Supplementary Material for: Multi-layer sequential network analysis improves protein 3D structural classification

Khalique Newaz[1,2,3,7], Jacob Piland[1,2,3], Patricia L. Clark[4], Scott J. Emrich[6], Jun Li[5], and Tijana Milenković[1,2,3,*]

[1]Department of Computer Science and Engineering
[2]Center for Network and Data Science
[3]Eck institute for Global Health
[4]Department of Chemistry and Biochemistry
[5]Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46556, USA
[6]Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996, USA
[7]Center for Data and Computing in Natural Sciences (CDCS), Institute for Computational Systems Biology, Universität Hamburg, Hamburg, 20146, Germany
[*]Corresponding author (email: tmilenko@nd.edu)

I Supplementary Sections

S1. Procedure to identify sequence non-redundant protein chains

Here, we complement Section 2.1 from the main paper by explaining how we obtain a non-redundant set of protein chains from a given set of protein chains. Given a set of protein chains, to select a set of sequence non-redundant protein chains, we only keep a set of protein chains such that each chain in the set is less than 90% sequence identical to any other chain in the set. Additionally, we make sure that the resulting set of chains keeps the maximum possible number of CATH and SCOPe protein domains. Specifically, we use MMSeq2 method-based [13] protein chain clusters data from the PDB. This data contains 63,421 clusters, where protein chains within a cluster are sequence redundant ($\geq$ 90% sequence identity) and proteins across clusters are sequence non-redundant ($<$ 90% sequence identity). We select a non-redundant set of protein chains as follows. If any one of our protein chains belongs to a cluster in which none of our other protein chains are present, we immediately keep it. Alternatively, if it belongs to a cluster having more than one of our proteins chains, then we keep the protein chain that results in the maximum number of CATH and SCOPe protein domains. Besides using MMSeq2 to obtain CATH- and SCOP-related datasets with <90% sequence identity, we use MMSeq2 to also obtain the Scop25% dataset with $\leq$ 25% sequence identity as follows. Given all SCOPe protein domains, we cluster the corresponding protein sequences using MMSeq2 based on 25% sequence identity, where protein chains within a cluster have $\geq$ 25% sequence identity and proteins across clusters have $<$ 25% sequence identity. Then, we follow the same procedure to select a non-redundant set of proteins as we do for the 90% sequence identity choice above, in order to obtain the Scop25% dataset.

S2. Features of protein structure networks

Here, we complement Section 2.2 from the main paper by explaining how we extract features from each of the considered four PSN types. Specifically, for each of the considered PSN types, we extract the corresponding feature using graphlets [7], which are established state-of-the-art network-based features. Here, we give a brief overview of graphlets and related concepts based on single-layer unweighted unordered networks. These concepts have been extended to other network types, such as, single-layer weighted unordered networks [2], single-layer unweighted ordered networks [5], and multi-layer sequential (or simply multi-layer) unweighted unordered networks [3].

Graphlets are small connected induced non-isomorphic k-node subgraphs of a network [9]. For example, there exist two 3-node graphlets, i.e., a 3-node path and a triangle. Similarly, there exist six 4-node and 21 5-node graphlets. Typically, due to high computational complexity (i.e., running time) of counting graphlets in large networks, up to 5-node graphlets are studied. Consequently, there exist $2 + 6 + 21 = 29$ 3-5-node graphlets. Typically, given a network, a graphlet-based network feature is based on counting the number occurrences of each of the considered graphlets in the network. However, a conceptually more advanced concept of node or edge orbits are often used, which we explain below.

Nodes in a given graphlet can be partitioned into one or more groups, where nodes within a group occupy the same topological position in the graphlet, while nodes across groups belong to different topological positions in the graphlet. For example, in the 3-node path graphlet, the two end nodes occupy the same topological position and are grouped together, while the middle node is in a group of its own. As another example, all three nodes of a triangle are in the same group. Each such group of nodes within a graphlet is called a node orbit. In total, there are 72 node orbits across the 29 3-5-node graphlets. Given the concept of node orbits, the topology of a given network is typically summarized in a node graphlet degree vector matrix (nGDVM) [10], as follows. First, graphlet degree vectors (GDVs) for each node in the network is computed. A GDV of a node counts the number of times the given node participates in each of the considered node orbits. Given this, an nGDVM of a network is defined as the collection of GDVs of each node in the network, where each row of the nGDVM is a GDV of a node and each column of the nGDVM is a node orbit. For 3-5-node graphlets and a network with N nodes, the dimension of the corresponding nGDVM is $N \times 72$.

