PREDICTING TRANSMEMBRANE PROTEIN TOPOLOGY FROM 3D STRUCTURE

Sitong Chen

Technical University of Denmark
s230027@dtu.dk

Xiaopeng Mao

Technical University of Denmark
s194408@dtu.dk

ABSTRACT

This paper presents a novel approach to infer protein topology using the state-of-the-art graph neural network (GNN), SchNet. The model is trained on the same dataset used to develop the recent DeepTMHMM model with 5-fold cross-validation. Unlike the conventional approaches based on using only the protein sequences or the α-carbons as features, we have decoded our classifier in this way, so all atom-level embeddings are used. Without applying any pre-trained weight, the final results have shown great potential that GNNs can be used for topological predictions.

***Index Terms*—** 3D protein structure, topology, graph neural network

***Github Repository Link*—** https://github.com/si-tong-chen/predicting-transmembrance-protein-topology-from-3D-structure

1. INTRODUCTION

Protein topology tells about the location of a particular protein, which can be inside, outside, or in-between the cell membrane (transmembrane protein) as illustrated in Figure 1. A transmembrane protein might serve as a receptor for cell-to-cell communication while a free-moving protein outside or inside a cell might play an important role in cellular transportation [1]. Currently, many machine learning-based models have been developed to predict this very topology by using 1D protein sequences, 3D protein structures, or both features. In [2], Bernsel *et al.* used an ensemble of 5 different 1D sequence machine learning models to predict the protein topology. In another study, Dobson *et al.* achieved a better result using a similar approach but with 10 models instead [3]. Recently, Hallgren *et al.* and M. Bernhofer *et al.* achieved excellent performances with the DeepTMHMM and TMbed models based on pre-trained protein Language Models (pLMs), respectively [4, 5]. In this study, we present a Graph Neural Network (GNN)-based method that directly uses 3D structural information obtained from the well-known Alphafold model together with an indirect application of the 1D protein sequence.

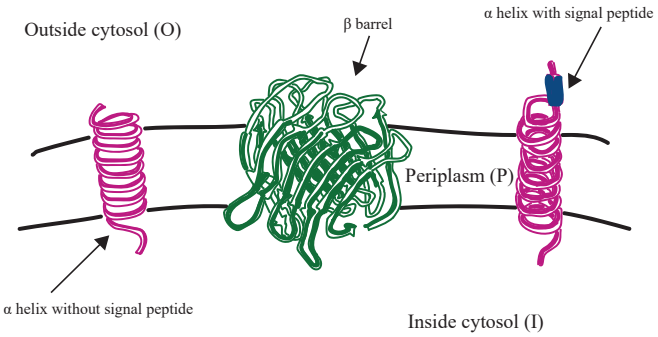

Fig. 1: Simple illustration explaining protein topologies. A protein can have different topologies depending on where it is. Please notice that the β barrel protein shown in the middle has more complicated topologies as its bottom part frequently changes location. On the other hand, α helices have relatively simpler topologies but they might have signal peptide bound to them, which needs to be distinguished. Illustration inspired by [1].

2. METHODOLOGY

Figure 2 presents the general approach throughout this study. The key parts of the process are described in the subsequent sections.

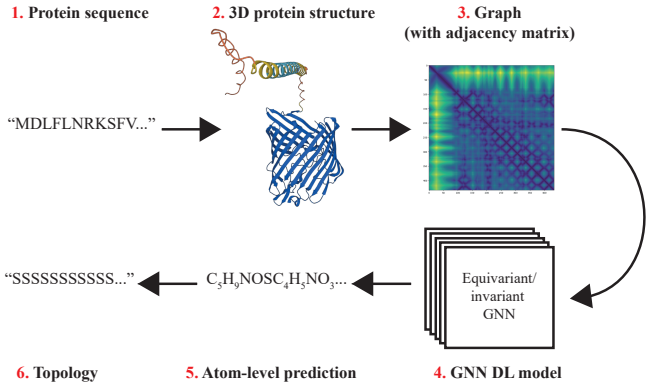

Fig. 2: Methodology throughout the project. All blocks are numbered for later reference. The 3D protein structure in block 2 is from AlphaDB 3D viewer.

