Evaluation and Ranking of Protein Docking Models

by 3D Convolutional Neural Networks

by

Sukejna Valjevac

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

September 3, 2020

Evaluation and Ranking of Protein Docking Models by 3D Convolutional Neural Networks

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Sukejna Valjevac

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Attila Gürsoy (Advisor)

_____

Prof. Özlem Keskin [Co-Advisor]

_____

Prof. Deniz Yüret

_____

Assoc. Prof. Nurcan Tunçbağ

Date: _____

ABSTRACT

Evaluation and Ranking of Protein Docking Models by 3D Convolutional Neural Networks

Sukejna Valjevac

Master of Science in Computer Science and Engineering

September 3, 2020

Biological processes depend on protein-protein interactions that occur through protein-protein interfaces. Identification of protein-protein interface regions is crucial to understand mechanisms of protein binding and predict new interactions. Therefore, it is critical to be able to determine protein-protein interfaces easily and reliably. To extract a protein interface, protein-protein complex structure is needed. Since even determining a single protein complex structure experimentally takes time and effort; it is promising to see that computational docking tools can provide thousands of possible complex structures, called decoys, in a short time. A challenge that comes with computational methods is to be able to discriminate near-native decoys among thousands proposed. Multiple methods have been developed for scoring and ranking the decoys to identify the biologically relevant ones that can be used in further biological research. In this thesis, we improved a three-dimensional convolutional neural network-based decoys` scoring and ranking approach called DeepInterface. We compared multiple convolutional neural network architectures and searched the hyperparameters` space to find the best performing DeepInterface model that resulted in having the VGG16 resembling architecture. We built positive datasets from the complexes stored in the Protein Data Bank, and negative datasets from the incorrect decoys stored in the PPI4DOCK and DOCKGROUND docking databases. We showed that the model we suggested can discriminate positive interfaces, similar to ones stored in the PDB, from the incorrect ones, according to the CAPRI criteria, with approximately 81% accuracy. We also compared our models with other decoys` scoring/ranking tools including IRAD and DOVE using ZDOCK docking benchmark 4.0 decoys and showed that our models are competitive. This method could be used to reduce the computational cost of the protein-protein interactions` predictions.

ÖZETÇE

Modellenmiş Protein-Protein Etkileşim Arayüzlerinin 3 Boyutlu Evrişimli Sinir Ağları ile Değerlendirilmesi ve Sıralanması

Sukejna Valjevac

Bilgisayar Bilimleri ve Mühendisliği, Yüksek Lisans

3 Eylül 2020

Biyolojik süreçler, protein-protein arayüzleri aracılığıyla meydana gelen protein etkileşimlerine bağlıdır. Protein-protein arayüzlerinin kolay ve güvenilir bir şekilde belirlenebilmesi, protein bağlanma mekanizmalarını anlamak ve yeni etkileşimleri tahmin etmek için çok önemlidir. Bir arayüz çıkarabilmek için, protein-protein kompleks yapısına ihtiyaç vardır. Tek bir protein kompleksinin yapısını belirlemek bile deneysel olarak uzun bir zaman ve büyük bir çaba gerektirdiğinden; hesaplamalı yöntemlerin binlerce olası protein kompleksi yapısını tahmin edebileceğini görmek umut vericidir. Fakat, hesaplamalı yöntemlerde karşımıza çıkan en büyük zorluk, yöntem tarafından önerilen binlerce protein kompleksi arasından biyolojik olanlarını ayırt edebilmektir. Günümüze kadar, biyolojik protein modellerini belirlemek için, dekoyları puanlayan ve sıralayan birçok yöntem geliştirilmiştir. Bu tezde, DeepInterface adı verilen üç boyutlu evrişimli sinir ağı tabanlı dekoy puanlama ve sıralama yaklaşımını geliştirdik. En iyi performans gösteren DeepInterface modelini bulmak için birden çok evrişimli sinir ağı mimarisini karşılaştırdık ve hiperparametre alanlarını taradık. Sonuç olarak, VGG16'nın mimarisine benzeyen yeni bir model ortaya çıkarttık. Protein Veri Bankasında (PDB) depolanan komplekslerden pozitif veri kümelerini ve PPI4DOCK ve DOCKGROUND veritabanlarında depolanan yanlış dekoylardan negatif veri kümelerini oluşturduk. Önerdiğimiz modelin, PDB'de depolananlara benzer pozitif protein arayüzlerini, yaklaşık %81 doğrulukla, CAPRI kriterlerine göre yanlış negatif protein arayüzlerinden ayırt edebildiğini gösterdik. Modellerimizi ZDOCK benchmark 4.0 üzerinden IRAD ve DOVE gibi diğer dekoyların puanlama / sıralama araçlarıyla da karşılaştırdık ve modellerimizin bu araçlarla yarışabilir nitelikte olduğunu gösterdik. Kullandığımız bu yöntem, protein-protein etkileşimlerini tahmin eden programların hesaplama maliyetini azaltmak için kullanılabilir.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CAPRI	Critical Assessment of Predicted Interactions
CNN	Convolutional Neural Network
Fnat	Fraction of native contacts
iRMSD	interface Root-Mean-Square Deviation
lRMSD	ligand Root-Mean-Square Deviation
NPV	Negative Predictive Value
PDB	Protein Data Bank
PPI	Protein-Protein Interaction
PRISM	Protein Interactions by Structural Matching
ReLU	Rectified Linear Unit
RMSD	Root-Mean-Square Deviation

Chapter 1:
INTRODUCTION

Protein-protein complex structures are crucial to understand mechanisms of protein binding and identify protein-protein interfaces. Knowledge of protein-protein interfaces aids characterization of cellular signaling pathways, and structure-based drug design. Therefore, it is important to be able to determine protein-protein interfaces easily and reliably. Since doing this experimentally takes time and effort, it is promising to see that computational docking tools can provide thousands of possible complex structures, decoys, in a short time. Multiple methods have been developed for scoring and ranking decoys to identify the reliable near-native complexes that can be used in further biological research.

In this thesis, we improved a 3D convolutional neural network-based decoys` scoring and ranking approach called DeepInterface. We compared multiple CNN architectures and searched the hyperparameters` space to find the best performing DeepInterface models. We showed that the models we suggested can discriminate positive interfaces from the incorrect ones with approximately 81% accuracy. We also compared our models with other decoys` scoring/ranking tools and showed that our models are competitive.

In Chapter 2, we give some background information and related work for the deep learning models we suggest.

In chapter 3, we discuss the datasets and training of the DeepInterface model. First, we give a detailed explanation of input data preparation and the datasets used for training, validation and testing of the model. Then, we discuss the search performed to find the best matching model architecture. Finally, we describe the hyperparameters` search and training procedure of the models.

In Chapter 4, we offer our results and performance analysis. First, we present the evaluation of the models we tried on the datasets and explain the output scores. Then, we assess the performance analysis of the models when used for ranking decoys produced by GRAMM-X for positive interfaces from Test datasets. And lastly, we compare our models with other decoys` ranking/scoring tools.

Finally, we wrap up this work with the conclusion giving a summary of the work done and results obtained.

Chapter 2:
LITERATURE REVIEW

2.1 Protein-Protein Interfaces

Proteins usually interact with each other to function. Nearly all processes in living organisms depend on the protein-protein interactions. Protein interactions occur through protein interfaces located on their surfaces [1] and consisting of interacting residues.

