et al.*, "Structure-based protein function prediction using graph convolutional networks," *Nature communications* **12**(1), 3168 (2021).

[51] C. Zhao, T. Liu, and Z. Wang, "PANDA2: protein function prediction using graph neural networks," *NAR genomics and bioinformatics* **4**(1), lqac004 (2022).

[52] J. X. Hu, C. E. Thomas, and S. Brunak, “Network biology concepts in complex disease comorbidities,” *Nature Reviews Genetics* **17**(10), 615–629 (2016).

[53] A. Rai, P. Shinde, and S. Jalan, “Network spectra for drug-target identification in complex diseases: new guns against old foes,” *Applied Network Science* **3**, 1–18 (2018).

APPENDIX

A. Graph Architectures

A.1 Graph Neural Networks

Graph Neural Networks (GNNs) are a class of neural networks that operate on data defined over graphs. Since their introduction (45), GNNs have shown outstanding results in a broad range of applications, from *computational chemistry* (46) to *protein folding* (6). The key idea is to exploit the inductive bias induced by the topology of graph-structured data to perform graph representation learning tasks.

A graph $G = (V, E)$ is a structure that consists of a set V of n nodes and a set of edges E. In this context, each node $v \in V$ is equipped with a d-dimensional feature vector $\mathbf{x}_v$, and these can be grouped into a feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ by stacking all the $n = |V|$ feature vectors vertically. The connectivity structure of G is fully captured by the adjacency matrix $\mathbf{A}$, in which the entry i, j of $\mathbf{A}$ is equal to 1 if node i is connected to node j and 0 otherwise. In GNNs, each layer consists of a nonlinear function that maps a feature matrix into a new (hidden) feature matrix, accounting for the pairwise relationships in the underlying graph captured by its connectivity. Formally,

$$\mathbf{H}^{(l)} = f(\mathbf{H}^{(l-1)}; \mathbf{A}) \tag{2}$$

where $\mathbf{H}^{(l)}$ is the hidden feature matrix at layer l and $\mathbf{H}^{(0)} = \mathbf{X}$. Among the plethora of neural architectures that have this structure, one of the most popular is the Graph Convolutional Network (36), which implements Equation 2 as

$$\mathbf{H}^{(l)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}) \tag{3}$$

where $\mathbf{W}^{(l)}$ is a learnable weight matrix, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\tilde{\mathbf{D}}$ is a diagonal matrix whose entries are $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ and σ is a pointwise nonlinear activation function (for example, Sigmoid, Tanh, ReLU).

A.2 Graph Attention Network

The Graph Attention Network (GAT) (37) is a type of GNN that uses attention mechanisms to capture dependencies between nodes in a graph. The key idea behind GATs is to learn a different weight for each neighboring node in the graph using a shared attention mechanism. This allows a GAT to attend to different parts of the graph when computing the representation of each node. The GAT layer can be mathematically expressed as

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l)}\mathbf{W}^{(l)}\mathbf{h}_j^{(l)}\right) \tag{4}$$

where $\mathbf{h}_i^{(l)}$ denotes the representation of node i at layer l, $\mathcal{N}(i)$ represents the set of neighbouring nodes of i, $\alpha_{ij}^{(l)}$ represents the attention weight between nodes i and j at layer l, $\mathbf{W}^{(l)}$ is the weight matrix at layer l, and σ is the activation function. The coefficients computed by the attention mechanism can be expressed as:

$$\alpha_{ij} = \frac{\exp\Big(\mathrm{LeakyReLU}\Big(\mathbf{a}^\top[\mathbf{W}^{(l)}\mathbf{h}_i^{(l)}||\mathbf{W}^{(l)}\mathbf{h}_j^{(l)}]\Big)\Big)}{\sum_{k\in\mathcal{N}(i)}\exp\Big(\mathrm{LeakyReLU}\Big(\mathbf{a}^\top[\mathbf{W}^{(l)}\mathbf{h}_i^{(l)}||\mathbf{W}^{(l)}\mathbf{h}_k^{(l)}]\Big)\Big)} \tag{5}$$

where $[\cdot||\cdot]$ denotes concatenation, $\mathbf{a}^\top$ is a trainable weight vector, and $\mathrm{LeakyReLU}$ is the Leaky Rectified Linear Unit activation function.

