Initiated Exhaustive search for SRS-MOTIF detection

Primary Author
A. Nishanth

Primary Supervisor
Dr. C. H. Chu
Co-Supervisor
Dr. Rasiah Loganantharaj
November 27, 2020

Contents

1 Level 2 **5**

1.1 Level 2's Project Outcome Abstract 5

1.2 Data set Preprocessing 6

1.2.1 Refining the selected PDB_{SITE}s 8

1.3 Introduction of Evaluation on Level 2 with Three-projections 8

1.3.1 Parameter selection for CNN-visual feature extraction towers 9

1.3.2 Change the feature (biochemical properties) representation models 10

2 Level 2 Quarter model Evaluation **14**

2.1 Quarter model Evaluation Introduction 14

2.2 Introduction to experiments of quarter model's approaches and classification 15

2.3 Data selection for approach 1 and 2 16

2.3.1 Approach 1 16

2.3.2 Approach 2 16

2.4 Classification 16

2.4.1 Functional classification of PDBs (isoforms) 17

2.4.2 Functional classification of primary structure 17

2.5 Quarter model evaluation summery 18

2.5.1 Randomized data evaluation 21

2.5.2 Locked data evaluation: best architecture selection and data representation selection 21

2.6 Rounding limitation due to deep learning model 28

2.7 Deep learning model architecture for quarter projections 30

2.8 Representations of Deep learning model architecture for quarter projections 30

2.8.1 Parameter selection 33

2.8.2 Calculation of number of parameters 34

2.9 Different kernel architectures of BIR (while include the conv kernel 7) 36

2.10 Level 2 results overview 36

2.11 Approach 1 results(10-fold cross validation with mixed primary structural PDBs) 37

2.11.1 Approach 1's functional classification of PDB_{SITE}s (isoforms) results 37

2.11.2 Approach 1's functional classification of gene's primary structure (using ensemble + direct) 38

2.12 Approach 2 results (the primary structural genes with their corresponding PDBs are separated as Train and test sets depending on the genes' primary structural information) 40
2.12.1 Approach 2's functional classification of PDBs (Isoforms) results 40
2.12.2 Approach 2's functional classification of gene's primary structure (using ensemble + direct) 41
2.13 Comparison with the available state-of-the-art methods for ONGO Vs TSG identification 42

A Sequence length Threshold Selection 48

B Resolution factor Selection for Normalization 50

C Surface $C\alpha$ Indexing 52

D Level 1 Property Tables 55

E General Biochemical properties for 21 Amino acids 57

F Towers with Different Parameters for each Projection 61
F.1 *Model_vin_4* 61
F.2 *Model_inception_vin_1*: with depth 8 and the kernels as 3 62
F.3 *Model_inception_all_depths_inception_vin_1* 63
F.4 *Model_inception_all_depths_inception_complic_vin_1* 64

G Parameter details of different tower selection presented in Appendix F 66

H Towers with same parameters for each projection 72
H.1 *Model_vin_4* 72
H.2 *Model_inception_vin_1* 73
H.3 *Model_inception_with_out_addition_vin_1* 74
H.4 *Model_inception_all_depths_inception_vin_1*: d1=4, d2=4, d3=8, d4=8 75
H.5 *Model_inception_all_depths_min_max_inception_vin_1*: d1=4, d2=4, d3=8, d4=8 76
H.6 *Model_inception_all_depths_inception_complic_vin_1*: d1=4, d3=8 77
H.7 *Model_inception_all_depths_min_max_inception_complic_vin_1* 78

I Level 2 data selection 80

J Hypothetical Example of Methods for Calculating Fractional Weights of Functional Primary Structures based on the PDBs' Overlapping Sequence Information 82
J.1 Hypothetical example of $Method_1$ 82
J.2 Hypothetical example of $Method_2$ and $Method_3$ 84

K Complexity of the Methods Classification Primary Structural Functional Detection 87
K.1 Complexity of $Method_1$ 87
K.2 Complexity of $Method_2$ and $Method_3$ 88

L Tier 1 and Tier 2 results annotations **89**
This document is part of my dessertation work.

Chapter 1

Level 2

Success of the basic level (Level 1 disscussed in previous chapter) lead to the Level 2 of evaluation on 3D classification between the ONGO vs TSG vs Fusion [76].

1.1 Level 2's Project Outcome Abstract

The Level 2 project obtained two classification approaches. In Approach 1 none of primary structural information is considered. Thus, all filtered and cleaned PDBs, are pooled together; then 10-fold cross validation is used to find the performance of deep convolutional neural network (DCNN) model in functional classification among these three classes. And in the Approach 2; the genes (primary structures) and their corresponding PDBs are separated as Train and test sets depending on the genes' primary structural information. A novel deep learning model combined with some methods, and for predicting ONGO, TSG and Fusion genes' primary structures' functionality from their corresponding Protein Data Bank (PDB) three dimensional structures' functional probability (obtained from DCNN trained by training set of Approach 2). The paper has implemented DCNN architecture (with inception and residual layers, and all the layers apart from last layer followed by "Swish" activation function), to classify the feature map sets extracted from the 3D protein structures. Each feature map set represents biochemical properties associated with the amino acid coordinates appearing on the outer surface (using the MSMS tool and an algorithm to find the surface amino acid coordinates) of protein's three-dimensional structure.

Specifically, some methods were developed, using the DCNN model's probabilities and primary structural information (sequence lengths of the corresponding classified PDBs) to classify the genes. The experimental results on the collected dataset, for Approach 1 (10-fold cross validation with mixed primary structural PDBs) for classifying ONGOs, TSGs, and Fusion demonstrate promising performance, with 93.5 % overall accuracy (where even Fusion class classified with 88.68 %). Since the Approach 2 test sets performance does not met the expectation on Fusion class. Thus, ONGO/ TSG classifications are evaluated. Where Approach 1 and Approach 2 DCNN AUROCs are 0.978 and 0.765, respectively. And the final genes' primary structure classification for Approach 1 and Approach 2 accuracies are 93.64 % and 74. 07 % with AUROCs of 0.989, and 0.879, respectively.

Classification of Tier 1 and Tier 2 of COSMIC [28] genes, by ensemble the results of our models

(10-best models DCNN obtained by Approach 1) and the methods, are attached in the link (`https://drive.google.com/drive/folders/1YCuMVPhAy7tIdGFiebInZ_6-kBAH1kMM?usp=sharing`). The success of our Approach 1 models warrants our future study to apply the same deep learning models to humans' (GRCh38) genes, for predicting their corresponding probabilities(by ensemble the 10-best models predictions) of functionality in the cancer drivers.

1.2 Data set Preprocessing

Fig. 1.1 visualizes the overall data flow of preprocessing. For humans (GRCh38), annotated cancer genes were downloaded from COSMIC v88 [28]. The experiment has focused on using machine learning technique to identify the genes' functional role (probability of ONGO, TSG, and Fusion) in cancer from their 3D structures. Thus, for preprocessing, the recent version of the COSMIC annotated gene lists' (corresponds to Tier 1) 80 ONGO, 141 TSG, and 94 Fusion genes were first extracted. Then the *"Entrez Gene Id"* of the ONGO, TSG, and Fusion (from the census) were used to map the details of PDB_{id}s through UniProt [34]. The mapping has *Gene-Symbols* with PDB_{id}s and their corresponding, start-end overlap sequence with the gene. This information was used to choose the filtered/ selected PDB (where the PDB means, the mapped satisfied PDB_{id} downloaded from RCSB Protein Data Bank website [23]); which were filtered with filters such as $X-ray$ diffraction, and gene overlap/ align sequence length more than the threshold sequence length (81); threshold length selection is reported in *Appendix* A .

As disscussed in previous chapter subsection **??** . The PDB format exists in standard formats such as macromolecular structure data which achieved by $X-ray$ diffraction, or NMR studies and some PDBs are cryoelectronic microscopy obtained structures. However, majority of PDBs belong to $X-ray$ defragment. Thus, only $X-ray$ diffraction PDB files were considered in this experiment.

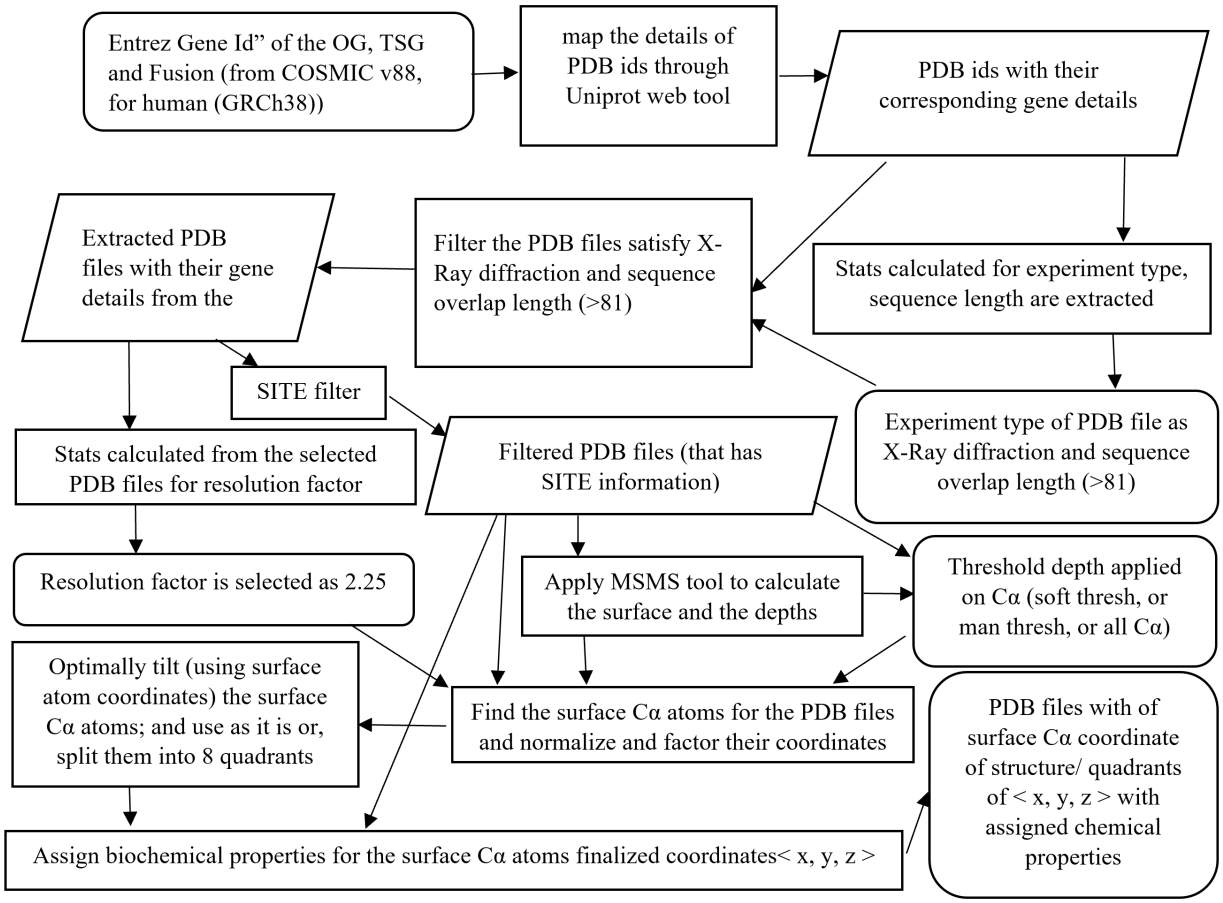
Consequently, the genes investigated in this experiment must have at least one PDB which satisfies all the threshold conditions such as obtained by $X-ray$ diffraction and overlapping sequence length > 81. The PDB files sequence's overlapping start-position and end-position were used to calculate the sequence length. If the mapped PDB_{id}s are quaternary; then they(PDBs) have more than one polypeptide chain (sequence start-end positions) of the gene's overall primary structure, then they were left out in the whole experiment.(for more details please check the *Chapter* **??** "Limitations and assumptions".)

The PDBs satisfying both $X-ray$ diffraction and overlapping sequence length > 81 were selected. Selected PDBs were used to obtain the resolution factor (normalized and factored by resolution factor 2.25, *Appendix* B covers the details of resolution factor selection); the particular normalization process ensuring that PDB structures with various resolutions could be accommodated without affecting the outcome of the experiment. From the selected PDBs, only those containing the *SITE* information were chosen for the analysis. As per [37] *SITE* information,

"specify residues comprising catalytic, cofactor, anticodon, regulatory or other important sites or environments surrounding ligands present in the structure."

Thus, these PDBs may likely play some role in Cancer. The PDB_{SITE} (where the PDB_{SITE} is PDB satisfying all $X-ray$ diffraction, overlapping sequence length > 81 and contain *SITE* feature)

Figure 1.1: : Data creation for deep learning model, COSMIC V88 census credit [28], and datamapping UNIPROT credit [34].



were used to train the CNN architectures (mentioned in this Chapter), and classify the genes (the gene at least has one PDB_{id} satisfying $X-ray$ diffraction, aligned/ overlapping sequence length > 81 and contain *SITE* information). These PDB_{SITE}'s surface $C\alpha$ atoms features (their coordinates with its chemical property and *SITE* information) were used to train the CNN architecture (as mentioned in section 1.3). *MSMS* tool was used to find the surface $C\alpha$ atoms; *Appendix* C, covers how the *"soft"* threshold depths such as depth of $C\alpha$ (7.2) and depth of residue (6.7) from the surface, combination depths (less than both threshold depths) used for surface atom selection. The selected surface $C\alpha$ atoms for PDB_{SITE} are used in all evaluations of this Chapter.

For the classification of primary structure, the 3-D structures' (PDB_{SITE}s') probabilities were weighted by three methods. To do this initially weights were calculated. Subsubsection 2.4.2 (*"functional classification of primary structure"*) explains the calculation of the weights from their overlap/ aligned sequence length of PDB_{SITE}s (those PDB_{SITE}s should be belong to the corresponding primary structure).

Table 1.1: Problematic primary structures (contain Overlapping PDB_{SITE}s) from V88 COSMIC census using UniPort mapping and processed PDB_{SITE}

Class	Gene symbol	Gene_ID	Number of PDBs		uni-gene (retired)
			group	overlapped	
ONGO	*H3F3A*	3020	9	9	Hs.726012
	HIST1H3B	8350	39	39	Hs.626666
	PIK3CA	5290	36	31	Hs.732394
	XPO1	7514	1	1	Hs.370770
TSG	*PIK3R1*	5295	38	31	Hs.734132
Fusion	*HLA-A*	3105	143	143	Hs.713441

[a] Since the lack of PDB_{SITE}s to train Fusion, and Fusion class is formed from ONGO or TSG primary structure; if all the processed PDB_{SITE}s in Fusion's primary structure group overlapped with other classes, then the primary structure group is removed.

E.g.: ONGO class' *Gene-Symbols PIK3CA*, has 36 PDB_{SITE}s, among them 31 PDB_{SITE}s also appear in TSG; thus these 31 PDB_{SITE}s are considered as overlapped.

1.2.1 Refining the selected PDB_{SITE}s

The PDB_{SITE}s are selected as shown in Fig. 1.1(Data creation for deep learning model). These PDB_{SITE}s may overlap in different classes (like one PDB_{SITE} may fall in ONGO & TSG, or Fusion & TSG or ONGO & Fusion; or ONGO & TSG & Fusion). Thus, these PDB_{SITE}s needed to be preprocessed and cleaned (to avoid noise to the models such as deep learning model and other) before defining the isoform (protein structure/ PDB) to class (ONGO/ TSG/ Fusion).

Only the primary structures having higher overlapping PDB_{SITE}s of ONGO and TSG are removed. From the experiment; the overlapping PDB_{SITE} contained primary structure details are shown in Table. 1.1; only the *Gene-Symbols* associates to, ONGO (*PIK3CA* corresponding *Entrez Gene Id* is 5290), and TSG (*PIK3R1* corresponding *Entrez Gene Id* is 5295) were removed. And these primary structures' corresponding group of PDB_{SITE}s were also totally removed for training and testing. The dataset of the PDB_{SITE}s after this separation is called preprocessed PDB_{SITE}s. Those preprocessed PDB_{SITE}s are obtained from the remaining primary structures' corresponding group of PDB_{SITE}s.

After the intial preprocessing of the PDB_{SITE}s, those PDB_{SITE}s of any overlaps among ONGO, TSG, and Fusion classes are also removed. The process ensure that cleaned PDB_{SITE}s never have PDB_{SITE}s overlapped among ONGO, TSG, and Fusion classes.

Hereonwards PDB_{SITE}s means preprocessed PDB_{SITE}s.

1.3 Introduction of Evaluation on Level 2 with Three-projections

This project is continuous of Level 1 project. The initial evaluation is conducted without separating the PDBs based on their corresponding primary structural information. In that evaluation from the selected PDB files, only those containing the *SITE* features were chosen for the analysis. The performance of the model is well for the *SITE* included dataset; on the other hand, primary structure wise separated PDBs are not obtained the expected performance (performance is like tossing the coin).

One of major problem here is lack of structure representation. The representation may lack due to

1. If the PDB is presented in different orientation the results created using the earlier models may not going to be rigid (may give different probability).
2. The biochemical features represent the residues may not effective; or the way the biochemical representation may need a change (like contrasting properties may have negative values instead of separate channel.)

To fix these problems two solutions are proposed accordingly,

1. Creating different models, while fixing the orientation problem.
2. Along with this new dataset is also created, where the 21'st amino acid is included. And the biochemical properties have changed and updated. The details of the new properties are explained in *Appendix* E .

Figure 1.2: Orientation fix by doing rotation with not affect the given scale.

To do this first find the covariance matrix of the surface $C\alpha$ coordinates.

$$X(number\,of\,coordinates \times 3) = surface\,C\alpha\,coordinates$$

$$Cov(X) = X^T X; \text{ find the covariance of the X} \tag{1.1}$$

Do the eigen value decomposition from the $Cov(X)$ obtained in equation (1.1), to find the eigen vector of X; in singular value decomposition of X can be obtained as

$$X(number\,of\,coordinates \times 3) = UDV^T$$

$$X^T X = VD^2V^T \tag{1.2}$$

Using the eigen vector obtained from equation (1.2) to do the rotation.

$$Y = XV$$

Verify, the if the rotation is preserving the X without affecting the scaling The rotation preserved the scaling. **Proof**

$$\begin{aligned} Y^TY &= (XV)^T XV \\ &= V^T X^T XV \\ &= V^T(UDV^T)^T(UDV^T)V \\ &= V^T VDU^T UDV^T V \qquad \text{(since } V^TV = 1 \text{ and } U^TU = 1\text{)} \\ &= D^2 \end{aligned}$$

Creating the dataset without affected by orientation, is obtained by the rotation as shown in Fig. 1.2. Using the dataset after the orientation changed, for the following models.

1.3.1 Parameter selection for CNN-visual feature extraction towers

Another question arises, after tilting the structures, their corresponding projections such as on $< x, y >$, $< y, z >$, and $< x, z >$ (as mentioned in subsection **??** "input feature maps") need to be handled

(weighted) differently or same by the CNN feature extraction towers of the model. (If the functionally important surface residues have any relation of occurrence with the orientation of the structure)

Models preferred to check the performance,

(a) Create the model with different parameters for each tower. And check the performance as it beat the expected performance. The detailed documentation about the models is in subsubsection 1.3.1 ("models with different parameters for each tower").

(b) Unfortunately, none of the models created part (a) mentioned in this Section give the performance as expected. Thus, the projection with same parameters (as done in Level 1; paper [33]) is applied and checked. Detailed explanation of these models is explained in subsubsection 1.3.1 ("models with same parameter for each tower").

Models with different parameters for each tower

The architecture is proposed to check structural functional contributes varies depends on the surface amino acids' presented orientation. To evaluate using different parameter selection for different orientation (projection after tilted).

General architecture of the model is shown in Fig. 1.3. The models created and run is detailed in *Appendix* F. And the default parameters tables of the model are shown in *Appendix* G. (And the codes can be found in GitHub link).

Sometimes the towers' minimum (MIN) and maximum (MAX) extracted feature may perform better. Just considering the MIN and MAX outputs only from the Tower data selection also evaluated. For this problem, the obtained performance in this way is worse than earlier (just using the extracted feature from Towers/ CNN feature extractors).

Models with same parameters for each tower

Here the models with same parameters for each tower. Thus, the model's each tower has same parameter. General architecture of the model is shown in Fig. 1.4. The models created and run is detailed in *Appendix* H. (And the codes can be found in GitHub link).

In these architectures also evaluated with only the MIN and Max outputs from the Tower data selection. Further combination of different CNN kernels also evaluated; only 3, 3, 5 ,5 kernels in inception model/ only 5 kernels instead of 3.

1.3.2 Change the feature (biochemical properties) representation models

Evaluation of the way of biochemical representation details are summarized in this section. Properties represented with 5 digits instead of using 9 digits (since image normalized it lie between $1 \div 255 = 3.921 \times 10^{-3}$; from this it can be seen, the 5 digits 392.1×10^{-5} capture the information).

Opposite properties are formed by binary representation, $0 - 1$, because, the property value is split to preserve the *Yes*, and *No*(in *pH* represented by *Acidic*, and *Base*, like wise dependent on the property's relation); some of the opposite property's (**E.g.:** *pH*) all values are not in binary category at all (*value not assigned:* "-"), and those properties are also represented by 0's. If binary representation $0 - 1$ hold together as one property without split. Then, these property relationships may have higher

Figure 1.3: : The general model with different parameters.

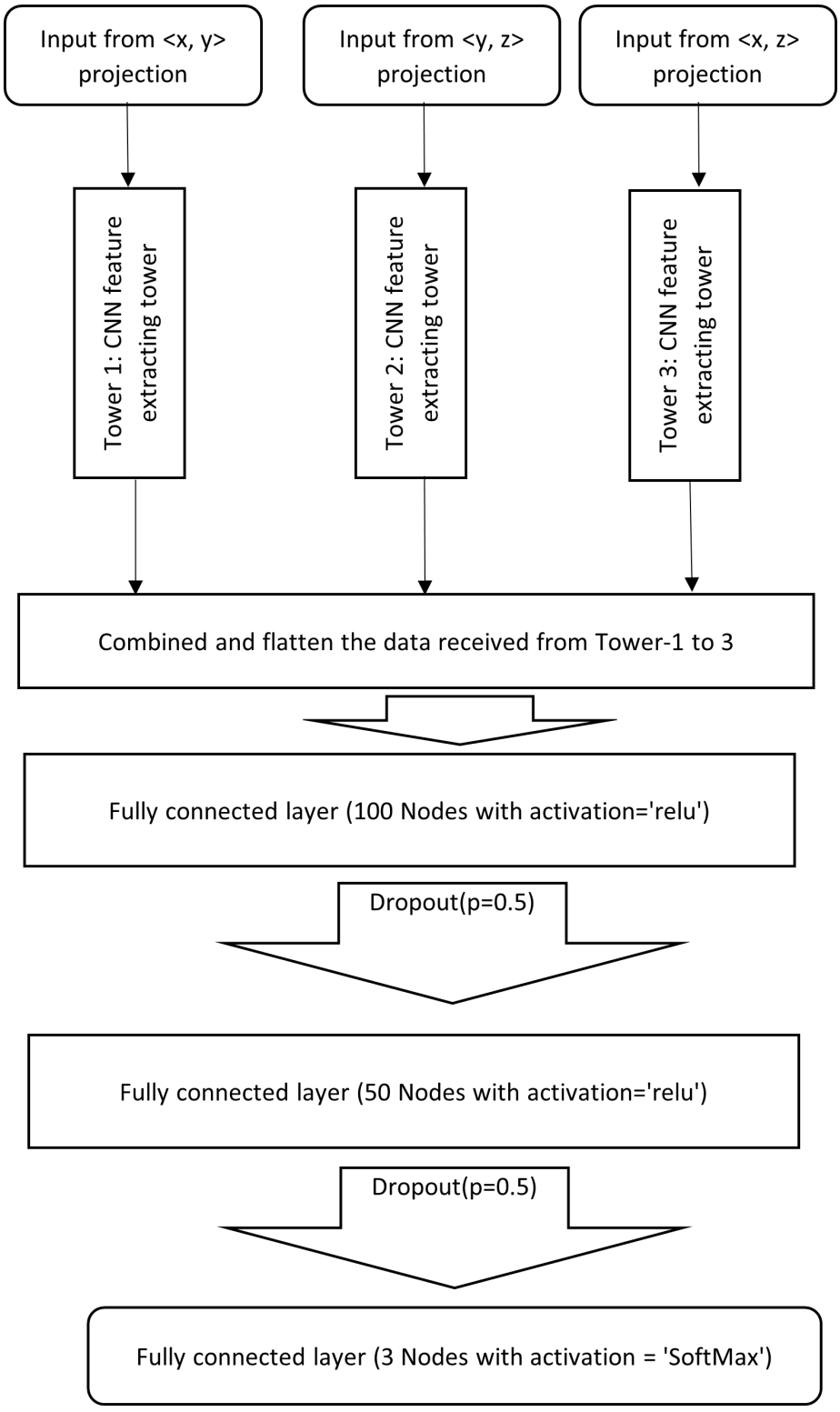
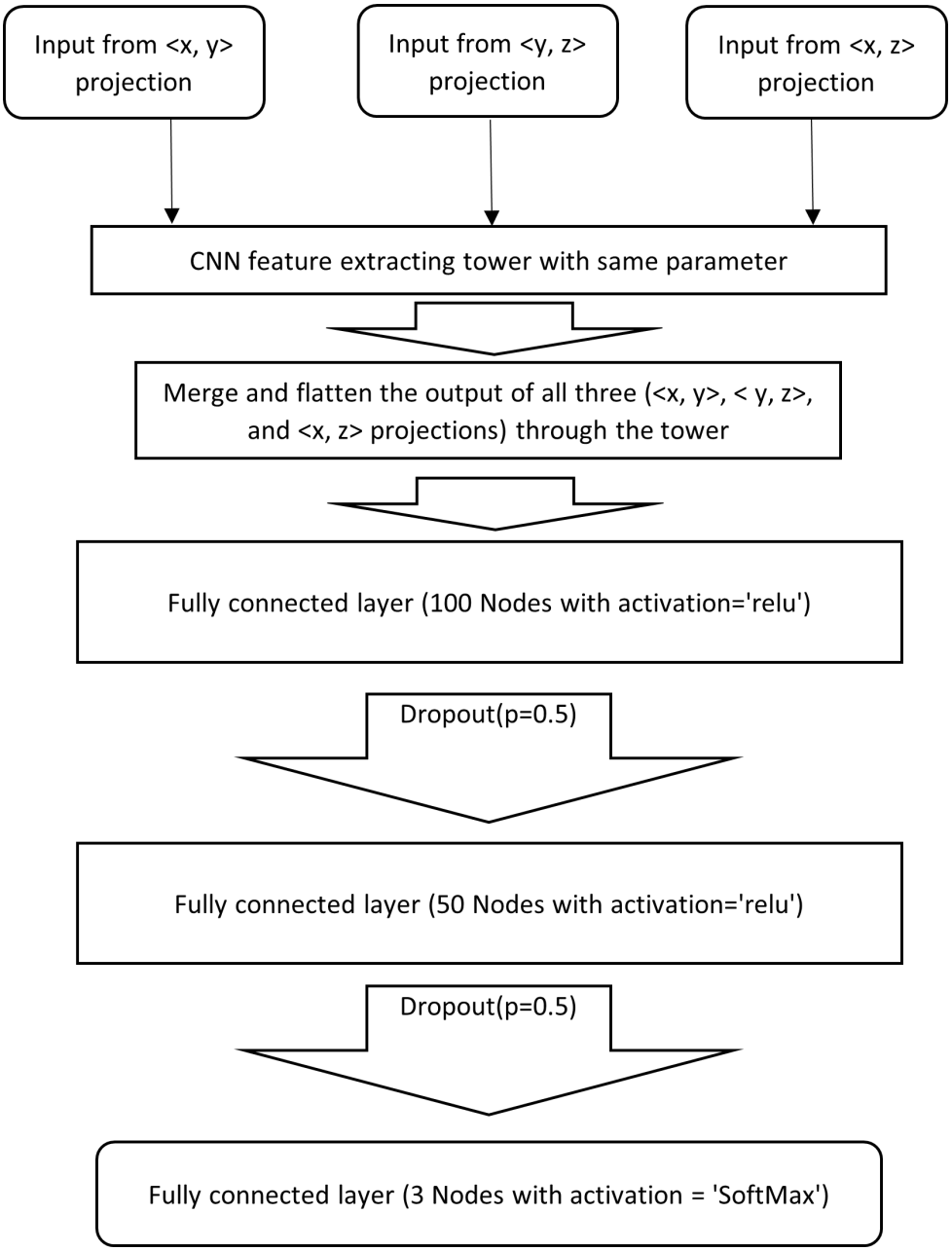


Figure 1.4: The general model with same parameters for tower.



chance to lose by the model, due to the ambiguity of 0s representation of the property. Because 0s holds the information of both *Acidic*, and *Base*, and the value not assigned as *Acidic*, or *Base*. On top of this the taken property represent by 0 is only considered (trained) by the bias of the Neural-network. Such that assigning one property value as 1 and the opposite as 0, gives a partial weightage in the property assignment. Thus, split them into two properties and assign 1 and 0, is one solution (as shown in *Appendix* D and *Appendix* E); Another solution is, one of the property representations, among them is factored by "-1" in the properties and merged as one property. This will make the

overall number of properties reduced.