Similar to nodes, the edges in a graphlet can be partitioned into groups (i.e., edge orbits) depending on their topological positions in the graphlet. In total, there are 68 edge orbits across the 29 3-5-node graphlets. Given edge orbits, the topology of a given network is summarized in an edge graphlet degree vector matrix (eGDVM) [12], which is defined in the similar manner as nGDVM. First, GDVs for each edge in the network is computed. A GDV of an edge counts the number of times the given edge participates in each of the considered edge orbits. Given this, an eGDVM of a network is defined as the collection of GDVs of each edge in the network, where each row of the eGDVM is a GDV of an edge and each column of the eGDVM is an edge orbit. For 3-5-node graphlets and a network with N nodes, the dimension of the corresponding eGDVM is $N \times 68$.

Given the above overview of some of the graphlet-related concepts, we now explain graphlet-based features that we consider for each of the considered four PSN types.

Features corresponding to a single-layer unweighted unordered PSN. Here, we consider both *nGDVM* and *eGDVM* corresponding to 3-5-node graphlets, as we define above. After we compute the nGDVM of a given PSN, following an established approach [14], we compute Pearson correlation between each pair of the considered node orbits (i.e., columns in nGDVM), to obtain a node orbit correlation matrix. Then, we take the upper triangular elements of this matrix as a feature vector. Similarly, after we compute the eGDVM of a given PSN, following an established approach [2], we compute Pearson correlation between each pair of the considered edge orbits (i.e., columns in eGDVM), to obtain an edge orbit correlation matrix. Then, we take the upper triangular elements of this matrix as a feature vector.

Feature corresponding to a single-layer weighted unordered PSN. With the hypothesis that single-layer weighted unordered PSNs can better capture the 3D structure of a protein than single-layer unweighted unordered PSNs, we recently proposed the concept of weighted graphlets (named *weGDVM*) [2], which can extract protein features from single-layer weighted unordered PSNs. Intuitively, weGDVM is similar to eGDVM, with a crucial difference that for a pair of edge and edge orbit, instead of counting how many times the edge participates in the given edge orbit as in eGDVM, it first computes the distribution of edge weights corresponding to the given edge orbit. Then, it compares this distribution of weights with the distribution of edge weights over the whole PSN using the Cramér-von Mises statistic, to obtain a single value for the given pair of edge and edge orbit. Given an *weGDVM*, following an established approach [2], we compute Pearson correlation between each pair of the considered edge orbits (i.e., columns in weGDVM), to obtain a correlation matrix, where we take the upper triangular elements of this correlation matrix as a feature vector.

Feature corresponding to a single-layer unweighted ordered PSN. Original graphlets capture 3D-structural but not sequence information of a protein. Hence, ordered graphlets were proposed that integrate the 3D-structural information with the sequence information of a protein [5]. Ordered graphlets are graphlets that have a node order based on the relative ordering of the corresponding amino acids in the protein sequence. We consider an ordered graphlet-based feature that is based on 3-4-node ordered graphlets, i.e., ordered graphlet frequency vector (oGFV), that performed better than or similar to state-of-the-art protein features in the task of PSC [6]. Given a PSN, position i in an oGFV contains total count of ordered graphlets of type i.

Features corresponding to a multi-layer unweighted unordered PSN. We use the concepts of dy-

namic graphlets [3] and graphlet orbit transition (GoT) [1] to extract features from multi-layer unweighted unordered PSNs. Note that we create multi-layer unweighted unordered PSNs in two different ways (see Section 2.2 in the main paper for details). For each of the two ways of creating such PSNs, we extract features using both dynamic graphlets and GoT. Hence, given a protein domain, we create four features corresponding to this PSN type. Below, we first explain dynamic graphlet-based features for each of the two strategies to create multi-layer PSNs. Then, we explain GoT-based features for each of the two strategies to create multi-layer PSNs.