A.3 GraphSAGE

GraphSAGE (38) is a type of GNN that learns node representations by aggregating information from the local neighborhood of each node. GraphSAGE learns a set of functions to aggregate the representations of a node's neighbors, and then combine them with the node's own representation to compute its updated representation. The GraphSAGE layer can be mathematically expressed as

$$h_i^{(l+1)} = \sigma\left(\mathbf{W}^{(l)}\cdot\mathrm{CAT}\left(\mathrm{AGG}\left(\mathbf{h}_j^{(l)} : j\in\mathcal{N}(i)\right), \mathbf{h}_i^{(l)}\right)\right) \tag{6}$$

where $\mathbf{h}_i^{(l)}$ denotes the representation of node i at layer l, $\mathcal{N}(i)$ represents the set of neighbouring nodes of i, AGG is a learnable aggregation function that combines the representations of a node's neighbors, CAT is the concatenation operation, $\mathbf{W}^{(l)}$ is the weight matrix at layer l, and σ is the activation function.

B. Distance Functions

The proposed approach is to map graphs into a continuous space so that the distance between embedded points is correlated to the distance between the original graphs measured by the TM-score. We explored different distance functions in the embedding space, and we give here their definitions. Given a pair of vectors **p** and **q** of dimension k, the definitions of the Manhattan, Euclidean, Square and Cosine distances are as follows:

$$\text{Manhattan: } d(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=0}^{k} |p_i - q_i|$$

$$\text{Euclidean: } d(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_2 = \sqrt{\sum_{i=0}^{k} (p_i - q_i)^2}$$

$$\text{Square: } d(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_2^2 = \sum_{i=0}^{k} (p_i - q_i)^2$$

$$\text{Cosine: } d(\mathbf{p},\mathbf{q}) = 1 - \tfrac{\mathbf{p}\cdot\mathbf{q}}{\|\mathbf{p}\|\|\mathbf{q}\|} = 1 - \tfrac{\sum_{i=0}^{k} p_i q_i}{\sqrt{\sum_{i=0}^{k} p_i^2}\sqrt{\sum_{i=0}^{k} q_i^2}}$$

C. Datasets

C1. Kinase Proteins

We downloaded the human proteome from UniProt [5] and sub-selected 512 protein kinases. We also used UniProt to download the PDB files for the kinases.

C2. SCOPe v2.07

The 40% identity filtered subset of SCOPe v2.07[6] is used to train and validate our approach. Out of the total of 14,323 domains, 1,058 domains were removed during the data collection process. The remaining 13,265 domains were used for training and testing. For both datasets, we computed ground truth TM-scores by performing all-against-all comparisons using TM-align (15). We used 80% of the comparisons for training and 20% for testing. We repeated all the experiments with 3 different seeds.

---

[5] https://www.uniprot.org

[6] https://scop.berkeley.edu/help/ver=2.07

D. Additional experiments and details

D1. TM-scores Predictions

We employed the trained GAT architectures from Table 2 to predict the TM-scores for the kinase pairs in the test set. Results of this evaluation, measured by Pearson correlation between model predictions and true TM-scores, are shown in Table 5. Using features learned by LLMs exhibits superior performance compared to other feature extraction methods. The highest score is reached with LSTM-extracted features and Euclidean geometry of the embedding space.

Table 5 Pearson correlation coefficients between predicted and actual TM-scores for the GAT model for different choices of node features and distance functions. We use gold ●, silver ●, and bronze ● colors to indicate the first, second and third best performances, respectively. The highest score is reached with LSTM-extracted features and Euclidean geometry.

