Prediction of protein-protein interaction with cosine matrices

Anton A. Novikov
Faculty of Applied Mathematics and Computer Science Belarusian State University
Minsk, Republic of Belarus
newnovnav@gmail.com

Alexander V. Tuzikov
United Institute of Informatics Problems National Academy of Sciences of Belarus Minsk, Republic of Belarus
tuzikov@newman.bas-net.by

Alexander V. Batyanovskii
United Institute of Informatics Problems
National Academy of Sciences of Belarus
Minsk, Republic of Belarus
alexanderSN@yandex.ru

***Abstract*—The protein-protein interaction prediction problem is one of the unsolved fundamental problems of bioinformatics and structural biology. A wide range of machine learning approaches has been developed, relying on prediction of protein-protein interaction interface. In this study we have tried a different approach to the problem. It relies on prediction of molecule centers displacement directions and their relative rotation. We present a novel protein structure representation with cosine matrices. These matrices can be considered as successors of widely used distance maps. They have useful properties such as rotation/shift invariance and self-correcting behavior. We developed a fully convolutional neural network architecture, which is able to predict dimer complexes (both homodimer and heterodimer). The model allowed to achieve 51% of correct predictions (59% for homodimers and 45% for heterodimers) for a test set of 5,854 complexes and 10 angstrom RMSD threshold.**

***Index Terms*—Protein-protein interaction, protein structure representation, cosine matrix, fully convolutional neural network**

I. Introduction

Proteins serve as the base of all known life and thus an object of wide practical and theoretical interest. Working with the protein molecules presents certain challenges. One of such challenges is prediction of protein-protein interactions. While a related problem of protein folding predicition has seen a recent major breakthrough [1], protein interaction prediction has also seen significant improvement in performance, although it is still not solved [2].

A protein is a polymer consisting of amino acid residues tied with peptide bonds. There are 20 amino acids which are regularly encountered in natural proteins. To describe a structure of a protein molecule four levels of the structure are defined. The primary level is defined as a sequence of amino acid residues which constitutes the protein. The secondary level is defined as a backbone conformation and consist of α-helices, β-sheets, unordered segments and other. The tertiery level is defined as a full 3-D model of the molecule with both the backbone and side-chains. The quaternary level is defined as full 3-D model of a protein complex and thus makes sense only for multimers.

Protein interaction prediction can be defined as quaternary structure deduction from primary, secondary and/or tertiary structure. Most popular approach is to predict interaction interface using some machine learning algorithm. The interface consists of pairs of interacting residues and can be represented by a binary contact map. Residues are considered interacting when distance between them is less than a chosen threshold (that can lay in range of 6-12 angstroms). When the interface is known, quaternary structure restoration can be treated as and solved as an optimization problem. In many studies interface prediction is the endpoint and no full quaternary structures are built. It presents a problem, since in those researches different metrics are used, and results can not be compared directly.

One method of this type is based on a representation of protein structure known as a distance map [3], [4]. As the definition suggests, distance maps are matrices consisting of distances between backbone atoms (alpha carbon or beta carbon) of the same molecule or two interacting molecules. The first of them are called intra-chain maps, and the second are called inter-chain maps. The main feature is that they are translation and rotation invariant. Using non-invariant representations, such as ordinary Cartesian coordinates, with neural networks often leads to unwanted side effects. This method takes an intra-chain distance map as input, combined with an amino acid sequence or multiple sequence alignment (MSA), and produces a binary contact map of the same size as the output. Here, the input and output are processed as multi-channel and binary image respectively. Processing of the input data is performed using convolutional neural networks (CNN), in particular, using fully convolutional neural networks (FCN) [5]. An important feature of FCNs is that they can handle input data of arbitrary size. This approach is best suited for predicting homodimeric complexes. While a

homodimeric complex can be represented by a single distance map, a heterodimeric complex requires a pair of maps of different sizes, so they cannot be stacked together. To avoid this shortcoming, the authors of [4] used only one of the two distance maps in the input data, but this does not seem to be a natural solution.