2.1. Dataset & 3D protein structure

The 1D protein sequences are the same as in the DeepTMHMM study. The corresponding 3D structures are predicted by Alphafold and acquired from the Alphafold Data Base (AlphafoldDB) [6, 7]. Protein sequences without their 3D structures are excluded from the study, Table 1 provides an overview of which protein types are included and whether or not their 3D predictions are available from the AlphaDB.

Table 1: Number and name of each protein type used in the DeepTMHMM study. *TM*: Transmembrane, *SP*: Signal Peptide. The numbers without parentheses indicate sequences with both 1D sequences and 3D structures available while the numbers within parentheses indicate proteins with only 1D sequences.

Protein type	alpha TM	alpha SP+TM	beta barrel	Globular	SP+Globular
Amount	383 (387)	102 (106)	82 (82)	1,982 (2,000)	994 (1,000)

The 3D protein structures are stored in `.pdb` files. Each `.pdb` file contains the protein 1D sequence, the atoms of the protein, the atom coordinates, and the prediction uncertainties. A graph in the form of an adjacency matrix can then be constructed based on the atom coordinates (see block **3** in Figure 2). The atom types include carbon (C), nitrogen (N), oxygen (O), and sulfur (S) while hydrogen (H) is omitted probably due to its relatively low importance. Upon data inspection, we observed that every residue (amino acid in bound form) would start with the nitrogen atom in the graph. This was later exploited for the decoder implementation used between blocks 5 and 6 in Figure 2.

2.2. Graph Neural Network

Generally, specific models are needed to process the graphs. A Graph Neural Network (GNN) is one of the models that takes a graph as input and returns as outputs a set of the so-called *node embeddings* corresponding to all the nodes of the graph and a *graph embedding* for the whole graph [8]. More generally, the GNN uses an AGGREGATE and an UPDATE function to perform the embedding calculations:

$$m_{\mathcal{N}(u)} = \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\}), \quad (1)$$

$$\text{UPDATE}(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}) = \sigma\left(\mathbf{W}_{\text{self}}\mathbf{h}_u + \mathbf{W}_{\text{neigh}}\mathbf{m}_{\mathcal{N}(u)}\right). \quad (2)$$

where u is the current node being calculated, $\mathcal{N}(u)$ is the neighborhood of u. v is a neighbor node to u and thus belongs to $\mathcal{N}(u)$, $\mathbf{h}_u$ is a so-called hidden node embedding, σ is a nonlinear function that can be e.g. a ReLU function while $\mathbf{W}_{\text{self}}$ and $\mathbf{W}_{\text{neigh}}$ are trainable parameters [8]. In the context of protein topology, we focus on the node embeddings. Figure 3 shows how the node embeddings and the graph embedding are obtained from the graph using the message passing mechanism.

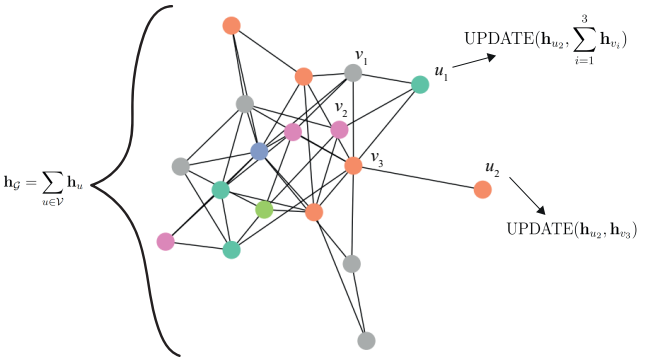

Fig. 3: Random graph illustrating the basic message passing of GNN. Notice that the AGGREGATE operation is part of the UPDATE function (indicated by a sum) and depends on the number of neighbors. The graph embedding is then a sum of all the node embeddings.