The representation without negative (*Appendix* D and *Appendix* E) give all possibilities of properties in 0 to 1, to use the useful combination of properties among them by force the model to train parameters effectively; but one question arises here, is useful information can feed it directly make the efficient model? (while assigning these properties another question arises, is assign 1 and -1 preferable in effectiveness?, because we use *Relu*, that gives positive outcome preferable (any way these can be done by negative weights); how we prioritize among *Yes* and *No* values of the properties.

- ✓ *Yes* column's 1's is factored by 1; thus 0 remain as it is
- ✓ *No* column's 1's is factored by (-1)

Is enough or, along with this assigning another property as

- ✓ *Yes* column's 1's is factored by -1
- ✓ *No* column's 1's is factored by 1

For an **E.g.:** In *Appendix* E (Table. E.1's column "Side-chain properties"),

- ✓ *Hydro-phobic*: *Yes* column's 1's is factored by 1
- ✓ *Hydro-phobic*: *No* column's 1's is factored by -1 One more possibility assigns the extra property like *No-Hydro-phobic*: where assign
- ✓ *No-Hydro-phobic*: *Yes* column's 1's is factored by -1
- ✓ *No-Hydro-phobic*: *No* column's 1's is factored by 1

All the properties are fell this kind of opposite behavior shown in *Appendix* E's Table. E.1's (Side-chain properties) can be changeable, such as,

- ✓ *Hydro-phobic*: *Yes* and *No*
- ✓ *Polar*: *Yes* and *No*
- ✓ *pH*: *Acidic* and *Base* (where the weights such as 0.3 and 0.6 also negated when it factored)
- ✓ *small*: *Yes* and *No*
- ✓ *Aliphatic* vs *Aromatic*: They are totally different in structure wise, thus those can be represented like binary

A property has opposite in *Appendix* Table.E.2's that can be changeable, that is

- ✓ *Gene expression and biochemistry*: *Yes* and *No*; and Conditionally may part as another group.

From the experimental outcomes, none of these feature representations have not shown any improvement in the performance at all.

Chapter 2

Level 2 Quarter model Evaluation

2.1 Quarter model Evaluation Introduction

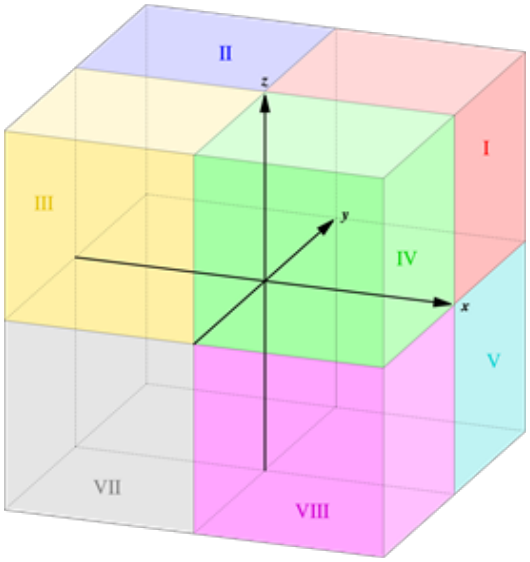
From the obtained results of experiment performed above (experiments performed with 3-projections) shows, none of the tilted PDBs does not show any improvement in performance at all.

There are chances such that projection of different surface $C\alpha$ residues may fall on top of each other (the one corner of principle direction edge may map near to the other corners of principle direction edge, this may give false information to the model). To fix this problem three tactics proposed,

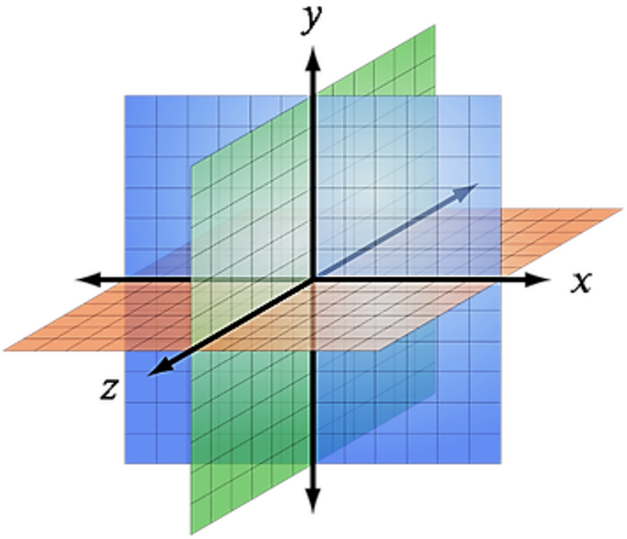
1. Best way to tackle this mapping problem, using 3D convolution, unfortunately that consume lot of memory; and it is not supported by current technology for this problem. Because one PDB structures surface $C\alpha$ coordinates with dimension consume around $\approx 2.7GB$ and $\approx 2.12GB$ for

Figure 2.1: Quadrarizaions visualization.

(a) How the quadrants separations visualization, Image credit: [21].



(b) How the quadrants projections going to be for minimum level visualization, Image credit: [29].



21 channels and 17 channels respectively; apart from this, the model's parameter calculations needs memory as well.

2. To avoid memory issue and almost preserve the positions, can be done by using the principally tilted coordinate and split the data into quadrants. Then using the projections (on $< x, y >$, $< y, z >$, and $< x, z >$) of quadrants separately. This may save the memory with preserving the positional information; and solve the problem of handling the PDBs' deposited in different principle direction/ orientation. (Fig. 2.1 visualize the quadrants).
3. Or tilt the PDB's principle direction into some 45° direction, but this approach does not preserve the positional information in all directions.

Since the tactic-2 is promising, tactic is evaluated, with different deep learning models as mentioned in flowing Section. As per the previous evaluations (mentioned in subsection 1.3.2) with different dataset creations with negation included datasets, did not show that much performance difference. Thus, the models proposed in this section, were only evaluated with positive normalized representations of general biochemical properties (as mentioned in *Appendix* E's proposed 21 amino acids' properties and the *Appendix* D's level 1 proposed properties). Many models(different architechtures) are proposed and evaluated in this section. The details of the evaluation results are presented in the following sections. Further, to elaborate the architecture only the selected model architectures are presented in section 2.7 .

Surprisingly, the quarter models outperformed even with different (randomized) cleaned (which is explained in 2.5.1 *Randomized data evaluation*) dataset each time. Different architectures (mention the section 2.7 and 2.8) are built and evaluated in the randomized data set. To select the model, the dataset is fixed (no randomization) and their performance is evaluated (mention the section 2.5.2). The details about selected models and dataset is explained in the following sections.

2.2 Introduction to experiments of quarter model's approaches and classification

Providing primary structure information, while separating PDB structures (3-D structures) for training may make a difference in functional detection, unless there are lot of PDB structures available. Thus, two approaches proposed to check the impact due to considering the primary structure information while separating PDB structures for training. In Approach 1, primary structure wise information is not considered at all, all the PDBs are pooled together; on the other hand, Approach 2 the PDBs are separated as training and testing based on primary structural information. Only Tier 1's COSMIC genes belongs to ONGO, TSG, and Fusion classes are used in both approaches.

In both approaches, two different functional classifications are done such as functional classification of isoforms (PDBs), and functional classification of primary structure (from the functional classification of isoforms results gained through deep learning model). Experiments' data selection is elaborated in section 2.3 . There are two different functional classifications, such as, Functional classification of PDBs (Isoforms), and Functional classification of primary structure. The details of the functional classifications are elaborated in section 2.4 .

In Approach 2 the PDBs are separated as training and testing based on primary structural information (this make sure same primary structural, preprocessed PDB files as mentioned in Section 1.2, not fall in both such as training, and testing as elaborated in section 2.3.2). Level 1(Chapter **??**) just pooled all the PDB files and randomly selected 85 % of the data (PDB files) for training and rest for testing; thus the model(proposed in [33] / Chapter **??** Level 1) only classify the isoforms (PDB files/3D structures).

2.3 Data selection for approach 1 and 2

Please check the *Appendix* I (Training and testing dataset selection from Tier 1) for more details about the number of primary structures and number of PDB files were used in the experiment.

2.3.1 Approach 1

To validate the different combinational (the blend of different primary structures') PDB_{SITE}s' contribution to the functionality among ONGO Vs TSG Vs Fusion, 10-fold cross validation is proposed. The PDB_{SITE}s are split into 10-folds balanced dataset classes and validated. Thus, cleaned (V88 mapping) ONGOs' 1031, TSGs' 706 and Fusions' 371, PDB_{SITE}s are shuffled separately (to randomize the PDB_{SITE}s distribution among each classes), split into 10-sets (while splitting make sure each has contains almost same number of PDB files).

2.3.2 Approach 2

The PDB_{SITE}s are related to the primary structural information. Therefore, the dataset for training and testing is selected based on the primary structural information. If the primary structure is chosen for training, then all the PDB_{SITE}s corresponding to that primary structure should be in the training set. As mentioned in subsection 1.2.1 cleaned dataset is obtained and used for training.

The removed/ cleaned data consists of 2108 PDB_{SITE}s belongs to 138 genes. From that, 1574 PDB_{SITE}s belong to 105 genes were used for training, and the remaining 33 genes and their corresponding 545 PDB_{SITE}s were used for testing. As mentioned in section 2.7 (and 2.8.2), the training set's 1574, 3D coordinate feature set was converted into three 2D feature sets ($1574 \times 24 = 37776$). Thus, 37776 with $128 \times 128 \times 21$ feature maps, were used to train the deep learning model. Please check the *Appendix* I for more details about the number of gene (primary structure) and number of PDB_{SITE}s used in the experiment.

2.4 Classification

Classification is performed in two ways such as classification of PDB_{SITE}s (Isoforms, as mentioned in subsection 2.4.1), and classification among primary structure (as mentioned in subsection 2.4.2). The ways are,

1. $Class\_classified$**:** Presenting the combination of classes (such that first check probabilities of classes higher than 40 %, if not check higher than 30 %)

2. *Most_probable_class:* Directly assign the highest probable class, such that it assigns mostly one class. Sometime two classes are assigned, because two classes share higher probability (like 50 % each or 40 % each **etc.**)

In both ways, five predicted outcomes are presented, such as *PDB/ Gene*, probability of three classes(*ONGO*, *TSG*, *Fusion*), and *Predicted class*.

2.4.1 Functional classification of PDBs (isoforms)

The deep learning model (BIR as mentioned in section 2.7) is used to predict the preprocessed PDB_{SITE}s (as mentioned in section 1.2). And the model output the PDB_{SITE}s' probabilities for each class (ONGO, TSG, and Fusion). In Approach 1, the classification of PDB_{SITE} is chosen when the PDB_{SITE} fell the validation set (among the 10-folds) for fare performance analysis. Classification of rest (Tier 2s' and remaining Tier 1s' gene groups apart from **ONGO**, TSG, and Fusion) of the COSMIC genes PDB_{SITE}s are obtained by ensemble the Approach 1's best 10-fold models' predictions.

2.4.2 Functional classification of primary structure

The primary structures sequence's specific portion may contribute specific role in functionality. This (primary structures sequence's specific portion's) role in functionality can be roughly/ rigidly (based on number of available aligned isoforms and the aligned portion) obtained by integrating the aligned (overlapping) isoforms (PDB structure) functionality and the overlapping sequence information. The overlapping sequence information must account the PDB structure's sequence overlapping length with the corresponding primary structure's sequence and number of other PDB structures shares that overlapping portion. If more than one PDB structures share one portion, then define method/ methods to give weightage to each of these shared PDB structures' functionalities on the primary structures' functionality.

The primary structures' corresponding PDB structures' functional classifications were obtained from subsection 2.4.1 . Sequence information (overlap/ aligned starting and ending position of sequence) for each PDBs were gained, while preprocessing (as mentioned section 1.2). There are plenty of Methods to consider the overlapping (aligned) sequence information to assign the function to the primary structure from their corresponding PDBs' functional probabilities (gained from deep learning model). In this experiment, only 3-Methods are proposed to predict the functional probability of the primary structure (with more than one PDB_{SITE}. Here PDB_{SITE} is the preprocess PDB witch obtained by $X - ray$ defragmentation, overlapping sequence length (>81), should have /emphSITE information, and cleaned/ only belongs to one class).

Methods proposed:

1. $Method_1$*:* This method uses the primary structure's all PDB_{SITE}s for fraction calculation, chooses the remaining highest length and calculate the fractions.
2. $Method_2$*:* This method uses non-overlapping PDB_{SITE}s to cover the primary structure much as possible
3. $Method_3$*:* This method uses overlapping PDB_{SITE}s to cover the primary structure much as possible

Figure 2.2 $Method_1$ (PseudoCode)

1: Highest lengthPDB_{SITE} is selected from the left-group.
2: Then the overlapping fractions (fraction of considered length overlapping with the highest length) of rest ofPDB_{SITE}s (the group ofPDB_{SITE}s for the primary structure which overlaps with selectedPDB_{SITE} excluding selectedPDB_{SITE}) with the selectedPDB_{SITE} are calculated.
3: From that start and end position of overlappingPDB_{SITE}s sequence length is updated (increased accordingly with the overlap with the selectedPDB_{SITE}s). Leave the highest lengthPDB_{SITE} and the remainingPDB_{SITE}s in the group is called as left group. If the left group has at least onePDB_{SITE}, then use left-group again from step 1.
4: Finalize overall fractions together, by pooling all the selected highest lengths in step 1. And their overlapping fractions contribution is calculated by normalize them (overlapping fractions) using their summation of selected highest lengths.

Finally, the *Ensemble*: This combine (average) all the methods mentioned above (such as $Method_1$, $Method_2$, and $Method_3$). If the primary structure has only one PDB_{SITE} then it directly took the probability value of PDB_{SITE}, those kinds of primary structures named as *Direct*. Else primary structure's probability is calculated by using all these three methods.

In these methods, primary structures were considered as groups, where "group" means each primary structure has PDB_{SITE}s (group) for the corresponding primary structure. A detailed explanation of three methods are given below. Further, hypothetical examples for each method is attached as *Appendix* J, where numerical values are used to make it clear. And the *Appendix* K summarizes the complexity of these proposed methods.

$Method_1$

The only algorithm analyzing all the PDB_{SITE}s in the group (belongs to each primary structure separately). Other methods are not using all PDB_{SITE}s in the group. The method extracts the highest length PDB_{SITE} for fraction calculation. The pseudo code for $Method_1$ is shown in Fig. 2.2.

$Method_2$

Uses dynamic programming to cover the maximum length of the primary structure sequence with non-overlapping PDB_{SITE}s. The pseudo code for $Method_2$ is shown in Fig. 2.3.

$Method_3$

The only difference from the $Method_2$ and $Method_3$, $Method_3$ chooses the maximum sequence cover as possible with allowing overlap, and small number of PDB_{SITE}s (their sequence information) as possible. Thus, negate the overlapped part of j^th PDB_{SITE}'s primary sequence at j step's back covering.

2.5 Quarter model evaluation summery

Due to lot of experiments done in this project the results are split and presented in the Tables with parts. Each tables' foot notes present, their corresponding locked feature/ data representation; further

Figure 2.3 $Method_2$ (PseudoCode)

1: Starting positions of PDB_{SITE} (for the group) are sorted in ascending order
2: Iterate until find $cover\,(m)$; {cover all the PDB_{SITE} ("m" is the total number of PDB_{SITE} in the group)}

Initializing condition for dynamic programming

$$cover\,(0) = 0$$

if the next PDB_{SITE} is not overlapped with the previous one (starting position of the next PDB_{SITE} is higher than the covered PDB_{SITE}.)

$$cover\,(n+1) = cover\,(n-j) + length\,(next\ PDB_{SITE})$$

if, it is overlapped, go back until, its non-overlap (example: if, it takes j steps to go back)

$$cover\,(n+1) = max \begin{cases} cover\,(n) \\ cover\,(n-j) + length\,(next\ PDB_{SITE}) \end{cases} \tag{2.1}$$

Exception: If two or more PDB_{SITE} has same start end position, both are considered as one for the calculation

Figure 2.4 $Method_3$ (PseudoCode)

$$cover\,(n+1) = max \begin{cases} cover\,(n) \\ cover\,(n-j) + length(nextPDB_{SITE}) - overlap(j^{th}PDB_{SITE}) \end{cases} \tag{2.2}$$

Where the $overlap(j^{th}PDB_{SITE})$ means the overlapping between the $cover(n-j)$'s PDB_{SITE}s group and the $j^{th}PDB_{SITE}$

Exception: As mentioned in the pseudoCode of $Method_2$ in Fig. 2.3, if two or more PDB_{SITE}s has same start end position, both are considered as one for the calculation

Table 2.1: Abbreviations of Deep models used in the Table of Level 2 experiment

Abbreviation	Description/ Details
BIR	Brain inception residual architecture is presented in the subsection 2.7
BIR_7x1_3x3 BIR_7x1 BIR_7	Brain inception residual architecture with kernel 7 presented in different ways as explained in the subsection 2.9
BI_O	Brain inception only architecture is BIR without residual
PIR	Parallel inception residual architecture is like all the towers (feature extractor for projections) have same parameters.
PI_O	PIR without residual(addition)
Parallel k-3 Parallel k-5 Parallel k-7	Level-1 architecture contains 24 projections with "swish" activation with the given kernel like k-3 mean equipped with 3×3 kernel for convolution layer

models name is presented in abbreviations are shown in Table. 2.1.

In order to understand these experiment results first recapping the data set extraction as men-

tioned in subsection (**??** "Basic understanding of data") retrieval the COSMIC [28] census' genes were mapped to PDB_{id} using mapping database UniProt [34]. These mapped PDB_{id} are download (retrieve) from protein data bank website[23]. These PDBs' surface $C\alpha$ functional contribution carried by the gene (to primary structure) is evaluated in different extent by two approaches. Approach 1 does not consider the primary structure while separating the PDBs, and Approach 2 considers primary structure information while splitting (separation) the PDBs as training and test set. In both approaches the PDBs which belongs to different functional class primary structures (overlap PDBs) are removed; the selected (non-overlapping) PDBs are used in these evaluations. Approach 1 does not consider primary structure while split the PDBs for training set; thus this Approach 1's models or data (feature) representation can be evaluated more effectively using 10-fold cross validation (the possibility of splitting the data in to 10- equal groups; each time one set used as test set). On the other hand, equal splitting is not possible, due to Approach 2 consider the primary structure details while splitting cause, different kind of functionality (with in one class the primary structures functionality contribution to that class may not equal) of the primary structures' PDBs placed in training set may give some bias; different genes (to primary structures) contain different number of PDBs, **etc.**. Thus, the Approach 1, and Approach 2 data separation and evaluation method is named as "10-fold" and "Primary structure separated" accordingly.

In all these experiments balanced dataset is used for training and evaluation. Since the number of PDBs belongs to each class (ONGO, TSG and Fusion) is different. To train the deep models, each epoch's batch's data contains the same number of PDBs fed from ONGO, TSG and Fusion.

Since the number of PDBs belongs to Fusion class is comparably low from ONGO class or TSG class. While construct the batch, mini batch with factor of five PDBs is considered. In each mini batch two PDBs from each ONGO, TSG class and one PDB from Fusion is retrieved, once all the PDBs in one class group is done, then randomized that class again before extracting the new PDBs to the mini batch. Since the number of PDBs belongs to ONGO is high in the considered dataset; the batch size is decided based on the number of PDBs belongs to ONGO class. To assign the batch size as factor of five and to cover all the PDBs in ONGO in one forward and backward propagation through neural network, the equation (2.3) is used.

$$Batch\ size = \ ceil\left(\frac{5\ \times\ number\ of\ ONGO\ train\ PDBs}{2}\right) \tag{2.3}$$

The Approach 1 evaluation may depend the 10-fold dataset creation. First each class PDBs are randomized, and then split into 10 equal groups; if the number of PDBs are not fitted to 10 group balance splits, then the initial groups have one extra PDB than the last group. Then the split groups of each class are joined to create the "balanced-10-fold data". Thus, this data creation may handle in two ways such as,

1. **Randomized data:** Each time create the "balanced-10-fold data" to train the model. In this way 10-groups have different combination of PDBs from ONGO, TSG and Fusion classes. Data representation/ feature representation is not random, it changed based on the evaluation.
2. **Locked data:** Once the "balanced-10-fold data" is created locked that data and use that data again and again to train different models. This will make sure, same combination of PDBs

Table 2.2: Random data results of Approach 1(10-fold) with soft-threshold and 21-channels

Deep-Models	PDB				Primary gene function			
	All three	ONGO	TSG	Fusion	All three	ONGO	TSG	Fusion
BIR	93.5	94.67	94.3	88.68	88.4	95.56	93.75	65.52
BIR_7x1_3x3	91.18	95.05	90.4	81.94	87.7	97.78	93.75	58.62
BIR_7x1	92.27	94.57	93.5	83.56	85.5	91.11	92.19	62.07
BIR_7	90.8	94.86	90.9	79.25	76.8	88.89	78.12	55.17
BI_O	90.65	92.34	94.2	79.25	80.4	86.67	90.62	48.28
PIR	90.23	92.34	93.6	77.9	80.4	86.67	95.31	37.93
PI_O	89.99	93.7	90.9	77.9	75.4	88.89	85.94	31.03
Parallel k-3	90.13	93.02	89.7	83.02	80.4	77.78	85.94	72.41
Parallel k-5	90.28	94.18	91.1	77.9	81.9	95.56	87.5	48.28
Parallel k-7	90.13	93.11	91.8	78.71	84.8	91.11	92.19	58.62

[a] Biochemical representation: 21 amino acids'
[b] SITE must: True
[c] Basic normalised prop Vs Normalised negation included property: Basic normalised prop

from ONGO, TSG and Fusion classes. In this evaluation sometimes data representations /feature representations were also locked separately, to evaluate the performance based on representations/ feature representations.

2.5.1 Randomized data evaluation

If the 10-fold set's evaluation/ performance in the randomized data set with different architectures and feature representation works well. This confirms the hypothesis "the functional contribution of the PDB structure is represented by the surface residues" more effectively. Because the performance is not affected by any data combination of PDBs, on different architectures.

From the Table. 2.2, and Table. 2.3 results of random data set with different architectures, three class PDB classification accuracy always lies above 89 % (or $\approx$ 90%) and primary structure function classification lie above 75 %. Among the whole evaluation with the surface define threshold as "soft threshold" with 21 channels and Brain inception residual architecture (BIR) architecture performed well (the results of the model are presented in Table. 2.2. These performances confirm the data representation (quarter model representation) is suited to confirm the hypothesis in Approach 1, as well as the performance is not affected by any data combination of PDB structures on different architectures.

2.5.2 Locked data evaluation: best architecture selection and data representation selection

This evaluation is performed to select the best feature representation and architecture selection. Locking (not varying) the data representation is categorized as

1. ***locked PDB selection:*** not randomizing the whole train and test set (here no change in the PDB occurrence in Train or Test set for each model.

Table 2.3: Random data results of Approach 1(10-fold) with soft-threshold and 17-channels

Deep-Models	PDB				Primary gene function			
	All three	ONGO	TSG	Fusion	All three	ONGO	TSG	Fusion
BIR	92.83	94.08	93.9	87.33	88.4	88.89	95.31	72.41
BIR _7X1_3X3	91.55	93.4	95.6	78.71	81.9	88.89	93.75	44.83
BIR _7X1	91.03	93.69	94.5	77.09	82.6	88.89	95.31	44.83
BIR _7	93.12	96.41	94.2	81.94	86.2	95.56	93.75	55.17
BIR_k3_k5-same	91.03	91.46	94.3	83.56	84.1	84.44	92.19	65.52
BI_O	89.27	93.79	87.7	79.78	84.1	95.56	87.5	58.62
PIR	90.18	91.17	93.6	80.86	76.1	82.22	85.94	44.83
PI_O	91.31	91.55	92.9	87.6	89.9	88.89	92.19	86.21
Parallel k-3	91.65	96.21	89.4	83.29	84.1	93.33	81.25	75.86
Parallel k-5	90.03	93.69	88.5	82.75	83.3	95.56	84.38	62.07
Parallel k-7	90.6	96.8	87.7	78.98	84.8	97.78	85.94	62.07

[a] Biochemical representation: 20 amino acids'
[b] SITE must: True
[c] Basic normalised prop Vs Normalised negation included property: Basic normalised prop

Table 2.4: Random data results of Approach 1(10-fold) with man-threshold

Amino acids	Channels	Deep-Models	PDB				Primary gene function			
			All three	ONGO	TSG	Fusion	All three	ONGO	TSG	Fusion
21	21	BIR	91.08	95.05	94.9	72.78	83.3	86.67	96.88	48.28
20	17	BIR_7X1_3X3	90.84	97.19	86.8	80.86	80.4	100	81.25	48.28
20		BIR_7X1	Since the outcome of BIR and BIR_7X1_3X3 not better than/nearly to soft threshold the random dataset evaluation is skipped; and these models are evaluated in Locked dataset evaluation.							
20		BIR_7								
21	21	BIR_7X1_3X3								
21		BIR_7X1								
21		BIR_7								

[b] SITE must: True
[c] Basic normalised prop Vs Normalised negation included property: Basic normalised prop

2. ***locking along the feature representation:*** The feature representation is fixed in all the experiments. Experiments fell in this category evaluates the best feature representation (by channels, these channels contain the selected surface residues by threshold condition to define the surface, and biochemical property assignments). Thus, this experiment evaluates the representations dependencies such as,

   (a) **Proposed threshold condition (soft) to define the surface:** to check and confirm the threshold condition is an effective threshold condition. Two main threshold conditions are experimented such as,

      (i) *Man:* threshold condition for surface $C\alpha$ is defined by the Manmade PDB structure threshold condition. Among COSMIC V88 obtained, mapped via UniProt and filtered (aligned primary structural length > 81, must has $SITE$ information and cleaned PDBs) PDB_{SITE}, only PDB_{SITE} *"3ZRC"* belongs to TSG class has $SITE$ information as

Author (like *"REMARK 800 EVIDENCE_CODE: AUTHOR"*). Thus, for define the threshold conditions such as $C\alpha$ threshold depth, and Residue threshold depth (definition of these threshold conditions are in the *Appendix* C) the maximum depth of all the $MOTIF$s ($SITE$s) reported in PDB_{SITE} *"3ZRC"* is considered accordingly. Such that, $C\alpha$ threshold depth: 37.54581036735755 and Residue threshold depth: 36.756521900380775 assigned.

(ii) *All $C\alpha$:* Considering all residues occurred in the PDB_{SITE}.

(b) **Proposed biochemical representation for residue with** $SITE$(here the $SITE$ details give the residue participate in any kind of interaction with other biomolecules [37])

(i) *21 channels:* 20 biochemical properties for amino acid + amino acid belongs to any $MOTIF$ ($SITE$). This feature representation includes the 21'st amino acids biochemical property as well. This 21'st amino acid *"Selenocysteine"* ("U") is confirmed as occurrence in human, and it's participation in cellular process [27]. For the time being only one of the PDB fell in our evaluation; in the future it will increased. This biochemical property representation (details about the property representation is in *Appendix* E) can be considered as future biochemical representation.

(ii) *20 channels:* the above (21-channel) representation without $SITE$ details. Here the PDBs without $SITE$ information are included.(And the PDBs satisfy all the other preprocessing steps and conditions).

(iii) *17 channels:* 16 biochemical properties (details about the property representation is in *Appendix* D) for amino acid + amino acid belongs to $SITE$. This only consider the 20 most generally used amino acids' properties.

(iv) *16 channels:* the above (17-channel) representation without $SITE$ details. Here the PDBs without $SITE$ information are included.(And the PDBs satisfy all the other preprocessing steps and conditions).

Architecture selection

Experiments fell in this category is aimed to select the best architecture among the proposed/ evaluated architectures. This selection/ evaluation is done in both Approach 1 and Approach 2.

In Approach 1, the PDB structures in the 10-fold set is locked (same 10-fold dataset used for Training and testing); thus, the performance may highly depend on the feature representation with best deep-model architecture in this evaluation. Thus, first lock the feature representation and evaluate the performance to choose the best proposed model.