Dynamic graphlet-based multi-layer PSN feature: Dynamic graphlets extend static graphlets to the dynamic context, where an additional information of the temporal order in which events (i.e., edges) appear in a dynamic network is added on to static graphlets. Dynamic graphlet degree vector (dGDV) counts how many times a node participates in each of the dynamic node orbits of the considered dynamic graphlets. As in the original dynamic graphlets paper [3], we use up to 4-node and 6-event graphlets, which gives 3,727-dimensional dGDV for a node. Below, we explain our procedure to get dynamic graphlet-based features for PSNs created using each of the two multi-layer PSN creation strategies.

Dynamic graphlet-based features of multi-layer PSNs created using strategy 1: To obtain a feature for a multi-layer PSN with N nodes, we compute dGDV for each node in the PSN, which gives us a dGDV matrix (dGDVM) of size $N \times 3,727$. Given all of the dGDVMs for all of the considered protein domains, we first remove all of those dynamic node orbits that are not present in any of the protein domains (i.e., multi-layer PSNs). We find that 3,516 out of 3,727 dynamic node orbits are never observed in any of the multi-layer PSNs of the considered CATH, SCOPe, or Astral protein domains. So, for every multi-layer PSN, we only keep information for $3,727 - 3,516 = 211$ types of dynamic node orbits. Consequently, for a protein domain of length N, this procedure transforms the dGDVM size from a dimension of $N \times 3,727$ to a "reduced" dimension of $N \times 211$.

Given all of the reduced dGDVMs of CATH, we transform each such dGDVM into a feature vector, as follows. First, for each dGDVM, we compute Pearson correlation between each pair of dynamic node orbits (i.e., columns of dGDVM) to obtain a correlation matrix. Then we take the upper triangular elements of this correlation matrix, which gives a feature vector of size $\binom{211}{2} = 22,155$. Second, given all such feature vectors, we perform principal component analysis (PCA), where we keep 90% variation between the vectors. We take this PCA-reduced version of the above feature vectors as feature vectors of the corresponding multi-layer PSNs. For each multi-layer PSN in CATH, this procedure results in a feature vector of size 166.

Given all of the reduced dGDVMs of SCOPe, we do the same as we do above for CATH, in order to obtain feature vectors of the corresponding multi-layer PSNs. For each multi-layer PSN in SCOPe, this procedure results in a feature vector of size 163.

Similarly, given all of the reduced dGDVMs of each of Astral and Scop25% datasets, we do the same as we do above for CATH or SCOPe, in order to obtain feature vectors of the corresponding multi-layer PSNs. For multi-layer PSN in the Astral and Scop25% datasets, this procedure results in a feature vector of size 118 and 123, respectively.

Dynamic graphlet-based features of multi-layer PSNs created using strategy 2: To obtain a feature for a multi-layer PSN with N nodes, we follow the same steps as above for multi-layer PSNs created using strategy 1, which transforms the original dGDVM size for each multi-layer PSN from a dimension of $N \times 3,727$ to a reduced dimension of $N \times 211$.

Given all of the reduced dGDVMs of CATH, we then follow the same steps as we do for multi-layer PSNs created using strategy 1 above. For each multi-layer PSN in CATH, this procedure results in a feature vector of size 270.

Given all of the reduced dGDVMs of SCOPe, we do the same as we do above for CATH, in order to obtain feature vectors of the corresponding multi-layer PSNs. For each multi-layer PSN in SCOPe, this procedure results in a feature vector of size 286.

Similarly, given all of the reduced dGDVMs of each of the Astral and Scop25% datasets, we do the same as we do above for CATH or SCOPe, in order to obtain feature vectors of the corresponding multi-layer PSNs. For multi-layer PSN in the Astral and Scop25% datasets, this procedure results in a feature vector of size 178 and 57, respectively.