The output of an *invariant* GNN always has the same order while the output of an *equivariant* GNN has the same order as the input order. In this study, we applied the state-of-the-art GNNs such as SchNet (invariant), EGNN (equivariant), and GCPNet (equivariant) to the topological task. However, SchNet was the only GNN that we managed to get meaningful results from.

2.3. Atom-to-residue translation

GNN outputs node embeddings correspond to each atom of the protein graph. This means that the output dimension is much greater compared to the topological labels, which are on the residue-level. To translate the embeddings from the atom-level to the residue-level, we devised a *major voting* approach, which is explained in Figure 4. The gist of major voting is that all atoms give a topological prediction. The topology that is agreed upon by all the atoms within the residue would be selected to represent the topology of the residue. In case of a tie between the atom predictions, either candidate was selected to represent the residue-level topology.

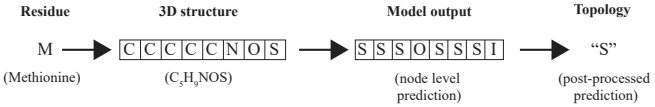

Fig. 4: Major voting approach used in this study. The final topological prediction is based on the majority of the atom-level topological predictions.

In practice, the atoms within a residue were grouped using the protein sequence and the residue chemical formulas. The starting point of every residue was found by searching

the nitrogen atom, which always served as the starting atom of each residue in the 3D graph as mentioned earlier. Subsequently, we could gather the other atoms within the residue to vote for the residue topology. More importantly, we created the atomic topology labels ourselves to align with the GNN predictions in the decoding stage before output.

Another approach is to train the GNN using the α-carbon embedding from each residue [9]. An α-carbon is the central atom linking to an amino group and a carboxyl group within an amino acid [1]. In this way, the output dimension based on the α-carbons would have the same dimension as the protein sequence. However, we only achieved similar or worse performance by using α-carbons compared to using the aforementioned major voting approach.

2.4. Training data distribution and Baseline model

The residue classes are *signal peptide (S)*, *inside cytosol (I)*, *alpha membrane part (M)*, *beta membrane part (B)*, *periplasm (P)* and *outside cell (O)* [4]. This study uses a 5-fold cross-validation (CV) to assess the model performance. We used the same splits from the DeepTMHMM study in such a setup so that the model was trained on the first three sets, validated on the fourth set, and tested on the fifth set. This process had been repeated five times and each time the model was validated and tested on different sets. The hyperparameter tuning was conducted only on the first fold setup because no major performance difference was observed with the other folds.

The model performance was compared to a baseline classifier, which predicted all test observations as belonging to the most frequently appearing class in the training set. We observed that in all the CV folds the class *inside cytosol (I)* was the most frequently appearing class and the class distribution was very similar for all the CV folds. The comparison was conducted using McNemars test [10]. Figure 5 shows the distribution of all the label classes in the first fold setup.

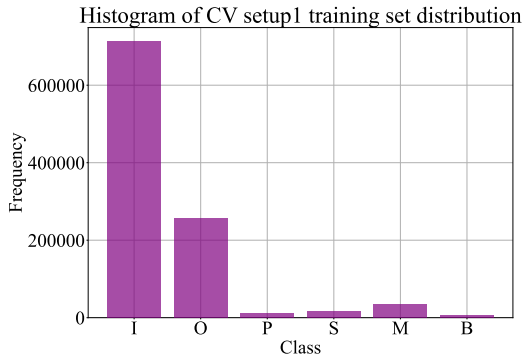

Fig. 5: Class distribution in the first fold of the 5-fold cross-validation. This distribution indicates that most protein parts intracellular proteins. The distributions in the other CV folds are very similar (not shown).