Feature	**Distance**			
	`Cosine`	`Euclidean`	`Manhattan`	`Square`
One-Hot	0.661	0.384	0.637	0.226
Physicochemical	0.463	0.358	0.534	0.166
BLOSUM	0.484	0.658	0.761	0.468
BERT features	0.849	0.870	0.837	0.785
LSTM features	0.861	0.879	0.858	0.839

D2. TM-scores Inference Times

Table 6 provides inference times for the different graph models and TM-align (15). We show the inference times on GPU and CPU for the graph models, and CPU time for TM-align. Time estimates for different percentages of the test set (20%, 10%, 5%) are reported. For the graph models, we also report standard deviations by estimating the times over 5 different runs. The GNN architectures are significantly faster than TM-align, even on CPU. Our approach represents a fast (Table 6) and accurate (Table 5) way to compute protein structural similarities even on large-scale datasets.

Table 6 Wall-clock estimates for the GNN models and TM-align on different percentages of the test set. Among the GNNs, GAT is the slowest at computing TM-scores, followed by GraphSAGE and GCN, both on GPU and CPU. However, TM-score computation with any of the GNN architectures is significantly faster than TM-align, even on CPU.

Test Size (%)	Model	GPU Inference (s)	CPU Inference (s)
26164 (20%)	GCN	88.3 ± 2.04	474.78 ± 1.98
	GAT	125.8 ± 2.26	1570.1 ± 3.21
	GraphSAGE	98.2 ± 3.46	618.2 ± 2.56
	TM-align	-	$57659.3 \approx 16$ hr
13082 (10%)	GCN	49.2 ± 0.53	231.1 ± 3.01
	GAT	59.3 ± 2.34	773.6 ± 3.14
	GraphSAGE	49.3 ± 0.04	313.9 ± 2.23
	TM-align	-	$29156.2 \approx 8$ hr
6541 (5%)	GCN	23.2 ± 0.18	119.6 ± 1.76
	GAT	30.1 ± 0.70	3882 ± 3.26
	GraphSAGE	25.6 ± 1.42	153.1 ± 3.01
	TM-align	-	$15019.9 \approx 4$ hr

D3. MSE Results on SCOPe Proteins

Table 7 reports the MSE scores for different distance functions and LSTM features on the SCOPe dataset. The best MSE is again reached with LSTM-extracted features and Euclidean geometry of the embedding space.

Table 7 MSE scores for different distance functions and LSTM features on the SCOPe dataset. We use gold ●, silver ●, and bronze ● colors to indicate the first, second and third best performances, respectively.

Model	Distance	MSE
GAT	`Cosine`	0.008048
	`Euclidean`	0.006294
	`Manhattan`	0.010655
	`Square`	0.008793

D4. Computational Resources, Code Assets and Data Availability

In all experiments we used NVIDIA® Tesla V100 GPUs with 5,120 CUDA cores and 32GB GPU memory on a personal computing platform with an Intel® Xeon® Gold 5218 CPU @ 2.30GHz

CPU running Ubuntu 18.04.6 LTS. Our code and the datasets used for evaluations are available on GitHub[7].

E. Bioinformatics applications

There are several areas of bioinformatics research where structural representation of proteins finds useful applications. We now give a few examples.

E1. Protein-protein Interaction

Proteins rarely carry out their tasks in isolation, but interact with other proteins present in their surroundings to complete biological activities. Knowledge of protein–protein interactions (PPIs) helps unravel cellular behaviour and functionality. Generating meaningful representations of proteins based on chemical and structural information to predict protein-pocket ligand interactions and protein-protein interactions is an essential bioinformatics task (47).

E2. Protein Function

The structural features of a protein determine a wide range of functions: from binding specificity and conferring of mechanical stability, to catalysis of biochemical reactions, transport, and signal transduction. While the experimental characterization of a protein's functionality is a challenging and intense task (48), exploiting graph representation learning ability to incorporate structural information facilitates the prediction of protein function (49; 50; 51).

E3. Small Molecules

The design of a new drug requires experimentalists to identify the chemical structure of the candidate drug, its target, its efficacy and toxicity and its potential side effects (52; 53). Because such

---

[7] https://github.com/cecca46/neural_embeddings

processes are costly and time consuming, drug-discovery pipelines employ in silico approaches. Effective representations of protein targets of small molecules (drugs) has the potential to dramatically speed up the field of drug discovery.