Another approach to predicting interaction is considered in this study. It is well suited for both homodimers and heterodimers. In the general case, the configurations of proteins in a dimer lie in a six-dimensional parameter space [6]. If one molecule has a fixed position, then the position of the other will be determined by rotation (3 parameters) and translation (3 parameters). Alternatively, we define the position using rotation (3 parameters), shear axis (2 parameters), and distance (1 parameter). At the same time, the direction of rotation and coupling is predicted, and the distance is reconstructed taking into account the van der Waals radii. This reconstruction is quite accurate, although it often results in an error of 1-3 angstroms RMSD. To implement the described method, a new representation was proposed, which also allows invariant translation/rotation of protein structures for both monomers and multimers.

II. Representation by cosine matrices

Assume there are two directed polylines L' and L'' in a k-dimensional Euclidian space. They consist of point sequences

$$(x'_0, x'_1, ..., x'_m), (x''_0, x''_1, ..., x''_n)$$

and corresponding vectors

$$(\vec{l'_1}, \vec{l'_2}, ..., \vec{l'_m}), (\vec{l''_1}, \vec{l''_2}, ..., \vec{l''_n})$$

(where $\vec{l'_i} = x'_i - x'_{i-1}$ and $\vec{l''_j} = x''_j - x''_{j-1}$ for i in range of $\{1, ..., m\}$ and j in range of $\{1, ..., n\}$).

We define a cosine matrix for the lines L' and L'' as an $m \times n$ real matrix of the form:

$$\begin{pmatrix} cos(\vec{l'_1}, \vec{l''_1}) & cos(\vec{l'_1}, \vec{l''_2}) & & cos(\vec{l'_1}, \vec{l''_n}) \\ cos(\vec{l'_2}, \vec{l''_1}) & cos(\vec{l'_2}, \vec{l''_2}) & & cos(\vec{l'_2}, \vec{l''_n}) \\ & & \ddots & \\ cos(\vec{l'_m}, \vec{l''_1}) & cos(\vec{l'_m}, \vec{l''_2}) & & cos(\vec{l'_m}, \vec{l''_n}) \end{pmatrix} \quad (1)$$

Let us rewrite the cosine matrix in another form. We have to normalize the vectors $\vec{l'_i}$ and $\vec{l''_j}$ into $\vec{e'_i} = \vec{l'_i}/|\vec{l'_i}|$ and $\vec{e''_j} = \vec{l''_j}/|\vec{l''_j}|$ and choose a Cartesian frame of reference. With the frame of reference, we write down the coordinates of the vectors $\vec{e'_i}$ and $\vec{e''_j}$ into row-matrices e'_i and e''_j. These rows are stacked in $3 \times m$ and $3 \times n$ matrices:

$$P_1 = (\, e'_1 \;\; e'_2 \;\; \cdots \;\; e'_m \,) \quad (2)$$

$$P_2 = (\, e''_1 \;\; e''_2 \;\; \cdots \;\; e''_n \,) \quad (3)$$

Since $cos(\vec{l'_i}, \vec{l''_j}) =< \vec{e'_i}, \vec{e''_j} >= (e'_i)^{\mathrm{T}} e''_j$ matrix (1) is rewritten as:

$$\begin{pmatrix} (e'_1)^{\mathrm{T}} e''_2 & (e'_1)^{\mathrm{T}} e''_2 & & (e'_1)^{\mathrm{T}} e''_n \\ (e'_2)^{\mathrm{T}} e''_2 & (e'_2)^{\mathrm{T}} e''_2 & & (e'_2)^{\mathrm{T}} e''_n \\ & & \ddots & \\ (e'_m)^{\mathrm{T}} e''_2 & (e'_m)^{\mathrm{T}} e''_2 & & (e'_m)^{\mathrm{T}} e''_n \end{pmatrix} =$$

$$= (\, e'_1 \;\; e'_2 \;\; \cdots \;\; e'_m \,)^{\mathrm{T}} (\, e''_1 \;\; e''_2 \;\; \cdots \;\; e''_n \,) =$$

$$= P_1^{\mathrm{T}} P_2 \quad (4)$$

Formula (4) is a more useful form of (1), since it allows analysis of the cosine matrices with the linear algebra calculations. Notations C_{P_1,P_2} and C_{P_1,AP_2} will be also used for cosine matrices. Here A is an orthogonal rotation matrix. The formula is used to encode protein complexes where both components are already known, therefore matrices P_1 and P_2 are also known.