There are multiple methods to extract protein-protein interfaces and some of them are accessible surface areas, Voronoi diagrams, and atomic distances. When atomic distances are used, two types of residues are said to comprise an interface, interacting residues and nearby residues. Residues belonging to two partner proteins are defined as interacting if the distance between them (any atom) is smaller than the 0.5 Å plus the sum of their van der Waals radii.

The residues that are surrounding the interacting residue located on the same protein and having their Cα's within 6 Å radius are called nearby residues [2].

Interfaces not necessarily consist only of continuous residues` segments, they may also have single residues included from any of the interacting proteins [3, 4]. Several interface properties help separate interface residues from the other surface residues. These properties at the first place include residues` physicochemical properties like the methylation and phosphorylation status, nonpolar/polar nature, number of H-bond donors and H-bond acceptors. At the same time, presence of hotspots, salt bridges, and different types of secondary structures in the interface area additionally help identification of the interface [1, 5-7]. There are multiple studies using these properties at the beginning of the interface prediction process [2, 8, 9, 10], or at the end, for scoring/ranking the predicted complexes [11, 12].

2.2 Prediction of Protein-Protein Interfaces

To define the protein-protein interface the knowledge of its complex 3D structure is required. 3D structure can be obtained experimentally in several ways, and in addition to classical methods, advances in powerful techniques such as cryo-electron microscopy (cryo-EM), help multiple protein complex structures to be determined every year. However, still many protein-protein complexes do not have needed structural

information. To help efforts done in experimental fields, computational protein-protein interaction prediction and docking approaches have been developed [13].

The docking methods can be divided into two categories, ab initio methods, that bring together two single protein structures not referring to any known protein complex structures, and template-based modelling methods, that use known local [2] or global [14] protein complex structures. Details of the existing ab initio methods vary. Used protein structure representations include voxel-based representation [15] and molecular surface-based representation [16]. In the docking pose search there are multiple choices, the most popular one is the Fast Fourier Transform [17, 18], other successful ones are particle swarm optimization [19], and geometric hashing [20, 16]. To give credit to protein flexibility, protein dynamics simulation [21] and normal mode analysis [22] have been used. Some additional advanced to the classical pairwise docking are peptide-protein docking [23, 24], multiple-chain docking [25-27], docking order prediction [28, 29], docking with disordered proteins [30], and docking modeling for cryo-EM maps [25, 31].

Once docking a protein pair is finished, as a result hundreds of decoys are produced. Not all of them are native-like structures. It is vital to rank the produced decoys and recognize the native-like ones among all.

2.3 Scoring Algorithms

Importance of the decoys` scoring has been recognized by the Critical Assessment of Prediction of Interactions (CAPRI), the docking prediction community-wide experiment [32]. CAPRI has a category concentrated on scoring methods evaluation, which asks participants to identify 10 native-like structures among thousands of decoys that are given by the organizers.

There are multiple methods currently available for scoring protein-protein complex structures based on their different properties including physics-based potentials [21, 33], knowledge-based statistical potentials [34, 35], interface residues evolutionary profiles [36], machine learning methods [37, 38, 39], interface shape [16], interface composition with residues contact preference [12], statistical learning using the native protein-protein interfaces [40],

DeepInterface [41], is the convolutional neural network (CNN) model suggested by Balci at al. for ranking of protein docking models and classifying them into non-native

or native categories. Training and test datasets are composed of protein-protein interface residue types and their atomic coordinates. In this study DeepInterface datasets, model and training are optimized to maximize the accuracy achieved on validation dataset, and the new model performance is measured on the well-known ZDOCK docking benchmark 4.0 [42].

One of the most known docking algorithms, ZDOCK, was developed by Chen et al. [43]. They compared four different scoring functions to find the best function to complement their FFT algorithm [43]. First, grid-based shape complementarity (GSC) scoring function which is commonly preferred by docking methods. According to GSC, the space in which two proteins interact is divided into grids, then clash check is done, and the level at which proteins fill the gaps avoiding clashes is determined. Penalization is done for clashes and rewards are given for tightly packed spaces. Second, pairwise shape complementarity (PSC) scoring function represents an improved form of GSC. PSC also rewards the matched pairs between receptor proteins and ligands. It pairs all ligand atoms inside the radius of the receptor atom plus a value determined by a user. Reward increases with the number of pairs present. PSC works better than GSC [43]. This is not a surprise since naturally two proteins will interact to pair their atoms to lower their complex energy. Third, desolvation is another part of the scoring function. Chen et al. used contact energy to calculate desolvation. Desolvation represents energy which is required to break a water-protein interaction and then form a protein-protein interaction. They used a set of previously calculated interaction energies to fetch desolvation scores. Combining desolvation energy with the PSC led to tremendous improvement when compared to PSC [43]. Fourth, electrostatic energy is the last part of the scoring function. Chen et al. used Coulomb's Law to add electrostatic energy component. They calculated and put in the electric potential energy of partial charges of ligand and receptor atoms. The best accuracy is obtained when PSC, electrostatic and desolvation energies are combined [43].

Pierce and Weng developed a scoring function called ZRANK [44] that includes van der Waals, desolvation, and detailed electrostatics, to rescore ZDOCK docking predictions. Scoring terms weights were optimized on a set of test cases then tested on a set of nonredundant independent cases. ZRANK proved to improve the rate of success over the ZDOCK rankings on a large benchmark.

Later, Pierce and Weng applied the ZRANK scoring algorithm together with the RosettaDock, protein docking program, to refine the scoring of protein docking models [45]. This was done in two stages, first reranking the ZDOCK global docking predictions by applying rigid-body and local side chain refinement using RosettaDock, second choosing the refined model according to the ZRANK [44] score. This algorithm is known as ZRANK2, and it proved to improve the rate of success over the ZRANK [44] rankings on a large benchmark.

Both ZRANK and ZRANK2 scoring functions only contain atom-based terms, which are usually more accurate on high quality structures when compared to residue-based potentials. However, residue-based potentials are better in tolerating the conformational changes, in cases of reranking and rigid body docking. At the same time when parametrization is done it is residue-specific, therefore biophysical interactions that might be missed by the atom-type in atom-based potentials could be captured by residue-based potentials. Vreven et al. combined atom-based potentials from ZRANK [45] with atom-based potential IFACE [46] and five residue-based potentials [47]. Three of residue-based potentials used are developed for protein-protein docking: Tobi and Bahar [48], Lu, Lu, and Skolnick [35], and Glaser, Steinberg, Vakser, and Ben-Tal [49], one of them is a pair potential developed for protein folding by Zhang, Kolinski, and Skolnick [50], and the last one is a non-pairwise potential which is based on interface propensities from Chakrabarti and Janin [51]. Next to these five residue-based potentials a variant of Zhang, Kolinski, and Skolnick [50] that uses a different definition for the interaction site was also used. All these residue-based potentials are different from each other in terms of their parameterization, how their contacts are defined, and the interaction sites number for each residue. Weights for combinations of terms used in the scoring function were optimized on the decoys generated by the ZDOCK [43], and cross-validation of the combinations was done using test cases from the docking benchmark 3.0 [52]. The resulting optimum scoring function combines the ZRANK [45] terms, van der Waals term separated into attractive and repulsive components, and electrostatics (short range and long range), with atom-based statistical potential IFACE [46], and a variant of Zhang, Kolinski, and Skolnick [50] residue-based potential. This function is called IRAD (integration of residue- and atom-based potentials for docking) [47].