2.5. Model specification

We applied a SchNet model with 6 layers, a hidden embedding size of 128, 128 convolutional filters, a maximum number of 32 neighbor nodes to be included for the message passing, and a dropout set to 0.8 right before the fully connected layers. A batch size of 1 was chosen because this gave the best and most stable performance. The Adam optimizer was used with a learning rate of $3 \cdot 10^{-4}$ and $L2$ regularization of $1 \cdot 10^{-4}$. Additionally, an exponential *learning rate scheduler* was applied with an exponential decay rate of 0.1. The learning scheduler was applied when the validation loss stopped decreasing, which was detected using *early stopping*.

In [5], M. Bernhofer and B. Rost applied a Gaussian smoothing filter with a kernel size of 7 and standard deviation (σ) of 1 to the node embeddings for each class before converting them into class probabilities. The intention of Gaussian smoothing during training was to eliminate the irregular and sudden appearance of residues. As for our study, we also observed that the residue topology predictions would sometimes change abruptly. Hence, we applied a Gaussian filter during the training and validation stage with a kernel size of 29 and σ of 5. Please notice that our input graph nodes are at atom-level, which means that a larger kernel size and σ are needed to achieve the same smoothing effect on residue-level.

We had experimented with all the existing pre-trained weights provided by the *ProteinWorkshop* framework [11], which were from the tasks *inverse folding*, *predicted local distance difference test (pLDDT) prediction*, *sequene denoising*, *structure denoiseing* and *torsional denoising*. Unfortunately, none of these weights improved the training. The Python library `wandb` was employed to monitor the learning and loss curves. The models were trained using Tesla A100 (32 GB and 80 GB) owned by the Technical University of Denmark (DTU) Compute department.

3. RESULTS

This section presents the main results obtained from SchNet and other relevant test results. Figure 6 shows the training and validation losses and residue-level accuracies for the five SchNet models. The residue-level accuracies are evaluated by directly comparing them to the class labels. It can be seen that all the loss and accuracy curves have similar tendencies, indicating that the variability is small in each CV setup.

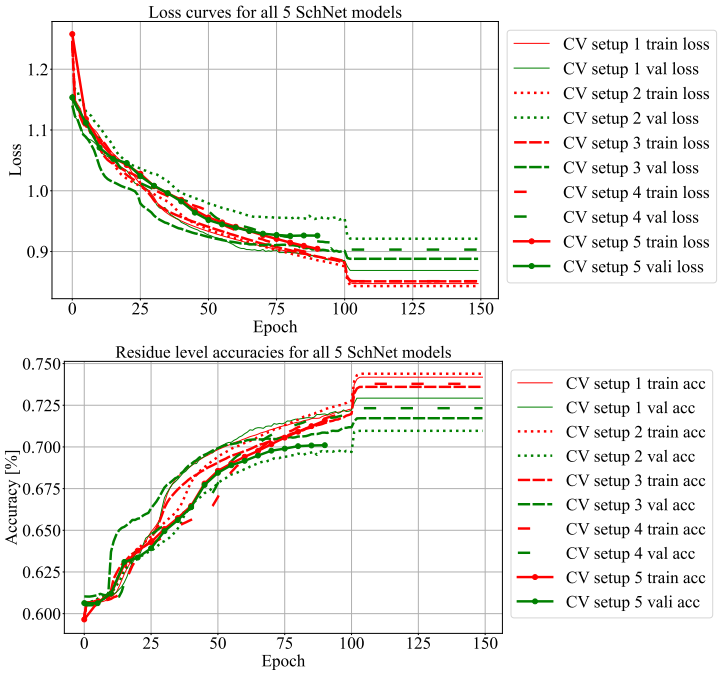

Fig. 6: a): Loss curves for the 5 SchNet models trained and validated on each of the five CV folds. b): Accuracy curves for the 5 SchNet models trained and validated on each of the five CV folds. The abrupt jump around epoch 100 is caused by the exponential decaying learning scheduler. Please notice that the training for CV setup 5 was early-stopped around epoch 90.