To fully represent the protein complexes with the matrices, it is necessary to introduce two special cases of C_{P_1,P_2}.

A. Matrix C_E

Assume there is only a single polyline. It is possible to build a cosine matrix, simply by taking it twice. Therefore (4) takes the form:

$$C_E = P^{\mathrm{T}} P \quad (5)$$

It is important to note that C_E is essentially the Gram matrix of the system $(\vec{e_1}, \vec{e_2}, ..., \vec{e_n})$ and thus has the corresponding properties.

B. Matrix C_v

Assume there are a polyline and a unitary vector $\vec{v}$. We can use vector sequence $(\vec{l'_1}, \vec{l'_2}, ..., \vec{l'_m})$ and sequence of size n $(\vec{v}, ..., \vec{v})$ to build a cosine matrix C_v of size $m \times n$.

Basic properties of cosine matrices:

1) All values of a cosine matrix lay in range $[-1, 1]$.
2) C_E main diagonal consists of ones.
3) Transposing cosine matrices is done as follows:

$$C^{\mathrm{T}}_{P_1,P_2} = C_{P_2,P_1} \quad (6)$$

$$C^{\mathrm{T}}_{P_1,AP_2} = C_{P_2,A^{\mathrm{T}}P_1} = C_{P_2,A^{-1}P_1} \quad (7)$$

$$C^{\mathrm{T}}_E = C_E \quad (8)$$

4) $rank(C_E) = rank(P)$. The polyline dimensionality equals to the rank of the cosine matrix.
5) All cosine matrices are independent on the Cartesian frame of reference choice.
6) The same isometric transformation applied to the both polylines do not change their cosine matrices.

III. Protein representation with cosine matrices

Cosine matrices are formally defined for abstract polylines. In context of protein structure representation, we use alpha carbon atoms as sequences of points. These atoms are mostly enough to represent tertiary structure, thereof a range of software exists [7] to restore full proteins from alpha carbons. When two molecules consisting of m and n residues are used one C_{P_1,P_2} of size $(m-1) \times (n-1)$ is produced. This fact affects the amino sequence encoding for the neural network input.

Each of three classes of matrices is utilized differently. C_E matrices are used for single molecules encoding. They can be

used for the protein folding prediction. As it has been previously said, for the protein interactions we represent complexes by defining rotations and the bonding axes. Matrix C_{P_1,P_2} is utilized for the rotation encoding and the bonding axis can be easily represented with C_v matrix. More specifically, the axis is represented with $m \times n$ C_{v_1} against the first molecule and $m \times n$ $C_{v_2}^{\mathrm{T}}$ against the second. Therefore, quaternary structures of dimer complexes can be represented with 3 matrices, which are in some sense 3-channel pseudo-images.