2.4 Machine Learning Algorithms

Machine learning is considered as a subset of artificial intelligence which studies algorithms that enhance through experience. Machine learning methods make use of statistical techniques to get inferences from data. These methods proved to work well especially with large datasets.

They have been applied to numerous science fields among which biology takes place as well. Machine learning algorithms are widely used in the computational biology area. This section summarizes the intersection between computational biology and machine learning concepts related to this research.

To better understand models created in this study, the first concept to describe are convolutional neural networks. Convolutional neural networks (CNNs) are a category of deep neural networks. They are mostly used for solving vision challenges. They have multiple advantages over fully connected multilayer perceptrons where all neurons from one layer are connected to all neurons from the next layer. CNNs do not need extensive input data pre-processing. Filters that used to be had-engineered are learned and feature extraction is done automatically. Filters or kernels are applied to all parts of the input and convolution operation is performed; this results in an abstracted feature map. In addition, usage of kernels decreases the number of parameters in the network greatly by assigning one parameter to more one input field while in fully connected networks multiple parameters were assigned to one input field. This leads to CNNs occupying less memory and taking less time for training.

One more advantage is pooling. CNNs include pooling layers in which multiple input values are down sampled to a single value, usually by taking average or maximum of that values. These layers help omit the redundancy in the input which leads to decrease in computational time and increase in robustness against noise. At the same time, by decreasing the input size they lead to decrease in number of parameters, less memory usage and faster training.

Over time different CNN architectures evolved. LeNet is a convolutional neural network architecture originally designed for hand-written digits` recognition by LeCun et al. [53]. It has a simple structure. It consists of three convolutional layers, two of which are followed by a pooling layer, last convolutional layer is followed by fully connected layer. This network started the era of fame for CNNs.

Visual Geometry Group (VGG) created a very deep convolutional neural network architecture that won ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2014. This network and its variations are known as VGG networks [54]. Instead of large receptive field filters, VGG networks use very small receptive field filters of size 3x3 with a stride of 1. In some places 1x1 convolution filters are used as well acting as linear input transformation followed by ReLU [55] unit which makes the decision function extra non-linear and does not change the receptive fields. These small size filters allow VGG networks to have numerous weight layers which adds to better performance. Most used VGG network architectures have eight convolutional layers or more, five maxpool layers, and three fully connected layers in the end [54].

Second winner of ILSVRC 2014 was a deep convolutional neural network proposed by Google research [56]. This network is called Google Net and is 22 weight layers deep. Google Net owes its deep architecture to methods like 1x1 convolution, global average pooling, inception modules, and auxiliary classifiers for training. 1x1 convolutional is used as an intermediate step between input and output layers of a normal convolutional layer to decrease the number of weight and bias parameters. At the end of a network global average pooling layer is applied before passing to fully connected layers; this layer takes input feature maps of larger size and averages them to 1x1 maps decreasing the number of parameters to train. Inside the inception modules 1x1, 3x3, 5x5 convolutions and 3x3 max pooling are performed in a parallel way to input and outputs are brought together to produce a final output. In this way having multiple convolution filter sizes object at multiple scales are handled better. Lastly, in the middle of the network there are two intermediate classifier branches used in training which losses are added up to total loss with a factor of 0.3. These layers are called auxiliary classifiers and help with vanishing gradient and provide some level of regularization.

The winner of ILSVRC 2015 was a deep residual neural network called ResNet [57]. He et al. observed that after a specific threshold as the layers number increases, the training error also increases. Also, they observed that most of the times shallower networks give better results than deeper networks, although just opposite is to be expected since addition of identity maps to a shallow network leads a deeper network. To be able to work with very deep architectures they made use of skip connections or identity mappings to jump over some layers. At the same time skipping some layers help with the problem of vanishing gradients and makes learning faster as there is a smaller number of

layers to go through. Therefore, residual neural networks although being deep have low complexity. Identity mapping has no parameters, it only adds the previous layer output to layer ahead. In case previous layer and layer ahead have different dimensions, identity mapping is multiplied by a weight matrix linear projection which leads to expansion of channels to match the residual layer. Skip connections add the previous layers` outputs to the stacked layers` outputs. The ResNet used in ILSVRC 2015 had 152 weight layers.

Machine learning has numerous applications in computational biology and bioinformatics. Large structure- or sequence-based datasets can be used for training deep learning and classical machine learning models to predict protein-protein interactions, classify protein folds, or fold proteins. While classical machine learning algorithms require highly engineered input data, deep learning methods have the benefit of direct usage of unprocessed raw data. The fast growth in the number of structural protein data deposited to protein data bank also allowed the faster deposit of protein-protein docking models. This as well resulted in multiple structural protein-protein interface datasets. While ProtCID [58] and ProtinDB [59] datasets contain all possible interfaces, the PIFACE [60] dataset groups the available interfaces according to their structural similarity.

The majority of CNN applications to protein fold and structure determination is with 1D or 2D CNNs which require reduced input representations of protein structure and sequence. One example is DNCON2 [61], a predictor of protein residue-residue contacts. Input features used are secondary structure values, predicted solvent accessibility, and coevolution features extracted using multiple sequence alignment. DeepContact [62] is another predictor of protein residue-residue contacts, where 1D and 2D representations of protein structure are extracted from protein structure and sequence. Fold determination and protein structure problems are not the only ones CNNs are applied to. For example, DeepFold [63] represents a protein structure motif extractor that can be used for identifying similar protein structures. 2D distance matrices extracted from protein 3D structures are used as input. The structural motifs learned by the DeepFold used as a pattern for protein structure representation. MaSIF [64] makes use of geometric deep learning to apply CNNs to protein data to capture molecular surface interaction fingerprints which are important for certain biomolecular interactions. MaSIF was proven to be useful for predicting protein-protein interaction sites and protein pocket-ligands, and for in ultrafast screening of protein surfaces to predict protein-protein complexes. As

input, protein structure angular and radial coordinates are used together with input features divided into geometric features comprising of shape index and distance-dependent curvature; and chemical features comprising of hydropathy, continuum electrostatics, and free electrons/protons.

In the field of bioinformatics, 3D CNNs have been used for analysis of protein functional sites [65], detection of secondary structures in cryo-EM maps [66], protein-drug interaction scoring [67], and quality evaluation of single protein structure models [68, 69]. For example, some approaches that use 3D protein structures as input and 3D CNNs as architecture types are DeepSite [70], 3DCNN [65], DOVE [38]. DeepSite [70] predicts binding sites in small molecule-protein interactions using 3D CNNs. Protein structures are represented as a 3D grids of 1x1x1 $Å^3$ voxels. Voxels are marked with pharmacophoric properties belonging to the atoms occupying the voxel based on the coordinates. Each pharmacophoric label represents a channel in the 3D CNN network. 3DCNN [65] uses protein structures as input to 3D CNN to learn the amino acids` environments that are later used for identifying the functionally important amino acids. Local regions or protein microenvironments are represented using a 3D grid with size 20 $Å^3$ which is placed around a central atom. There are four channels used, one for each of the atoms.