3.1. Topological evaluation

The evaluation criteria for two same protein topologies are defined according to [4] as the following:

- The N-terminal topology must be the same.
- The predicted labels and the ground truth labels must overlap with 5 residues for α-helices and 2 for β-strands (sub-part of β-barrel).

The test results are gathered from the five test sets of the 5-fold CV, so the topological predictions are evaluated using the above-mentioned criteria for all the five protein types. In Table 2, it can be seen that only very few topologies have been predicted for *Globular* and *SP+Globular* by the trained SchNet models. However, the average correctly predicted residues per protein is still much higher for SchNet than for the baseline. The average correctly predicted residues per protein are also calculated for the DeepTMHMM predictions to show how far away our models are from achieving a similar performance as DeepTMHMM.

Table 2: Topological and residue-level evaluation performed on the protein types *alpha TM*, *SP+TM*, *beta barrel*, *Globular* and *SP+Globular* presented in Table 2. Please notice that the overlap criterion is set to be 5 by default for the topological evaluation of *Globular* and *SP+Globular* as these proteins contain both α-helices and β-sheets [1].

Protein types / Metrics	alpha TM	alpha SP+TM	beta barrel	Globular & SP+Globular
Correct topology by SchNet [%]	0	0	0	3.6
Correct topology by baseline [%]	0	0	0	0
Correct topology by DeepTMHMM [%]	83.5	92.5	80.3	*95.6
Average correct residue labels per protein by SchNet [%]	59.5	50.8	26.4	75.2
Average correct residue labels per protein by baseline [%]	39.8	17.3	0.0	66.6
Average correct residue labels per protein by DeepTMHMM [%]	88.4	94.5	87.6	96.9

*Calculated by our own using an overlap of 5 residues.

3.2. Comparison between baseline and SchNet using McNemars test

To verify the validity of our trained GNNs, we compared the baseline with the SchNet models using the McNemars test. In Table 3, the comparison is presented for each of the 5-fold CV setups. The SchNet models have better residue-level accuracies compared to the baseline. Meanwhile, 0 is always outside the 95 % confidence intervals (CIs), and the p-values are always strongly significant as well. Overall, all this indicates that the SchNet models are better suited for topological prediction than the baseline.

Table 3: McNemars test and the corresponding 95 % confidence intervals (CIs) between the baseline and the trained SchNet model for each CV setup. The trained SchNet is always superior regardless of which setup.

CV fold / Metrics	CV setup 1	CV setup 2	CV setup 3	CV setup 4	CV setup 5
Baseline residue-level accuracy [%]	68.0	69.3	69.4	67.5	69.3
SchNet residue-level accuracy [%]	75.2	76.9	75.5	76.3	75.6
Comparison between baseline and SchNet (p-value)	0.00	0.00	0.00	0.00	0.00
Comparison between baseline and SchNet (95 % CI)	[-0.074, -0.071]	[-0.078, -0.075]	[-0.062, -0.059]	[-0.089, -0.086]	[-0.063, -0.061]

Finally, it is noteworthy that we initially used a hold-out approach to train and validate the GNN model on only *Globular* and *SP+Globular* proteins and tested on the remaining three protein types. However, the result was worse compared to the 5-fold CV.

3.3. Results from α-carbon method and other models

Figure 7 shows the SchNet performance trained on CV setup 1 using the previously mentioned α-carbon approach. While

the loss curves converge at a lower value compared to the major voting approach, the actual number of the correctly predicted residue and the number of correct topological predictions do not improve. These results are not included to avoid confusion. Figure 8 shows the performance of GCPNet and EGNN trained on CV setup 1. It can be seen that both models fail to learn the topology from the data because they have approximately constant losses and accuracies. It has to be emphasized that we were unable to train GCPNet and EGNN with large batch sizes due to their demanding memory requirements. The numerical prediction results from the two models are below the baseline predictions and thus not presented here.