A. Textures on the cosine matrices

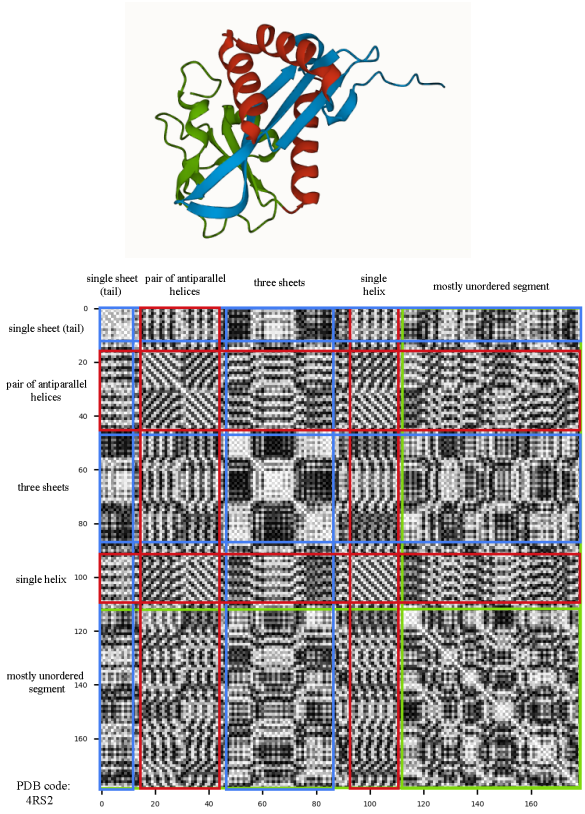

Fig. 1. Secondary structures to textures mapping.

In the context of neural networks, it is better to think of the cosine matrix as a grayscale image rather than a mathematical matrix. Because the secondary structures of proteins are quite regular, they create regular and distinct textures in the image. Here we list the most common textures and their meaning.

- The "Parallel Rods" texture appears where two spirals overlap. The inclination and color of these stripes depend on the angle between the spirals. If the spirals are parallel, then the rods are parallel to the main diagonal; if the spirals are antiparallel, then the rods are parallel to the minor diagonal.
- Solid texture appears in areas where two sheets overlap. The color of these areas depends on the angle between the sheets. When they are parallel, the color is white (values around 1), when the sheets are anti-parallel, the color is black (values around -1).
- The "wave" texture appears where the sheet and spiral overlap.
- When an area consists of many small solid, "banded" and "wavy" rectangles, it usually means that the area is an overlap of an unordered segment with something

All this means is that cosine matrices encode secondary and tertiary structure as image texture. We will later show that the coordinates of alpha carbons (in some reference frame) can be extracted from cosine matrices using ordinary linear algebra operations.

IV. Matrices decoding

There are several algorithms we use to decode cosine matrices and get back protein structures from them. Most important ones are extracting polyline from C_E, decoding column-vector v from C_v with known P, extracting rotation matrix A from C_{P_1,AP_2} with P_1 and P_2 known and decoding both P_1 and P_2 from C_{P_1,P_2}.

These algorithms are suitable not only for the exact cosine matrices defined above, but also for their approximations and noisy cases due to the self-correctness property of cosine matrices. This is important because neural networks are approximators by design and usually do not make absolutely accurate predictions. When an imprecise matrix is decoded, it can also be converted to an accurate matrix with decoded polylines.

The algorithms are based on the fact that the distance between neighboring alpha carbon atoms is constant and equal to 3.8 angstroms. We also included a bumping algorithm that is used to determine the distance between interacting molecules. We use the following notation:

- $T_X = (XX^{\mathrm{T}})^{-1}X$
- $[X]^2$ – is element-wise matrix square
- $sqrt(X)$ – is element-wise matrix square root
- $diag(v)$ – is diagonal matrix with elements of vector v on the main diagonal

A. Extracting alpha carbons positions from C_E

Since the matrices C_E are Gram matrices, extracting a polyline is finding a vector realization. Here we assume that P is a full-rank matrix and all eigenvalues of C_E are distinct. This is true for all protein molecules in the data set. Suppose we have a matrix $\tilde{C}_E$, which is not necessarily an exact cosine matrix and may be noisy or the result of a neural network. In such cases, a reliable algorithm for obtaining coordinates is as follows:

1. Let $\hat{C}_E \leftarrow \frac{1}{2}(\tilde{C}_E^{\mathrm{T}} + \tilde{C}_E)$ (symmetrisation)
2. Compute $V = (v_1\ v_2\ \cdots\ v_k) \in \mathbb{R}^{n \times k}$ matrix of eigenvectors and eigenvalues $\lambda_1, \lambda_2, ..., \lambda_k > 0$
3. Let $\hat{P} \leftarrow diag(\pm\sqrt{\lambda_1}, \pm\sqrt{\lambda_2}, ..., \pm\sqrt{\lambda_k})V^{\mathrm{T}}$
4. Let P be $\hat{P}$ with all column-vectors normalized
5. Multiply P with length constant d and compute the cumulative sum along rows

The result is a alpha carbon polyline in which the first atom is at the zero point.