DOVE [38] predicts if the docking model has an acceptable quality or not based on the CAPRI criteria [32]. As input features atom coordinates and distance-dependent contact potentials, GOAP [71] and ITscore [34] are used. Therefore, there are six input channels, four for carbon, nitrogen, oxygen, and other atoms, and other two for GOAP and ITscore values. Input is represented as grid of 20x20x20 dimensions with voxel size of 1 Å resulting in 20 $Å^3$ cube or a voxel size of 2 Å resulting in 40 $Å^3$ cube input representation. Cube is placed at protein-protein interface which is defined as heavy atoms located within 10 Å to any heavy atoms belonging to other protein from the complex. DOVE has a simple 3D CNN architecture resembling the LeNet architecture [53] composed of three convolutional layers, two max-pooling layers and two fully-connected layers. Dataset used for training DOVE is based on the ZDOCK benchmark 4.0 [42]. Protein docking models classified as correct according to CAPRI [32] criteria were used as positive samples while those classified as incorrect according to CAPRI [32] criteria were used as negative samples. Since number of correct decoys in the benchmark is much lower than the number of incorrect decoys, data augmentation was done on

correct samples. Each correct sample was placed in different 24 orientations on a grid, by applying 90-degree rotations to the original coordinates around z-axis (4 orientations) and putting each of the six cube faces upwards (totalling in 24 orientations).

DeepInterface [41], uses 3D protein structures like raw input data as well. Data input is done through 40 channels, one for each amino acid type (except Glycine) and one for C alpha (Cα) locations, resulting in 20 channels for one protein and 20 channels for other interacting protein in the complex. Each channel input is a 3D grid having 25x25x25 dimensions with voxel size of 2x2x2 $Å^3$ dimensions. Grid covers the interface region of a protein-protein complex. DeepInterface is different from the approaches above in the way raw data is represented, its channels represent residue types, not atom types, pharmacophoric properties, or contact potentials. DeepInterface is trained with approximately 255,000 protein-protein interfaces collected from the PDB for positive samples and from docking models set PPI4DOCK [72] for negative samples. It is validated and tested on approximately 19,000 protein-protein interfaces collected from the PDB for positive samples and from docking models set DOCKGROUND [73] for negative samples.

In this study DeepInterface [41] datasets, model and training are optimized to maximize the accuracy achieved on validation dataset, and the new model performance is measured on well-known ZDOCK docking benchmark 4.0 [42].

Chapter 3:
METHODS

DeepInterface [41] is a deep learning model used to identify if a docked complex is a native-like structure or not. In this thesis, DeepInterface datasets, architecture and hyperparameters are optimized to maximize the accuracy achieved on test dataset, and the new models performance is measured using well-known ZDOCK docking benchmark 4.0 [42]. In this chapter, new model architectures and training procedure of an updated DeepInterface, the datasets used for training and testing, and performance analyses done are described.

3.1 DeepInterface Datasets

The data used for training and testing the DeepInterface model comes from four different resources: PPI4DOCK [72], DOCKGROUND [73], Protein Data Bank (PDB) [74], and PIFACE [60]. Docking models datasets DOCKGROUND [73] and PPI4DOCK [72] are used as negative data sources. On the other hand, complexes from PIFACE [60] which contains structures from PDB [74] deposited until 2012 are used as positive data together with complexes deposited in the PDB after 2012 fetched directly from the PDB. Validation and test datasets consist of structures from different sources than training dataset to show the model robustness.

Interfaces coming from a biological complex deposited in PDB are considered as positive. Interfaces coming from complexes classified as “incorrect” by CAPRI [32] criteria are considered as negative. Incorrect complexes have Fnat (fraction of native contacts) value lower than 0.1, lRMSD (ligand root mean square deviation) greater than 10 Å, and iRMSD (interface root mean square deviation) greater than 4 Å. Interface definition for both positive and negative complexes is the same, residues from one chain closer than 12 A to residues of another chain, single or multiple, build an interface.

PPI4DOCK [72] is a docking benchmark with 1,417 non-redundant unbound models that have less than 70% sequence similarity and can each produce up to 54000 decoys. There are five steps in selection procedure of 1,417 unbound models. Initially, using InterEvol [75] database, 5,641 heterodimeric interfaces were obtained to create PPI4DOCK dataset. These interfaces consist of two one-chain proteins that have less than

70% sequence similarity. To generate the standard PPI4DOCK dataset five steps were followed.

First, Yu and Guerois filtered the initial 5,641 interfaces based on the following criteria which defines the rules of a "good-quality" homodimer: it should contain ≤ 20 chains in its biological assembly, its X-ray resolution should be ≤ 3.25 Å, its interface should have size > 300 $Å^2$, it should not have any interfacial ligands, and according to NOXclass [76] it should be predicted as biological not crystal.

Second, on every side of the 3,157 interfaces, Yu et al performed the HHsearch [77] to search for potential homologs. Candidates were grouped such that every candidate in the group has more than 50% HHsearch probability and more than 20% sequence similarity with respect to the reference interface chain, and in total more than twenty aligned columns. Groups of candidates did not have more than 60 templates at the end.

Third, Yu et al. selected "good" candidates from the groups of candidates. A "good" candidate is defined by the following criteria: it should have X-ray resolution < 4Å, it should contain less than 20 chains in its biological assembly, it should contain minimum 40% of interfacial residues according to CAPRI's [32] interface residue definition that describes them like two atoms (non-hydrogen) belonging to different chains which are split up by a distance ≤ 5Å, it should have ≥ 0.6 TM-score given by TM-align [78] run on the candidate model and its reference protein which indicates that both structures are from the same fold/family, it should be a representative of a proper unbound structure, which was confirmed by iAlign [79] confidence scores to guarantee distinct interfaces. Afterwards, the candidate with the highest TM-score was picked as the "best" candidate for a template.

Fourth, Yu et al used RosettaCM [80] to produce a homology model for heterodimeric interfaces which had valid templates assigned for both interface sides. After this step, the dataset consisted of 1,531 modelled interactions.

Fifth step was the evaluation of the model. For the models where multiple spatially separated sub-segments were present in the one chain, Yu et al. stripped all sub-segments except the one with the highest number of contacts with the other chain. If this led to the decrease in interface size by 50% or larger, or if it left the model with residue number less than 30 model was removed. Using TM-align, x-ray structures of the matching target subunits were aligned with the stripped models. Those models that had sequence similarity less than 50% and TM-score less than 0.5 were removed. In the end, Yu et al

created “superimposed decoys” by using TM-align for superimposing the rest of the homology models with their reference bound structures. Those cases that had interface size smaller than 300Å or had less than 5 contacting residues were removed.

To sum up, PPI4DOCK dataset resulted in 1,417 models that had the same sequences, residue numbers, and chain IDs like their reference structures. Using ZDOCK 3.0.2 [43] up to 54,000 decoys were produced for each model. ZDOCK operates using a Fast Fourier Transform (FFT) docking algorithm. ZDOCK tries all possible rotations and translations of ligand unit onto receptor unit using a unit step. It also provides CAPRI evaluation for each structure. Obtained decoys were categorized according to CAPRI criteria as ‘High’, ‘Medium’, ‘Acceptable’ and ‘Incorrect’. All decoys that have Fnat > 10% were stored and can be acquired from the PPI4DOCK dataset. Around 110 incorrect structures were selected from each model having ligand in different positions. In total, 149,557 negative interfaces were selected from PPI4DOCK benchmark.