GoT-based multi-layer PSN feature: GoT of a node counts for every pair of considered original graphlet orbit types, how many times one orbit type changes (i.e., "transitions") into the other in a dynamic network. As in the original GoT publication [1], and for a fair comparison with dGDV, we use 4-node graphlets with 121 orbit transitions, resulting in a 121-dimensional GoT node feature. Below, we explain our procedure to get GoT-based features for PSNs created using each of the two strategies to create multi-layer PSNs.

GoT-based features of multi-layer PSNs created using strategy 1: To obtain a feature for a multi-layer PSN with N nodes, we compute GoT for each node in the PSN, which gives us a GoT matrix (GoTM) of size $N \times 121$. Given all of the GoTMs for all of the considered protein domains, we remove all of those orbit

transitions that are not present in any of the protein domians (i.e., multi-layer PSNs). We see that 110 out of 121 orbit transitions are never observed in any of the multi-layer PSNs of the considered CATH, SCOPe, or Astral protein domains. So, for every multi-layer PSN, we only keep information for $121 - 110 = 11$ types of orbit transitions. Consequently, for a protein domain of length N, this procedure transforms the GoTM size from a dimension of $N \times 121$ to a reduced dimension of $N \times 11$.

Given all of the reduced GoTMs of domains in CATH, we transform each of such GoTMs into a feature vector, in the similar manner as we do for dynamic graphlets. First, for each GoTM, we compute Pearson correlation between each pair of orbit transitions (i.e., columns of GoTM) to obtain a correlation matrix. Then we take the upper triangular elements of this correlation matrix, which gives a feature vector of size $\binom{11}{2} = 55$. Second, given all such feature vectors, we perform principal component analysis (PCA), where we keep 90% variation between the vectors. We take this PCA-reduced version of the above feature vectors, as feature vectors of the corresponding multi-layer PSNs. For each multi-layer PSN in CATH, this procedure results in a feature vector of size six.

Given all of the reduced GoTMs of SCOPe, we do the same as we do above for CATH, in order to obtain feature vectors of the corresponding multi-layer PSNs. For each multi-layer PSN in SCOPe, this procedure results in a feature vector of size six.

Similarly, given all of the reduced GoTMs of each of the Astral and Scop25% datasets, we do the same as we do above for CATH or SCOPe, in order to obtain feature vectors of the corresponding multi-layer PSNs. For multi-layer PSN in the Astral and Scop25% datasets, this procedure results in a feature vector of size six and five, respectively.

GoT-based features using multi-layer PSNs created using strategy 2: To obtain a feature for a multi-layer PSN with N nodes, we follow the same steps as above for multi-layer PSNs created using strategy 1, which reduces the original GoTM size of each multi-layer PSN from a dimension of $N \times 211$ to a reduced dimension of $N \times 11$.

Given all of the reduced GoTMs of CATH, we then follow the same steps as we do for multi-layer PSNs created using strategy 1 above. For each multi-layer PSN in CATH, this procedure results in a feature vector of size six.

Given all of the reduced GoTMs of SCOPe, we do the same as we do above for CATH, in order to obtain feature vectors of the corresponding multi-layer PSNs. For each multi-layer PSN in SCOPe, this procedure results in a feature vector of size six.

Similarly, given all of the reduced GoTMs of each of the Astral and Scop25% datasets, we do the same as we do above for CATH or SCOPe, in order to obtain feature vectors of the corresponding multi-layer PSNs. For multi-layer PSN in the Astral and Scop25% datasets, this procedure results in a feature vector of size six and five, respectively.

S3. Selection of the best of the two considered unweighted single-layer PSN features

As described in the main paper, we compare our proposed multi-layer PSNs with original PSNs in the task of PSC. However, as explained in Supplementary Section S2, we evaluate two graphlet-based graphlet features for original PSNs, i.e., eGDVM and nGDVM. Hence, we first determine which of the two features perform statistically significantly better than the other. To do this, we first evaluate each of the two features on the considered 72 considered datasets in the task of PSC and obtain the corresponding misclassification rates (Section 2.3 in the main paper). Then, we compare the two features by evaluating whether the misclassification rates of each one of them are significantly better (i.e., have lower values) than the corresponding misclassification rates of the other, using paired Wilcoxon rank sum test. Hence, we obtain two p-values corresponding to the above two tests, which we correct using False Discovery Rate (FDR) to obtain the corresponding q-values. We find that eGDVM significantly (q-value of $\leq 10^{-6}$) outperforms nGDVM (Supplementary Fig. S2). Hence, we use eGDVM as the feature for original PSNs.