Mainly two feature representations are locked/ selected: the different between the selected feature representation is, one with 21 amino acid biochemical representation, and another with 20 amino acid biochemical representation. The selected feature representations' corresponding evaluation results are presented in the Table. 2.5, and Table. 2.6 accordingly. From the results of both Tables(Table. 2.5, and Table. 2.6), among all the models evaluated, the BIR model with 21 channel representation works well (in both PDB classification and primary structure classification from the predictions). The BIR model's performance is comparably higher in both classifications, with 17 channel representation

Table 2.5: Locked PDB selection results of Approach 1(10-fold) with soft-threshold and 21-channels

Deep-Models	PDB				Primary gene function			
	All three	ONGO	TSG	Fusion	All three	ONGO	TSG	Fusion
BIR	93.5	94.67	94.3	88.68	88.41	95.56	93.75	65.52
BIR_7X1_3X3	90.13	93.02	90.7	81.13	80.43	97.78	84.38	44.83
BIR_7X1	91.89	96.8	89.4	83.02	78.99	93.33	75	65.52
BIR_7	90.46	94.86	89	81.13	78.26	91.11	81.25	51.72
BI_O	91.32	93.6	92.2	83.29	80.43	93.33	87.5	44.83
PIR	90.09	93.89	88.2	83.02	83.33	95.56	87.5	55.17
PI_O	90.65	94.86	92.4	75.74	76.09	93.33	85.94	27.59
Parallel k-3	89.52	95.44	86.5	78.71	75.36	88.89	78.12	48.28
Parallel k-5	89.85	94.96	87.1	80.86	83.33	97.78	81.25	65.52
Parallel k-7	89.61	94.08	87.7	80.86	81.16	95.56	79.69	62.07

[a] Biochemical representation: 21 amino acids'
[b] SITE must: True
[c] Basic normalised prop Vs Normalised negation included property: Basic normalised prop

Table 2.6: Locked PDB selection results of Approach 1(10-fold) with soft-threshold and 17-channels

Deep-Models	PDB				Primary gene function			
	All three	ONGO	TSG	Fusion	All three	ONGO	TSG	Fusion
BIR	92.12	93.59	95.6	81.4	84.78	84.44	96.88	58.62
BIR_7X1_3X3	92.17	95.44	92.5	82.48	87.68	97.78	92.19	62.07
BIR_7X1	91.65	94.76	91.9	82.48	83.33	95.56	87.5	55.17
BIR_7	91.36	95.44	90.5	81.67	86.23	97.78	90.62	58.62
BI_O	91.5	92.33	93.6	85.18	86.96	93.33	89.06	72.41
PIR	89.99	92.91	92.8	76.55	75.36	84.44	87.5	34.48
PI_O	90.79	94.56	89.9	81.94	82.61	97.78	81.25	62.07
Parallel k-3	89.65	95.05	88.8	76.28	78.26	93.33	79.69	51.72
Parallel k-5	89.99	95.05	88.2	79.25	81.88	95.56	81.25	62.07
Parallel k-7	90.75	94.85	89.8	81.13	80.43	95.56	84.38	48.28

[a] Biochemical representation: 20 amino acids'
[b] SITE must: True
[c] Basic normalised prop Vs Normalised negation included property: Basic normalised prop

results as shown in Table. 2.6. Thus, this evaluation confirms BIR architecture is better among all the proposed and evaluated architectures.

In Approach 2, the PDB structures in the training set is locked (same PDB dataset used for Training and testing; because the PDB structures are split based on the primary structural deatils); thus, the performance may highly depend on the feature representation with best deep-model architecture in this evaluation. Thus, first lock the feature representation, and evaluate the performance to choose the best proposed model.

Mainly two feature representation are locked/ selected (feature representation with 21 or 20 amino acid biochemical representation). The selected feature representation evaluation results are presented

Table 2.7: Locked primary structure for train data results of Approach 2(Primary structure separated) with soft-threshold and 21-channels

Deep-Models	PDB				Primary gene function			
	All three	ONGO	TSG	Fusion	All three	ONGO	TSG	Fusion
BIR	70.66	73.93	68.4	0	51.5	53.85	71.43	0
BIR _7X1_3X3	69.56	73.03	65.8	0	51.5	61.54	64.29	0
BIR _7X1	76.2	81.8	60.8	5.56	63.6	84.62	71.43	0
BIR_7	74.72	78.43	70.9	0	66.7	84.62	78.57	0
BI_O	68.27	72.81	58.2	0	39.4	38.46	57.14	0
PIR	66.97	69.66	67.1	0	60.6	76.92	71.43	0
PI_O	71.22	73.93	72.2	0	60.6	69.23	78.57	0
Parallel k-3	68.08	70.79	68.4	0	57.6	53.85	85.71	0
Parallel k-5	70.48	73.26	70.9	0	63.6	84.62	71.43	0
Parallel k-7	70.66	75.06	62	0	57.6	76.92	64.29	0

[a] Biochemical representation: 21 amino acids'
[b] SITE must: True
[c] Basic normalised prop Vs Normalised negation included property: Basic normalised prop

Table 2.8: Locked primary structure for train data results of Approach 2(Primary structure separated) with soft-threshold and 17-channels

Deep-Models	PDB				Primary gene function			
	All three	ONGO	TSG	Fusion	All three	ONGO	TSG	Fusion
BIR	78.04	83.6	64.6	0	60.6	84.62	64.29	0
BIR_7X1_3X3	71.59	75.51	65.8	0	54.6	69.23	64.29	0
BIR_7X1	72.14	75.06	72.2	0	54.6	46.15	85.71	0
BIR_7	72.32	75.28	70.9	5.56	48.5	53.85	64.29	0
BI_O	77.12	82.47	64.6	0	54.6	84.62	50	0
PIR	68.08	69.44	76	0	54.6	61.54	71.43	0

[a] Biochemical representation: 20 amino acids'
[b] SITE must: True
[c] Basic normalised prop Vs Normalised negation included property: Basic normalised prop

in Table. 2.7, and Table. 2.8. From the results of these Tables, among all the models, the BIR model with 17 channel representation works well in PDB classification; and BIR_7 model with 21 channel representation works well in primary structure classification. Thus, this evaluation also supports BIR architecture is better among all the proposed and evaluated architecture.

Data representation selection

To find the best feature representation, the all the categories mentioned in *"locking along the feature representation"* should be locked and experimented independently. Further the category *"Basic normalised prop Vs Normalised negation included property"* can be excluded in these experiments. Because, it is already verified in the subsection 1.3.2 , and variations on categories of feature repre-

Table 2.9: Locked primary structure for train data results of Approach 2(Primary structure separated) with man-threshold and 21-channels

Deep-Models	PDB				Primary gene function			
	All three	ONGO	TSG	Fusion	All three	ONGO	TSG	Fusion
BIR	91.56	93.02	94.5	81.94	80.43	84.44	90.62	51.72
BIR _7X1_3X3	92.27	95.44	93.1	81.94	79.71	91.11	87.5	44.83
BIR _7X1	90.94	93.4	91.4	83.29	84.78	93.33	89.06	62.07
BIR _7	90.8	95.64	90.5	77.9	79.71	95.56	85.94	41.38

[a] Biochemical representation: 21 amino acids'
[b] SITE must: True
[c] Basic normalised prop Vs Normalised negation included property: Basic normalised prop

Table 2.10: Locked primary structure for train data results of Approach 2(Primary structure separated) with man-threshold and 17-channels

Deep-Models	PDB				Primary gene function			
	All three	ONGO	TSG	Fusion	All three	ONGO	TSG	Fusion
BIR	91.31	96.99	90.1	77.9	81.16	97.78	81.25	55.17
BIR _7X1_3X3	92.22	95.34	92.4	83.29	86.23	93.33	92.19	62.07
BIR _7X1	90.93	94.27	88.8	85.71	79.71	93.33	81.25	55.17
BIR _7	92.69	96.02	92.6	83.56	84.06	93.33	87.5	62.07

[a] Biochemical representation: 20 amino acids'
[b] SITE must: True
[c] Basic normalised prop Vs Normalised negation included property: Basic normalised prop

sentation does not give a bigger performance improvement/ decrement at all. Thus, this category can be skipped.

This selection/ evaluation also carried on both Approach 1 and Approach 2.

Evaluation of *"Proposed threshold condition"* To evaluate the selected soft condition for *"Proposed threshold condition"*. Let us check the other thresholds conditions such as,

(a) **Man:** To define the threshold, use the statistics from PDB with $SITE$ deposited by biologist, since the threshold condition is comparably less strict (assign more surface $C\alpha$ atoms) than the soft threshold.

(b) **All $C\alpha$:** whole PDB structure (not selecting the surface $C\alpha$ atoms). This condition does not check the surface at all.

If these conditions outperform the defined *"soft"* condition only, it can be said, the definition criteria of surface amino acids are not effective; or surface amino acids contribution to the function is less. Because the *"soft"* threshold condition is the strongest condition to define surface $C\alpha$ atoms among the conditions (*"Man"*, and *"All $C\alpha$"*) proposed/ evaluated; and the *"soft"* threshold condition get the performance with fewer neighborhood amino acids information presence for 3D structure (PDBs).

Table 2.11: Locked primary structure for train data results of Approach 2(Primary structure separated) with All $C\alpha$-threshold and 21-channels

Deep-Models	PDB				Primary gene function			
	All three	ONGO	TSG	Fusion	All three	ONGO	TSG	Fusion
BIR	91.75	94.18	91.8	84.91	84.06	91.11	92.19	55.17
BIR _7X1_3X3	91.79	94.76	93.3	80.59	84.78	95.56	93.75	48.28
BIR _7X1	91.22	96.02	89.7	80.86	80.43	91.11	84.38	55.17
BIR _7	93.03	95.93	91.9	87.06	81.88	97.78	79.69	62.07

[a] Biochemical representation: 21 amino acids'
[b] SITE must: True
[c] Basic normalised prop Vs Normalised negation included property: Basic normalised prop

Table 2.12: Locked primary structure for train data results of Approach 2(Primary structure separated) with All $C\alpha$-threshold and 17-channels

Deep-Models	PDB				Primary gene function			
	All three	ONGO	TSG	Fusion	All three	ONGO	TSG	Fusion
BIR	91.88	95.24	93.3	79.78	85.51	91.11	95.31	55.17
BIR _7X1_3X3	91.84	94.27	92.5	83.83	86.23	97.78	89.06	62.07
BIR _7X1	92.07	93.79	93.8	84.1	84.78	91.11	92.19	58.62
BIR _7	92.6	96.21	92.9	81.94	84.78	97.78	90.62	51.72

[a] Biochemical representation: 20 amino acids'
[b] SITE must: True
[c] Basic normalised prop Vs Normalised negation included property: Basic normalised prop

Approach 1 evaluate the performance among four BIR-based models such as BIR, BIR_7X1_3X3, BIR_7X1, and BIR_7. And the evaluation results of Approach 1 are presented in Tables. (2.5 and 2.6) and Tables.(2.9, 2.10, 2.11, and 2.12). From the evaluations results none of them break the performance of both (PDB and Primary structural function) classifications by BIR obtained with *"soft"* condition as shown in the Table. 2.5.

Approach 2's evaluates the performance only with the BIR model since it shows remarkable performance than other models. And the evaluation result of Approach 2 is presented in Table. 2.13. From the evaluation 21 amino acid feature representation break the performance of classification by BIR obtained with *"soft"* condition as shown in Table. 2.7. On the other hand, the evaluation 20 amino acid feature representation fell in middle between *"Man"* and *"All $C\alpha$"* conditions' performance of classification by BIR obtained with *"soft"* condition as shown in Table. 2.8. However, this cannot be enough to question the *"soft"* threshold condition; since these results are comparably low due to lack of PDBs to training the model; and all these results are near by 3 % in PDB classification.

Evaluation of *"SITE condition"* The evaluation is done in only Approach 2, as shown in Table. 2.13 the results are extremely low for without $SITE$ condition. Even some performance did not reach the 60 % at all. Thus, $SITE$ information is must in the data representation.

Table 2.13: Locked primary structure for train data results of Approach 2(Primary structure separated)

soft stat(old) / Man	amino acids	*SITE* must	PDB				Primary gene function			
			All three	ONGO	TSG	Fusion	All three	ONGO	TSG	Fusion
All $C\alpha$	21	TRUE	72.19	75.11	72.2	0	57.6	61.54	78.57	0
Man	21		73.48	77.13	69.6	0	60.6	76.92	71.43	0
All $C\alpha$	20		79.74	84.75	69.6	0	54.6	61.54	71.43	0
Man	20		72.88	75.73	73.4	0	54.6	53.85	78.57	0
All $C\alpha$	21	FALSE	Overall accuracy < 60 and ONGO accuracy < 65, TSG accuracy < 65							
Man	21									
soft(old)	21		62.5	68.37	68.3	0	53.1	63.64	71.43	0
All $C\alpha$	20		62.2	68.37	67.3	0	50	54.55	71.43	0
Man	20		63.72	71.94	65.4	0	50	72.73	57.14	0
soft(old)	20		64.33	69.39	72.1	0	65.6	90.91	78.57	0

[c] Basic normalised prop Vs Normalised negation included property: Basic normalised prop
[d] Deep-Models: BIR (Brain inception residual) only evaluated
[e] Amino acids 21 with $SITE$ must coloumn,as TRUE, and FALSE presented by 21, and 20 channels accordingly
[f] Amino acids 20 with $SITE$must coloumn,as TRUE, and FALSE presented by 17, and 16 channels accordingly without SITE 16 channels

2.6 Rounding limitation due to deep learning model

As mentioned in the limitations (Chapter **??**). These experiments uses the deep learning models, and the PDB structures presented images (projections of 3D structures); thus while mapping 3D structures cartesian coordinates to images, rounding error is unavoidable unless using 0.1 Å for pixel for the current deposits (since PDB structures are in different resolutions) but this may cost lot of memeory due to the image size.

Currently proposed experiments (of Level-2) with the normalization procedure proposed in *Appendix* C to transform the PDB to another coordinate template(since the template proposed to preserve the Amino acid position on the surface of the protein structure/ PDB) uses $24 \times 128 \times 128$ pixels. If the PDB structure with resolution as 1 Å is, normalized and presented by 0.1 Å resolution per pixel, may lead to $24 \times 568 \times 568$; because $(128 \div 2.25) \times 10 \approx 568$; please check the *Appendix* C for 2.25 factor selection.

Thus, the current memory usage for the PDB is factored by ≈ 20 due to $(568 \div 128)^2$; and this lead to computation cost.

The factorization 2.25 may avoid some errors. As it clearly seen the factorization make the residues further away by factor of 2.25, thus it may reduce the rounding error cause in higher resolution deposits $(1 < (2.25 \div resolution))$. On the other hand, lower resolution deposits $(1 > (2.25 \div resolution))$ may have higher rounding error.

So, in order to evaluate the template proposed for Level 2 and the rounding errors some experiments done with both extreme rounding (with ceil and floor), with the locked dataset (as mentioned in section 2.5.2) and BIR model. As mentioned in the section 2.5.2 the locked dataset is selected from random dataset evaluation performance best model (BIR) data set which results is highlighted* in Table. 2.14.

From the overall results both floor rounded models and ceil model with 17 channels representation

Table 2.14: Rounding error impact with different ways of rounding on SITE included soft-threshold representation

Locked	amino acids	Rounding	Channels	PDB				Primary gene function			
				All three	ONGO	TSG	Fusion	All three	ONGO	TSG	Fusion
✓	21	floor	21	93.26	95.15	94.48	85.71	88.41	93.33	96.9	62.07
✓	21	near*	21	93.5	94.67	94.33	88.68	88.41	95.56	93.8	65.52
✓	21	near	21	91.32	93.6	92.21	83.29	80.43	93.33	87.5	44.83
✓	21	ceil	21	90.7	92.53	93.34	80.59	80.43	93.33	89.1	41.38
✗	21	ceil	21	91.98	94.28	92.21	85.18	85.51	91.11	90.6	65.52
✓	20	floor	17	91.12	96.41	86.4	85.44	81.16	91.11	81.3	65.52
✓	20	near	17	92.12	93.59	95.61	81.4	84.78	84.44	96.9	58.62
✓	20	ceil	17	92.93	94.85	96.74	80.32	84.78	86.67	95.3	58.62

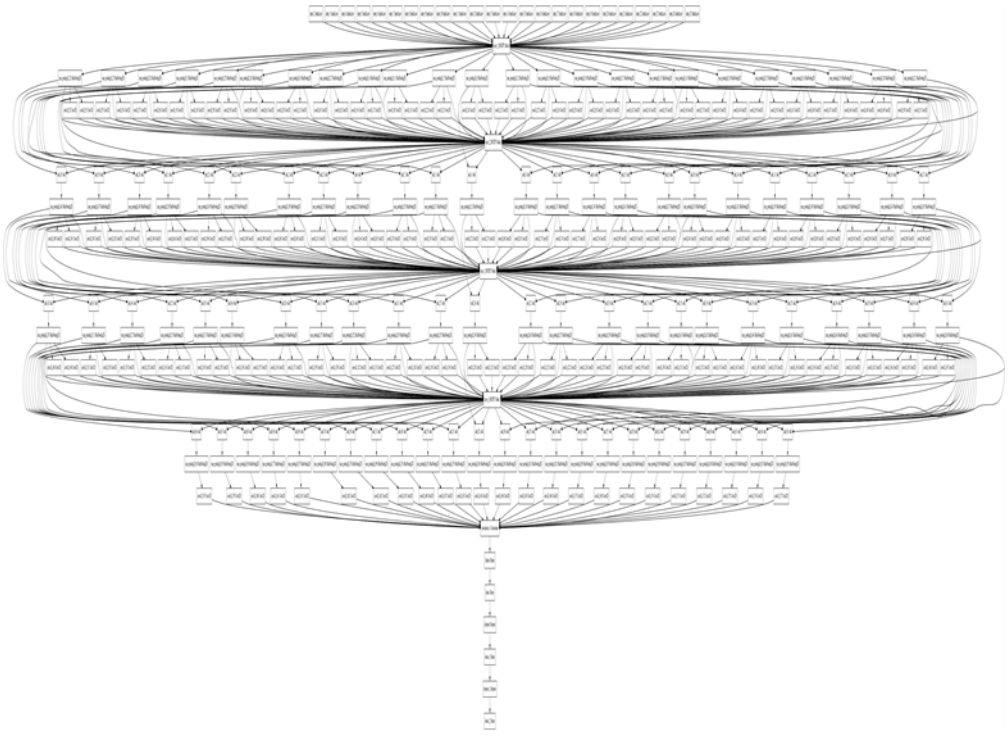
[c] Basic normalised prop Vs Normalised negation included property: Basic normalised prop
[d] Deep-Models: BIR (Brain inception residual) only evaluated
[g] near* locked data set is used; and the Locked coloumn ticked based on the same data set is used as the validation split; or splitted again (shufulled), it may endup in different validation group.
[h] rounding coloumn specify the round methods used, ceil: all fractions are rounded to higest nearest integer, floor: all fractions are rounded to smallest nearest integer, near: all fractions are rounded to nearest integer; e.g: 1.4 is ceil, near and small rounded to 2,1,and 1 accordingly.

does not show much variation. But the performance of ceil model with 21 channels representation classification of primary structure wise not met the expectations. To elaborate, another results of same near rounded locked dataset on the same architecture's results is shown for 21 channels. There the PDB wise classification is little low and primary structure wise functional classification is low as ceil rounded representation.

Even though the locked dataset forced to use the same validating groups to keep the groups of ONGO, TSG and Fusion PDB_{SITE}s as much as possible, while training the deep learning model random combination among those groups is not always same. This may cause loss of functional encoding capture by the model. Further of lacking Fusion class PDB_{SITE}s may cause the deep learning models highly dependent on initial conditions of weights with the random combination of the PDB_{SITE}s (mix of balanced PDB_{SITE}s from all three classes) provided while training the model. So, if the models reinitiated and trained again may beat the performance obtained by best model (BIR) which results is highlighted* in Table. 2.14. So, it is clear the rounding error does not make much impact. (the random combination data with rounded ceil of 21 channels representation obtain almost same performance as other rounded fixed data set).

2.7 Deep learning model architecture for quarter projections

Figure 2.5: Brain inception residual full architecture.



2.8 Representations of Deep learning model architecture for quarter projections

Different deep learning architectures are evaluated. Inception [31] and residual [11] networks are well known for, improvement of performance compare to general CNN (without inception or residual connection). Activation function $Swish$ [22] performs better than Rectified Linear Unit ($ReLU$) in deeper networks (per [22]

"Like ReLU, Swish is unbounded above and bounded below. Unlike ReLU, Swish is smooth and nonmonotonic.".)

Thus, especially the architechtures are evaluated with/ without naïve inception_V1 [31] in Layer_1 (as shown in Fig. 2.6), inception with dimensionality reduction [31], and with/ without residual connection (as shown in Fig. 2.6 or Fig. 2.8). And these architectures' almost all layers are equipped with $Swish$ activation function. ($ReLU$ is equipped sometimes in the feature extraction layers after $Max - pool$; $Softmax$ is always equipped in the final of fully connected output layer).

Architecture construction also considers (evaluated) the parameter selection; use same parameters for CNN feature extractions' same layer (inception module) of all projections or not (for an example, layer 2 is shown Fig. 2.7, here the dimensionality reduction CNNs are not same even in the same $2^n d$ layer's for each projections feature extractor; likewise, this same kind of architecture is repeated

Figure 2.6: *Layer_1_INCEPT* model.

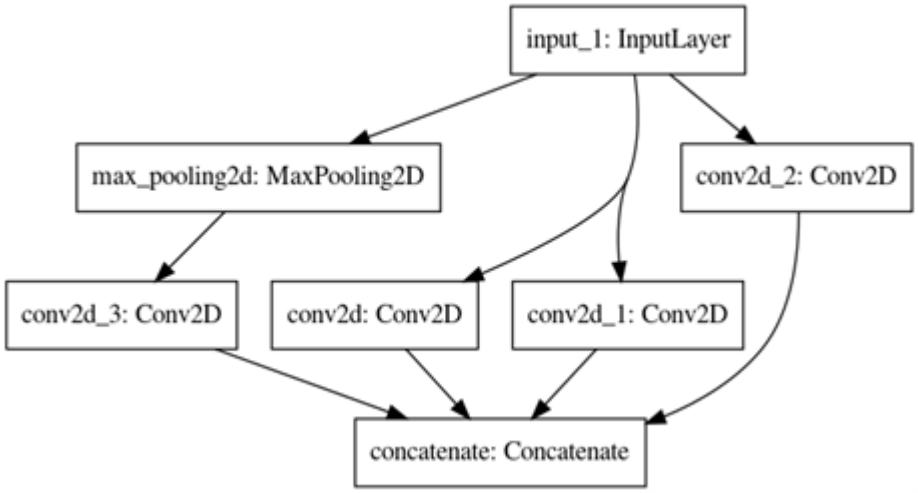
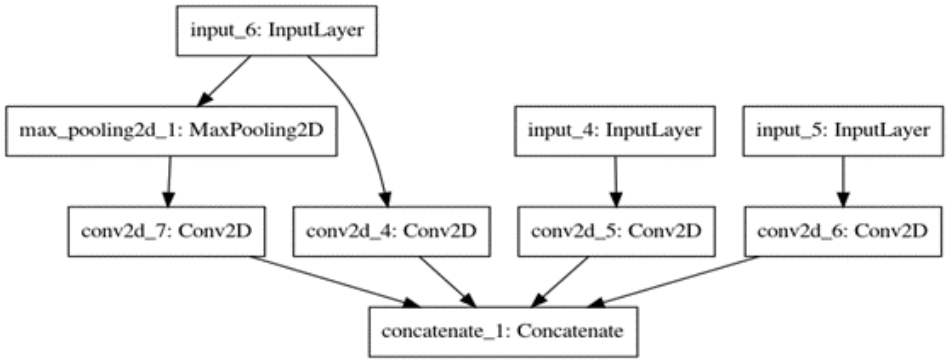


Figure 2.7: Example of Layer_2_INCEPT Model is given, Likewise the Layer_3_INCEPT, and Layer_4_INCEPT models also have same architecture.

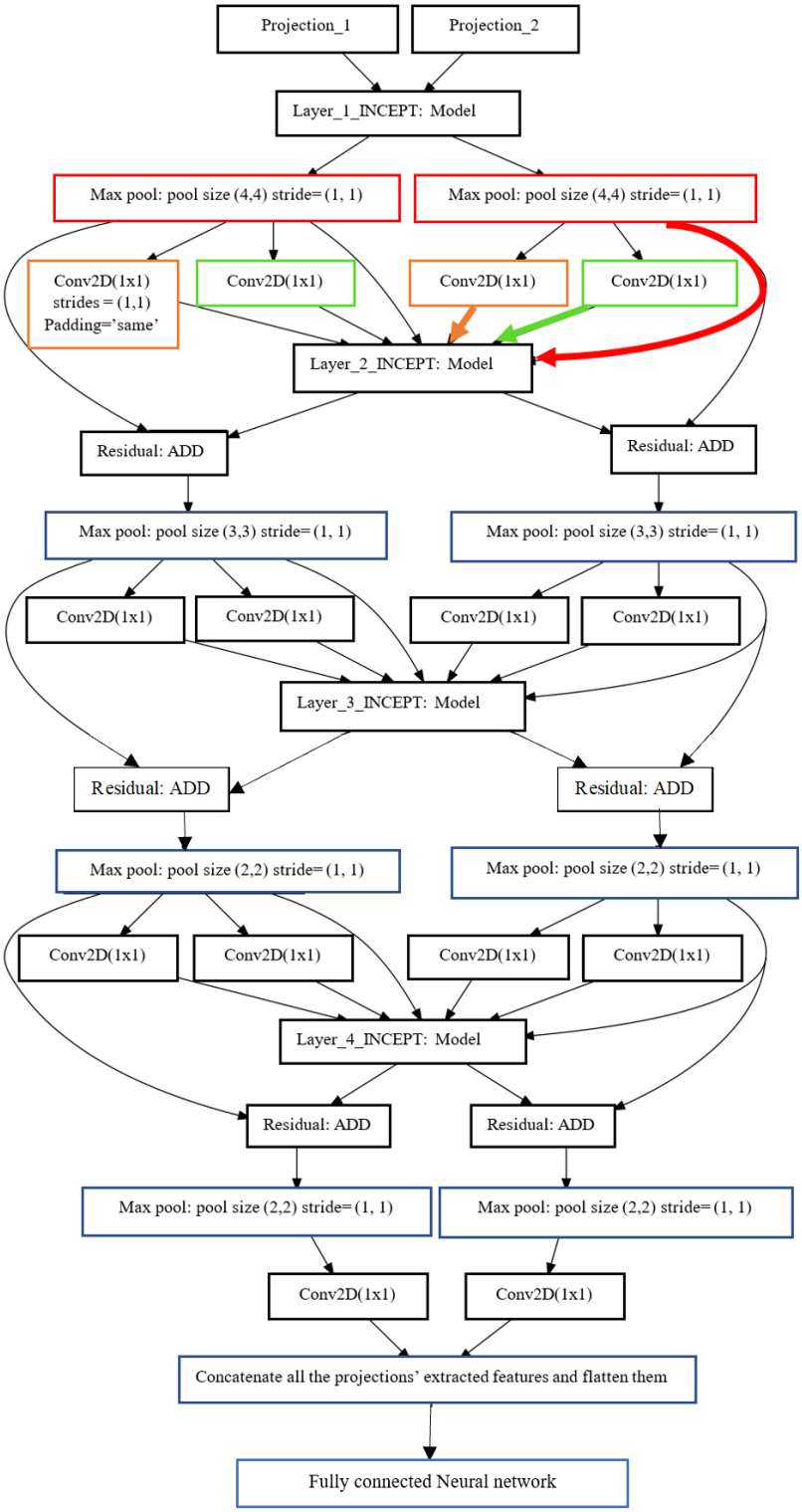


with different parameters in layers 3 and 4). To check the base line performance and parameter selection same depth naive DCNNs (without inception and residual) are experimented with different CNN kernels.

From the controlled experiments the model presented in Fig. 2.5, has highest performance; the model must extract the feature from 24 projections. Each projection has feature maps; likewise, 24 projections are fed as shown in Fig. 2.5 (21-channel information is obtained by combining, *SITE* information, and biochemical properties of amino acids as mentioned in *Appendix* E ; and tactics 2 of section 2.1 introduction elaborates about 24 projections). As shown in Fig. 2.5, the DCNN receives 24 projections' of feature maps in parallel and performs a multi-class (ONGO Vs TSG Vs Fusion) classification.

The model is named as "Brain-inception-residual" (BIR); since the architecture is almost (In order to mimics brain functionality, increase deepness/ number of layers then fully connected layers can be added in the deeper layers as mentioned in [31] and predictions in each level can be evaluated; further inception modules like [30] can be swap with the current inception modules to reduce the parameters)

Figure 2.8: Brain inception residual overall architecture with 2-projections.