DOCKGROUND [73], an unbound docking benchmark, is similar to PPI4DOCK. DOCKGROUND [73] dataset contains docking decoys sets or scoring benchmarks, and docking benchmark sets, that can be used for different tasks. Docking benchmarks contain sets of simulated unbound, unbound, and model structures. On the other hand, docking decoy sets are mostly used for testing ranking/scoring functions, which leads to such datasets offering numerous poses for each protein-protein complex, where one of them has a near-native structure and other are false positives.

There are two subsets in the docking decoy set. These are two unrelated decoy sets. One of them consists of 101 structures (at least one structure is near-native) for each of 61 unbound complexes taken from docking benchmarks in DOCKGROUND. The second one consists of 10 structures (at least one structure is near-native) for each of 351 unbound complexes. GRAMM-X [81], FFT-based tool, was used for decoys generation. DOCKGROUND benchmark does not provide CAPRI [32] evaluation for the structures, however it provides Fnat, iRMSD, and lRMSD values for them therefore CAPRI evalution can be calculated. Structures that are classified as incorrect by CAPRI evaluation were selected.

PIFACE [60] dataset contains protein-protein interfaces obtained using protein structures from PDB deposited before 2012. Cukuroglu et al. made use of structural alignment tools and networks algorithms to create unique clusters by grouping the interfaces according to their structural similarity. First, all protein structures having more

than one chain and stored in PDB before January 14th, 2012 were fetched. This resulted in 45,491 structures. Second, all possible two-chain interactions within each structure were listed. This resulted in 622,321 such interactions. To determine which pairs physically interact Cukuroglu et al. applied NACCESS [82] to protein structures to calculate their accessible surface area (ASA). They claim that existence of a significant difference between the ASA value of the protein/chain in a complex and the ASA value of the protein/chain as a monomer indicates that the two chains represent interacting partners. This resulted in 130,209 possible interfaces/ interactions.

Cukuroglu et al. used structural alignment tool called Multiprot [83] which is sequence-independent to calculate the pair-wise alignment. They defined the interface similarity measure as shown in the equation 3.1, where M is the number of residues matched by Multiprot`s alignment, and Ni is the number of residues belonging to ith interface. Those interface pairs that had highly unbalanced lengths were discarded to accelerate the similarity calculations.

$$S_{1,2} = \frac{M}{\min(N_1, N_2)} \tag{3.1}$$

Afterwards, a graph was created, where each interface is a node, and each edge represents the pair-wise similarity measure between the matching interfaces pair. Cukuroglu et al. used Girvan and Newman method [84] to cluster the graph. This method finds communities consisting of densely connected nodes in a graph. The output is a set of clusters/communities containing structurally similar interfaces inside the set of all interfaces.

To summarize, the PIFACE dataset [60] interfaces were extracted from approximately 80,000 PDB files deposited to the PDB until 2012. It consists of approximately 130,000 interfaces clustered using structural similarities into approximately 26,000 clusters. Interfaces that are already extracted and stored in PIFACE were included into positive dataset. At the same time to enlarge the positive dataset, 16,249 interfaces were extracted from PDB files that are deposited to the PDB after 2012.

When training the DeepInterface models, PIFACE and PPI4DOCK datasets were used to train the models, while DOCKGROUND dataset and the PDB files deposited after 2012 were used to validate and test the models. Using datasets with different distribution for validation and testing of the model from the distribution used for training is to test how well the models generalize to structures from different datasets.

Although the training and test datasets are taken from different resources; in terms of similar protein-protein structures redundancy was still present between the datasets. To avoid any biased results this redundancy was removed. The redundant samples were discarded from the validation and test datasets. A redundant sample is identified as a sample from validation or test dataset having more than 40% sequence similarity when compared to one or more samples coming from the training set.

After the interfaces were collected from all data sources and redundancy was removed from test and validation datasets, total number of 274,108 interfaces for training the models was obtained. 137,054 interfaces in the training dataset are positive, and the same number of them is negative. To summarize, training dataset contains 127,790 positive interfaces coming from PIFACE and 127,790 negative interfaces coming from PPI4DOCK; validation and test datasets contain 4,632 negative interfaces from DOCKGROUND, and 4,632 positive interfaces from PDB [74] files deposited after 2012. Final composition of training, validation and test datasets is shown in Table 3.1.

Table 3.1: Interfaces and their sources in training, validation, and test datasets.

	PIFACE	PPI4DOCK	DOCKGROUND	PDB2012	Total
Training	127,790	127,790	-	-	255,580
Validation	-	-	4,632	4,632	9,264
Test	-	-	4,632	4,632	9,264
Total	127,790	127,790	9,264	9,264	274,108

Input protein-protein structures are processed before they are used for training or testing the DeepInterface models. First, interface extraction according to PRISM [2] criteria is done. Second, interfaces are made translation and rotation independent. This is done by translating the midpoint of two interface sides` centroids to the origin, then line connecting centroids is aligned to the x-axis by rotating the interface, and last interface is rotated to make the furthest atom from origin to point in the direction of the positive z-direction. Third, a matrix of 40 x (25x25x25) dimensions is created. Input has 40 channels, each being a 3D 25x25x25 cube. Each cube cell is a 2x2x2 $Å^3$ space. Each side of interface is represented by 20 channels, one channel for all residues` Cα‘s locations, and 19 channels for the locations of each residue type (except Glycine) side chains (represented as one Cβ atom at the side chain`s centroid). In cases that an interface cannot

fit the 50x50x50 $Å^3$ cube it was discarded from the dataset. Input data format for DeepInterface is shown in Figure 3.1.

Locations of the amino acids are considered like 3D Gaussian distributions where coordinates of the amino acid are taken as mean and standard deviation is calculated using van der Waals radii of corresponding amino acid type. The reason behind usage of Gaussian distributions is to present the proteins` dynamic nature and smooth the noise in data. To fill the cells of the cube, distributions are quantized.

