ScanNet Architecture

Module Name	Input Module(s)	Input Size	Output Size	Parameters	Constraints	Initialization	Padding Value	Trainable	Comments
AA Point Cloud	/	/	[2 Naa +1, 3] (float)	/	/	/	0.0	No	For each amino acid, 3D coordinates of its Calpha and Side Chain Center of Mass (SCoM); for Glycin, virtual SCoM placed such that x(SCoM)+x(previous N)+ x(C) = 3 x(Calpha). One extra virtual Calpha introduced at the **beginning** of the protein to construct the frame of the first aa
AA Sequence Index	/	/	[Naa,] (int)	/	/	/	-1	No	The position of each amino acid along the sequence, starting from 0. We do not rely on the PDB indexing.
AA Frame Index Triplet	/	/	[Naa,3] (int)	/	/	/	-1	No	For each amino acid, the indices in **AA Point Cloud** of the three points used to construct the frame. The three points are: the Calpha (frame origin O), the SCoM (Calpha-SCoM = z direction), the **previous** Calpha along the backbone (defines Oxz plane)
AA Attributes	/	/	[Naa,21] (float)	/	/	/	0	No	For each amino acid, the corresponding Position Weight Matrix (21-valued probability distribution); see methods for construction. For the evolution-free model, the amino acid type in one-hot encoded format (20 dimensions)
Atom Point Cloud	/	/	[Natom+Nvirtualatom,3] (float)	/	/	/	0	No	The 3D coordinates of each **heavy** atom (excluding hydrogens). In addition, we add virtual atoms (effectively hydrogen atom coordinates) next to the atoms that have only one covalent bond so as to construct their frames.
Atom Sequence Index	/	/	[Natom,] (int)	/	/	/	-1	No	For each **heavy** atom, the position of its corresponding amino acid along the sequence starting from 0. We do not rely on the PDB indexing
Atom Frame Index Triplet	/	/	[Natom,3] (int)	/	/	/	-1	No	For each **heavy** atom, the indices in the **Atom Point Cloud** of the three points used to construct the frame. The three points are the atom itself (frame origin O), and its two covalently bonded heavy atoms; if it has only one covalent bond, use a virtual atom. The rule for ordering the neighbors are as follows: if not sidechain ring, use the protein tree indexing (next atom for z axis, previous atom for xz plane) otherwise, see full correspondence available in the file protein_chemistry.py
Atom Attributes	/	/	[Natom,] (int)	/	/	/	0	No	The atom attribute; since hydrogen atoms are not explicitly included, we use atom types augmented by the number of bound hydrogens. This gives 12 categories: C,CH,CH2,CH3,CPi (aromatic ring), O,OH,N,NH,NH2, S, SH. Ranges from 1 to 12
AA Frame Computation	AA Point Cloud; AA Frame Index Triplet	[2Naa+1,3]; [Naa,3]	[Naa,4,3]	/	/	/	/	No	Compute one frame (= origin + 3 directions) per amino acid
AA Attribute Embedding	AA Attributes	[Naa,21]	[Naa,32]	Output dimension: 32	/	Default (glorot_uniform)	/	Yes	Dense (no bias) + BatchNorm (no rescaling) + ReLU module, 32 filters
Atom Frame Computation	Atom Point Cloud; Atom Frame Index Triplet	[Natom+Nvirtualatom,3]; [Natom,3]	[Natom,4,3]	/	/	/	/	No	Compute one frame (= origin + 3 directions) per atom
Atom Attribute Embedding	Atom Attributes	[Natom,]	[Natom,12]	/	Fixed to one-hot encoding	Fixed to one-hot encoding	/	No	Converts atomic categorical variable into vector embedding with Keras Embedding Layer. Fixed to identity so as to obtain sparse atomic SCAN filters
Atom Neighborhood Computation	Atom Frame Computation; Atom Attribute Embedding	[Natom,4,3]; [Natom,12]	[Natom,16,3]; [Natom,16,12]	Nature of local coordinate: euclidian Number of neighbors: 16	/	/	/	No	For each atom, finds the K=16 closest neighbors (including itself), compute their **Euclidian** coordinates in the local frame (first output) and copy their attributes (second output)
Atom Neighborhood Embedding	Atom Neighborhood Computation	[Natom,16,3]; [Natom,16,12]	[Natom,128]	Number of gaussian kernels: 32 Covariance matrix type: full Number of filters: 128 Sparse regularization penalty L12 = 2e-3	Each row of the dense layer matrix has fixed norm = Ngaussians/ Nneighbors=sqrt(2)	Gaussian kernels are initialized by fitting a Gaussian Mixture Model to a set of neighborhoods Default for dense kernel (glorot_uniform)	/	Yes	Atomic SCAN filters.