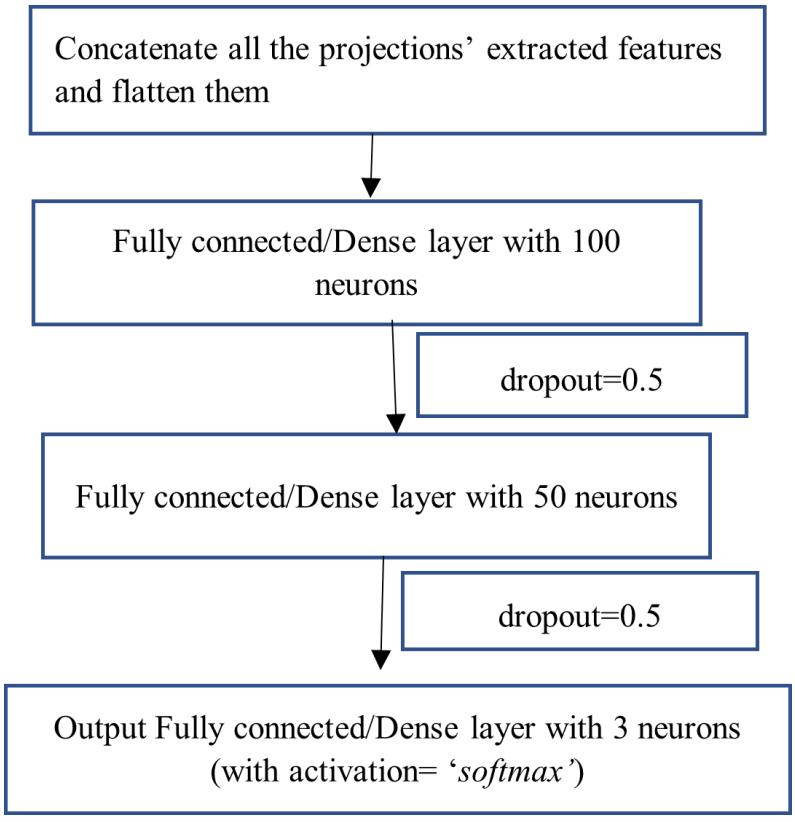


following the brain architecture (because the architecture reuses the same inception feature extractor in different projections; while use different dimensionality reduction networks in each projections to find, if their relative positional information).

2.8.1 Parameter selection

In order to illustrate the model's overall architecture, only two projections' feature extraction is shown in Fig. 2.8 (likewise rest of the 22 projections are going to be followed/ fed in visual feature extraction, and then extracted visual features are concatenated, and flatten before feed in to the Fully connected NN). The model's visual feature extraction inception layers' modules are shown in Fig. 2.6, and Fig. 2.7. The final fully connected architecture is shown in Fig. 2.9; these experiments used 50 % dropout in the fully connected layers to control probably overfitting.

Figure 2.9: Fully connected Neural network of Brain inception residual.



The parameters assigned for these modules (of the brain inception residual model) are mentioned in Table. 2.15. The Network section in Table. 2.15 implies parameters applicable network modules.

Inception modules of layer 1 and layers 2-4 are shown in Fig. 2.6 and Fig. 2.7, respectively. The number of generated feature maps of inception module's CNN with kernel, is d_CNN_inc_i (therefore d_CNN_inc_1, d_CNN_inc_3, and d_CNN_inc_5 are inception modules' CNN kernels with respectively). Each inception module's total generated feature map is 256 (d_CNN_inc_1 + d_CNN_inc_3 + d_CNN_inc_5 + d_CNN_1_max). These 256-feature maps are added with the previous layers' Max-

Table 2.15: The brain inception residual DCNN's parameters

Network	Layers	Parameters	Values	Activation
Inception	1-4	d_CNN_inc_1	128	*Swish*
	1-4	d_CNN_inc_3	64	*Swish*
	1-4	d_CNN_inc_5	32	*Swish*
	1-4	Max Pool	3×3	N/A
	1-4	d_CNN_1_max	32	*ReLu*
layer reducer	$1-\triangleright 2$	Max Pool (and stride)	4×4	N/A
	$2-\triangleright 3$	Max Pool (and stride)	3×3	N/A
	$3-\triangleright 4$	Max Pool (and stride)	2×2	N/A
	$4-\triangleright$feature	Max Pool (and stride)	2×2	N/A
	1,2	d_lay_1_to_2	32	*Swish*
	3,4	d_lay_3_to_4	64	*Swish*
Dense	5	Hidden Neurons	100	*Swish*
	6	Hidden Neurons	50	*Swish*
Final layer (7)		Neurons	3	*Softmax*

[a] N/A means Not applicable for that parameter

Pool's 256-feature maps as shown in Fig. 2.8 (the blocks named as "Residual: ADD"). Since the 256-feature maps is expensive to feed in each layers continuously, the layer reducers extract the most wanted feature maps for such as d_lay_1_to_2 (for layer 1 and 2 to end up with 32 features maps) and d_lay_3_to_4(for layer 3 and 4 end up with 64 features maps); and Max pooling in the layer reducers uses the same pool size for their strides(**E.g.:** in "layer 1 to 2 reducer" or "$1-\triangleright 2$" uses Max pool with 4×4 filter and 4×4 stride) with valid padding. All the CNNs' and rest of the Max Pools' (excluding layer reducers) have same padding, and their stride size is 1×1.

2.8.2 Calculation of number of parameters

The inception residual convolution/ pooling layers extract $24 \times 2 \times 2 \times 64 = 6144$ visual features. The model has $2,925,751$ trainable parameters. And the parameter calculation is shown in Table. 2.16.

For Approach 2's total number of train data's feature ($Number_{features}$)

$$\begin{aligned} Number_{features} &= Number\ of\ PDBs \times projections \times \\ &\quad size_x \times size_y \times channels(feature) \\ &= 1628 \times 24 \times 128 \times 128 \times 21 \\ &\approx 1.34 \times 10^{10} \end{aligned}$$

Ratio of $Input features \div Trainable parameter$ is shown below as

$$\begin{aligned} Ratio_{feturetoparam} &\approx \frac{(1.34 \times 10^{10})}{2,925,751} \\ &\approx 4580 \end{aligned}$$

likewise, Approach 1's total number of train data's feature is 1.56×10^{10} and $Ratio_{feturetoparam}$ is $\approx 5,332$.

Table 2.16: The brain inception residual DCNN's number of trainable parameter calculation

Layer wise CNN		1-D CNN reducer	Single projection feature extractor (with CNN)	Applying same (1)/ parallel		Fully connected layer parameters calculator					Total parameters
				Number of NN	Total feature extractor param	Flatten		layer_1	layer_2	Softmax layer	Total fully connected parameters	
Layer_1			32,512	1	32,512	6,144	neurons	100	50	153	619,703	2,925,751
		8,224		48	394,752		Weights	614,500	5,050			
Layer_2			85,248	1	85,248							
		8,224		48	394,752							
Layer_3			85,248	1	85,248							
		16,448		48	789,504							
Layer_4			129,280	1	129,280							
		16,448		24	394,752							

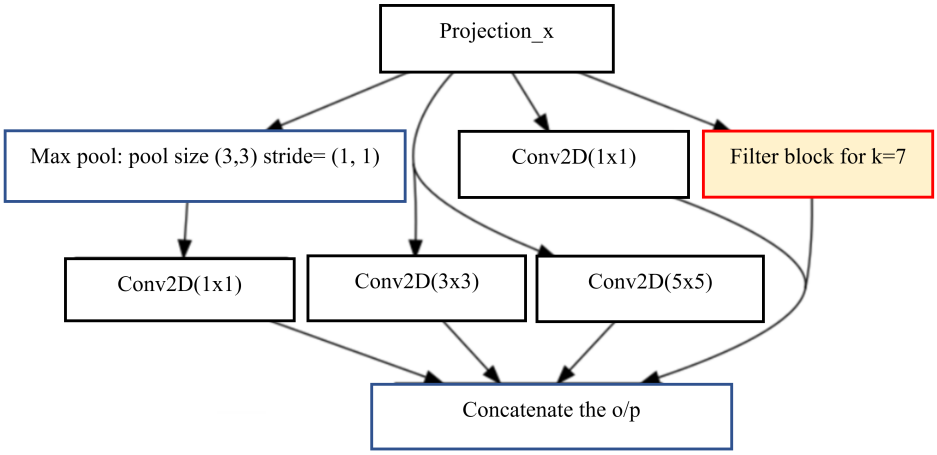
[a] Applying same (1)/ parallel feature extractors', Parallel: different models for each data given
[b] The same number of parameters is brain inception without residual as well
[c] Proof of parameter calculations
https://drive.google.com/open?id=1yCiy7a6UGdoriNw6mtXf-CXnbDV8doiO,
https://drive.google.com/open?id=15AD41y8xLiTpUiMesWvrBPFMeItlySM,
https://drive.google.com/open?id=1W1XkSRLdw-iOCwcUDabwyZpXejVwCyM,
https://drive.google.com/open?id=1vzBLUUukkr5_ObDFonAYGPvzl11Y3SSwO,

2.9 Different kernel architectures of BIR (while include the conv kernel 7)

Plenty of ways to build the kernel architecture as discussed in [31], [30] **etc.**. The inception networks of the architecture can be extended with kernel 7. There are plenty of ways to place this 7×7 kernel. Here, experimented with only three (these three captures almost different ways of architecture design of convolution with k=7) of them.

1. ***BIR_7:*** One is just using 7×7 kernel followed by 3×3 and 5×5 the structure. (to capture the general old way)
2. ***BIR***_7×1***:***To use 7×7 kernel; if those replaced by 7×1 and 1×7 kernels like ($n \times 1$ and $1 \times n$) as discussed in [30]. (to capture the 7×1 followed by 1×7 convolution replace the 7×7 filter only using 14 weights instead of 49 weights)
3. ***BIR***_7×1_3×3***:*** Use 7×1 and 1×7 kernels and followed by 3×3 kernel give weightage. (to capture the 7×1 followed by 1×7 convolution and followed by 3×3 filter to extract important features after extracted).

Figure 2.10: Layer_1_INCEPT: Model with Filetr block k=7.



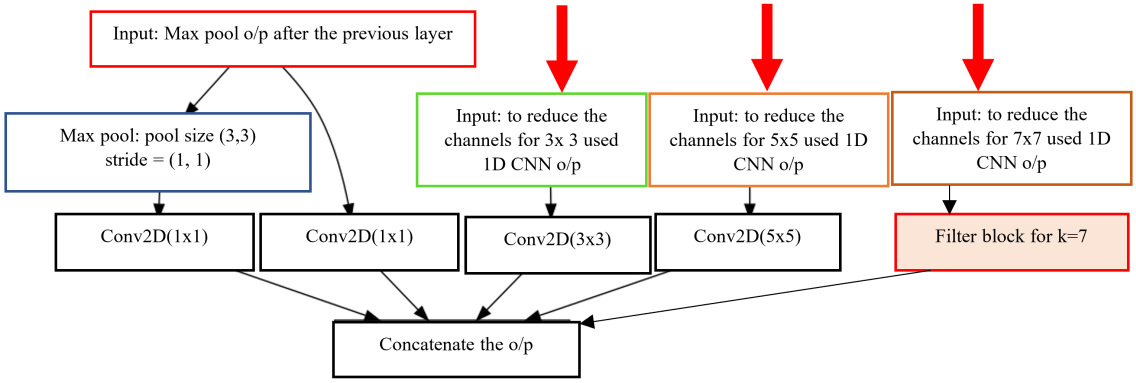
To show the architecture of them; general architecture frames as shown in Fig. 2.10, and Fig. 2.11a are used. In those architecture only the "Filter block for k=7" is changed in each of these networks. For an example the first architecture(Model-1_of_k_7) uses only cov2D(7x7) kernel so if it is placed in the 2.11a then the architecture would be look like as shown in 2.11b. Likewise, the "Filter block for k=7" of Model-2_of_k_7 and Model-3_of_k_7 is shown in Fig. 2.11c and Fig. 2.11d respectively.

2.10 Level 2 results overview

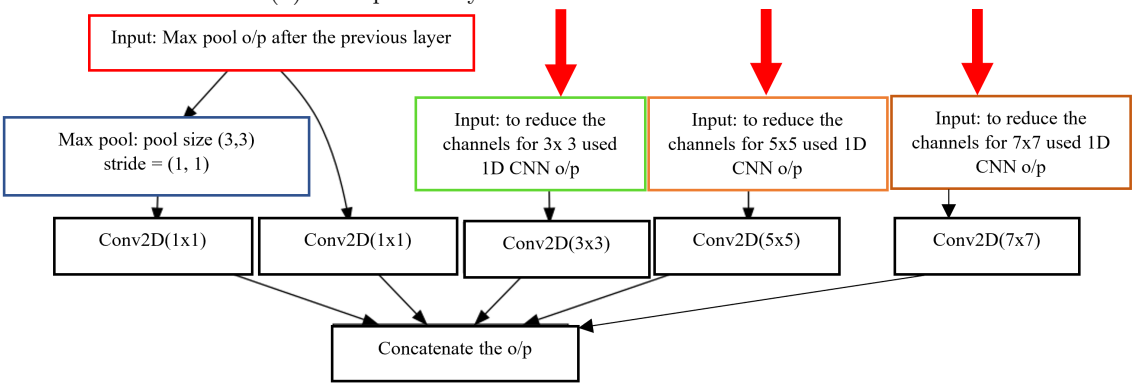
All these results are obtained from Tier 1 (confirmed by experimental, the genes data credit goes to COSMIC [28]) genes belongs to only ONGO or TSG or Fusion. Further these classification's diagnostic ability is verified by Receiver operating characteristic (ROCs) and their corresponding area under the curve (AUROC). Where, macro averaging means give all classes the same weightage.

Figure 2.11: Examples of Layer_2_INCEPT Model is given, Likewise the Layer_3_INCEPT, and Layer_4_INCEPT Models also have same architecture

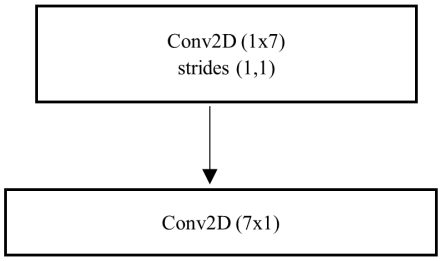
(a) Example of Layer_2_INCEPT Model with Filter block for k=7



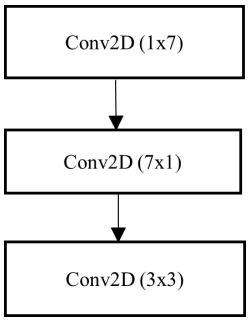
(b) Example of Layer_2_INCEPT Model with raw CNN



(c) Filter block for k=7 for BIR_7x1.



(d) Filter block for k=7 for BIR_7x1_3x3



2.11 Approach 1 results(10-fold cross validation with mixed primary structural PDBs)

2.11.1 Approach 1's functional classification of PDB_{SITE}s (isoforms) results

The PDB_{SITE} wise classification of Approach 1's confusion matrix is shown in Fig. 2.12. Functional 10-fold cross validation accuracies of the following classes ONGO, TSG, Fusion and all combined, are 94.67 %, 94.33 %, 88.68 % and 93.5 % respectively.

Table 2.17: Approach 1's(10-fold cross validation's)Area under the curves for Functional classification of PDB_{SITE}s'(using DCNN model) and gene's Primary structure (using DCNN and Ensamble+direct methods)

AUROCs of 10-fold Classif-ication			Functional classification of PDB_{SITE}s'					Functional classification of gene's Primary structure (using Ensemble + direct)				
ONGO	TSG	Fusion	ONGO	TSG	Fusion	Micro average	Macro average	ONGO	TSG	Fusion	Micro average	Macro average
✓	✓	✓	0.974	0.976	0.961	0.977	0.971	0.98	0.948	0.832	0.941	0.923
✓	✓	✗	0.978	0.978	N/A	0.979	0.978	0.988	0.988	N/A	0.988	0.99
✓	✗	✓	0.975	N/A	0.975	0.986	0.975	0.92	N/A	0.92	0.925	0.927
✗	✓	✓	N/A	0.981	0.981	0.986	0.981	N/A	0.861	0.861	0.911	0.877

Figure 2.12: Confusion matrix of Approach 1 classification of PDBs in 10-fold crossvalidation on

(a) ONGO Vs TSG Vs Fusion

valid	ONGO	TSG	Fusion
ONGO	976	32	22
TSG	45	666	20
Fusion	10	8	329

(b) ONGO Vs TSG.

valid	Given	
	ONGO	TSG
ONGO	985	35
TSG	46	671

(c) ONGO Vs Fusion.

valid	ONGO	Fusion
ONGO	1003	35
Fusion	28	336

(d) TSG Vs Fusion.

valid	TSG	Fusion
TSG	683	38
Fusion	23	333

Only ONGO vs TSG is calculated by normalizing the ONGO and TSG probabilities while neglecting the Fusion class (Note, if one PDB_{SITE} classified as Fusion class only then corresponding PDB_{SITE}'s probability is given to ONGO and TSG as 0.5 to each); which is shown in Fig. 2.12b. Likewise, binary classification among ONGO Vs Fusion and TSG Vs Fusion also performed; and shown in Fig. 2.12c and Fig. 2.12d, respectively. Where, over all accuracies of ONGO vs TSG: 93.64 %, ONGO vs Fusion: 94.07 %, and TSG vs Fusion: 94.41 % respectively.

ROCs of the classification of PDB_{SITE}s are shown in Fig. 2.13, with their corresponding AUROC. Further, the summery of the AUROCs is mentioned in Table. 2.17; where all the AUROCs are above 0.96 and, binary classification of ONGO Vs TSG's AUROC is .978.

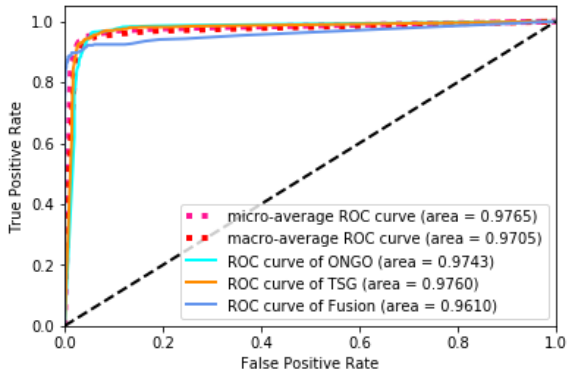
2.11.2 Approach 1's functional classification of gene's primary structure (using ensemble + direct)

Genes' primary structure wise functional classification (using an ensemble method and direct) of Approach 1's confusion matrix is shown in Fig. 2.14. ONGO, TSG, Fusion and all combined 10-fold cross validation accuracies obtained from functional classification of primary structures'(Genes'), are 95.56 %, 93.75 %, 65.52 %, and 88.41% respectively.

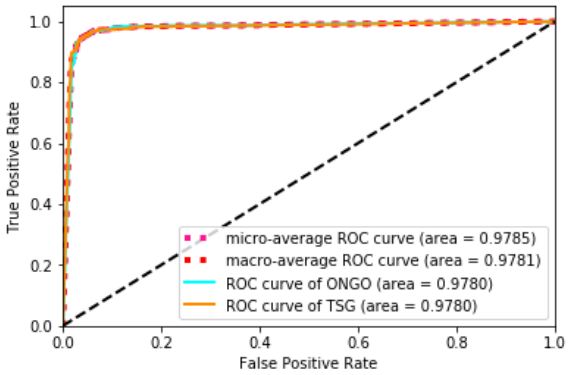
Only ONGO vs TSG is calculated by normalizing the ONGO and TSG probabilities while neglecting

Figure 2.13: ROC curve of Approach 1(10-fold) classification of fully cleaned PDB_{SITE}

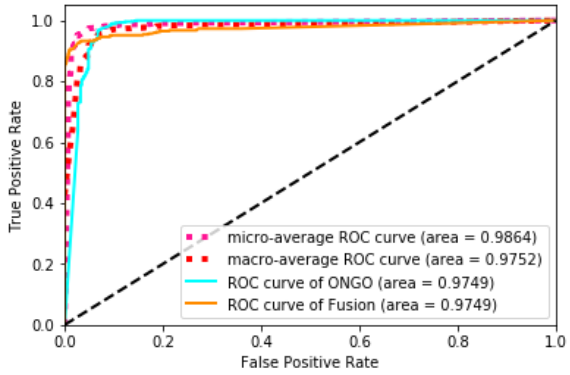
(a) ONGO Vs TSG Vs Fusion .



(b) ONGO Vs TSG only, here the Fusion probability is omitted



(c) ONGO Vs Fusion only, here the TSG probability is omitted.



(d) TSG Vs Fusion only, here the ONGO probability is omitted

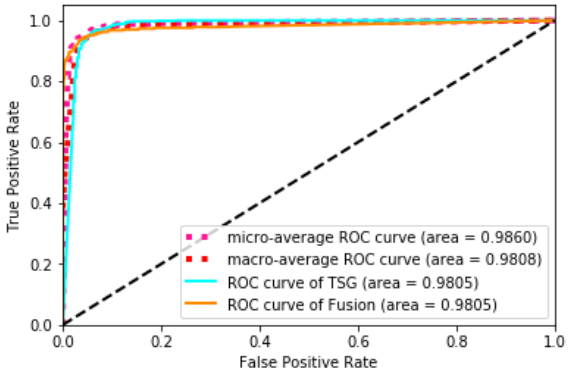


Figure 2.14: Confusion matrix of Approach 1 classification of primary structures with 10-fold cross-validation and methods(from Ensemble and Direct) on

(a) ONGO Vs TSG Vs Fusion

valid	ONGO	TSG	Fusion
ONGO	43	3	3
TSG	2	60	7
Fusion	0	1	19

(b) ONGO Vs TSG

valid	Given	
	ONGO	TSG
ONGO	43	4
TSG	2	60

(c) ONGO Vs Fusion

valid	ONGO	Fusion
ONGO	45	9
Fusion	0	20

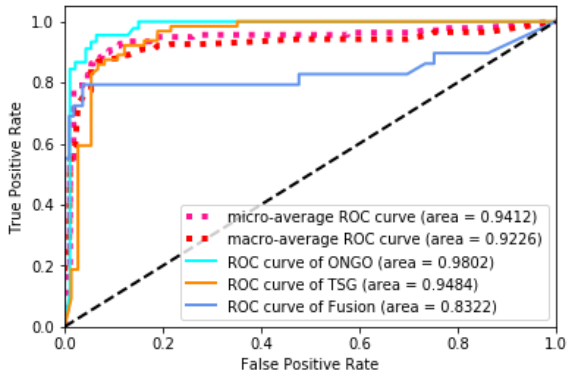
(d) TSG Vs Fusion

valid	TSG	Fusion
TSG	63	9
Fusion	1	20

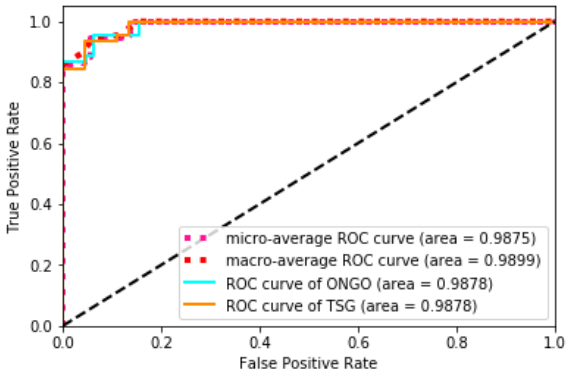
the Fusion class (same probability calculation as mentioned previous section). Binary classifications' confusion matrices are shown in Fig. 2.14 (exclude Fig. 2.14a). Where, binary classifiers' over all accuracies are acrshortONGO vs TSG: 94.5 % acrshortONGO vs Fusion: 87.8 % and TSG vs Fusion: 89.24 % respectively.

Figure 2.15: 10-fold cross validation primary structure of gene's functional classification (from Ensemble and Direct) on

(a) ONGO Vs TSG Vs Fusion .



(b) ONGO Vs TSG only, here the Fusion probability is omitted



ROCs of the Genes' primary structure wise functional classification are shown Fig. 2.15, with their corresponding AUROC. Further, the summery of the AUROCs is mentioned in Table. 2.17; where, only ONGO Vs TSG classifiers' AUROC is .988.

2.12 Approach 2 results (the primary structural genes with their corresponding PDBs are separated as Train and test sets depending on the genes' primary structural information)

2.12.1 Approach 2's functional classification of PDBs (Isoforms) results

The PDB_{SITE} wise classification of Approach 2's Test set confusion matrix is shown in Fig. 2.16. From that none of the test set of PDB_{SITE}s' functionality are distinguished by the DCNN (classifier of Approach 2) as Fusion class.

Figure 2.16: Confusion matrix of Approach 2 classification on ONGO Vs TSG Vs Fusion of the

(a) training set.

Train	ONGO	TSG	Fusion
ONGO	486	9	110
TSG	15	535	77
Fusion	0	0	140

(b) validating set

valid	ONGO	TSG	Fusion
ONGO	83	1	2
TSG	5	87	12
Fusion	0	0	12

(c) test set.

Test	Given		
	ONGO	TSG	Fusion
ONGO	326	20	6
TSG	119	59	12
Fusion	0	0	0

But the DCNN (classifier of Approach 2) have some capability to distinguish PDB_{SITE} functional

classification between ONGO and TSG. Binary classification of ONGO vs TSG (following the same probability normalization as mentioned in subsection 2.11.1 "Approach 1's functional classification of PDBs (Isoforms) results") for the test set, and validation set got 73.47 %, and 96.59 % respectively. The accuracy of 73.47 % give a validity of informational gain in ONGO Vs TSG even primary structural gene wise separation presented in Train and Test set (some primary structures in Trian set and Test set has same kind of functionality). If the number of PDBs increased, then the classifiers may perform better in functionally classifying ONGO and TSG, even if they separated by primary structure wise.

However, as expected it has some capability of separate Fusion class in validation set (Binary classification of ONGO vs Fusion, and TSG vs Fusion got over all accuracy as 84.21 % and 87.72 % accordingly). The possible reason behind the lack of performance comparable to the Approach 1; is due to lack of number of PDBs to train the DCNN (since considerable amount of PDBs were allocated to test; such as 24 % primary structural genes were used in Test set, that contains 34 % of PDBs in whole filtered and cleaned PDB set) of Approach 2.

Classification of PDB_{SITE}s' ROCs are shown in Fig. 2.17, with their corresponding AUROC. Fig. 2.17a, Fig. 2.17c, and Fig. 2.17e represents ONGO Vs TSG Vs Fusion, for Training, Validation and Test sets accordingly. Fig. 2.17b, Fig. 2.17d, and Fig. 2.17f represents binary classification of ONGO Vs TSG, for Training, Validation and Test sets accordingly.

2.12.2 Approach 2's functional classification of gene's primary structure (using ensemble + direct)

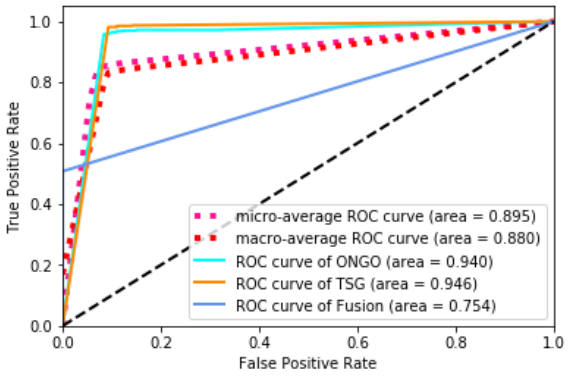
Since none of the PDB_{SITE}s of test set are classified as Fusion class by the DCNN(as shown in Fig.2.16c), the methods not going to predict the Fusion class in Test set at all. But the ONGO Vs TSG performance can be evaluted for the primary structural Function prediction as shown in Fig. 2.18a (since no difference in confusion matrixes of binary classification; the 3-class confusion matrixes are only presented). Further, Fig. 2.18b shows the Training sets primary structural Function predictions; here the combined Train and Validation PDB_{SITE}s classified functional details are used.

The ONGO vs TSG vs Fusion primary structural function prediction of Test sets' ROCs are shown in 2.19a; further binary classification of ONGO vs TSG is shown in Fig. 2.19b.