S4. The Logistic Regression framework

Here, we complement Section 2.3 from the main paper by explaining how we train a classifier for a given dataset and a given protein feature. For a given dataset, we train an LR classifier corresponding to each of the considered eight protein features. Consequently, we obtain eight trained LR classifiers for each dataset. For a given dataset and a given feature, we train the corresponding classifier in the following manner.

Because PSC is a multi-class problem (i.e., it deals with the classification of data among ≥ 2 classes), we use the one-vs-rest scheme to train an LR classifier. That is, given a PSN dataset which has C number

of protein structural classes, we train C different binary LR models corresponding to the C different protein structural classes. Each such binary model can be explained as follows.

Given n augmented feature vectors of the form $x_j = [1, f_1, ..., f_k]$ (where, $j = 1, 2, ..., n$ and f_1 to f_k are the K features in a feature vector), an LR binary model aims to learn optimal values of each parameter β_k ($k = 0, 1, ..., K$) in the parameter vector $\beta = [\beta_0, ..., \beta_K]$. An LR binary model learns the optimal values for these parameters by minimizing the cost function $\xi(\beta, X, c)$, where, X is the set of all n data points in a dataset (i.e., a dataset) and c are their corresponding class labels. For this study, we use the L_2-Ridge regularized loss function (1).

$$\xi(\beta, x, c) = \frac{\lambda}{2}\beta^T\beta + \sum_{j=1}^{n} \log(1 + e^{-c_j \beta^T x_j}). \tag{1}$$

Here, $\frac{\lambda}{2}\beta^T\beta$ is the regularization term to avoid over-fitting, and λ is the regularization hyper-parameter.

Given a test feature vector y, a learned LR classifier (i.e., C trained binary LR models) estimates the "likelihood" for each of the C classes to be the true label for y using equation (2).

$$\sigma(t)_i = \frac{1}{1 + e^{-t}}, \tag{2}$$

Here, $t =< \beta, y >$ is the inner-product of the learned parameter vector β and the test feature vector y, and $\sigma(t)_i$ represents the likelihood score that the input vector y belongs to class i ($i = 1, 2, ..., C$).

We use the scikit-learn package in Python [8], in order to implement our LR classifier.