Attention Pooling	Atom Neighborhood Embedding; Atom Sequence Index;
	AA Sequence Index
		[Natom,128];
		[Natom,1];
		[Naa,1];
			[Naa,64]	Number of attention heads: 64
				Number of attention outputs: 64
				Sparse regularization penalty L12 = 2e-3
					The attention output matrix has fixed norm 1	The attention coefficient matrix is initialized as zeros (i.e. each atom contribute equally to its parent amino acid, naive averaging)	/	Yes	Aggregates the learnt atomic SCAN embedding at the amino acid level. This is done by an attention pooling with multiple heads; namely each of the 64 output is a weighted average (via attention coefficients) of one projection of the SCAN embedding. The attention mechanism allows to construct features that average information across all atoms (e.g. accessible surface area) as well as features focusing on specific part of the amino acid (e.g. backbone-related features or side chain-related features).

Concatenate	Attention Pooling;
	AA Attribute Embedding
		[Naa,32];
		[Naa,64];
			[Naa,96]	/	/	/	/	No	Concatenate the amino acid embeddings learnt from a) the PWM and b) the atomic scale

Amino Acid Neighborhood Computation	AA Frames; Concatenate	[Naa,4,3] [Naa,96]	[Naa,16,3] [Naa,16,96]	Nature of local coordinate: euclidian Number of neighbors: 16	/	/	/	No	For each amino acid, finds the K=16 closest neighbors (including itself), compute their **Euclidian** coordinates in the local frame (first output) and copy their attributes (second output)
AA Neighborhood Embedding	Amino Acid Neighborhood Computation	[Naa,16,3] [Naa,16,96]	[Naa,64]	Number of gaussian kernels: 32 Covariance matrix type: full Number of filters: 64 Sparse regularization penalty L12 = 2e-3	Each row of the dense layer matrix has fixed norm = Ngaussians/ Nneighbors=sqrt(2)	Gaussian kernels are initialized by fitting a Gaussian Mixture Model to a set of neighborhoods Default for dense kernel (glorot_uniform)	/	Yes	Amino acid SCAN filters.
SCAN Attribute Embedding	AA Neighborhood Embedding	[Naa,64]	[Naa,32]	Output dimension: 32	/	Default (glorot_uniform)	/	Yes	Non-linear embedding of the Amino Acid SCAN filters output. Its purpose is to a) reduce dimensionality before applying the neighorhood attention layer and b) non-linearly recombine the filter activities.

Graph Neighborhood Computation	SCAN Attribute Embedding;
	AA Frames;
	AA Sequence Index
		[Naa,32];
		[Naa,4,3];
		[Naa,]
			[Naa,32,5];
			[Naa,32,32]
				Mixed local coordinates:
				- Calpha-Calpha distance
				- 3 Angles < Calpha_1 SCoM_1| Calpha_2 SCoM_2>, < Calpha_1 Calpha_2 | Calpha_1 SCoM_1> , < Calpha_2 Calpha_1 | Calpha_2 SCoM_2>
				- Sequence index distance
				Number of neighbors: 32
					/	/	/	No	For each amino acid, finds the K=32 closest neighbors (including itself), compute their **graph** coordinates in the local frame (first output) and copy their attributes (second output)

Neighborhood Attention	Graph Neighborhood Computation	[Naa,32,5]; [Naa,32,32]	[Naa,2]	Number of gaussian kernels: 32 Covariance matrix type: full Number of dimensions per graph edge: 1 Number of attention heads: 1 Number of attention outputs: 2	/	The Gaussian kernel and the dense matrix (for graph embedding) are initialized as a parametric approximation of the **label autocorrelation function** C(d, angle_1,angle_2,angle_3, sequence distance) = PearsonCorr (A_i, A_j | d(O_i,O_j)), see neighborhoods.py file. Default (glorot_uniform)	/	Yes	Goal is to locally average the predictions over the protein graph. The graph itself is learnt from Calpha-Calpha distance, angles and sequence index distance, and can **take both positive and negative values**. Intuitively, this is because the side chains of two consecutive amino acids point in opposite directions, and therefore the labels should be anti-correlated. An attention mechanism is required because some residues are "drivers" of PPI (i.e. hotspots) and hence have strong influence over their neighbors whereas others are "passengers" (i.e. involved in a PPI because of a neighboring hotspot).
Softmax	Neighborhood Attention	[Naa,2]	[Naa,2]	/	/	/	/	No	Converts output into a 2-state probability distribution. Note that we used the softmax with two outputs in conjunction with categorical cross-entropy loss rather than a logistic function with one output and binary cross-entropy even though we are performing binary prediction. This allows to account for i) missing labels (O_i = [0,0]) and ii) residue-dependent weight (O_i = [w_i,0] or [0,w_i]). The later case arises either when we use class weights and/or the protein serialization trick, where several small proteins with different weights are grouped in a single instance.