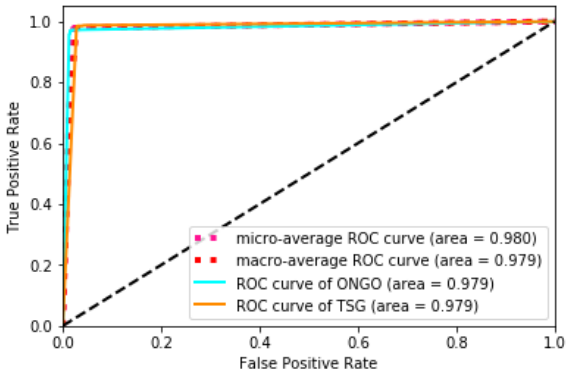
From confusion matrix of Test set(Fig. 2.16c), the capability of distinguish between classes is very low, the classifier is almost biased to TSG, due to the lack of PDBs in TSG compare to ONGO, in training data creation higher priority is given to TSG(if not, it does not classify TSG at all; due to biased of higher number of ONGO PDB_{SITE}s; $\approx$ 28% primary structures of ONGO placed in Test set). If the number of the primary structures of TSG incresed, then the classifier may perform better. Even in the training set's Fusion class classififction is very low, but the Fusion class classification performance is better in Approach 1(10-fold crossvalidation); this validates, if more PDB_{SITE}s are there then the Approach 2 can applicable to check the performance. As per the statistics from RCSB PDB databank[`http://www.rcsb.org/stats/growth/xray` last accessed on 10-25-2019 at 8.47p.m] in Future Approach 2 may applicable.

Figure 2.17: Approach 2's ROC curve of fully cleaned PDB_{SITE}s' of

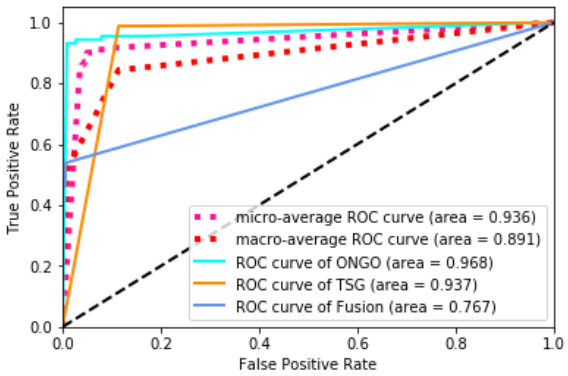
(a) training set on ONGO Vs TSG Vs Fusion.



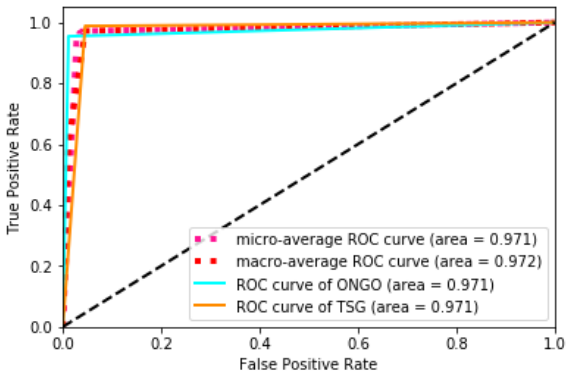
(b) training set on ONGO Vs TSG.



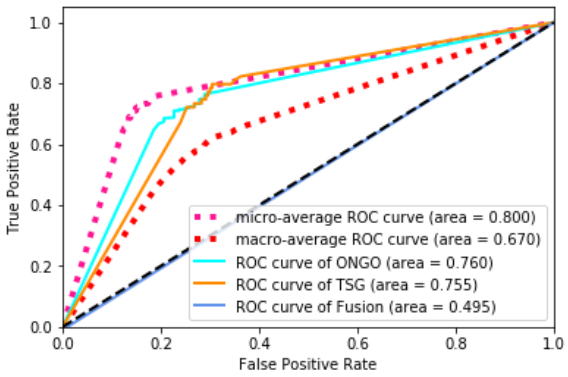
(c) validation set on ONGO Vs TSG Vs Fusion.



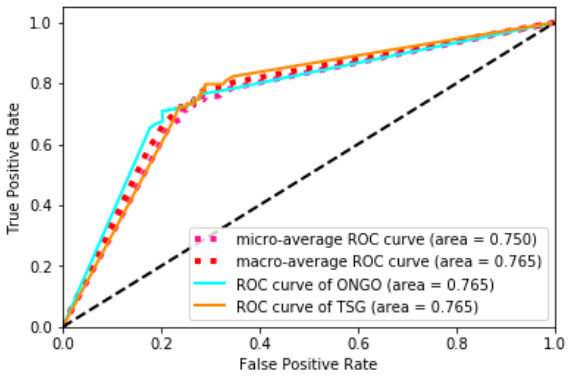
(d) validation set on ONGO Vs TSG.



(e) test set on ONGO Vs TSG Vs Fusion.



(f) test set on ONGO Vs TSG.



2.13 Comparison with the available state-of-the-art methods for ONGO Vs TSG identification

Since already developed methods [13] and [33] were focused on classifying ONGO Vs TSG or ONGO Vs TSG Vs Unknown; In all these methods, Fusion class is not considered at all . Thus, Table. 2.18 compares the AUROC values reported in this study with the AUROC values reported by the

Figure 2.18: Confusion matrix of Approach 2 classification of primary structures with DCNN(obtained by train set) and methods(from Ensemble and Direct) on

(a) Test set.

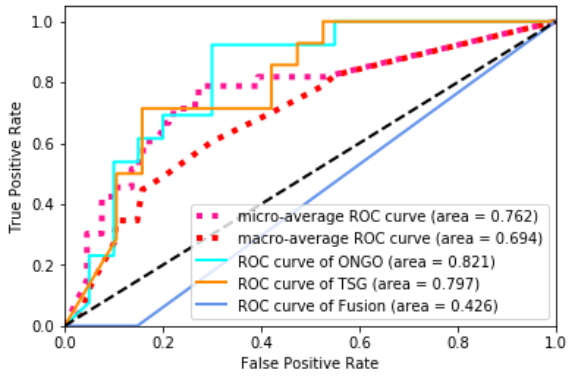
Test	Given		
	ONGO	TSG	Fusion
ONGO	7	1	2
TSG	6	13	4
Fusion	0	0	0

(b) Training set.

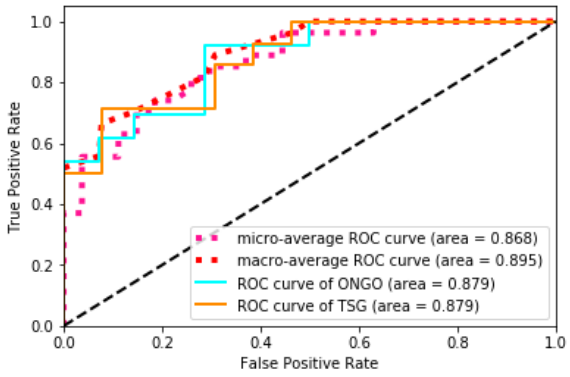
Train	Given		
	ONGO	TSG	Fusion
ONGO	31	0	7
TSG	1	50	13
Fusion	0	0	3

Figure 2.19: Approach 2's ROC curve of primary structure of gene's functional classification test set of

(a) ONGO Vs TSG Vs Fusion.



(b) ONGO Vs TSG



state-of-art statistical methods [13], [32], [10] for ONGO vs TSG identification, and the [33].

From Table. 2.18, Level 2's DCNN (BIR model) applied to the PDB structure, performance in this study is far better than Level 1 [33]; because this study only considering the PDB structure's SITE information. The Approach 1's outperforms (since minimum AUROC of Approach 1 is 0.94) all (maximum AUROC among other is 0.924) the other state-of-art statistical methods' ONGO Vs TSG identification (reported in [13] and including [13]), and the Level 1 [33]. Even the Approach 2's lowest AUROC is 0.754; because the trainset test set split based on primary structure reduce the number of PDBs to train as well as test; this would be overcome in the future by growing PDB deposits.

Table 2.18: AUROC of ONGO Vs TSG identification using the statistical methods reported by [13], and our models from Level 1 [33], and Level 2 [18].

Method			Description	AUROC
Truncation Rate			of truncating events*	0.922
Unaffected Residues			Intra-gene mutation clustering/ recurrence*	0.479
VEST Mean			Functional impact bias*	0.710
Patient Distribution			Bias in patient labels*	0.556
Cancer Type Distribution			Cancer type bias*	0.612
Oncodrive-fm			Gonzalez-Perez and Lopez-Bigas [10]	0.725
OncodriveCLUST			Tamborero et al.,[32]	0.597
Random Forest			Ensemble on the first 5 methods (*)	0.924
Level 1 PDB vise functional classification			DCNN applied to the PDB structure	0.887
Level 2 PDB_{SITE} vise functional classification			*Approach* 1_{DCNN}	0.978
			Approach 2_{DCNN}	0.765
Proposed primary structure vise Functional classification of genes	Dynamic progr--amming methods	$Method_1$**	*Approach* 1_{DCNN}	0.997
			Approach 2_{DCNN}	0.871
		$Method_2$**	*Approach* 1_{DCNN}	0.94
			Approach 2_{DCNN}	0.754
		$Method_3$**	*Approach* 1_{DCNN}	0.979
			Approach 2_{DCNN}	0.761
		Ensemble on the last 3 methods (**)	*Approach* 1_{DCNN}	0.99
			Approach 2_{DCNN}	0.833
	(using Ensemble + direct)		*Approach* 1_{DCNN}	0.989
			Approach 2_{DCNN}	0.879

[a] *Approach* 1_{DCNN} means using the PDB_{SITE} probabilities obtained using the Level 2's Approach 1 DCNN/ BIR model's classification(for fare comparison, PDB_{SITE} fell in the validation set only taken) are used by the methods to functionally classify the primary structures.

[b] *Approach* 2_{DCNN} means using the PDB_{SITE} probabilities obtained using the Level 2's Approach 2 DCNN/ BIR model's(trained by Approach 2's training set) classification(of test set) are used by the methods to functionally classify the Test sets' primary structures.

Bibliography

[1] Atomic weights and isotopic compositions for all elements.

[2] contour length in polymers. In Miloslav Nič, Jiří Jirát, Bedřich Košata, Aubrey Jenkins, and Alan McNaught, editors, *IUPAC Compendium of Chemical Terminology*. IUPAC, 2.1.0 edition.

[3] Proteinogenic amino acid.

[4] Pamela C. Champe , Richard A. Harvey, and Denise R. Ferrier. *Chapter 20: Amino Acid Degradation and Synthesis*. Number 3.

[5] Sri Rama Koti Ainavarapu, Jasna Brujić, Hector H. Huang, Arun P. Wiita, Hui Lu, Lewyn Li, Kirstin A. Walther, Mariano Carrion-Vazquez, Hongbin Li, and Julio M. Fernandez. Contour length and refolding rate of a small protein controlled by engineered disulfide bonds. 92(1):225–233.

[6] Alexandre Ambrogelly, Sotiria Palioura, and Dieter Söll. Natural expansion of the genetic code. 3(1):29–35. Place: United States.

[7] Byung Jin Byun and Young Kee Kang. Conformational preferences and pK(a) value of selenocysteine residue. 95(5):345–353. Place: United States.

[8] Albert Erives. A model of proto-anti-codon RNA enzymes requiring l-amino acid homochirality. 73(1):10–22.

[9] Tom Goddard. Surface algorithms.

[10] Abel Gonzalez-Perez and Nuria Lopez-Bigas. Functional impact bias reveals cancer drivers. 40(21):e169–e169. Edition: 2012/08/16 Publisher: Oxford University Press.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition.

[12] Lukasz P. Kozlowski. Proteome-pI: proteome isoelectric point database. 45:D1112–D1116.

[13] Runjun D. Kumar, Adam C. Searleman, S. Joshua Swamidass, Obi L. Griffith, and Ron Bose. Statistically identifying tumor suppressors and oncogenes from pan-cancer genome-sequencing data. 31(22):3561–3568.

[14] J. Kyte and R. F. Doolittle. A simple method for displaying the hydropathic character of a protein. 157(1):105–132. Place: England.

[15] Lehninger, Albert L., Nelson, David L., Cox, and Michael M. *Lehninger Principles of Biochemistry*. 3 edition.

[16] Alexey V. Lobanov, Anton A. Turanov, Dolph L. Hatfield, and Vadim N. Gladyshev. Dual functions of codons in the genetic code. 45(4):257–265.

[17] Uwe Meierhenrich. *Amino acids and the asymmetry of life : caught in the act of formation.* Springer.

[18] Anandanadarajah Nishanth, Chee Hung Chu, and Rasiah Loganantharaj. An integrated deep learning and dynamic programming method for predicting tumor suppressor genes, oncogenes, and fusion from pdb structures. Under review with Computers in Biology and Medicine; submitted on 17-Oct-2020 and manuscript number is CIBM-D-20-02914.

[19] C. Nick Pace, Gerald R. Grimsley, and J. Martin Scholtz. Protein ionizable groups: pK values and their contribution to protein stability and solubility. 284(20):13285–13289.

[20] Rob Phillips. *Physical biology of the cell.* Garland Science, second edition edition.

[21] pinimg.com. xyz quadrants.

[22] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions.

[23] Rcsb.org. Rcsb protein data bank.

[24] Victor Rodwell. *Harper's illustrated biochemistry.* Mcgraw-Hill Education.

[25] Michael Rother and Joseph A. Krzycki. Selenocysteine, pyrrolysine, and the unique energy metabolism of methanogenic archaea. 2010.

[26] Michel Sanner. The reduced surface computation has been implemented in a program called MSMS which has been presented at the 11th symposium on computational geometry held in vancouver BC canada. the abstract of that communication is available online as well as a PostScript version of the full communication. page 2.

[27] Rachel L. Schmidt and Miljan Simonović. Synthesis and decoding of selenocysteine and human health. 53(6):535–550.

[28] Zbyslaw Sondka, Sally Bamford, Charlotte G. Cole, Sari A. Ward, Ian Dunham, and Simon A. Forbes. The COSMIC cancer gene census: describing genetic dysfunction across all human cancers. 18(11):696–705.

[29] Study.com. xyz quadrants projections.

[30] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826. IEEE.

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9. IEEE.

[32] David Tamborero, Abel Gonzalez-Perez, and Nuria Lopez-Bigas. OncodriveCLUST: exploiting the positional clustering of somatic mutations to identify cancer genes. 29(18):2238–2244.

[33] A. Tavanaei, N. Anandanadarajah, A. Maida, and R. Loganantharaj. A deep learning model for predicting tumor suppressor genes and oncogenes from pdb structure. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 613–617, 2017.

[34] The UniProt Consortium. UniProt: a worldwide hub of protein knowledge. 47:D506–D515.

[35] Richard L. Thurlkill, Gerald R. Grimsley, J. Martin Scholtz, and C. Nick Pace. pK values of the ionizable groups of proteins. 15(5):1214–1218.

[36] UCSF Computer Graphics Laboratory. Introduction to protein data bank format.

[37] wwPDB. Protein data bank contents guide: Atomic coordinate entry format description.

[38] V. R. Young. Adult amino acid requirements: the case for a major revision in current recommendations. 124(8):1517S–1523S. Place: United States.

Appendix A

Sequence length Threshold Selection

Already filtered PDBs (PDBs obtained by $X - Ray$ diffraction) were only considered here. As mentioned in section 1.2 "Data Set Preprocessing" (the mapped data from UniProt [34], has Gene_Symbols with PDB_{id}s and their corresponding, start-end overlap sequence with the gene). From that, overlapped/ aligned sequence length for each PDB with their genes details was gained by, negating the starting position of the PDB's aligned primary structural position (sequence starting position number) from ending position of the PDB's aligned primary structural position (sequence ending position number) for corresponding gene's primary structure. Since some of the PDBs' overlapping/ aligning sequence length is too small, thus the contribution to the Gene from the isoform is less. To extract useful PDBs, the threshold length should be obtained without losing too many PDBs, and reduce the noise created by small sequence length PDBs. If the PDBs have more than one sequence length, then those PDBs are left in the whole experiment as mentioned in Chapter 1.2 "Limitations and assumptions".

To select the overlapping sequence threshold length, all the dataset (ONGO, TSG, and Fusion) with the sequence length were pooled together and combinedly considered. Table. A.1 shows, how many PDBs fell between the overlapping sequence lengths (Bin). From the Table. A.1, the threshold 81 chosen, because this make sure more than 82 % PDB ids remained for further processing. Thus, it reduces the noise created by the PDB_{id}s which has overlap sequence length < 81s.

Table A.1: Number of PDB_{id}s with the sequence overlapping length

Bin	Frequency	cumulative frequency	Fraction covered by cumulative
1	0	0	0.00
21	494	0	0.00
41	66	560	15.52
61	26	586	16.24
81*	34	620	17.18
162	1225	1845	51.12
243	622	2467	68.36
324	452	2919	80.88
405	405	3324	92.10
486	72	3396	94.10
567	60	3456	95.76
648	73	3529	97.78
729	8	3537	98.00
810	2	3539	98.06
891	0	3539	98.06
972	14	3553	98.45
1053	11	3564	98.75
1134	36	3600	99.75
1215	8	3608	99.97
More	1	3609	100.00

[a] Bin: represents the threshold sequence lengths for PDB_{id}s

[b] Frequency: Number of PDB_{id}s has sequence length between the threshold length of previous Bin and the current Bin.

[c] Cumulative frequency: Number of PDB_{id}s has sequence length up to the Bin given.

[d] Fraction covered by cumulative: Up to what percentage of all PDB_{id}s covered by the threshold length (given by the Bin).
E.g.: take the row "Bin" 81, which is highlighted* in the table above. The PDB_{id}'s overlapping sequence length (with the corresponding Gene_Symbol) has threshold up to 81. "Frequency" section has 34 PDB_{id}s, those PDB_{id}s (34) has overlapping sequence length between 61 <= sequence overlapping length < 81. "Cumulative frequency" section has 620 PDB_{id}s. Those PDB_{id}s' (620) overlapping sequence lengths were lower than 81 (sequence overlapping length < 81). "Fraction covered by cumulative" section has 17.18 % fraction of PDB_{id}s. Those PDB_{id}s' fraction (17.18 %) was obtained by cumulative frequency/ Sum of PDBs that means ($100 \times (620 \div 3609) = 17.18$ %)

Appendix B

Resolution factor Selection for Normalization

Since the PDBs (Isoforms) are gathered from different sources, the resolutions for each of them is also different. Hence, it must be normalized; however, the normalization reduces the positional information of the PDBs. Extracted coordinates of PDBs must be maintained at least some of the original positional information, thus it must be factored without much altering the surface aminoacid location. To find the factor, the resolution of each chosen PDBs were pooled as shown in Table. B.1 and Fig. B.1 (obtained from Table. B.1; to visualize the resolution distribution). The significant factor as "2.25" is chosen, because the mean of the distribution is 2.258, the standard deviation is 0.566 and the most frequent bins are 2,and 2.5, and the mean of the selected bins is 2.25 $((1.5 + 2 + 2.5 + 3) \div 4 = 2.25)$.

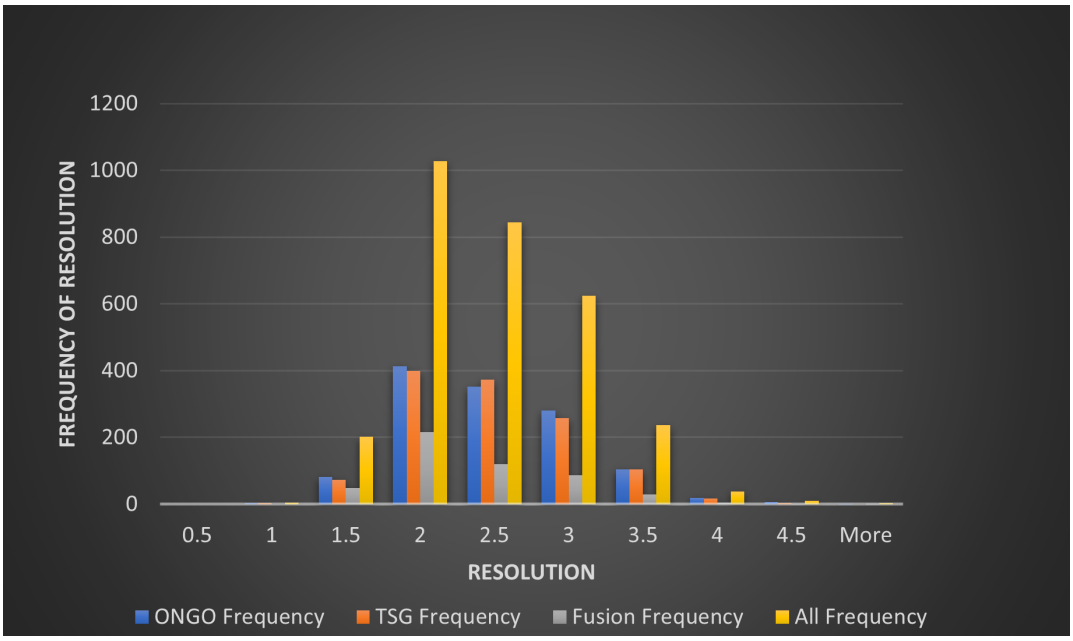
Table B.1: Frequency of resolution of PDB files for ONGO, TSG and Fusion of Tier 1

Resolution Bin	Frquency of				general bin frequency percentage	Cumulative
	ONGO	TSG	Fusion	All		
0.5	0	0	0	0	0.00%	0.00%
1	2	2	0	4	0.13%	0.13%
1.5	82	72	48	202	6.76%	6.89%
2	413	399	216	1028	34.39%	41.28%
2.5	351	373	120	844	28.24%	62.63%
3	281	257	87	625	20.91%	90.43%
3.5	104	104	29	237	7.93%	98.36%
4	18	16	4	38	1.27%	99.63%
4.5	6	2	1	9	0.30%	99.93%
More	1	0	1	2	0.07%	100.00%

[a] These bins represent the occurrence of resolution in PDBs,**E.g.:** Bin 2 contains the resolution of PDB files between the count of 1.5 and 2 (1.5 < resolution of PDB <= 2).

The normalization and factorization done, to make the PDBs with different resolutions (of Angstrom/ Å) to another template of scaled coordinates system. Such that these PDBs' surface representations are not much depend on the resolution of angstrom at all. The purpose of making these coordinate to

Figure B.1: Resolution of PDBs with their corresponding occurance(frequency)



another template, is to present all the PDBs in same scale of representation. Because the $C\alpha$ atoms are used to represent the surface aminoacid position, but the amino-acid molucule covers a average space(3.6Å) [5]. And the orientation it present makes the $C\alpha$'s position of aminoacid on the PDB structure also makes difference. Further, the purpose of the experiment is to show the surface amino acids in protein functionality. Lower resolution by general normalisation(multiplication by the resolution) may makes these aminoacids position further away (even though they present near). Thus, the coordinates are devided by their resolution, and then multiplied by 2.25 (this makes sure no positional information is loss due to this, kind of projection template). This coordinate template bridge the lower resolution PDBs' coordinates and higher dimentional PDBs in same space to represent their aminoacids. In this way final $< x, y, z >$ coordinates were found.

According to [5],

"the measured contour length per amino acid from five different methods $(4.0 \pm 0.2)\,\AA$ *is longer than the end-to-end length obtained from the crystal structure (3.6Å)"*

Keyword: contour length: according to [2],

"The maximum end-to-end distance of a linear polymer chain. For a single-strand polymer molecule, this usually means the end-to-end distance of the chain extended to the all-trans conformation. For chains with complex structure, only an approximate value of the contour length may be accessible. The sum of the lengths of all skeletal bonds of a single-strand polymer molecule is occasionally termed 'contour length'. This use of this term is discouraged."

Appendix C

Surface $C\alpha$ Indexing

The main piece of the experiment is surface atom detection. If proper surface atoms are not defined, then the inefficiency propagates through the deep learning model for predicting PDBs. As per the work [33] try to develop an algorithm that determines the surface $C\alpha$ atoms of the PDB. Instead of generating/ improving surface detection algorithm better to use the already well-known advanced tool (MSMS [26]). To define surface atoms, there are newly developed tools as shown in [9] available ,however MSMS is used widely. On the other hand, the main objective is, find the amino acids on the surface of given PDB-structure. So, very smaller details of the atoms present on the surface is not essential in the Level-2 experiments.

As mentioned in [9],

"MSMS allows computing very efficiently triangulations of Solvent Excluded Surfaces"

The MSMS-tool only calculates the solvent excluded surface of the PDB-structure, but it does not explicitly mention the surface $C\alpha$ atoms. Thus, to find the surface $C\alpha$ atoms, two main ways considered with the tool such as,

1. ***The depth of each*** $C\alpha$ ***atoms from the surface atoms:*** which is calculated by choosing the minimum distance between the surface and the $C\alpha$ atoms, and then define that distance as the $C\alpha$ atoms' depth from the surface.

2. ***The depth of each residue from the surface atoms:*** which is calculated by choosing the minimum distances between the surface and all atoms in the residue, and then average that distance as the residues' depth from the surface.

For the surface calculation, all the atoms in the PDB are considered by MSMS tool. Afterward, for the depth's calculation of the $C\alpha$ atoms, the PDBs' (Chain ID assignment's) ATOM record was only considered, thus the (Chain ID assignment's) $HETATM$ record was not included. Since, $HETATM$ records are non-polymer structures thus, it does not have amino acid information. However, $HETATM$ also have $C\alpha$ atoms, because they have nonstandard residues [36]. For our experiment the chemical properties are allocated according to the standard amino acid details (these are allocated to $C\alpha$ atoms of the PDBs' (Chain ID assignment's) ATOM record).

To define (or select) the depths threshold for selection of surface $C\alpha$ atoms, only the PDB_{SITE}s (which satisfied up to SITE records) were considered. From that, to choose the threshold depth (the combination of depths, such as depth of $C\alpha$ atoms and the depth of residue) for surface; two stats are mainly considered such as,

✓ Check how much percentage of SITE groups' $C\alpha$ atoms missed when the threshold value is assigned.

✓ Check how much percentage of $C\alpha$ atoms chosen for the PDB_{SITE} while assigning that threshold.

The overall objective is choosing the threshold combination without losing (more than) approximately 10 % (factor of ten widely used in engineering,**E.g.:** power gain is expressed in decibels(dB) represent in factor of 10 and in telecommunication power gain with Rule of 10s; as well as in biology, **E.g.:** rule of 10 or 10 % rule in energy pyramid, burns rule of tens **etc.**) SITE information and choose the important $C\alpha$s on surface.(which means the minimum number of $C\alpha$ atoms as possible to represent the surface).

To select the threshold ranges (bins). First, all the SITE's depths were pooled. Then $C\alpha$'s threshold bins selected from only considering the $C\alpha$ threshold depth based on loss of the SITE structures between 7 %-11 %, and then the residue depth's threshold bin selection is produced separately as the same way as, depth of $C\alpha$ atom bins.

For the threshold bins of $C\alpha$ atoms depth selection; only the depths of $C\alpha$ atoms from the surface, which has loss of atoms in SITE range [7 %, 8 %, 9 %,10 %,11 %] were considered. From the experiment 7.2, 6.9, 6.7, 6.5, and 6.4 are chosen for the $C\alpha$ threshold depth, which thresholds causes 7 %, 8 %, 9 %,10 %, and 11 % loss of SITE atoms accordingly.

For the threshold bins of residue depth selection; only the depths of residue from the surface, which has loss of atoms in SITE range [7 %, 8 %, 9 %,10 %,11 %] were considered. From the experiment 6.9, 6.7, 6.6, 6.5, and 6.3 are chosen for the residue threshold depth, which thresholds causes 7 %, 8 %, 9 %,10 %, and 11 % loss of SITE atoms accordingly.

After these bins were selected; both bins of thresholds were linked. Where the percentage of MOTIF atoms missed, was checked with the selection of surface using these threshold combinations. Thus, there are totally 25 combinations were checked as shown in the Table. C.1.

Table C.1: MOTIF Missing atoms %, while varying the thresholds of depths

		Threshold of residue depth				
		6.9	6.7	6.6	6.5	6.3
Threshold of $C\alpha$ depth	7.2	9.10914	*9.99487	10.5837	11.1871	12.5445
	6.9	9.90982	10.6431	11.1373	11.6748	12.8863
	6.7	10.6896	11.3106	11.7471	12.206	13.3019
	6.5	11.7551	12.2493	12.6055	12.9826	13.9132
	6.4	12.4258	12.8333	13.1334	13.4768	14.3144

Threshold combinations as *highlighted, the threshold depth of $C\alpha$ as 7.2 and the threshold depth of residue as 6.7 is worked well as shown in Table. C.1. However, the highlighted portion thresholds were assigned separately to find the percentage of $C\alpha$ atoms chosen for surface.

The ratio of missing MOTIF atoms near to 10 % and the percentage of $C\alpha$ atoms missed is checked as shown in Table. C.2.