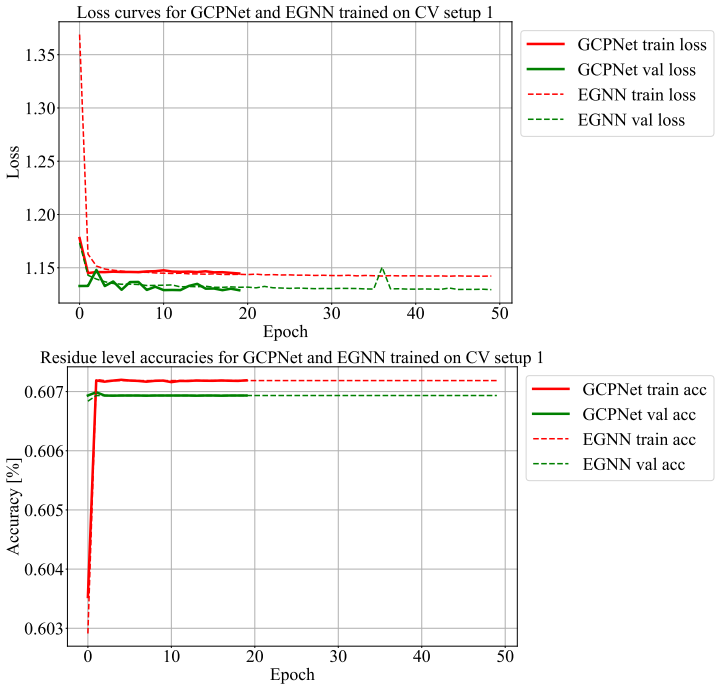

Fig. 8: Training and validation curves of GCPNet and EGNN trained and validated on CV setup 1 with the proposed major voting method. Both models seem to have failed to learn the representation from the CV setup 1 dataset. Gaussian smoothing is applied to the models.

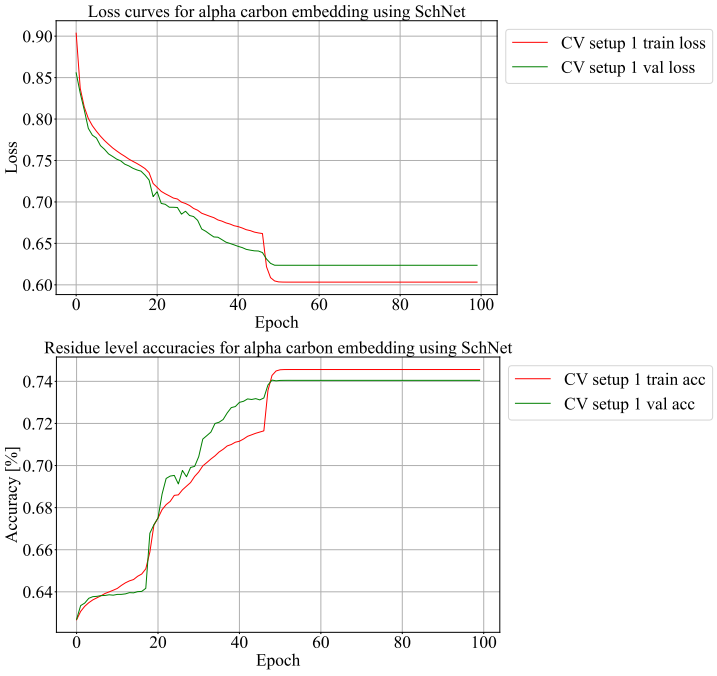

Fig. 7: SchNet trained and validated on CV setup 1 using α-carbon approach with Gaussian smoothing. The hyperparameters are fine-tuned and an exponential learning schedule is applied.