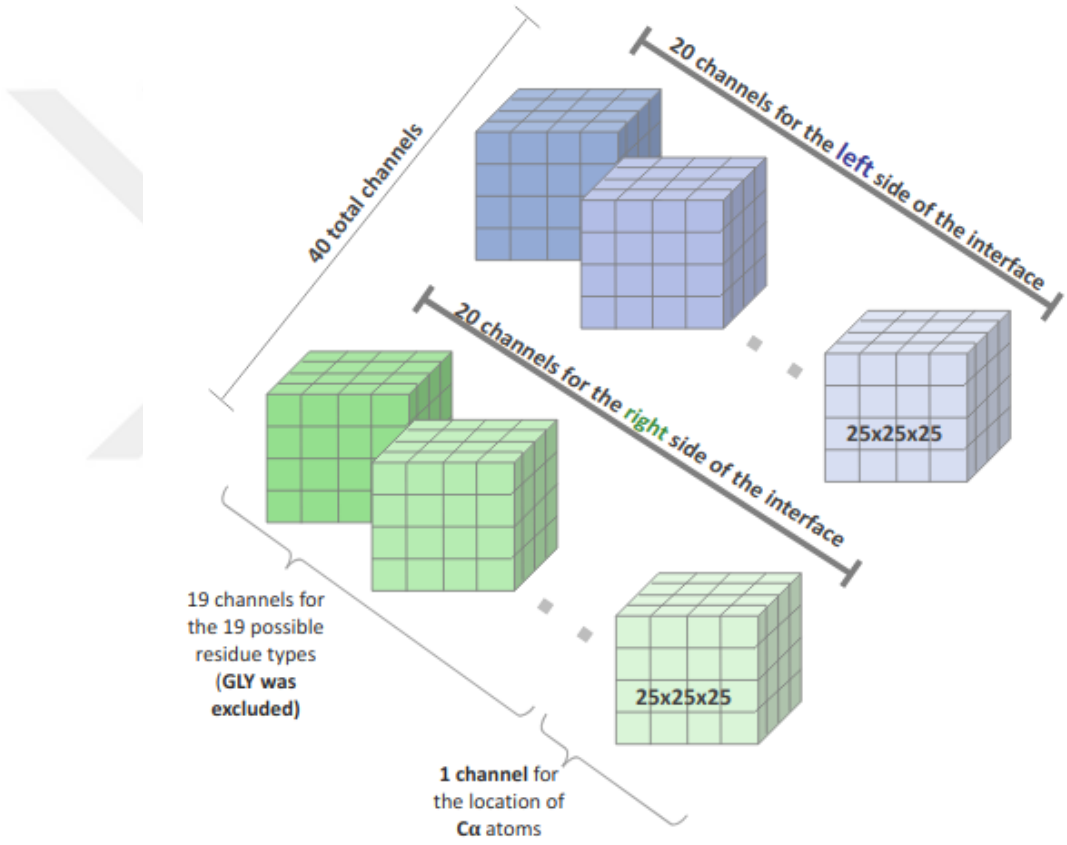

Figure 3.1: Input data format for DeepInterface.

3.2 Models

DeepInterface [41] model created by Balci et al. is deep CNN having 4 convolutional layers and 2 fully connected-layers following them. Each convolutional layer is followed by batch normalization [87] layer, and a rectifier linear unit (ReLU) [55] activation. In addition, to improve generalization the last convolutional fourth layer is followed by a global average pooling layer. Last layer in the network is a softmax layer

which outputs two scores, one for positive and one for negative class. Figure 3.2 shows the DeepInterface [41] model architecture. It should be mentioned that this model was developed in Julia programming language [85] and using framework for deep learning called “Knet” [86].

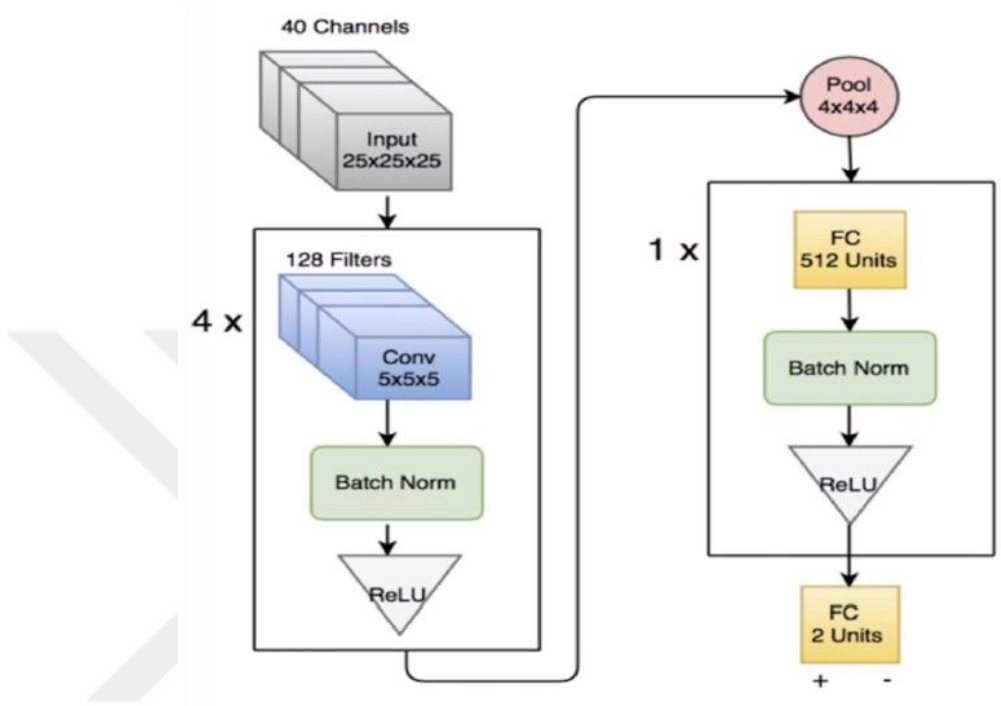

Figure 3.2: Model architecture of DeepInterface by Balci et al. [41]

In this study DeepInterface [41] model is changed to minimize the running time of the training and maximize the accuracy achieved on validation dataset. The best performing model after the systematic architecture search performed was selected. Our network architecture resembles the VGG [54] model architecture with 16 weight layers and batch normalization used. The major difference between the architecture of VGG model (VGG16) [54] and the architecture of our model (VGG16 DeepInterface) is that the first max pooling layer in the VGG16 model is removed which leads to a total number of 13 convolutional layers and 4 max pooling layers followed by 1 average pooling layer, and 3 fully connected layers. Max pooling layers are good for addressing the sparsity of our input data and better generalization, however since the original VGG16 model has 5 max pooling layers and input size of 224x244 dimensions, but the input data size for DeepInterface is 25x25x25, one of the max pooling layers had to be removed or its dimensions had to be changed. All convolutional layers have 3x3x3 filters with stride and padding set to 1. All max pooling layers have 2x2x2 filters with stride and padding set to

2. Each convolution is followed by a batch normalization [87] and rectifier linear unit (ReLU) activation [55]. Order of the layers is as follows: 2 convolutional layers with 64 filters, 2 convolutional layers with 128 filters, 1 max pooling layer, 3 convolutional layers with 256 filters, 1 maxpooling layer, 3 convolutional layers with 512 filters, 1 maxpooling layer, 3 convolutional layers with 512 filters, and 1 maxpooling layer. Global average pooling at the end of convolutional and maxpooling layers converts the 512 1x1x1 spatial feature maps to 512 global features. Global pooling is followed by 2 fully connected layers with 4096 hidden units, each followed by ReLU and Dropout (p = 0.5) [88] layers. The final fully connected layer outputs two scores, one for positive and one for negative class. Figure 3.3 shows our best model`s architecture.

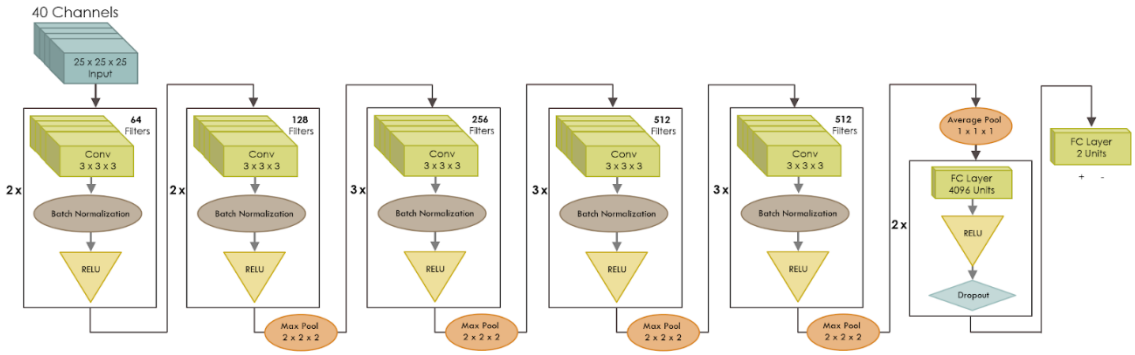

Figure 3.3: Architecture of our highest validation accuracy DeepInterface model having VGG16 resembling architecture [54].