Table C.2: Percentage of $C\alpha$ atoms chosen for threshold conditions

Classes	$C\alpha$ threshold depth	Residue threshold depth		
		6.9	6.7	6.6
ONGO	7.2	94.241	93.742	93.4545
	6.9	93.7017	93.3184	93.1038
	6.7	93.2101	92.8974	92.7255
TSG	7.2	95.2399	94.8002	94.5313
	6.9	94.7557	94.4162	94.193
	6.7	94.3065	94.0331	93.8488
Fusion	7.2	93.7676	93.0892	92.7362
	6.9	93.3871	92.8099	92.4906
	6.7	93.0233	92.5351	92.2521
All	7.2	94.41617	93.87713	93.574
	6.9	93.94817	93.51483	93.26247
	6.7	93.5133	93.1552	92.94213

Table C.3: Factors for final threshold selection

		Threshold of residue depth		
		6.9	6.7	6.6
Threshold of C_α depth	7.2	0.011889	2.076456	0.01831
	6.9	0.118032	0.01663	0.00943
	6.7	0.01551	0.00819	0.00616

To choose the threshold, a factor is calculated as shown in equation (1). The highest factor's threshold combination was selected. The factor based on the missing percentage MOTIF surface atoms should be nearer to 10 % and it must choose the minimum percentage of surface $C\alpha$ atoms as possible. Thus,

$$Batch\ size = \ ceil\left(\frac{5\ \times\ number\ of\ ONGO\ train\ PDBs}{2}\right) (1)$$

Factors' results are shown in Table. C.3. For an example, the combination of threshold depth of $C\alpha$ as 7.2, threshold depth of residue as 6.7's Factor $= |\frac{1}{((10-9.99487)\times 93.87713)}| = 2.076456$

From the results, the combination of threshold depth of $C\alpha$ as 7.2 and a threshold depth of residue as 6.7 were chosen. The surface $C\alpha$ atom satisfy both depth threshold ($C\alpha$'s depth from the surface less than these depths accordingly) chosen to calculate(mapped to the template) the final surface $C\alpha$ coordinate $< x, y, z >$ as mentioned in *Appendix* B.

The results mentioned in subsection 2.5.2 (subpragraph *"Proposed threshold condition"*) supports the threshold selection of surface works. In the progress the all $C\alpha$ used and the *"Man"* threshold(which is less than the software/ *"soft"* threshold since only one man reported SITE information is available for selected ONGO and TSG class PDBs) are compared with selected SOFTWARE threshold named as (named as *"soft"* thresh in subsection 2.5.2).

As expected *"soft"* thresh works, because the chosen factor is near to factor of ten; factor of ten widely used in engineering(**E.g.:** power gain is expressed in decibels(dB) represent in factor of 10; in telecommunication power gain with Rule of 10s), as well as in biology(**E.g.:**rule of 10 or 10 % rule in energy pyramid, burns rule of tens**etc.**).

Appendix D

Level 1 Property Tables

Table D.1: Amino acids with properties (first half)

Amino acid short form		Arg	Lys	Asp	Glu	Gln	Asn	His
Code		R	K	D	E	Q	N	H
Web reference	charged	1	1	1	1	0	0	0
	Polar	0	0	0	0	1	1	1
	Hydrophobic	0	0	0	0	0	0	0
Soluble reference	Hydrophobic	0	0	0	0	0	0	0
	Moderate	0	0	0	0	0	0	1
	Hydrophilic	1	1	1	1	1	1	0
	polar	0	0	0	0	1	1	0
	Aromatic	0	0	0	0	0	0	0
	Aliphatic	0	0	0	0	0	0	0
	Acid	0	0	1	1	0	0	0
	Basic	1	1	0	0	0	0	1
	Negative charge	0	0	1	1	0	0	0
	Neutral	0	0	0	0	1	1	0
	Positive charge	1	1	0	0	0	0	1
	$pK_a - NH_2$	9.09	10.28	9.6	9.67	9.13	8.8	8.97
	$pK_a - COOH$	2.18	8.9	1.88	2.19	2.17	2.02	1.78
Normalized	$pK_a - NH_2$	0.15	0.75	0.4	0.44	0.17	0	0.09
	$pK_a - COOH$	0.07	1	0.02	0.07	0.06	0.04	0.01

Table D.2: Amino acids with properties (second half)

Amino acid short form		Ser	Thr	Tyr	Cys	Met	Trp	Ala	Ile	Lue	Phe	Val	Pro	Gly
Code		S	T	Y	C	M	W	A	I	L	F	V	P	G
Web reference	charged	0	0	0	0	0	0	0	0	0	0	0	0	0
	Polar	1	1	1	1	1	1	0	0	0	0	0	0	0
	Hydrophobic	0	0	0	0	0	0	1	1	1	1	1	1	1
Soluble reference	Hydrophobic	0	0	1	0	0	1	1	1	1	1	1	1	1
	Moderate	0	0	0	1	1	0	0	0	0	0	0	0	0
	Hydrophilic	1	1	0	0	0	0	0	0	0	0	0	0	0
	polar	1	1	0	1	1	0	0	0	0	0	0	0	0
	Aromatic	0	0	1	0	0	1	0	0	0	1	0	0	0
	Aliphatic	0	0	0	0	0	0	1	1	1	0	1	0	0
	Acid	0	0	0	0	0	0	0	0	0	0	0	0	0
	Basic	0	0	0	0	0	0	0	0	0	0	0	0	0
	Negative charge	0	0	0	0	0	0	0	0	0	0	0	0	0
	Neutral	1	1	1	1	1	1	1	1	1	1	1	1	1
	Positive charge	0	0	0	0	0	0	0	0	0	0	0	0	0
	$pK_a - NH_2$	9.15	9.12	9.11	10.78	9.21	9.39	9.87	9.76	9.6	9.24	9.72	10.6	9.6
	$pK_a - COOH$	2.21	2.15	2.2	1.71	2.28	2.38	2.35	2.32	2.36	2.58	2.29	1.99	2.34
Normalized	$pK_a - NH_2$	0.18	0.16	0.16	1	0.21	0.3	0.54	0.48	0.4	0.22	0.46	0.91	0.4
	$pK_a - COOH$	0.07	0.06	0.07	0	0.08	0.09	0.09	0.08	0.09	0.12	0.08	0.04	0.09

Appendix E

General Biochemical properties for 21 Amino acids

Table E.1: Part one of normalized general properties

Amino acid	Short	Abbrev	Side-chain properties											
			Hydro-phobic		$pKa^{\S}$	Polar		pH		Small				
			yes	No		yes	No	acidic	basic	Yes	No	Aromatic	Aliphatic	
Alanine	A	Ala	1	0	0.191	0	1	0	0	1	0	0	1	
Cysteine	C	Cys	1	0	0.156	0	1	1	0	1	0	0	0	
Aspartic acid	D	Asp	0	1	0.162	1	0	1	0	1	0	0	0	
Glutamic acid	E	Glu	0	1	0.171	1	0	1	0	0	1	0	0	
Phenylalanine	F	Phe	1	0	0.179	0	1	0	0	0	1	1	0	
Glycine	G	Gly	1	0	0.191	0	1	0	0	1	0	0	0	
Histidine	H	His	0	1	0.146	1	0	0	0.3	0	1	1	0	
Isoleucine	I	Ile	1	0	0.189	0	1	0	0	0	1	0	1	
Lysine	K	Lys	0	1	0.176	1	0	0	0.6	0	1	0	0	
Leucine	L	Leu	1	0	0.189	0	1	0	0	0	1	0	1	
Methionine	M	Met	1	0	0.173	0	1	0	0	0	1	0	1	
Asparagine	N	Asn	0	1	0.174	1	0	0	0	1	0	0	0	
Proline	P	Pro	1	0	0.159	0	1	0	0	1	0	0	0	
Glutamine	Q	Gln	0	1	0.176	1	0	0	0	0	1	0	0	
Arginine	R	Arg	0	1	0.148	1	0	0	1	0	1	0	0	
Serine	S	Ser	0	1	0.178	1	0	0	0	1	0	0	0	
Threonine	T	Thr	0	1	0.170	1	0	0	0	1	0	0	0	
Selenocysteine	U	Sec	0	1	0.155	0	1	1	0	1	0	0	0	
Valine	V	Val	1	0	0.194	0	1	0	0	1	0	0	1	
Tryptophan	W	Trp	1	0	0.200	0	1	0	0	0	1	1	0	
Tyrosine	Y	Tyr	0	1	0.179	1	0	0.3	0	0	1	1	0	
Normalisation way explained			Not needed binary		Normalized*	Not needed binary		Between 0-1; and 0.3 , 0.6 shows the weightage		Not needed binary				

Table E.2: Part two of normalized general properties

Amino acid	Short	Abbrev.	General chemical properties				Gene expression and biochemistry			Mass spectrometry	
					pK_1	pK_2	Essential in humans				
			Avg. mass (Da)	pI	$(\alpha - COOH)$	$(\alpha - {}^{+}NH_3)$	Yes	No	Conditionally	Mon. mass§ (Da)	Avg. mass (Da)
Alanine	A	Ala	0.436	0.399	0.833	0.581	0	1	0	0.382	0.382
Cysteine	C	Cys	0.593	0.278	0.182	1.000	0	0	1	0.554	0.554
Aspartic acid	D	Asp	0.652	0.000	0.288	0.596	0	1	0	0.618	0.618
Glutamic acid	E	Glu	0.720	0.038	0.455	0.379	0	0	1	0.693	0.693
Phenylalanine	F	Phe	0.809	0.334	0.606	0.298	1	0	0	0.790	0.790
Glycine	G	Gly	0.368	0.406	0.833	0.535	0	0	1	0.306	0.306
Histidine	H	His	0.760	0.601	0.000	0.308	1	0	0	0.737	0.736
Isoleucine	I	Ile	0.642	0.405	0.788	0.525	1	0	0	0.608	0.608
Lysine	K	Lys	0.716	0.853	0.545	0.172	1	0	0	0.688	0.688
Leucine	L	Leu	0.642	0.399	0.803	0.515	1	0	0	0.608	0.608
Methionine	M	Met	0.731	0.365	0.500	0.283	1	0	0	0.704	0.705
Asparagine	N	Asn	0.647	0.324	0.515	0.000	0	1	0	0.613	0.613
Proline	P	Pro	0.564	0.436	0.227	0.970	0	1	0	0.522	0.522
Glutamine	Q	Gln	0.716	0.354	0.561	0.207	0	1	0	0.688	0.688
Arginine	R	Arg	0.853	1.000	0.030	0.136	0	0	1	0.839	0.839
Serine	S	Ser	0.515	0.358	0.591	0.247	0	1	0	0.468	0.468
Threonine	T	Thr	0.583	0.348	0.439	0.192	1	0	0	0.543	0.543
Selenocysteine	U	Sec	0.823	0.331	0.167	0.646	0	1	0	0.811	0.806
Valine	V	Val	0.574	0.398	0.894	0.515	1	0	0	0.532	0.532
Tryptophan	W	Trp	1.000	0.384	1.000	0.348	1	0	0	1.000	1.000
Tyrosine	Y	Tyr	0.887	0.353	0.606	0.247	0	0	1	0.876	0.876
Normalisation way explained			This part factored by the 1/max_Avg_mass	Normalized*			Not needed binary			This part factored by the 1/max_Avg_mass	

Here the new amino acid "Selenocysteine" is added along with the 20 amino acids, because of it's contribution to human health [27]. And now the "Selenocysteine" is represented by "U". And the new general normalized properties of the 21 amino acids is shown in Table. E.1 and Table. E.2(since the number of properties are high for visualization, thus the table is split into two parts). Earlier property representation shown in (*Appendix* D) also used normalized data (but directly 9 digits used; and the Tables' only present 3 digits for visualization).

One Example PDB, "5WLB" in training class ONGO, that has 21^{st} amino acid in the PDB file thus it is not satisfied by 17 properties.

And the Normalized* columns represent the property values are just normalized as shown in the equation below,

$Normalized_{value} = \frac{(value\ -\ min_{value})}{(max_{value} - min_{value})}$

Where the min_{value} is the minimum value of the given property, and max_{value} is the maximum value of the given property.

The property values are directly obtained from the summery in [3], since these property values are already obtained and categorized as

✓ General chemical properties

✓ Side-chain properties

✓ Gene expression and biochemistry

✓ Mass spectrometry

✓ Stoichiometry and metabolic cost in cell

from the literatures [1, 4, 6–8, 12, 14–17, 19, 20, 24, 25, 35, 38] , I just took the property values from summery in [3], and check the property satisfy (have a value for the property) for all the 21-amino acids; among 21 amino acids, even if one amino acid not has the value then the property is left. Such that 20 amino acids properties are obtained and tabled as shown in Table. E.1 and Table. E.2.

Appendix F

Towers with Different Parameters for each Projection

F.1 *Model_vin_4*

The kernels for each tower is different like 3, 5, 7 in all depths Same kind of model with little variation model's architecture as shown in the Figure below. I have tried the model with residual connections as well.

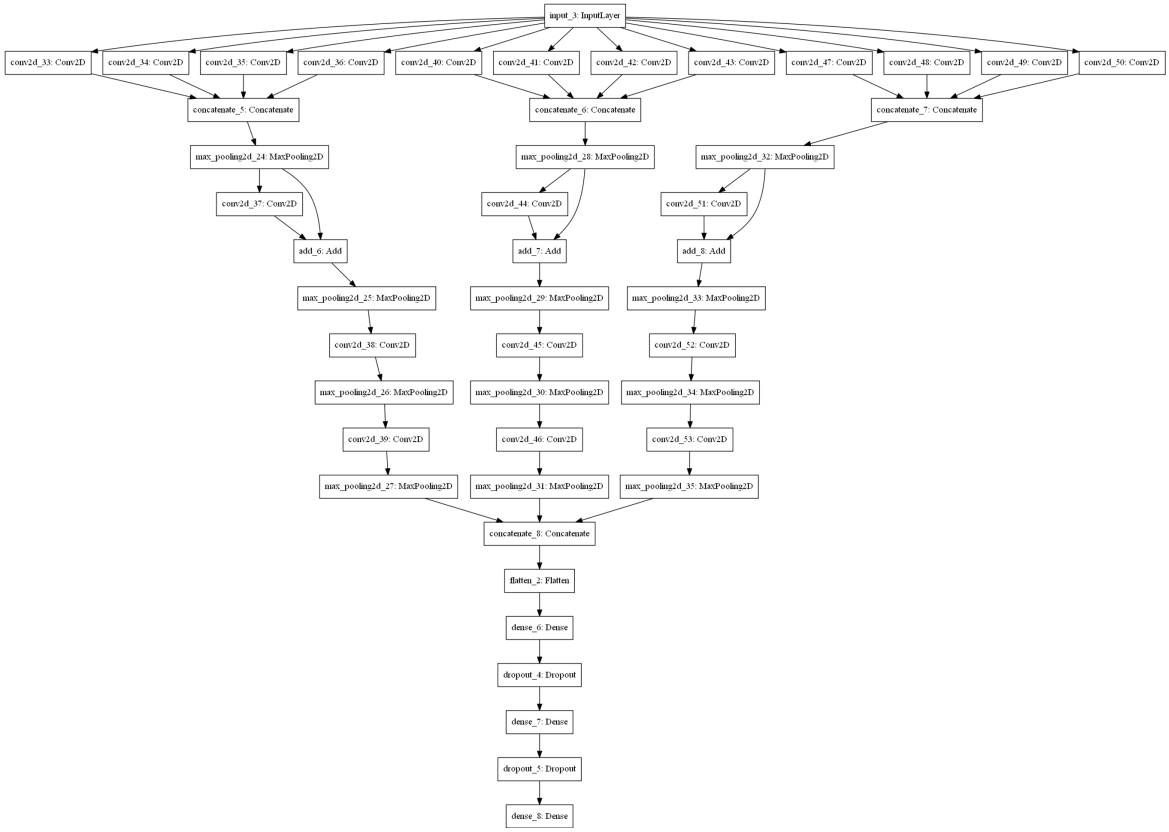
Figure F.1: *model_vin_4* Architecthture.

Total params: 1,337,175 **Trainable params:** 1,337,175

F.2 *Model_inception_vin_1*: with depth 8 and the kernels as 3

Same kind of model with little variation model's architecture: I have tried the models with the kernels as (3,3) and only (5,5) in inception since those only gave better performance in the in initial evaluations of models; give much depths depends on the number of properties); Further inception without addition in the second layer; just normal inception in the beginning only.

Figure F.2: *model_inception_vin_1* Architecthture.

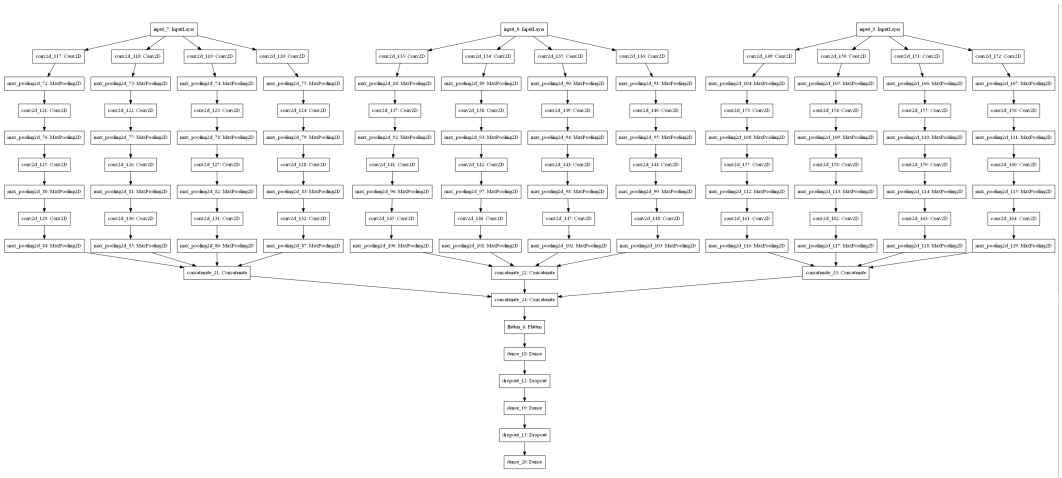


Total params: 957,527 **Trainable params:** 957,527

F.3 *Model_inception_all_depths_inception_vin_1*

Same kind of model with little variations model's architecture: I have tried the model with the kernels as (3,3 and only 5,5 in inception since those only gave better performance in initial evaluations give much depths depends on the number of properties).

Figure F.3: *model_inception_all_depths_inception_vin_*1 Architecthture.



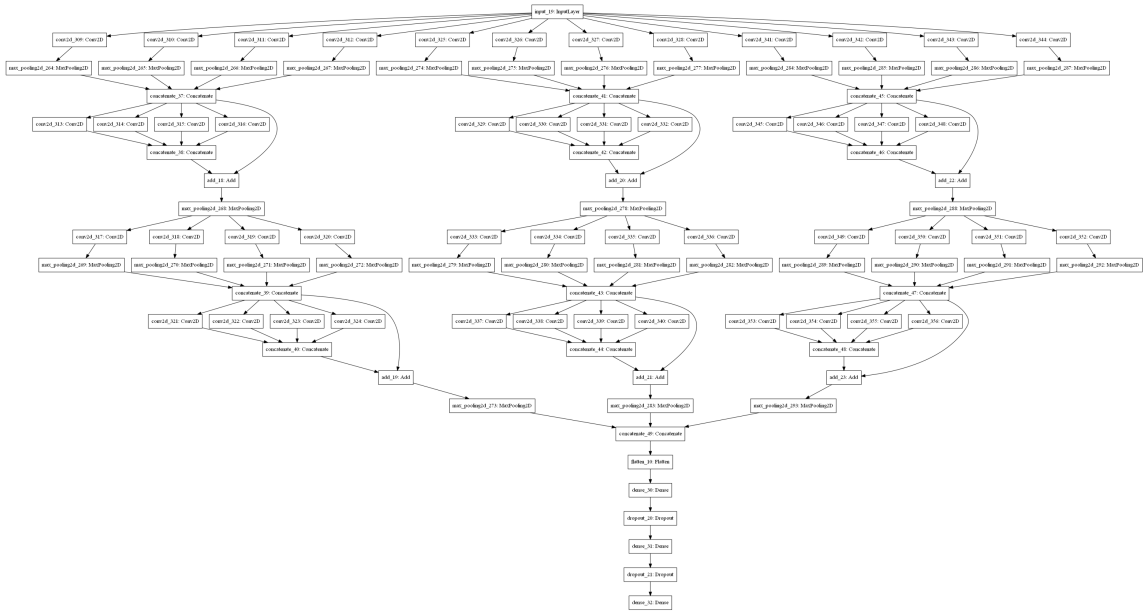
Total params: 984,407
Trainable params: 984,407
Tried models and parameters:
All depths are 16 then **trainable params:** 1,208,951
Depths are 16,8,8,16 then **trainable params:** 988,343

F.4 *Model_inception_all_depths_inception_complic_vin_*1

Depth of layer 1 & 2 -assigned as 8 for each kernel projections (end up in 32 each layer), and Depth of layer 3 & 4 -assigned as 16 (end up in 64 each layer).

I have tried the model with the kernels as (3,3 and only 5,5 in inception since those only gave better performance in initial evaluations give many depths depends on the number of properties). Tried without residual as well.

Figure F.4: *model_inception_all_depths_inception_complic_vin_*1 Architecthture.



Total params: 1,645,655
Trainable params: 1,645,655
When all depth for kernels as 16: 64 for each layer **Total params:** 2,342,519 When, depth for kernels in layer1 and layer2 as 16: 64 for each layer and depth for kernels in layer3 and layer4 as 32: 128 **Total params:** 5,049,335; **Trainable params:** 5,049,335

Appendix G

Parameter details of different tower selection presented in Appendix F

Table G.1: *model_vin_4*: Parameter details.

Layer	(type)	Output Shape	Number of Parameters	Connected
input_1	(InputLayer)	(None, 200, 200, 17)	0	
conv2d	(Conv2D)	(None, 200, 200, 32)	26688	input_1[0][0]
conv2d_4	(Conv2D)	(None, 200, 200, 32)	13632	input_1[0][0]
conv2d_8	(Conv2D)	(None, 200, 200, 32)	4928	input_1[0][0]
max_pooling2d	(MaxPooling2D)	(None, 50, 50, 32)	0	conv2d[0][0]
max_pooling2d_4	(MaxPooling2D)	(None, 50, 50, 32)	0	conv2d_4[0][0]
max_pooling2d_8	(MaxPooling2D)	(None, 50, 50, 32)	0	conv2d_8[0][0]
conv2d_1	(Conv2D)	(None, 50, 50, 32)	50208	max_pooling2d[0][0]
conv2d_5	(Conv2D)	(None, 50, 50, 32)	25632	max_pooling2d_4[0][0]
conv2d_9	(Conv2D)	(None, 50, 50, 32)	9248	max_pooling2d_8[0][0]
add	(Add)	(None, 50, 50, 32)	0	conv2d_1[0][0]
	max_pooling2d[0][0]			
add_1	(Add)	(None, 50, 50, 32)	0	conv2d_5[0][0]
	max_pooling2d_4[0][0]			
add_2	(Add)	(None, 50, 50, 32)	0	conv2d_9[0][0]
	max_pooling2d_8[0][0]			
max_pooling2d_1	(MaxPooling2D)	(None, 25, 25, 32)	0	add[0][0]
max_pooling2d_5	(MaxPooling2D)	(None, 25, 25, 32)	0	add_1[0][0]
max_pooling2d_9	(MaxPooling2D)	(None, 25, 25, 32)	0	add_2[0][0]
conv2d_2	(Conv2D)	(None, 25, 25, 64)	100416	max_pooling2d_1[0][0]
conv2d_6	(Conv2D)	(None, 25, 25, 64)	51264	max_pooling2d_5[0][0]
conv2d_10	(Conv2D)	(None, 25, 25, 64)	18496	max_pooling2d_9[0][0]
max_pooling2d_2	(MaxPooling2D)	(None, 12, 12, 64)	0	conv2d_2[0][0]
max_pooling2d_6	(MaxPooling2D)	(None, 12, 12, 64)	0	conv2d_6[0][0]
max_pooling2d_10	(MaxPooling2D)	(None, 12, 12, 64)	0	conv2d_10[0][0]
conv2d_3	(Conv2D)	(None, 12, 12, 64)	200768	max_pooling2d_2[0][0]
conv2d_7	(Conv2D)	(None, 12, 12, 64)	102464	max_pooling2d_6[0][0]
conv2d_11	(Conv2D)	(None, 12, 12, 64)	36928	max_pooling2d_10[0][0]
max_pooling2d_3	(MaxPooling2D)	(None, 6, 6, 64)	0	conv2d_3[0][0]
max_pooling2d_7	(MaxPooling2D)	(None, 6, 6, 64)	0	conv2d_7[0][0]
max_pooling2d_11	(MaxPooling2D)	(None, 6, 6, 64)	0	conv2d_11[0][0]
concatenate	(Concatenate)	(None, 18, 6, 64)	0	max_pooling2d_3[0][0]
	max_pooling2d_7[0][0]			
	max_pooling2d_11[0][0]			
flatten	(Flatten)	(None, 6912)	0	concatenate[0][0]
dense	(Dense)	(None, 100)	691300	flatten[0][0]
dropout	(Dropout)	(None, 100)	0	dense[0][0]
dense_1	(Dense)	(None, 50)	5050	dropout[0][0]
dropout_1	(Dropout)	(None, 50)	0	dense_1[0][0]
dense_2	(Dense)	(None, 3)	153	dropout_1[0][0]
Total params	1,337,175			
Trainable params	1,337,175			

Table G.2: *model_inception_vin_1*: Parameter details.

Layer	(type)	Output Shape	Number of Parameters	Connected
input_6	(InputLayer)	(None, 200, 200, 17)	0	
conv2d_96	(Conv2D)	(None, 200, 200, 8)	1232	input_6[0][0]
conv2d_97	(Conv2D)	(None, 200, 200, 8)	3408	input_6[0][0]
conv2d_98	(Conv2D)	(None, 200, 200, 8)	6672	input_6[0][0]
conv2d_99	(Conv2D)	(None, 200, 200, 8)	11024	input_6[0][0]
conv2d_103	(Conv2D)	(None, 200, 200, 8)	1232	input_6[0][0]
conv2d_104	(Conv2D)	(None, 200, 200, 8)	3408	input_6[0][0]
conv2d_105	(Conv2D)	(None, 200, 200, 8)	6672	input_6[0][0]
conv2d_106	(Conv2D)	(None, 200, 200, 8)	11024	input_6[0][0]
conv2d_110	(Conv2D)	(None, 200, 200, 8)	1232	input_6[0][0]
conv2d_111	(Conv2D)	(None, 200, 200, 8)	3408	input_6[0][0]
conv2d_112	(Conv2D)	(None, 200, 200, 8)	6672	input_6[0][0]
conv2d_113	(Conv2D)	(None, 200, 200, 8)	11024	input_6[0][0]
concatenate_17	(Concatenate)	(None, 200, 200, 32)	0	conv2d_96[0][0]
	conv2d_97[0][0]			
	conv2d_98[0][0]			
	conv2d_99[0][0]			
concatenate_18	(Concatenate)	(None, 200, 200, 32)	0	conv2d_103[0][0]
	conv2d_104[0][0]			
	conv2d_105[0][0]			
	conv2d_106[0][0]			
concatenate_19	(Concatenate)	(None, 200, 200, 32)	0	conv2d_110[0][0]
	conv2d_111[0][0]			
	conv2d_112[0][0]			
	conv2d_113[0][0]			
max_pooling2d_60	(MaxPooling2D)	(None, 50, 50, 32)	0	concatenate_17[0][0]
max_pooling2d_64	(MaxPooling2D)	(None, 50, 50, 32)	0	concatenate_18[0][0]
max_pooling2d_68	(MaxPooling2D)	(None, 50, 50, 32)	0	concatenate_19[0][0]
conv2d_100	(Conv2D)	(None, 50, 50, 32)	9248	max_pooling2d_60[0][0]
conv2d_107	(Conv2D)	(None, 50, 50, 32)	9248	max_pooling2d_64[0][0]
conv2d_114	(Conv2D)	(None, 50, 50, 32)	9248	max_pooling2d_68[0][0]
add_15	(Add)	(None, 50, 50, 32)	0	conv2d_100[0][0]
	max_pooling2d_60[0][0]			
add_16	(Add)	(None, 50, 50, 32)	0	conv2d_107[0][0]
	max_pooling2d_64[0][0]			
add_17	(Add)	(None, 50, 50, 32)	0	conv2d_114[0][0]
	max_pooling2d_68[0][0]			
max_pooling2d_61	(MaxPooling2D)	(None, 25, 25, 32)	0	add_15[0][0]
max_pooling2d_65	(MaxPooling2D)	(None, 25, 25, 32)	0	add_16[0][0]
max_pooling2d_69	(MaxPooling2D)	(None, 25, 25, 32)	0	add_17[0][0]
conv2d_101	(Conv2D)	(None, 25, 25, 64)	18496	max_pooling2d_61[0][0]
conv2d_108	(Conv2D)	(None, 25, 25, 64)	18496	max_pooling2d_65[0][0]
conv2d_115	(Conv2D)	(None, 25, 25, 64)	18496	max_pooling2d_69[0][0]
max_pooling2d_62	(MaxPooling2D)	(None, 12, 12, 64)	0	conv2d_101[0][0]
max_pooling2d_66	(MaxPooling2D)	(None, 12, 12, 64)	0	conv2d_108[0][0]
max_pooling2d_70	(MaxPooling2D)	(None, 12, 12, 64)	0	conv2d_115[0][0]
conv2d_102	(Conv2D)	(None, 12, 12, 64)	36928	max_pooling2d_62[0][0]
conv2d_109	(Conv2D)	(None, 12, 12, 64)	36928	max_pooling2d_66[0][0]
conv2d_116	(Conv2D)	(None, 12, 12, 64)	36928	max_pooling2d_70[0][0]
max_pooling2d_63	(MaxPooling2D)	(None, 6, 6, 64)	0	conv2d_102[0][0]
max_pooling2d_67	(MaxPooling2D)	(None, 6, 6, 64)	0	conv2d_109[0][0]
max_pooling2d_71	(MaxPooling2D)	(None, 6, 6, 64)	0	conv2d_116[0][0]
concatenate_20	(Concatenate)	(None, 18, 6, 64)	0	max_pooling2d_63[0][0]
	max_pooling2d_67[0][0]			
	max_pooling2d_71[0][0]			
flatten_5	(Flatten)	(None, 6912)	0	concatenate_20[0][0]
dense_15	(Dense)	(None, 100)	691300	flatten_5[0][0]
dropout_10	(Dropout)	(None, 100)	0	dense_15[0][0]
dense_16	(Dense)	(None, 50)	5050	dropout_10[0][0]
dropout_11	(Dropout)	(None, 50)	0	dense_16[0][0]
dense_17	(Dense)	(None, 3)	153	dropout_11[0][0]
Total params	957,527			
Trainable params	957,527			

Table G.3: *model_inception_all_depths_inception_vin_1*: Parameter details.