B. Extracting vector v from C_v

If matrix P is known then vector v encoding is quite simple:

1. Let $\Upsilon \leftarrow (PP^{\mathrm{T}})^{-1}PC_v = T_P C_v$
2. Compute the sum of Υ along rows into column-vector $\tilde{v}$
3. Let $\hat{v}$ be normalization of $\tilde{v}$

The vector $\tilde{v}$ is the result. Note that $(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}Y$ specifies multivariate linear regression in explicit form (least squares). Therefore, this algorithm is essentially just linear regression.

C. Extracting A from C_{P_1,AP_2} with known P_1 and P_2

If the tertiary structures of both interacting molecules are known, then it is easy to construct the matrices P_1 and P_2. When the rotation is predicted, it results in the matrix C_{P_1,AP_2}, where A is the orthogonal rotation matrix. The matrix A can be extracted using the following algorithm:

1. Let $\tilde{A} \leftarrow T_{P_1} C_{P_1,AP_2} T_{P_2}^{\mathrm{T}}$
2. Compute singular value decomposition (SVD) of $\tilde{A}$: $\tilde{A} = UDV^{\mathrm{T}}$, there $D = diag(\sigma_1, \sigma_2, ..., \sigma_k)$ and U and V are orthogonal
3. Let $\hat{D} \leftarrow diag(sign(\sigma_1), sign(\sigma_2), ..., sign(\sigma_k))$
4. Let $\hat{A} \leftarrow U\hat{D}V^{\mathrm{T}}$, $\hat{A}$ is now orthogonal

The result is matrix $\hat{A}$.

This is a general algorithm that is most often used for heterodimers. For homodimers $P_1 = P_2 = P$ and the matrix A is symmetric with the eigenvalues $(1, -1, -1)$, and so the algorithm can be easily modified to ensure that the resulting estimate $\hat{A}$ has the required properties. This can be done by symmetrisation of $\hat{A}$ and using eigenvalues/eigenvectors instead of SVD. We also note that the first step of the algorithm is essentially linear regression applied twice.

D. Extracting P_1 and P_2 from C_{P_1,P_2}

It is possible to extract both P_1 and P_2 only from C_{P_1,P_2}. There are two methods for this: one is exact and the other is approximate. This approximation is more computationally efficient, but with modern computing power it does not make much difference. Here is the exact algorithm:

1. Let X_1 be matrix of $C_{P_1,P_2}C_{P_1,P_2}^{\mathrm{T}}$ eigenvectors and X_2 be matrix of $C_{P_1,P_2}^{\mathrm{T}}C_{P_1,P_2}$ eigenvectors
2. Let $Y_1 \leftarrow X_1^{\mathrm{T}}C_{P_1,P_2}$ and $Y_2 \leftarrow (C_{P_1,P_2}X_2)^{\mathrm{T}}$
3. Let $Z_1 \leftarrow T_{Y_2} C_{P_1,P_2} T_{Y_1}^{\mathrm{T}}$
4. Let Z_2 be matrix of $Y_1 Y_1^{\mathrm{T}}$ eigenvectors
5. Let $Z_3 \leftarrow Z_2^{\mathrm{T}} Z_1 Y_2$
6. Sum $T_{[Z_3]^2}$ along rows into a column-vector s
7. Let $\hat{P}_1$ be $diag(sqrt(s))Z_3$ with normalized column-vectors
8. Let $\hat{P}_2$ be $T_{\hat{P}_1} C_{P_1,P_2}$ with normalized column-vectors

The result is a pair of matrices $\hat{P}_1$ and $\hat{P}_2$.

All four described algorithms have the property of self-correction. It is based on SVD robustness and linear regression. We tested these algorithms with white noise of various distributions and amplitudes and saw the stability of the cosine matrices.