4. DISCUSSION

The comparison between the trained SchNet models and the baseline model shows that SchNet models are better at determining the residue-wise topology classes. Nonetheless, the overall topological prediction is far from satisfactory. One of the main reasons for this could be that the model was trained from scratch without using any pre-trained weight. The current state-of-the-art models like DeepTMHMM and TMbed are based on sophisticated protein language models (pLMs), which are pre-trained on billions of protein sequences [5]. Furthermore, the time and computational constraints prevented us from experimenting with other GNNs, which might be more well-suited for the topology task. Finally, the amounts of the protein types *alpha TM*, *alpha SP+TM* and *beta barrel* are way less compared to *Globular* and *SP+Globular* as shown in Table 1. This data imbalance most likely has contributed to the fact that the model is better at predicting the topologies for *Globular* and *SP+Globular* than for *alpha TM*, *alpha SP+TM* and *beta barrel*. Finally, it is important to emphasize that the 3D structures predicted by Alphafold also contain uncertainties, which inevitably would limit the performance of the models that are trained upon AlphaDB.

5. CONCLUSION

In this study, we combined knowledge from biology, dynamic programming, and deep learning to apply state-of-the-art GNNs such as SchNet, EGNN, and GCPNet to identify protein topology. Specifically, we developed the major voting method to infer the residue-wise topological labels by using a consensus of the atom-level topological predictions. We also explored another well-known method based on extracting the feature from the α-carbons, but this method did not show to improve the performance compared to our major voting approach. Within the computational budget, only SchNet gave the best results although its performance was still far away from the state-of-the-art protein models such as DeepTMHMM and TMbed. Nonetheless, with more pretrained GNNs becoming available concerning the topological task, the performance would certainly be improved soon.

6. ACKNOWLEDGMENT

We would like to sincerely thank Felix Teufel for his supervision, advice, and provision of relevant materials during the project.

7. REFERENCES

[1] Bruce Alberts, Dennis Bray, Karen Hopkin, Alexander D Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter, *Essential cell biology*, Garland Science, 2015.

[2] Andreas Bernsel, Håkan Viklund, Aron Hennerdal, and Arne Elofsson, "Topcons: consensus prediction of membrane protein topology," *Nucleic acids research*, vol. 37, no. suppl_2, pp. W465–W468, 2009.

[3] László Dobson, István Reményi, and Gábor E Tusnády, "Cctop: a consensus constrained topology prediction web server," *Nucleic acids research*, vol. 43, no. W1, pp. W408–W412, 2015.

[4] Jeppe Hallgren, Konstantinos D Tsirigos, Mads Damgaard Pedersen, José Juan Almagro Armenteros, Paolo Marcatili, Henrik Nielsen, Anders Krogh, and Ole Winther, "Deeptmhmm predicts alpha and beta transmembrane proteins using deep neural networks," *BioRxiv*, pp. 2022–04, 2022.

[5] Michael Bernhofer and Burkhard Rost, "Tmbed: transmembrane proteins predicted through language model embeddings," *BMC bioinformatics*, vol. 23, no. 1, pp. 326, 2022.

[6] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al., "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[7] Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, et al., "Alphafold protein structure database: massively expanding the structural coverage of protein-sequence space with high-accuracy models," *Nucleic acids research*, vol. 50, no. D1, pp. D439–D444, 2022.

[8] William L Hamilton, *Graph representation learning*, Morgan & Claypool Publishers, 2020.

[9] Anonymous, "Evaluating representation learning on the protein structure universe," in *Submitted to The Twelfth International Conference on Learning Representations*, 2023, under review.

[10] Tue Herlau, Mikkel N Schmidt, and Morten Mørup, "Introduction to machine learning and data mining," *Lecture notes of the course of the same name given at DTU (Technical University of Denmark)*, 2022.

[11] Arian Jamasb, "Pre-trained weights for protein workshop (0.0.1) [data set]," Zenodo, 2023, https://doi.org/10.5281/zenodo.8287754.