Layer	(type)	Output Shape	Number of Parameters	Connected
input_16	(InputLayer)	(None, 200, 200, 17)	0	

Table H.4 *model_inception_all_depths_inception_vin_1*: Parameter details (continued)

Layer	(type)	Output Shape	Number of Parameters	Connected
input_17	(InputLayer)	[(None, 200, 200, 17	0	
input_18	(InputLayer)	[(None, 200, 200, 17	0	
conv2d_261	(Conv2D)	(None, 200, 200, 8)	1232	input_16[0][0]
conv2d_262	(Conv2D)	(None, 200, 200, 8)	3408	input_16[0][0]
conv2d_263	(Conv2D)	(None, 200, 200, 8)	6672	input_16[0][0]
conv2d_264	(Conv2D)	(None, 200, 200, 8)	11024	input_16[0][0]
conv2d_277	(Conv2D)	(None, 200, 200, 8)	1232	input_17[0][0]
conv2d_278	(Conv2D)	(None, 200, 200, 8)	3408	input_17[0][0]
conv2d_279	(Conv2D)	(None, 200, 200, 8)	6672	input_17[0][0]
conv2d_280	(Conv2D)	(None, 200, 200, 8)	11024	input_17[0][0]
conv2d_293	(Conv2D)	(None, 200, 200, 8)	1232	input_18[0][0]
conv2d_294	(Conv2D)	(None, 200, 200, 8)	3408	input_18[0][0]
conv2d_295	(Conv2D)	(None, 200, 200, 8)	6672	input_18[0][0]
conv2d_296	(Conv2D)	(None, 200, 200, 8)	11024	input_18[0][0]
max_pooling2d_216	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_261[0][0]
max_pooling2d_217	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_262[0][0]
max_pooling2d_218	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_263[0][0]
max_pooling2d_219	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_264[0][0]
max_pooling2d_232	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_277[0][0]
max_pooling2d_233	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_278[0][0]
max_pooling2d_234	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_279[0][0]
max_pooling2d_235	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_280[0][0]
max_pooling2d_248	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_293[0][0]
max_pooling2d_249	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_294[0][0]
max_pooling2d_250	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_295[0][0]
max_pooling2d_251	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_296[0][0]
conv2d_265	(Conv2D)	(None, 50, 50, 8)	584	max_pooling2d_216[0][0]
conv2d_266	(Conv2D)	(None, 50, 50, 8)	1608	max_pooling2d_217[0][0]
conv2d_267	(Conv2D)	(None, 50, 50, 8)	3144	max_pooling2d_218[0][0]
conv2d_268	(Conv2D)	(None, 50, 50, 8)	5192	max_pooling2d_219[0][0]
conv2d_281	(Conv2D)	(None, 50, 50, 8)	584	max_pooling2d_232[0][0]
conv2d_282	(Conv2D)	(None, 50, 50, 8)	1608	max_pooling2d_233[0][0]
conv2d_283	(Conv2D)	(None, 50, 50, 8)	3144	max_pooling2d_234[0][0]
conv2d_284	(Conv2D)	(None, 50, 50, 8)	5192	max_pooling2d_235[0][0]
conv2d_297	(Conv2D)	(None, 50, 50, 8)	584	max_pooling2d_248[0][0]
conv2d_298	(Conv2D)	(None, 50, 50, 8)	1608	max_pooling2d_249[0][0]
conv2d_299	(Conv2D)	(None, 50, 50, 8)	3144	max_pooling2d_250[0][0]
conv2d_300	(Conv2D)	(None, 50, 50, 8)	5192	max_pooling2d_251[0][0]
max_pooling2d_220	(MaxPooling2D	(None, 25, 25, 8)	0	conv2d_265[0][0]
max_pooling2d_221	(MaxPooling2D	(None, 25, 25, 8)	0	conv2d_266[0][0]
max_pooling2d_222	(MaxPooling2D	(None, 25, 25, 8)	0	conv2d_267[0][0]
max_pooling2d_223	(MaxPooling2D	(None, 25, 25, 8)	0	conv2d_268[0][0]
max_pooling2d_236	(MaxPooling2D	(None, 25, 25, 8)	0	conv2d_281[0][0]
max_pooling2d_237	(MaxPooling2D	(None, 25, 25, 8)	0	conv2d_282[0][0]
max_pooling2d_238	(MaxPooling2D	(None, 25, 25, 8)	0	conv2d_283[0][0]
max_pooling2d_239	(MaxPooling2D	(None, 25, 25, 8)	0	conv2d_284[0][0]
max_pooling2d_252	(MaxPooling2D	(None, 25, 25, 8)	0	conv2d_297[0][0]
max_pooling2d_253	(MaxPooling2D	(None, 25, 25, 8)	0	conv2d_298[0][0]
max_pooling2d_254	(MaxPooling2D	(None, 25, 25, 8)	0	conv2d_299[0][0]
max_pooling2d_255	(MaxPooling2D	(None, 25, 25, 8)	0	conv2d_300[0][0]
conv2d_269	(Conv2D)	(None, 25, 25, 16)	1168	max_pooling2d_220[0][0]
conv2d_270	(Conv2D)	(None, 25, 25, 16)	3216	max_pooling2d_221[0][0]
conv2d_271	(Conv2D)	(None, 25, 25, 16)	6288	max_pooling2d_222[0][0]
conv2d_272	(Conv2D)	(None, 25, 25, 16)	10384	max_pooling2d_223[0][0]
conv2d_285	(Conv2D)	(None, 25, 25, 16)	1168	max_pooling2d_236[0][0]
conv2d_286	(Conv2D)	(None, 25, 25, 16)	3216	max_pooling2d_237[0][0]
conv2d_287	(Conv2D)	(None, 25, 25, 16)	6288	max_pooling2d_238[0][0]
conv2d_288	(Conv2D)	(None, 25, 25, 16)	10384	max_pooling2d_239[0][0]
conv2d_301	(Conv2D)	(None, 25, 25, 16)	1168	max_pooling2d_252[0][0]
conv2d_302	(Conv2D)	(None, 25, 25, 16)	3216	max_pooling2d_253[0][0]
conv2d_303	(Conv2D)	(None, 25, 25, 16)	6288	max_pooling2d_254[0][0]
conv2d_304	(Conv2D)	(None, 25, 25, 16)	10384	max_pooling2d_255[0][0]
max_pooling2d_224	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_269[0][0]
max_pooling2d_225	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_270[0][0]
max_pooling2d_226	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_271[0][0]
max_pooling2d_227	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_272[0][0]
max_pooling2d_240	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_285[0][0]
max_pooling2d_241	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_286[0][0]
max_pooling2d_242	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_287[0][0]
max_pooling2d_243	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_288[0][0]
max_pooling2d_256	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_301[0][0]
max_pooling2d_257	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_302[0][0]
max_pooling2d_258	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_303[0][0]
max_pooling2d_259	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_304[0][0]
conv2d_273	(Conv2D)	(None, 12, 12, 16)	2320	max_pooling2d_224[0][0]

Table H.4 *model_inception_all_depths_inception_vin_1*: Parameter details (continued)

Layer	(type)	Output Shape	Number of Parameters	Connected
conv2d_274	(Conv2D)	(None, 12, 12, 16)	6416	max_pooling2d_225[0][0]
conv2d_275	(Conv2D)	(None, 12, 12, 16)	12560	max_pooling2d_226[0][0]
conv2d_276	(Conv2D)	(None, 12, 12, 16)	20752	max_pooling2d_227[0][0]
conv2d_289	(Conv2D)	(None, 12, 12, 16)	2320	max_pooling2d_240[0][0]
conv2d_290	(Conv2D)	(None, 12, 12, 16)	6416	max_pooling2d_241[0][0]
conv2d_291	(Conv2D)	(None, 12, 12, 16)	12560	max_pooling2d_242[0][0]
conv2d_292	(Conv2D)	(None, 12, 12, 16)	20752	max_pooling2d_243[0][0]
conv2d_305	(Conv2D)	(None, 12, 12, 16)	2320	max_pooling2d_256[0][0]
conv2d_306	(Conv2D)	(None, 12, 12, 16)	6416	max_pooling2d_257[0][0]
conv2d_307	(Conv2D)	(None, 12, 12, 16)	12560	max_pooling2d_258[0][0]
conv2d_308	(Conv2D)	(None, 12, 12, 16)	20752	max_pooling2d_259[0][0]
max_pooling2d_228	(MaxPooling2D	(None, 6, 6, 16)	0	conv2d_273[0][0]
max_pooling2d_229	(MaxPooling2D	(None, 6, 6, 16)	0	conv2d_274[0][0]
max_pooling2d_230	(MaxPooling2D	(None, 6, 6, 16)	0	conv2d_275[0][0]
max_pooling2d_231	(MaxPooling2D	(None, 6, 6, 16)	0	conv2d_276[0][0]
max_pooling2d_244	(MaxPooling2D	(None, 6, 6, 16)	0	conv2d_289[0][0]
max_pooling2d_245	(MaxPooling2D	(None, 6, 6, 16)	0	conv2d_290[0][0]
max_pooling2d_246	(MaxPooling2D	(None, 6, 6, 16)	0	conv2d_291[0][0]
max_pooling2d_247	(MaxPooling2D	(None, 6, 6, 16)	0	conv2d_292[0][0]
max_pooling2d_260	(MaxPooling2D	(None, 6, 6, 16)	0	conv2d_305[0][0]
max_pooling2d_261	(MaxPooling2D	(None, 6, 6, 16)	0	conv2d_306[0][0]
max_pooling2d_262	(MaxPooling2D	(None, 6, 6, 16)	0	conv2d_307[0][0]
max_pooling2d_263	(MaxPooling2D	(None, 6, 6, 16)	0	conv2d_308[0][0]
concatenate_33	(Concatenate)	(None, 6, 6, 64)	0	max_pooling2d_228[0][0]
	max_pooling2d_229[0][0]			
	max_pooling2d_230[0][0]			
	max_pooling2d_231[0][0]			
concatenate_34	(Concatenate)	(None, 6, 6, 64)	0	max_pooling2d_244[0][0]
	max_pooling2d_245[0][0]			
	max_pooling2d_246[0][0]			
	max_pooling2d_247[0][0]			
concatenate_35	(Concatenate)	(None, 6, 6, 64)	0	max_pooling2d_260[0][0]
	max_pooling2d_261[0][0]			
	max_pooling2d_262[0][0]			
	max_pooling2d_263[0][0]			
concatenate_36	(Concatenate)	(None, 18, 6, 64)	0	concatenate_33[0][0]
	concatenate_34[0][0]			
	concatenate_35[0][0]			
flatten_9	(Flatten)	(None, 6912)	0	concatenate_36[0][0]
dense_27	(Dense)	(None, 100)	691300	flatten_9[0][0]
dropout_18	(Dropout)	(None, 100)	0	dense_27[0][0]
dense_28	(Dense)	(None, 50)	5050	dropout_18[0][0]
dropout_19	(Dropout)	(None, 50)	0	dense_28[0][0]
dense_29	(Dense)	(None, 3)	153	dropout_19[0][0]
Total params	984,407			
Trainable params	984,407			

Table G.4: *model_inception_all_depths_inception_complic_vin_1*: Parameter details.

Layer	(type)	Output Shape	Number of Parameters	Connected
input_16	(InputLayer)	(None, 200, 200, 17)	0	
conv2d_453	(Conv2D)	(None, 200, 200, 8)	1232	input_22[0][0]
conv2d_454	(Conv2D)	(None, 200, 200, 8)	3408	input_22[0][0]
conv2d_455	(Conv2D)	(None, 200, 200, 8)	6672	input_22[0][0]
conv2d_456	(Conv2D)	(None, 200, 200, 8)	11024	input_22[0][0]
conv2d_469	(Conv2D)	(None, 200, 200, 8)	1232	input_22[0][0]
conv2d_470	(Conv2D)	(None, 200, 200, 8)	3408	input_22[0][0]
conv2d_471	(Conv2D)	(None, 200, 200, 8)	6672	input_22[0][0]
conv2d_472	(Conv2D)	(None, 200, 200, 8)	11024	input_22[0][0]
conv2d_485	(Conv2D)	(None, 200, 200, 8)	1232	input_22[0][0]
conv2d_486	(Conv2D)	(None, 200, 200, 8)	3408	input_22[0][0]
conv2d_487	(Conv2D)	(None, 200, 200, 8)	6672	input_22[0][0]
conv2d_488	(Conv2D)	(None, 200, 200, 8)	11024	input_22[0][0]
max_pooling2d_354	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_453[0][0]
max_pooling2d_355	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_454[0][0]
max_pooling2d_356	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_455[0][0]

Table G.4 *model_inception_all_depths_inception_complic_vin_1*: Parameter details (continued)

Layer	(type)	Output Shape	Number of Parameters	Connected
max_pooling2d_357	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_456[0][0]
max_pooling2d_364	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_469[0][0]
max_pooling2d_365	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_470[0][0]
max_pooling2d_366	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_471[0][0]
max_pooling2d_367	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_472[0][0]
max_pooling2d_374	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_485[0][0]
max_pooling2d_375	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_486[0][0]
max_pooling2d_376	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_487[0][0]
max_pooling2d_377	(MaxPooling2D	(None, 50, 50, 8)	0	conv2d_488[0][0]
concatenate_76	(Concatenate)	(None, 50, 50, 32)	0	max_pooling2d_354[0][0]
	max_pooling2d_355[0][0]			
	max_pooling2d_356[0][0]			
	max_pooling2d_357[0][0]			
concatenate_80	(Concatenate)	(None, 50, 50, 32)	0	max_pooling2d_364[0][0]
	max_pooling2d_365[0][0]			
	max_pooling2d_366[0][0]			
	max_pooling2d_367[0][0]			
concatenate_84	(Concatenate)	(None, 50, 50, 32)	0	max_pooling2d_374[0][0]
	max_pooling2d_375[0][0]			
	max_pooling2d_376[0][0]			
	max_pooling2d_377[0][0]			
conv2d_457	(Conv2D)	(None, 50, 50, 8)	2312	concatenate_76[0][0]
conv2d_458	(Conv2D)	(None, 50, 50, 8)	6408	concatenate_76[0][0]
conv2d_459	(Conv2D)	(None, 50, 50, 8)	12552	concatenate_76[0][0]
conv2d_460	(Conv2D)	(None, 50, 50, 8)	20744	concatenate_76[0][0]
conv2d_473	(Conv2D)	(None, 50, 50, 8)	2312	concatenate_80[0][0]
conv2d_474	(Conv2D)	(None, 50, 50, 8)	6408	concatenate_80[0][0]
conv2d_475	(Conv2D)	(None, 50, 50, 8)	12552	concatenate_80[0][0]
conv2d_476	(Conv2D)	(None, 50, 50, 8)	20744	concatenate_80[0][0]
conv2d_489	(Conv2D)	(None, 50, 50, 8)	2312	concatenate_84[0][0]
conv2d_490	(Conv2D)	(None, 50, 50, 8)	6408	concatenate_84[0][0]
conv2d_491	(Conv2D)	(None, 50, 50, 8)	12552	concatenate_84[0][0]
conv2d_492	(Conv2D)	(None, 50, 50, 8)	20744	concatenate_84[0][0]
concatenate_77	(Concatenate)	(None, 50, 50, 32)	0	conv2d_457[0][0]
	conv2d_458[0][0]			
	conv2d_459[0][0]			
	conv2d_460[0][0]			
concatenate_81	(Concatenate)	(None, 50, 50, 32)	0	conv2d_473[0][0]
	conv2d_474[0][0]			
	conv2d_475[0][0]			
	conv2d_476[0][0]			
concatenate_85	(Concatenate)	(None, 50, 50, 32)	0	conv2d_489[0][0]
	conv2d_490[0][0]			
	conv2d_491[0][0]			
	conv2d_492[0][0]			
add_36	(Add)	(None, 50, 50, 32)	0	concatenate_76[0][0]
	concatenate_77[0][0]			
add_38	(Add)	(None, 50, 50, 32)	0	concatenate_80[0][0]
	concatenate_81[0][0]			
add_40	(Add)	(None, 50, 50, 32)	0	concatenate_84[0][0]
	concatenate_85[0][0]			
max_pooling2d_358	(MaxPooling2D	(None, 25, 25, 32)	0	add_36[0][0]
max_pooling2d_368	(MaxPooling2D	(None, 25, 25, 32)	0	add_38[0][0]
max_pooling2d_378	(MaxPooling2D	(None, 25, 25, 32)	0	add_40[0][0]
conv2d_461	(Conv2D)	(None, 25, 25, 16)	4624	max_pooling2d_358[0][0]
conv2d_462	(Conv2D)	(None, 25, 25, 16)	12816	max_pooling2d_358[0][0]
conv2d_463	(Conv2D)	(None, 25, 25, 16)	25104	max_pooling2d_358[0][0]
conv2d_464	(Conv2D)	(None, 25, 25, 16)	41488	max_pooling2d_358[0][0]
conv2d_477	(Conv2D)	(None, 25, 25, 16)	4624	max_pooling2d_368[0][0]
conv2d_478	(Conv2D)	(None, 25, 25, 16)	12816	max_pooling2d_368[0][0]
conv2d_479	(Conv2D)	(None, 25, 25, 16)	25104	max_pooling2d_368[0][0]
conv2d_480	(Conv2D)	(None, 25, 25, 16)	41488	max_pooling2d_368[0][0]
conv2d_493	(Conv2D)	(None, 25, 25, 16)	4624	max_pooling2d_378[0][0]
conv2d_494	(Conv2D)	(None, 25, 25, 16)	12816	max_pooling2d_378[0][0]
conv2d_495	(Conv2D)	(None, 25, 25, 16)	25104	max_pooling2d_378[0][0]
conv2d_496	(Conv2D)	(None, 25, 25, 16)	41488	max_pooling2d_378[0][0]
max_pooling2d_359	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_461[0][0]
max_pooling2d_360	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_462[0][0]
max_pooling2d_361	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_463[0][0]
max_pooling2d_362	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_464[0][0]
max_pooling2d_369	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_477[0][0]
max_pooling2d_370	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_478[0][0]
max_pooling2d_371	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_479[0][0]
max_pooling2d_372	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_480[0][0]
max_pooling2d_379	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_493[0][0]

Table G.4 *model_inception_all_depths_inception_complic_vin_1*: Parameter details (continued)

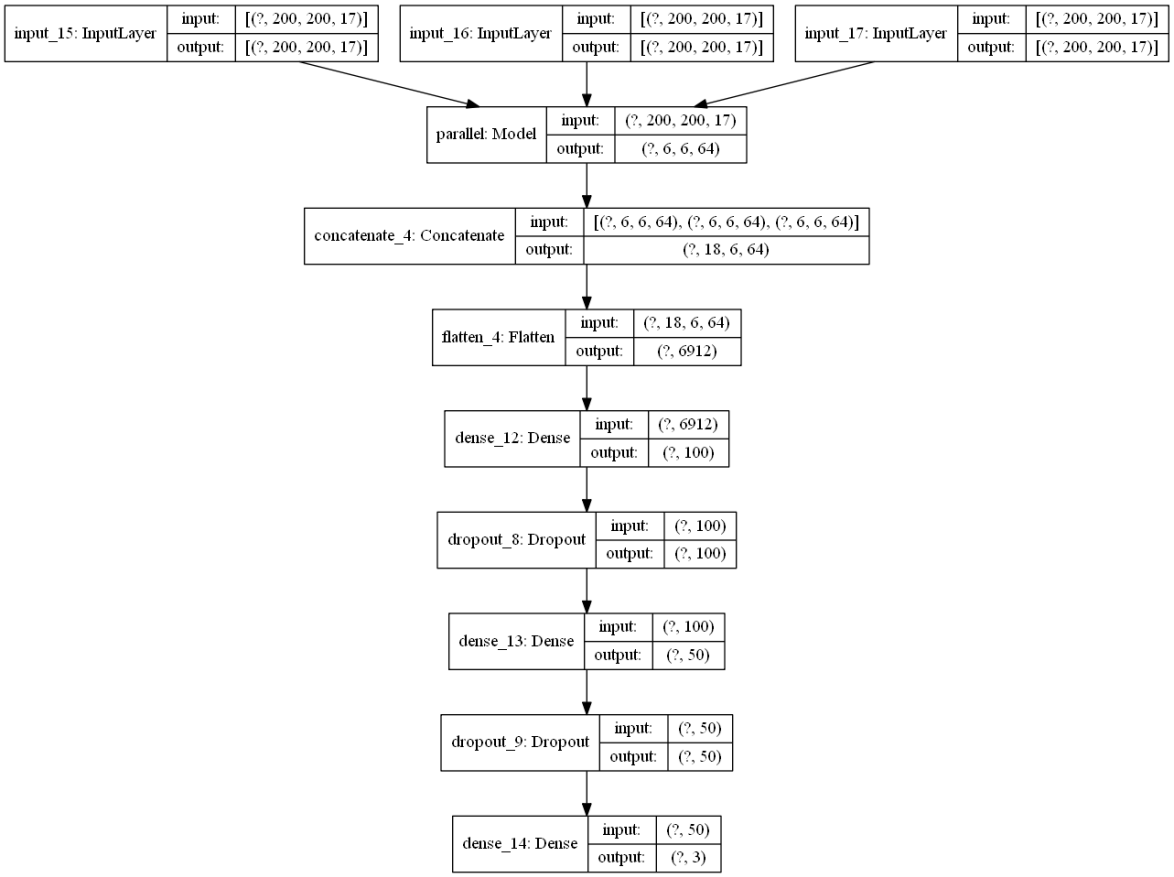
Layer	(type)	Output Shape	Number of Parameters	Connected
max_pooling2d_380	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_494[0][0]
max_pooling2d_381	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_495[0][0]
max_pooling2d_382	(MaxPooling2D	(None, 12, 12, 16)	0	conv2d_496[0][0]
concatenate_78	(Concatenate)	(None, 12, 12, 64)	0	max_pooling2d_359[0][0]
	max_pooling2d_360[0][0]			
	max_pooling2d_361[0][0]			
	max_pooling2d_362[0][0]			
concatenate_82	(Concatenate)	(None, 12, 12, 64)	0	max_pooling2d_369[0][0]
	max_pooling2d_370[0][0]			
	max_pooling2d_371[0][0]			
	max_pooling2d_372[0][0]			
concatenate_86	(Concatenate)	(None, 12, 12, 64)	0	max_pooling2d_379[0][0]
	max_pooling2d_380[0][0]			
	max_pooling2d_381[0][0]			
	max_pooling2d_382[0][0]			
conv2d_465	(Conv2D)	(None, 12, 12, 16)	9232	concatenate_78[0][0]
conv2d_466	(Conv2D)	(None, 12, 12, 16)	25616	concatenate_78[0][0]
conv2d_467	(Conv2D)	(None, 12, 12, 16)	50192	concatenate_78[0][0]
conv2d_468	(Conv2D)	(None, 12, 12, 16)	82960	concatenate_78[0][0]
conv2d_481	(Conv2D)	(None, 12, 12, 16)	9232	concatenate_82[0][0]
conv2d_482	(Conv2D)	(None, 12, 12, 16)	25616	concatenate_82[0][0]
conv2d_483	(Conv2D)	(None, 12, 12, 16)	50192	concatenate_82[0][0]
conv2d_484	(Conv2D)	(None, 12, 12, 16)	82960	concatenate_82[0][0]
conv2d_497	(Conv2D)	(None, 12, 12, 16)	9232	concatenate_86[0][0]
conv2d_498	(Conv2D)	(None, 12, 12, 16)	25616	concatenate_86[0][0]
conv2d_499	(Conv2D)	(None, 12, 12, 16)	50192	concatenate_86[0][0]
conv2d_500	(Conv2D)	(None, 12, 12, 16)	82960	concatenate_86[0][0]
concatenate_79	(Concatenate)	(None, 12, 12, 64)	0	conv2d_465[0][0]
	conv2d_466[0][0]			
	conv2d_467[0][0]			
	conv2d_468[0][0]			
concatenate_83	(Concatenate)	(None, 12, 12, 64)	0	conv2d_481[0][0]
	conv2d_482[0][0]			
	conv2d_483[0][0]			
	conv2d_484[0][0]			
concatenate_87	(Concatenate)	(None, 12, 12, 64)	0	conv2d_497[0][0]
	conv2d_498[0][0]			
	conv2d_499[0][0]			
	conv2d_500[0][0]			
add_37	(Add)	(None, 12, 12, 64)	0	concatenate_78[0][0]
	concatenate_79[0][0]			
add_39	(Add)	(None, 12, 12, 64)	0	concatenate_82[0][0]
	concatenate_83[0][0]			
add_41	(Add)	(None, 12, 12, 64)	0	concatenate_86[0][0]
	concatenate_87[0][0]			
max_pooling2d_363	(MaxPooling2D	(None, 6, 6, 64)	0	add_37[0][0]
max_pooling2d_373	(MaxPooling2D	(None, 6, 6, 64)	0	add_39[0][0]
max_pooling2d_383	(MaxPooling2D	(None, 6, 6, 64)	0	add_41[0][0]
concatenate_88	(Concatenate)	(None, 18, 6, 64)	0	max_pooling2d_363[0][0]
	max_pooling2d_373[0][0]			
	max_pooling2d_383[0][0]			
flatten_13	(Flatten)	(None, 6912)	0	concatenate_88[0][0]
dense_39	(Dense)	(None, 100)	691300	flatten_13[0][0]
dropout_26	(Dropout)	(None, 100)	0	dense_39[0][0]
dense_40	(Dense)	(None, 50)	5050	dropout_26[0][0]
dropout_27	(Dropout)	(None, 50)	0	dense_40[0][0]
dense_41	(Dense)	(None, 3)	153	dropout_27[0][0]
Total params	1,645,655			
Trainable params	1,645,655			

Appendix H

Towers with same parameters for each projection

H.1 *Model_vin_4*

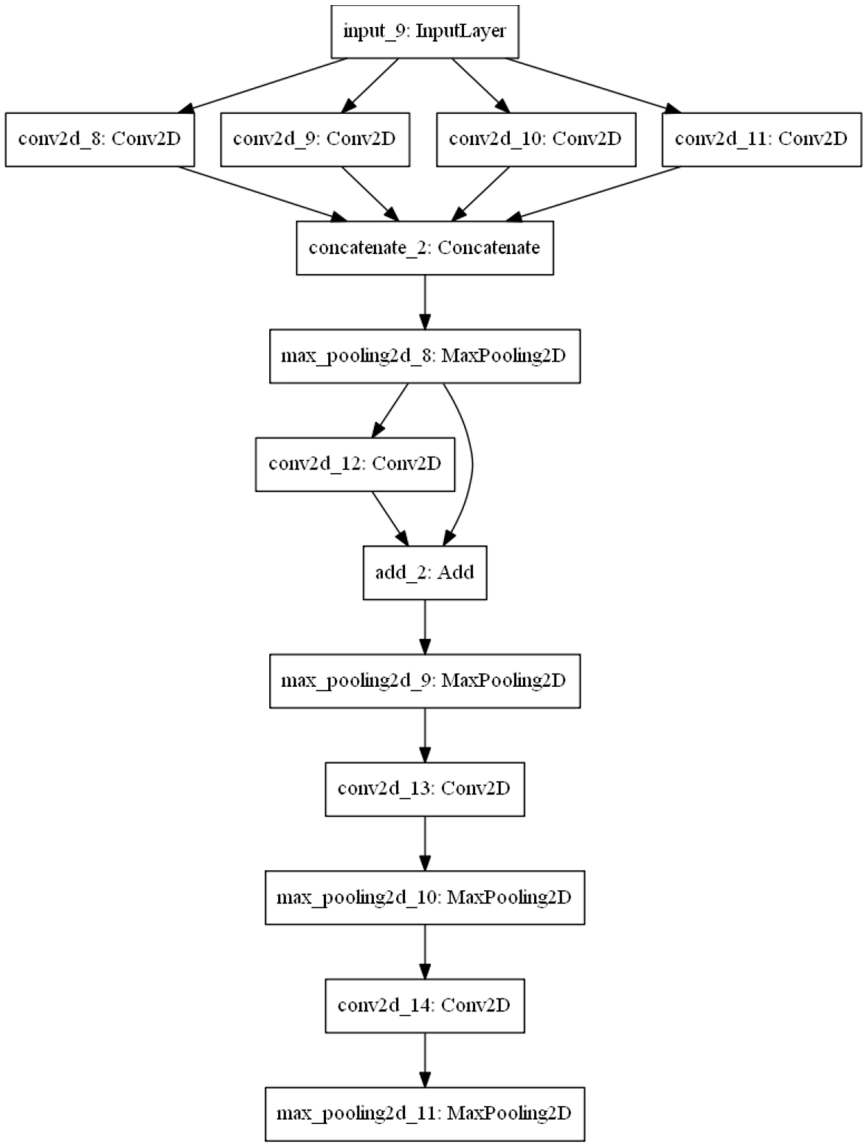
Figure H.1: *model_vin_4* Architecthture.