E. Bumping algorithm

A simple iterative algorithm is used to find the distances between molecules. The first shift is sought at which the van der Waals radii of alpha carbon atoms will not be violated (this radius is 1.7 angstroms). The algorithm is as follows:

1. Fix the position of one of the molecules and move the second to align their geometric centers
2. Calculate all the distances between the chains, if all of them are not less than 3.4 angstroms and 90% of them not less than 3.6 angstrom, then the algorithm stops
3. Shift the second molecule by 0.1 angstroms in the direction of the bond and proceed to step 2.

The result is assumed to be the final quaternary structure prediction.

V. Neural network architecture

Two convolutional neural networks were built. The first of them is an autoencoder and consists of an encoder and a decoder. Both the encoder and decoder are fully convolutional 1D and 2D networks. The encoder takes an amino acid sequence as input and creates an intermediate 256-mer code from it. The outer product of the code is then performed with itself and the result is sent to the decoder. This block creates a single grayscale image. The network was trained to predict C_E and hence the protein folding.

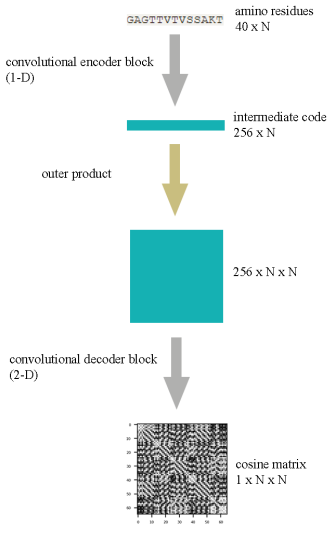

Fig. 2. First stage neural network scheme.

The second neural network is also an autoencoder. It reuses the already trained blocks from the previous step, but introduces two new blocks: a recorder and a reformer (both also

FCN). The recoder takes the intermediate code and the amino acid sequence of the protein molecule and recodes them into another intermediate code. When two interacting molecules act independently, their new codes are converted by the outer product into a single code. The reformer then converts the code into 3 codes, each of which is then processed by the decoder and thus produces a 3-channel image corresponding to C_{P_1,P_2}, C_{υ_1} and $C^{\mathrm{T}}_{\upsilon_2}$ matrices. This predicts the desired quaternary structure.

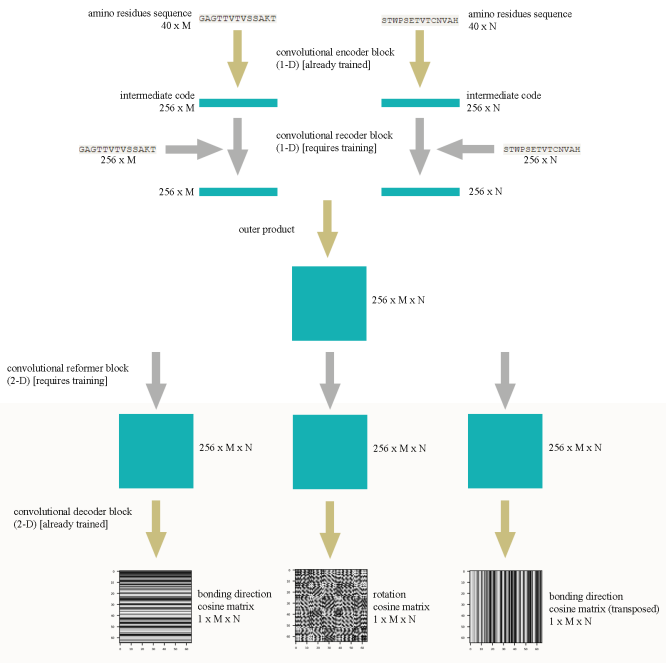

Fig. 3. Second stage neural network scheme.

The reason for this dual architecture is that the dataset of individual protein structures is much larger than the dataset of dimeric complexes. The idea is to obtain an intermediate code containing general information about the tertiary structure of a protein and then use it to predict interactions. Once the code is generated, a second round of training is performed on a smaller set of dimer data. Therefore, it is possible to predict the quaternary structure based on the primary structure. Known tertiary structures do not participate in the input data of neural networks, but they are used to obtain the matrices P_1 and P_2 and use them in decoding C_{P1,AP_2}.