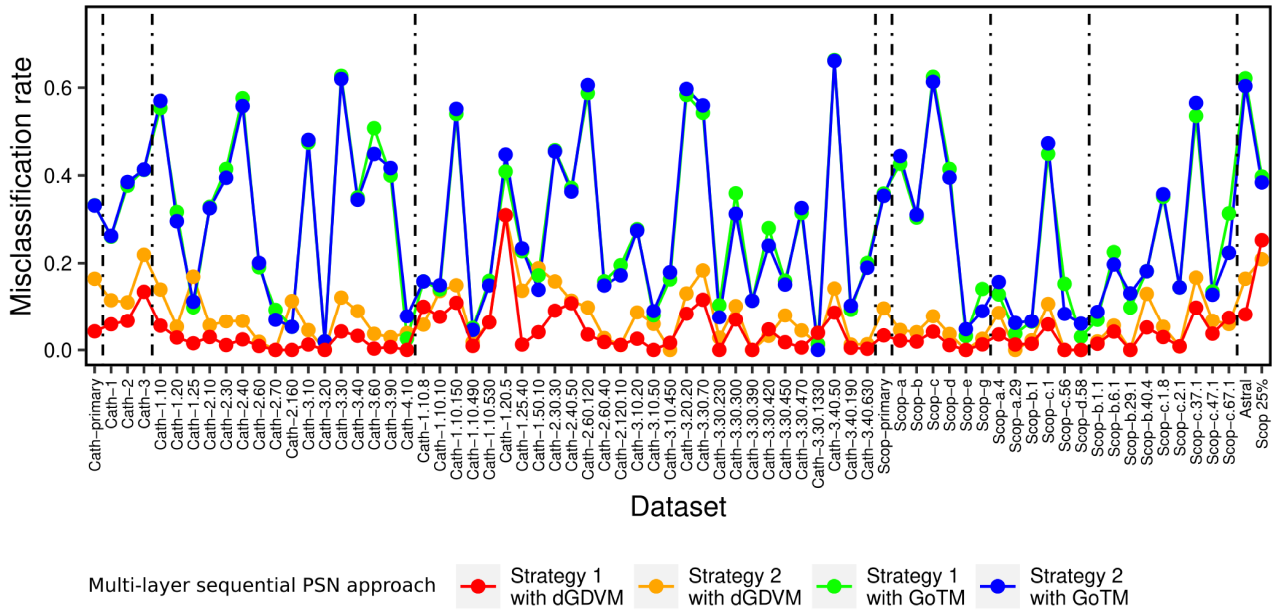
For a given PSN dataset and a PSN feature, we evaluate the performance of the feature using 5-fold cross-validation. That is, first, we divide the PSN dataset into five equal-sized folds (or subsets), where in each subset we keep the same proportion of different protein structural classes as present in the initial PSN dataset. Second, for each such subset, we use the PSNs in the subset as the test data and the union of PSNs in the remaining four subsets as the training data.

Before we train an LR model using a training data and use it to predict classes of PSNs in the corresponding test data, we use the training data itself in a 5-fold cross validation manner [4, 11] to perform hyper-parameter tuning, i.e., to choose an "optimal" value for the regularization hyper-parameter (λ). We perform linear search on 10 equally-spaced log-scaled values between 2^{-8} and 2^8 to find an optimal value. We say that a given value is an "optimal" choice for λ if, on average over 5-folds, the corresponding value results in the best classification performance based on average accuracy over 5-folds. Accuracy measures the percentage of all proteins from the test data that are classified into their correct protein structural classes. Note that we choose the "optimal" value for λ based on just the training data, which is independent of the test data. We use this "optimal" λ value to train the classifier using all of the training data, and then evaluate the performance of the classifier using a separate test data.

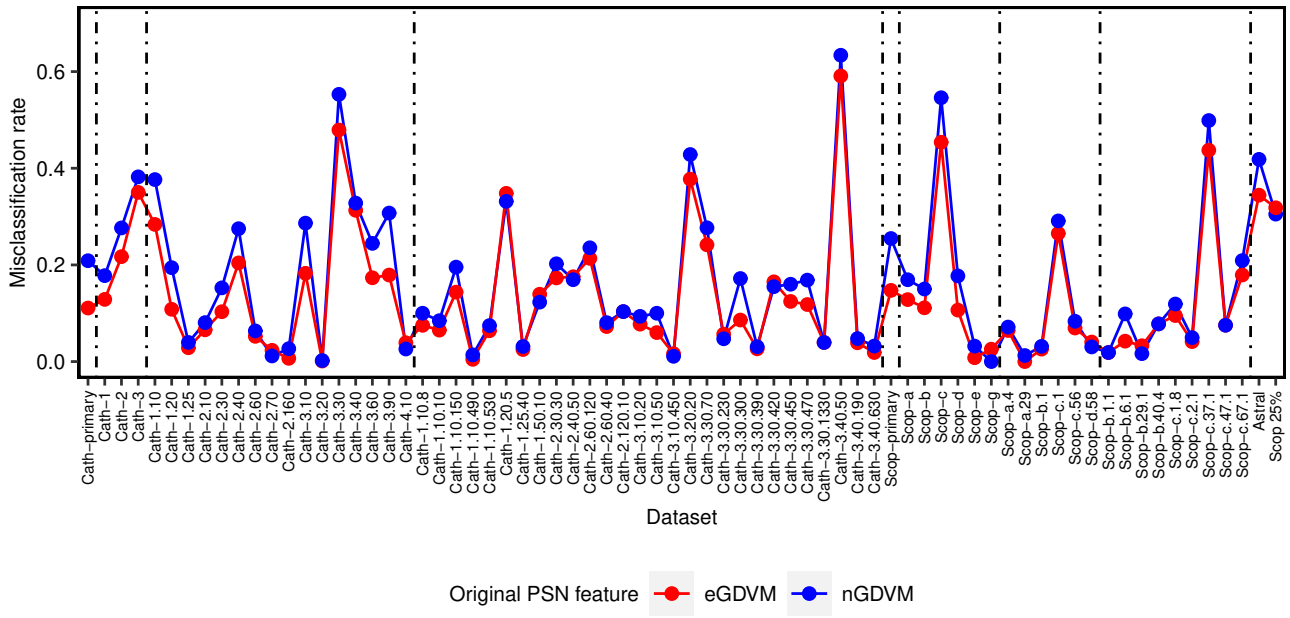
S5. Statistical significance for CATH/SCOPe hierarchy levels of a set of datasets