All 2D layers in the original VGG16 model were converted to their 3D versions to accommodate the 3D input data. Number of input classes was changed from 1,000 to 2 since our model predicts two classes, and the number of input channels was changed from 3 used for RGB pictures to 40 to accommodate the data input.

However, in order to find the best performing architecture explained above, that minimizes the training running time and maximizes the accuracy achieved on validation dataset, systematic architecture search was performed. Through our experiments, classical Lenet-like CNN architectures [53], deep residual neural networks with ResNet resembling architecture [57], deep inception networks with Google Net resembling architecture [56], and deep CNNs with VGG resembling architecture [54] were all tried with various layers` and filter sizes` combinations. However, we observed that the model with architecture resembling VGG16 described in previous paragraph is overfitting less and has better convergence. Results obtained when other architectures used are as well discussed in the next chapter.

Firstly, when models with architectures resembling VGG [54] were tried, as already mentioned, to accommodate the input size of 25x25x25 dimensions removal of at least one max pooling layer from VGG architecture [54] had to be done. Different combinations of maxpooling layers` removal were tested to check which one results in the highest validation accuracy. This resulted in the conclusion that the removal of the first max pooling layer gives the highest validation accuracy. Also, different filter sizes were experimented upon, and original filter size with 3x3x3 dimensions resulted in the highest validation accuracy.

Secondly, when models with architectures resembling ResNet [57] were tried, different layer combinations of original ResNet [57] were tested. The best performing DeepInterface model with architecture resembling ResNet [57] has 18 weighted layers. ResNet models with higher number of weighted layers were also tested, but this number of layers have the best results with our data. The major difference between the architecture of original Resnet model (ResNet18) [57] and the architecture of our model (ResNet DeepInterface) is that the max pooling layer after the first convolution in the ResNet18 model is removed which leads to a total number of 17 convolutional layers followed by 1 average pooling layer, 1 fully connected layer, and 1 softmax layer. First convolutional layer has 7x7x7 filters with stride 2 and padding 3. Other convolutional layers have 3x3x3 filters with stride and padding set to 1. Order of the layers is as follows: 1 convolutional layer with 64 filters, 2 x 2 convolutional layers with 64 filters, 2 x 2 convolutional layers with 128 filters, 2 x 2 convolutional layers with 256 filters, 2 x 2 convolutional layers with 512 filters, 1 average pooling layer, 1 fully connected layer, and softmax layer. Global average pooling at the end of convolutional layers converts the 512 1x1x1 spatial feature maps to 512 global features. Global pooling is followed by 1 fully connected layer with two units. This fully connected layer is followed by softmax layer which outputs two scores, one for positive and one for negative class. The evaluation metrics of ResNet DeepInterface model are presented in the next section.

Thirdly, when models with architectures resembling GoogleNet [56] were tried, different layer combinations of original GoogleNet [56] were tested. The best performing DeepInterface model with architecture resembling GoogleNet [56] has 22 weighted layers, counting parallel convolution as one. The major difference between the architecture of original GoogleNet model [56] and the architecture of our model (GoogleNet DeepInterface) is that the first and third max pooling layers are removed

which leads to a total number of 22 convolutional layers (counting parallel convolutions as one), 12 max pooling layers, followed by 1 average pooling layer, 1 fully connected layer, and 1 softmax layer. Order of the layers is as follows: 3 convolutional layers, 1 max pooling layer, 7 inception modules, 1 max pooling layer, 2 inception modules, 1 average pooling layer, 1 fully connected layer, and softmax layer. Global average pooling at the end of convolutional layers converts the 1024 1x1x1 spatial feature maps to 1024 global features. Global pooling is followed by 1 fully connected layer with two units. This fully connected layer is followed by softmax layer which outputs two scores, one for positive and one for negative class. The evaluation metrics of GoogleNet DeepInterface model are presented in the next section.

All models in this study are developed using PyTorch [89], a deep learning library implemented in Python.

3.3 Training

While training our models, to find the best training procedure that minimizes the training running time and maximizes the accuracy achieved on validation dataset, systematic hyperparameters` space search was performed considering following hyperparameters: learning rate, dropout [87], L2 regularization, batch size, optimizer choice, and learning rate decay. First learning rate search was done and learning rate of 0.001 was selected for resulting in highest validation accuracy. Second, to address overfitting, different dropout [87] probabilities and L2 regularization factors were tried. Dropout of 0.2 and L2 regularization of 0.0001 resulted in the highest validation accuracies. Third, different optimizers including Adam, stochastic gradient descent (SGD), and SGD with Nesterov momentum [93] were tried. Usage of Adam or SGD with Nesterov momentum 0.9 were resulted in better validation accuracy values compared to the usage of only SGD. Fourth, different batch sizes were tried and size 32 resulted in the highest validation accuracy. Last, when the learning was performed on more epochs, applying learning rate decay in case the validation loss increases according to previous epoch validation loss value was tried using different learning rate decay factors, and the best validation accuracy was obtained with learning rate decay factors of 0.4 and 0.9. Our models` training usually converged in approximately 10 epochs.

DeepInterface [41] model created by Balci et al. is trained using a learning rate of 0.001 and Adam [91] optimizer. The data is split into mini batches of size 32, and it is shuffled for each training epoch. Based on validation loss increase, a learning rate decay of factor 0.5 is applied. The convolutional layers` weights are initialized using Kaiming distribution [91] and the fully connected layers` weights are initialized using Xavier distribution [92].

VGG16 DeepInterface model is trained using a learning rate of 0.001 and Adam optimizer [90]. The data is split into mini batches of size 32, and it is shuffled for each training epoch. Learning rate is decayed by a factor of 0.4 whenever validation loss increases compared to the previous epoch. L2 regularization of 0.0001 is used. The convolutional layers` weights are initialized using Kaiming distribution [91] and the fully connected layers` weights are initialized using standard normal distribution.

ResNet DeepInterface model is trained using a learning rate of 0.001 and Adam optimizer [90]. The data is split into mini batches of size 32, and it is shuffled for each training epoch. Learning rate is not decayed. L2 regularization of 0.000001 is used. The convolutional layers` weights are initialized using Kaiming distribution [91].

GoogleNet DeepInterface model is trained using a learning rate of 0.001 and Adam optimizer [90]. The data is split into mini batches of size 32, and it is shuffled for each training epoch. Learning rate is not decayed. L2 regularization of 0.000001 is used. The convolutional layers` weights and the fully connected layers` weights are initialized using standard normal distribution truncated to [-2, 2] interval.

Chapter 4:
RESULTS AND DISCUSSION

4.1 Model Evaluation

In this section evaluation metrics of DeepInterface models are shown. We also present multiple performance analyses in the following sections to compare our models with other protein docking models` ranking methods: ZRANK [44], IRAD [47], and DOVE [38].