VI. Training and Results

A. First stage training

For training at the first stage, a training set of 309,505 molecules and a test set of 77,419 molecules were used. All molecules were at least 65 residues in size. The network was trained for 80 epochs and a loss function based on the Structural Similarity Index Score (SSIM) [8] was used. It is differentiable and easy to interpret and is therefore well suited for this task.

For each training step, a protein was selected from the data set and a random contiguous segment of 65 residues was cut from it. Based on them, the 40×64 binary code of the amino acid sequence and the true tertiary structure of the 64×64 C_E matrix were built. The binary code was fed to the input and the matrix 64×64 $\tilde{C}_E$ was obtained at the output. A comparison of C_E and $\tilde{C}_E$ as grayscale images was then performed to obtain the training loss.

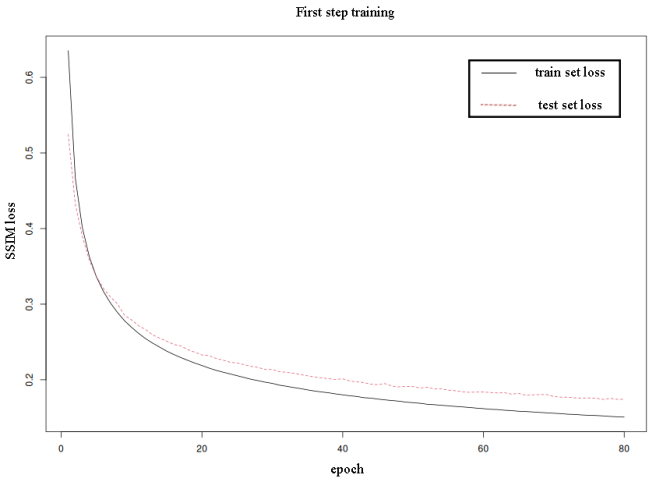

Fig. 4. First stage training (fold).

B. Second stage training

For training at the second stage, a training set of 23,344 complexes and a test set of 5,854 complexes were used. For each of them, both components had a size of at least 65 residues. The network was trained for 100 epochs.

For each training step, a dimeric protein complex was selected from the data set, the binding direction of the proteins (the axis connecting the geometric centers of the molecules) was calculated, and a random contiguous segment of 65 residues was cut from each protein. Based on them, two 40×64 binary codes and 3 matrices for encoding the quaternary structure were constructed: 64×64 true C_{P_1,P_2} matrix and C_{υ_1}, $C^{\mathrm{T}}_{\upsilon_2}$ of the same size (actually $\upsilon_2 = -\upsilon_1$). A pair of binary codes was used as input and the output was $3 \times 64 \times 64$ prediction $(\tilde{C}_{P_1,P_2}, \tilde{C}_{\upsilon_1}, \tilde{C}^{\mathrm{T}}_{\upsilon_2})$. Then a comparison was made between $(C_{P_1,P_2}, C_{\upsilon_1}, C^{\mathrm{T}}_{\upsilon_2})$ and $(\tilde{C}_{P_1,P_2}, \tilde{C}_{\upsilon_1}, \tilde{C}^{\mathrm{T}}_{\upsilon_2})$ as RGB images to obtain training loss.

This network tends to overfit. The reason for this is most likely a general lack of data. Both training and test sets were evaluated. To do this, complete complexes were taken, their second molecule was randomly rotated, and then complete amino acid codes of sizes $40 \times (m-1)$, $40 \times (n-1)$ and two true matrices were constructed P_1 and P_2 (their sizes are $3 \times (m-1)$ and $3 \times (n-1)$). Then the prediction was performed $(\tilde{C}_{P_1,AP_2}, \tilde{C}_{\upsilon_1}, \tilde{C}^{\mathrm{T}}_{\upsilon_2})$ and extracted $\hat{A}$, $\hat{\upsilon_1}$, $\hat{\upsilon_2}$ consistent with each other. The output of the neural network was decoded with known P_1 and P_2. The predicted rotation and bumping