Here, we complement Section 3.4 from the main paper by explaining how we examine statistical significance of the enrichment of a set of datasets in hierarchy levels of CATH/SCOPe. We measure enrichment of a set of datasets in a CATH/SCOPe hierarchy level using the hypergeometric test. Given the observed number of occurrences of datasets from a CATH/SCOP hierarchy level in a set of the given size, the hypergeometric test measures the probability of getting the same or higher number of occurrences of datasets from the same CATH/SCOP hierarchy level in a randomly chosen set of the same size. The latter is selected by chance from a set of background datasets, which consists of all of the considered 72 CATH and SCOPe datasets in our study.

II Supplementary Figures



Supplementary Fig. S1: Misclassification rates of our proposed multi-layer PSN types over all 72 datasets.



Supplementary Fig. S2: Misclassification rates of original PSNs using eGDVM and original PSNs using nGDVM over all 72 datasets.

References

[1] David Aparício, Pedro Ribeiro, and Fernando Silva. Graphlet-orbit transitions (GoT): A fingerprint for temporal network comparison. *PLOS ONE*, 13(10):e0205497, 2018.

[2] Hongyu Guo, Khalique Newaz, Scott Emrich, Tijana Milenković, and Jun Li. Weighted graphlets and deep neural networks for protein structure classification. *arXiv preprint arXiv:1910.02594*, 2019.

[3] Yuriy Hulovatyy, Huili Chen, and Tijana Milenković. Exploring the structure and function of temporal networks with dynamic graphlets. *Bioinformatics*, 31(12):i171–i180, 2015.

[4] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[5] Noël Malod-Dognin and Nataša Pržulj. GR-Align: fast and flexible alignment of protein 3D structures using graphlet degree similarity. *Bioinformatics*, 30(9):1259–1265, 2014.

[6] Khalique Newaz, Mahboobeh Ghalehnovi, Arash Rahnama, Panos J. Antsaklis, and Tijana Milenković. Network-based protein structural classification. *Royal Society Open Science*, 7(6):191461, 2020.

[7] Khalique Newaz and Tijana Milenković. Graphlets in network science and computational biology. *Analyzing Network Data in Biology and Medicine: An Interdisciplinary Textbook for Biological, Medical and Computational Scientists*, page 193, 2019.

[8] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[9] N. Pržulj, D. G. Corneil, and I. Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 07 2004.

[10] Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 01 2007.

[11] Steven L Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data mining and knowledge discovery*, 1(3):317–328, 1997.

[12] R. W. Solava, R. P. Michaels, and T. Milenković. Graphlet-based edge clustering reveals pathogen-interacting proteins. *Bioinformatics*, 28(18):i480–i486, 09 2012.

[13] Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, 35(11):1026–1028, Nov 2017.

[14] Ömer Nebil Yaveroğlu, Noël Malod-Dognin, Darren Davis, Zoran Levnajic, Vuk Janjic, Rasa Karapandza, Aleksandar Stojmirovic, and Nataša Pržulj. Revealing the hidden language of complex networks. *Scientific Reports*, 4(1):4547, Apr 2014.