Total params: 766,103 **Trainable params:** 766,103

H.2 *Model_inception_vin_1*

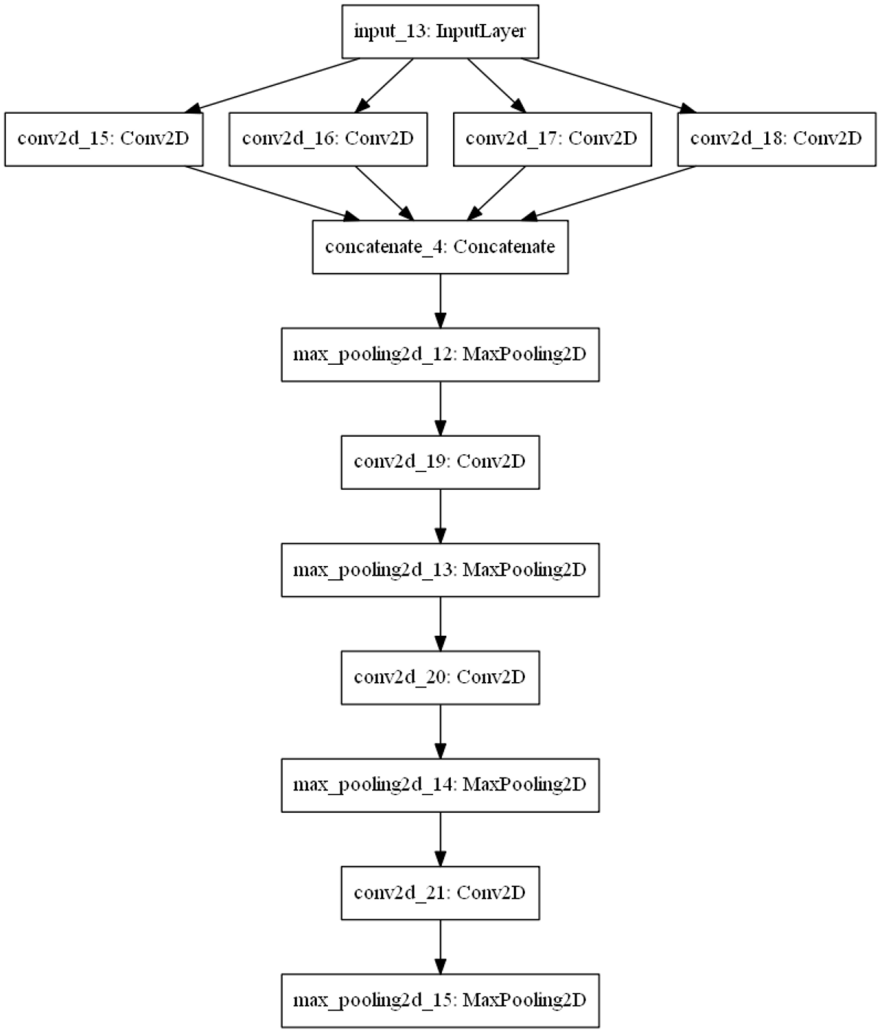
Figure H.2: *model_inception_vin_1* Architecthture.



Total params: 783,511 **Trainable params:** 783,511

H.3 *Model_inception_with_out_addition_vin_1*

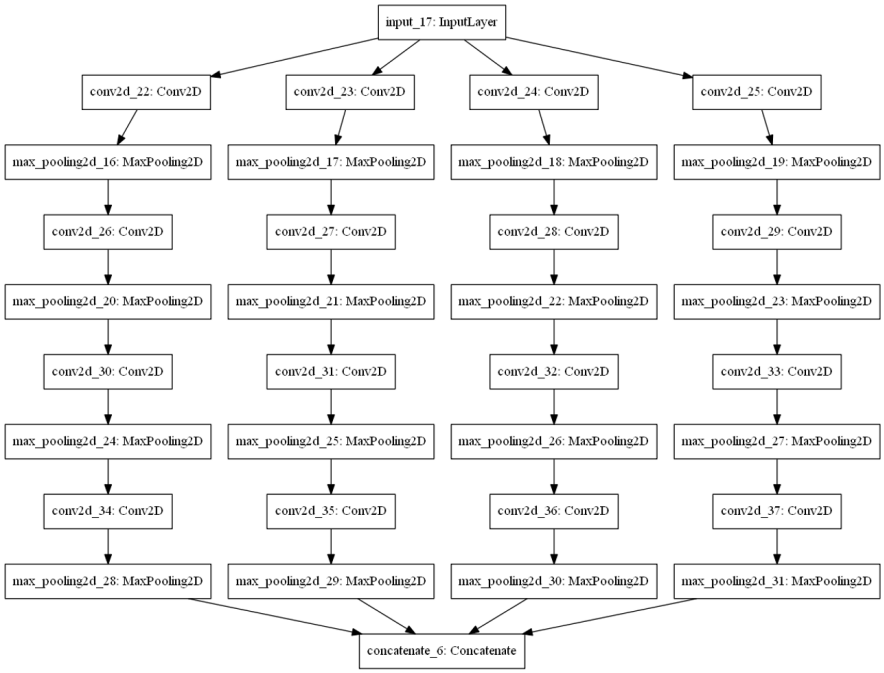
Figure H.3: *model_inception_with_out_addition_vin_1* Architecthture.



Total params: 783,511 **Trainable params:** 783,511

H.4 *Model_inception_all_depths_inception_vin_1*: d1=4, d2=4, d3=8, d4=8

Figure H.4: *model_inception_all_depths_inception_vin_1* Architecthture.



Total params: 380,519 **Trainable params:** 380,519

H.5 *Model_inception_all_depths_min_max_inception_vin_1*: d1=4, d2=4, d3=8, d4=8

Figure H.5: *model_inception_all_depths_min_max_inception_vin_1* Architecthture.

Total params: 226,087 **Trainable params:** 226,087

H.6 *Model_inception_all_depths_inception_complic_vin_1*: d1=4, d3=8

Figure H.6: *model_inception_all_depths_inception_complic_vin_1* Architecthture.

Total params: 435,623 **Trainable params:** 435,623

H.7 *Model_inception_all_depths_min_max_inception_complic_vin_1*

Figure H.7: *model_inception_all_depths_min_max_inception_complic_vin_1* Architecthture.

Total params: 226,087 **Trainable params:** 226,087

Appendix I

Level 2 data selection

To elaborate the data selections, from the Table. I.1 just take the number of gene (primary structure) and PDBs details of 21- amino acid. The removed/ cleaned data consists of 2108 PDB_{SITE}s belongs to 138 genes. In approach 2 of Level 2, from the 2108 PDBs, 1574 PDB_{SITE}s belong to 105 genes were used for training, and the remaining 33 genes and their corresponding 545 PDB_{SITE}s were used for testing as shown in Table. I.1. As mentioned in subsection 2.3.2 , the training set's 1574, 3D coordinate feature set was converted into 24, 2D feature sets ($1574 \times 24 = 37776$). Thus, 37776 with $128 \times 128 \times 21$ feature maps, were used to train the deep learning model. V88 gene PDB details are frozen; then used to train the models reported in this Table. I.1 and evaluations as mentioned in subsection 2.3.2 .

Table I.1: Training and testing dataset selection from Tier 1.

Classes of Tier 1	Data purpose	SITE (thresh81)		preprocessed		cleaned preprocessed PDBs	
		Number of Genes	Number of PDBs	Number of Genes	Number of PDBs	21 amino acids (21-channels)	20 amino acids (17-channels)
ONGO	train	N/A	N/A	32	590	589	588
	test	N/A	N/A	13	448	445	445
	Overall	49	1121	45	1035	1031	1030
TSG	train	N/A	N/A	50	639	632	632
	test	N/A	N/A	14	79	79	79
	Overall	65	751	64	713	706	706
Fusion	train	N/A	N/A	23	403	353	353
	test	N/A	N/A	6	18	18	18
	Overall	30	425	29	421	371	371
All	train	N/A	N/A	105	1632	1574	1573
	test	N/A	N/A	33	545	542	542
	Overall	144	2212	138	2169	2108	2107

Problematic (contain Overlapping PDBs) primary structures are evaluated separately, not in neither train nor test nor validation. Those primary structures' (are reported in subsection 1.2.1 "Refining the PDB data files" Table. 1.1) and their corresponding PDBs (has SITE information and

primary structural length¿81) are evaluated separately. Thus, the column "SITE (thresh > 81)" only reporting overall number of gene and PDB, the train and test are Not-Applicable (N/A).

Some the dataset numbers train, test, overall and clean different, due to these five reasons.

1. Training and Testing set separation is based on Primary structural separation, thus some Training and Testing PDBs has overlapping of eight PDBs. Here, three of them belongs to ONGO and five of them belong to TSG.

2. In ONGO in training class 590 PDBs satisfied the condition but for the deep learning model 589 PDBs are used; since PDB file "4MDQ" not able produce results through $MSMS$ tool.

3. In ONGO in testing class 448 PDBs satisfied the condition but the deep learning model classified only 445 PDBs. Among these three left the PDBs "3GT8" and "3LZB" has unknown amino acids so those left and PDB "721P" has no single $C\alpha$ surface atom by the threshold condition as mentioned in section *Appendix* A .

4. In TSG "2H26" has unknown amino acids. PDBs "5C0B" and "5C0C" have size issues as their corresponding highest eigen vector's (x size) is 143 and 147 accordingly the feature size for 1 of the 8th quadrants is 128. And 4 PDBs are in overlap, altogether 7 PDBs.

5. In Fusion, the 50 PDBs belongs to overlapping with other classes.

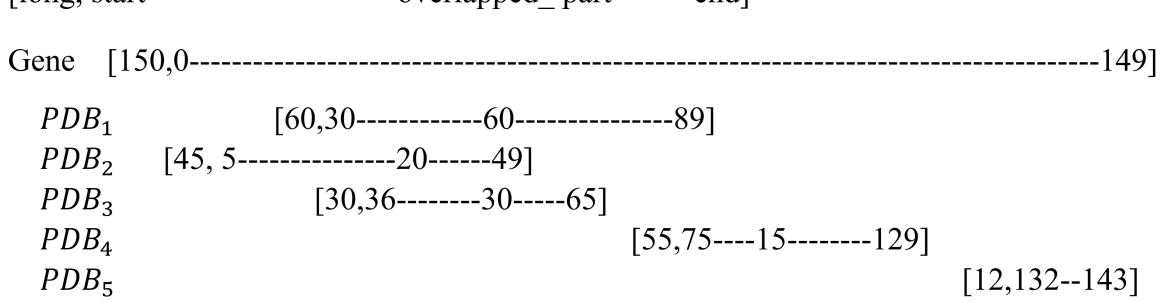
Appendix J

Hypothetical Example of Methods for Calculating Fractional Weights of Functional Primary Structures based on the PDBs' Overlapping Sequence Information

J.1 Hypothetical example of $Method_1$

It finds the longest overlapped PDB, and then use that to give the fractions for the rest of the PDBs.

Figure J.1: $Method_1$ hypothetical example



This algorithm first chooses the PDB_1 because that is the longest among all,Then calculate the fraction of the rest of the PDBs overlapped with that (PDB_1)

✓ PDB_1:1

✓ PDB_2:$\frac{20}{60} = 0.333$

✓ PDB_3:$\frac{30}{60} = 0.5$

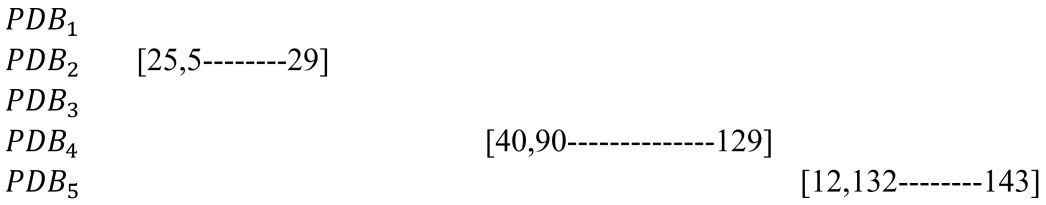
✓ PDB_4:$\frac{15}{60} = 0.25$

✓ PDB_5:0

with these fractions the length considered also saved to calculate the overall probability; here the length is PDB_1's, that is 60.

Then calculate the remaining length after removing the overlapped part. This will lead to go to PDB_4 ,because that has the highest length 40.

Figure J.2: $Method_1$ hypothetical example after first calculation



Then follow the same procedure to calculate the fractions

✓ PDB_1:0

✓ PDB_2:0

✓ PDB_3:0

✓ PDB_4:1

✓ PDB_5:0

With these fractions the length considered also saved to calculate the overall probability, after done every calculation of two steps:

$$\frac{60(P[PDB_1] + 0.33P[PDB_2] + 0.5P[PDB_3] + 0.25P[PDB_4])}{(60 + 40 + ...)} + \frac{40(P[PDB_4])}{(60 + 40 + ...)}$$

Fully gone through this example final equation of method ones fractional weights for the PDB group look like,

$$\frac{weight\ of\ PDB_1\ fract\ prob + 40(P[PDB_4]) + 20(P[PDB_2]) + 12(P[PDB_5])}{(60+40+25+12)}$$

where,

$$weight\ of\ PDB_1\ fract\ prob = \Bigg(60(P[PDB_1] + 0.33P[PDB_2] + 0.5P[PDB_3] + 0.25P[PDB_4]) \Bigg)$$

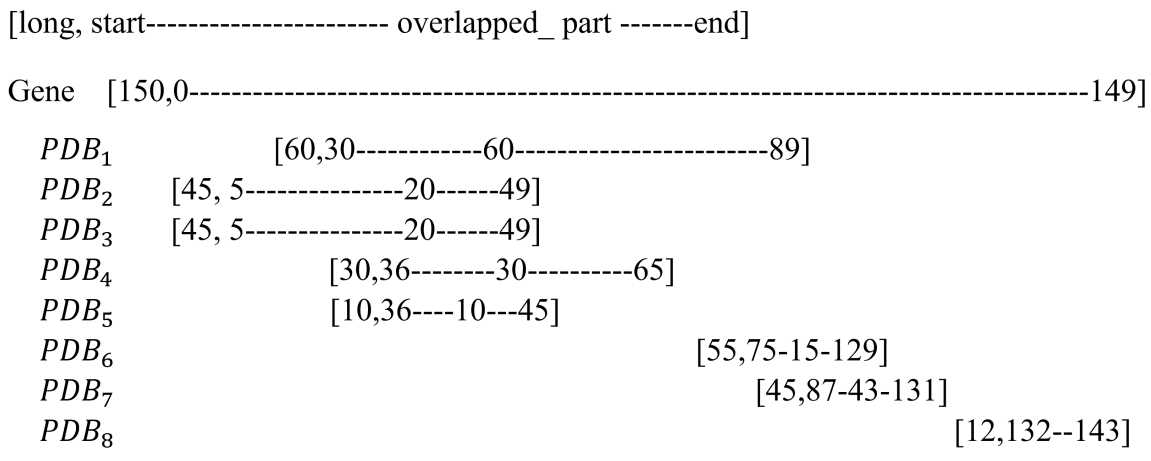
J.2 Hypothetical example of $Method_2$ and $Method_3$

$Method_2$ and $Method_3$ is mainly focused on covering the maximum gene as possible with the PDB_{id} of that gene's group, but how we choose covering the maximum gene with those PDBs such as,

✓ $Method_2$ uses non-overlapping PDBs to cover the gene

✓ $Method_3$ uses overlapping PDBs to cover the gene

Figure J.3: $Method_2$'s or $Method_3$'s hypothetical example



Both algorithms use the unique starting point and unique ending point for calculation. $Method_2$ uses non-overlapping PDB's to cover the gene

✓ cov (0) = 0

✓ cov (1) = 45; PDB_2 and PDB_3 (finally we use this information to give the vote for that part by the probabilities)

✓ cov (2) = 60; PDB_1

✓ cov (3) = 60; PDB_1 when checking PDB_4

✓ cov (4) = 60; PDB_1 when checking PDB_5

✓ cov (5) = 55+45 = 100; ((PDB_2 and PDB_3) from cov (1), and PDB_6)

✓ cov (6) = 55+45 = 100; ((PDB_2 and PDB_3), and PDB_6) when checking PDB_7

✓ cov (7) = 112; ((PDB_2 and PDB_3), and PDB_6) from cov (6), and PDB_8)

$$Weightage\ Method_2 = \left(\frac{45(0.5P[PDB_2] + 0.5P[PDB_3])}{112} + \frac{55P[PDB_6]}{112} + \frac{12P[PDB_8]}{112} \right)$$

$Method_3$ uses overlapping PDB's to cover the gene, since the overlapping allowed and cover the primary structure with small number as possible, to avoid more conditions while running the dynamic programming approach, implementation of the algorithm is done with another dynamic programming approach and then go through the final outcome of the dynamic program to find the final solution of the algorithm (since the overlapping is allowed both of these proposed will return same outcome). thus, the implemented algorithm follows the dynamic programming as

Figure J.4: Implemented algorithm

$$cover\,(n+1) = max \begin{cases} cover\,(n) & \\ cover\,(n-1) & +length\,(next\ PDB_{SITE}) \\ & -\ overlaped\_length_{with\ previous\ one} \end{cases}$$

where the $overlaped\_length_{with\ previous\ one}$ is the overlapping length between the $next\,PDB_{SITE}$, and previous PDB_{SITE}.

✓ cov (0) = 0

✓ cov (1) = 45;(PDB_2 and PDB_3) (finally we use this information to give the vote for that part by the probabilities)

✓ cov (2) = 85; ((PDB_2 and PDB_3) and PDB_1)

✓ cov (3) = 85; ((PDB_2 and PDB_3) and PDB_1), when checking PDB_4

✓ cov (4) = 85;((PDB_2 and PDB_3) and PDB_1), when checking PDB_5

✓ cov (5) = 125; ((PDB_2 and PDB_3) and PDB_1 from cov (4), and PDB_6)

✓ cov (6) = 127; ((PDB_2 and PDB_3), PDB_1, PDB_6 from cov (5) and PDB_7)

✓ cov (7) = 139; ((PDB_2 and PDB_3), PDB_1, PDB_6, PDB_7 from cov (6) and PDB_8)

It is clearly seen the PDB_6, is already covered by PDB_1, and PDB_7. All these kinds of PDB_{SITE}s fell in the last group (in this example cov (7) is the last group). So, in the implementation go through

the last group, checked the whether the PDB_{SITE} is redundant, if redundant PDB_{SITE} found then removed.

So final group of the implementation cov is,

✓ cov [implementation final] = 139; ((PDB_2 and PDB_3), PDB_1, PDB_7 and PDB_8)

Let us check the original proposed dynamic programming approach, the difference starts from

✓ cov (6) = 127;((PDB_2 and PDB_3) and PDB_1 from cov (4), and PDB_7)

And then updated in the following groups, in this example only one group is updated as,

✓ cov (7) = 139; ((PDB_2 and PDB_3), PDB_1, PDB_7 from cov (6) and PDB_8)

Either in proposed or implemented way the outcome is same. Thus, the final $Method_3$ fractional wight as

$$Method3_{fin} = \left(\frac{\left(45 - \frac{20}{2}\right)(0.5\ P[PDB_2] + \ 0.5\ P[PDB_3]) + \left(60 - \frac{20}{2} - \frac{2}{2}\right) P[PDB_1]}{139} + \frac{\left(45 - \frac{2}{2}\right)\ P[PDB_7]\ +\ 12\ P[PDB_8]}{139} \right)$$

Appendix K

Complexity of the Methods Classification Primary Structural Functional Detection

To calculate the complexity, for hypothetical model assume a primary structure contain maximum "n" number of PDBs. Since all these methods use same kind of data structure to do the calculation. The complexity of the common calculation is given below.

✓ Complexity of running time is given in Font color red

✓ Complexity of memory is given in *Font color green

K.1 Complexity of $Method_1$

Figure K.1 $Method_1$ (PseudoCode) representation 2

1: $high_{length}$=1;{intailised with non zero number to start; and $un_{overlap}$ group contain all the PDBs overlapped with the primary structure}
2: **while** $high_{length} > 0$;{$\mathcal{O}(n)$ or $\Omega(1)$} **do**
3: $high_{length}$, $sel\_PDB_{id} = select\_pdb\_with\_high_{length}(un_{overlap})$;{$\mathcal{O}(n)$ or $*\mathcal{O}(n)$}
4: **end while**
5: **if** $high_{length} == 0$;{$\mathcal{O}(1)$} **then**
6: **break**;{$\mathcal{O}(1)$}
7: **else**
8: $un_{overlap}, over_{sat} = method_1\_overlap_{sel}(sel\_PDB_{id}, un_{overlap}, high_{length})$; {$\mathcal{O}(n)$ or $*\mathcal{O}(n)$}
9: **end if**

In pseudo code representation 2 for $Method_1$

line 2: if none of the PDBs' primary structure is overlapped with each then the while loop iterates "n" times thus upper bound at $\mathcal{O}(n)$. If highest length primary structure PDB is overlapped with all the rest of the PDBs then while loop end in first iteration, so it is lower bound is $\Omega(1)$.

line 3: function "$select\_pdb\_with\_high_{length}$" finds the PDBs' primary structures highest length among them it goes through the unoverllaped PDBs($un_{overlap}$: mean it not overlapped part with the previous highest length PDB's primary structure/ previous sel_PDB_{id}). It need to go through all in first iteration($\mathcal{O}(n)$), and space to store the length thus higher bound is $*\mathcal{O}(n)$. And the number of non-overlaps PDBs is reducing over each run.

line 8: "$method_1\_overlap_{sel}$" function find the fractions of overlapped primary structures with the highest length and update the information in "$over_{sat}$", thus it need to go through the all the overlapped PDBs to find the fractions thus higher bound(if all PDBs overlapped with the sel_PDB_{id}) is $\mathcal{O}(n)$.

So, the time complexity of $Method_1$ is $\mathcal{O}(n^2)$ or $\Omega(n)$ and the space complexity is $*\mathcal{O}(n)$.

K.2 Complexity of $Method_2$ and $Method_3$

In pseudo code for $Method_2$ and $Method_3$ Starting positions of PDB_{id}s (for the group) are sorted in ascending order: that only taking sorting complexity like $\mathcal{O}(n \log n)$ can be achieved by quick sort, but here two or more PDB_{id}s has same start end position both it considered as one for the calculation thus, PDBs primary structures with same starting position and end position calculation need $\mathcal{O}(n^2)$ time complexity, and space $*\mathcal{O}(n)$ complexity.

Both $Method_2$ and $Method_3$, need doubly nested loop structure to perform the bottom up dynamic programming approach, thus it need $\Theta(n^2)$ time complexity. Since both needed to store the indexes of each values of steps to go back and find out so the space complexity is always upper bounded by $*\mathcal{O}(n^2)$.

But after the calculation both needed to check the overlapped portion; that needs $\mathcal{O}(n^3)$ time complexity; since it have to go through each groups created by bottom up fashion and go through all PDBs' primary structures to assign weights depends on do these PDBs' primary structures overlap or not(that checking need another loop). Space complexity remains same. Thus $Method_2$'s and $Method_3$'s time complexity is $\mathcal{O}(n^3)$, and the space complexity is $*\mathcal{O}(n^2)$.

Appendix L

Tier 1 and Tier 2 results annotations

This appendix covers the Tier 1 and Tier 2 overall classification annotation presented in the link, `https://drive.google.com/drive/folders/1YCuMVPhAy7tIdGFiebInZ_6-kBAH1kMM?usp=sharing`.

Table L.1: Annotation of Tier 2's Not-annotated genes

Probabilities of direct pickles				
Uni gene ID	OG	TSG	Fusion	Class
1500	0.46	0.28	0.26	ONGO_TSG_Fusion
1630	0.82	0.17	0.01	ONGO
5579	0.95	0.04	0.01	ONGO
6000	0.23	0.48	0.29	TSG_Fusion
Probabilities from Ensemble				
Uni gene ID	OG	TSG	Fusion	Class
1558	0.89	0.06	0.05	ONGO
2042	0.72	0.19	0.09	ONGO
2045	0.44	0.5	0.06	ONGO_TSG
2316	0.65	0.29	0.06	ONGO_TSG
286	0.23	0.59	0.18	TSG
30835	0.42	0.5	0.08	ONGO_TSG
3685	0.31	0.66	0.04	ONGO_TSG

To show the results (in the link) predicted by the proposed model (BIR) and methods. The Table. L.1 presents the results of Not-annotated genes of Tier 2 (from COSMIC V91 [28] data) using V88 Approach 1's best BIR models. Annotations of these predicted structures' given probability can be used by users to assign classes depends on their requirement.

For an example, take the gene-id 3685 (it's gene symbol is $ITGAV$) if the user just need the mainly contributing class then the GeneID can be showing higher probability to TSG class as 0.66 (which is > 0.5), on the other hand if the user need multi classification then the ONGO can be included with TSG; since the ONGO probability is 0.31 (which > 0.25).

Thus, here the annotation is for multi class classification. And the classes are annotated as shown in the

Figure L.1 PseudoCode for annotating classes

1: **if** $P_{ONGO} >= min_{thr_{prob}} and P_{TSG} >= min_{thr_{prob}} and P_{Fusion} >= min_{thr_{prob}}$: **then**
2: **return** "ONGO_TSG_Fusion"
3: **else if** $P_{ONGO} >= min_{thr_{prob}} and P_{TSG} >= min_{thr_{prob}} and P_{Fusion} < min_{thr_{prob}}$: **then**
4: **return** "ONGO_TSG"
5: **else if** $P_{ONGO} >= min_{thr_{prob}} and P_{TSG} < min_{thr_{prob}} and P_{Fusion} >= min_{thr_{prob}}$: **then**
6: **return** "ONGO_Fusion"
7: **else if** $P_{ONGO} < min_{thr_{prob}} and P_{TSG} >= min_{thr_{prob}} and P_{Fusion} >= min_{thr_{prob}}$: **then**
8: **return** "TSG_Fusion"
9: **else if** $P_{ONGO} >= min_{thr_{prob}} \times 2$: **then**
10: **return** "ONGO"
11: **else if** $P_{TSG} >= min_{thr_{prob}} \times 2$: **then**
12: **return** "TSG"
13: **else if** $P_{Fusion} >= min_{thr_{prob}} \times 2$: **then**
14: **return** "Fusion"
15: **else**
16: **Raise class not defined**
17: **end if**

pseudocode Fig. L.1. There

✓ $min_{thr_{prob}}$: Minimum threshold probability (assigned as 0.25).

✓ P_{ONGO} : Predicted probability of ONGO class (**E.g.:**for GeneID 3685 probability is 0.31).

✓ P_{TSG} : Predicted probability of TSG class (**E.g.:** for GeneID 3685 probability is 0.66).

✓ P_{Fusion} : Predicted probability of Fusion class (**E.g.:** for GeneID 3685 probability is 0.04).

The probabilities should be added to 1.0. Since the probabilities are rounded to 2^{nd} digit, due to three class sometimes these rounds up end with 1.01 or 0.99. From the example presented here if the probabilities are rounded to 3^{rd} digit 0.306, 0.656 and 0.038; then the addition is end up with 1.0.