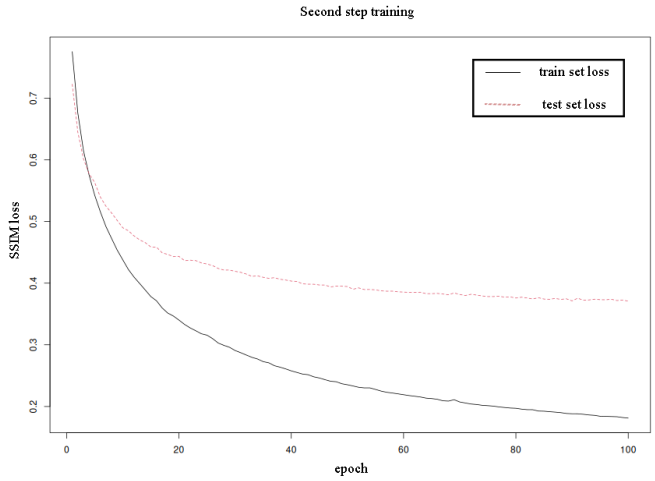

Fig. 5. Second stage training (interaction).

algorithm were then applied. A comparison was made between the predicted and true (unrotated) quaternary structures using the RMSD calculated for the second molecule (the first one was still fixed). The prediction was considered correct if the RMSD was less than 10 angstroms. For the test set, an overall performance of 51% correct predictions was achieved (59% for homodimers and 45% for heterodimers). For the training set, performance was 67% (70% for homodimers and 64% for heterodimers). It seems possible to increase the result by 5-8%, since previously a performance of 64% had already been achieved on the predecessor of the described network, but that network was not suitable for predicting heterodimers and required a C_E matrix as input data.

VII. Conclusion

Using the described approach, a result close to modern performance [2] was obtained. It seems that it is limited by the amount of data available, but as already said that there are opportunities for its improvement. In our opinion, the introduced cosine matrices are well suited for representing protein molecules and can find their place in bioinformatics.They can be useful as an input/output representation and for visualization. Together with SSIM, they can also be used for structural comparisons of molecules.

Code availability

Source code of the neural network and the algorithms implementation are available on GitHub with a link: https://github.com/SwampHiker/COSMAP/

References

[1] J. Jumper, et al. "Highly accurate protein structure prediction with AlphaFold," Nature, 2021. doi: 10.1038/s41586-021-03819-2.

[2] J. Durairaj, D. de Ridder, A. van Dijk, "Beyond sequence: Structure-based machine learning," Computational and Structural Biotechnology Journal, vol. 21, 630-643, doi:10.1016/j.csbj.2022.12.039, 29 Dec. 2022.

[3] A. Hadarovich, A. Kalinouski, A. Tuzikov, "Deep Learning Approach with Rotate-Shift Invariant Input to Predict Protein Homodimer Structure", in Bioinformatics Research and Applications: 16th International Symposium, ISBRA 2020, Moscow, Russia, December 1–4, 2020, Z. Cai, I. Mandoiu, G. Narasimhan, P. Skums, X. Guo., Eds. Lecture Notes in Computer Science, vol 12304. Springer, Cham. doi: 10.1007/978-3-030-57821-3_27, 2020.

[4] Z. Guo, J. Liu, J. Skolnick, J. Cheng, "Prediction of inter-chain distance maps of protein complexes with 2D attention-based deep neural networks," Nat Commun 13, art. 6963, 2022. doi: 10.1038/s41467-022-34600-2.

[5] J. Long, E. Shelhamer, T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," arXiv:1411.4038v2, 2015. doi: 10.48550/arXiv.1411.4038.

[6] T. Vreven, H. Hwang, Z. Weng, "Integrating atom-based and residue-based scoring functions for protein-protein docking," Protein science: a publication of the Protein Society, vol. 20(9), pp. 1576–1586, 2011. doi: 10.1002/pro.687.

[7] A. Badaczewska-Dawid, A. Kolinski, S. Kmiecik, "Computational reconstruction of atomistic protein structures from coarse-grained models", Computational and Structural Biotechnology Journal, vol. 18, pp. 162-176, 2020. doi: 10.1016/j.csbj.2019.12.007.

[8] Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861.