We trained our DeepInterface models using 127,790 protein interfaces from PIFACE [60] as positive training samples and 127,790 protein interfaces from PPI4DOCK [72] as negative training samples. The size of training dataset is 255,580 interfaces. We validated and tested our DeepInterface models using 18,582 protein interfaces from PDB as positive samples and 18,582 protein interfaces from DOCKGROUND [73] as negative samples. Validation and test datasets were uniformly sampled from these two resources. The size of validation dataset is 9,264 interfaces and the size of test dataset is 9,264 interfaces as well. We used validation dataset to evaluate the model while training and pick the best performing one.

During the systematic architecture and hyperparameters` space search for models with architectures resembling deep CNN network VGG [54] various layers` and filter sizes` combinations were tried and multiple hyperparameters were tuned including learning rate, dropout [87], L2 regularization, batch size, optimizer choice, and learning rate decay. Among all tried models the model with the highest validation accuracy is explained below. Evaluation metrics for the other models that had validation accuracy > =0.81 can be found in Table A.1, and their means and standard deviations can be found in Figure A.1 - Figure A.18.

The evaluation metrics of our highest validation accuracy model shown in Figure 3.3, which is resembling VGG16 architecture [54] are given in Table 4.1. Evaluation metrics for the training, validation and test datasets are given separately. Validation accuracy converged to 81% after 40 epochs of training. From the Table 4.1 it can be seen that final model accuracy on the training dataset is 0.97, on the validation dataset is 0.81, and on the test dataset is 0.81.

In all cases, recall values are high meaning that most of the positives are predicted correctly. However, specificity and precision values are lower in validation and test cases, indicating that the model does not manage to find all negatives and tends to predict output as positive. Since positive interfaces in both training and validation/test datasets come from the same source, the PDB repository, it is to expect that model will easily detect positives. However, negative interfaces come from PPI4DOCK decoys in training and DOCKGROUND decoys in validation/test cases, both using different docking tools, which might have a negative impact on model`s detection of negative decoys. Still high NPV values in all cases show that if a model predicts one decoy to be negative it is most probably so. Model could be used for elimination of incorrect decoys.

Table 4.1: Evaluation metrics of DeepInterface VGG16 model.

	Accuracy	Loss	Recall	Specificity	Precision	NPV
Training Dataset	0.971	0.080	0.970	0.972	0.972	0.970
Validation Dataset	0.806	0.909	0.936	0.676	0.743	0.913
Test Dataset	0.812	0.920	0.939	0.684	0.748	0.918

The evaluation metrics of the best DeepInterface models with VGG16 [54] resembling architecture that had the validation accuracy > =0.81 and were the best among other similar models when trying different hyperparameters are given next.

First, the evaluation metrics of the DeepInterface model that had the highest validation accuracy when different VGG layer combinations were tried is given in Table 4.2. Model accuracy on the training dataset is 0.91, on the validation dataset is 0.81, and on the test dataset is 0.81. This model has a VGG19 similar architecture with the fifth max pooling layer removed and batch normalization applied.

Table 4.2: Evaluation metrics of DeepInterface VGG19 model with fifth max pooling layer removed.

	Accuracy	Loss	Recall	Specificity	Precision	NPV
Training Dataset	0.909	0.229	0.913	0.906	0.906	0.913
Validation Dataset	0.810	0.865	0.885	0.735	0.770	0.864
Test Dataset	0.814	0.887	0.885	0.742	0.775	0.866

Second, the evaluation metrics of the DeepInterface model that had the highest validation accuracy when filter size in convolutional layers of VGG models were changed from 3x3x3 to 5x5x5 is given in Table 4.3. Model accuracy on the training dataset is 0.95, on the validation dataset is 0.81, and on the test dataset is 0.81. This model has VGG13 similar architecture with the second max pooling layer removed and batch normalization applied.

Table 4.3: Evaluation metrics of DeepInterface VGG13 model with second max pooling layer removed and filter size 5x5x5.

	Accuracy	Loss	Recall	Specificity	Precision	NPV
Training Dataset	0.950	0.135	0.949	0.951	0.951	0.949
Validation Dataset	0.810	1.106	0.932	0.689	0.750	0.910
Test Dataset	0.814	1.199	0.932	0.696	0.754	0.910

Third, the evaluation metrics of the DeepInterface model that had the highest validation accuracy when L2 regularization was applied is given in Table 4.4. Model accuracy on the training dataset is 0.92, on the validation dataset is 0.81, and on the test dataset is 0.81. This model has VGG16 similar architecture with the first max pooling layer removed and L2 regularization of factor 0.0001 applied.

Table 4.4: : Evaluation metrics of DeepInterface VGG16 model with first max pooling layer removed and L2 regularization 0.0001 applied.

	Accuracy	Loss	Recall	Specificity	Precision	NPV
Training Dataset	0.923	0.201	0.922	0.924	0.924	0.922
Validation Dataset	0.807	0.690	0.916	0.697	0.752	0.892
Test Dataset	0.810	0.695	0.914	0.706	0.756	0.892

Fourth, the evaluation metrics of the DeepInterface model that had the highest validation accuracy when Dropout regularization was applied is given in Table 4.5. Model accuracy on the training dataset is 0.95, on the validation dataset is 0.81, and on the test dataset is 0.81. This model has VGG13 similar architecture with the first, third and fifth max pooling layers removed and Dropout regularization with 0.2 probability applied.

Table 4.5: Evaluation metrics of DeepInterface VGG13 model with first, third and fifth max pooling layers removed and Dropout regularization 0.2 applied.

	Accuracy	Loss	Recall	Specificity	Precision	NPV

Training Dataset	0.918	0.211	0.923	0.914	0.915	0.922
Validation Dataset	0.807	0.552	0.906	0.707	0.756	0.883
Test Dataset	0.809	0.543	0.902	0.715	0.760	0.880

Fifth, the evaluation metrics of the DeepInterface model that had the highest validation accuracy when learning rate decay was applied is given in Table 4.6. Model accuracy on the training dataset is 0.93, on the validation dataset is 0.82, and on the test dataset is 0.82. This model has VGG16 similar architecture with the first max pooling layer removed and learning rate with factor 0.6 applied when validation loss would increase through 7 epochs.

Table 4.6: Evaluation metrics of DeepInterface VGG16 model with first max pooling layer removed and learning rate 0.6 applied.

	Accuracy	Loss	Recall	Specificity	Precision	NPV
Training Dataset	0.931	0.181	0.931	0.932	0.932	0.931
Validation Dataset	0.815	0.903	0.918	0.713	0.762	0.897
Test Dataset	0.817	0.983	0.918	0.717	0.765	0.897

And finally, sixth, the evaluation metrics of the DeepInterface model that had the highest validation accuracy when different batch sizes for data input were tried is given in Table 4.7. Model accuracy on the training dataset is 0.93, on the validation dataset is 0.82, and on the test dataset is 0.82. This model has VGG16 similar architecture with the first max pooling layer removed, and the batch size used was 32.

Table 4.7: Evaluation metrics of DeepInterface VGG16 model with first max pooling layer removed and batch size 32 used.

	Accuracy	Loss	Recall	Specificity	Precision	NPV
Training Dataset	0.923	0.201	0.922	0.924	0.924	0.922
Validation Dataset	0.807	0.690	0.916	0.697	0.752	0.892
Test Dataset	0.810	0.695	0.914	0.706	0.756	0.892

Next, the evaluation metrics of architectures different than VGG are discussed. The evaluation metrics of the DeepInterface [41] model created by Balci et al. with architecture resembling classical Lenet-like CNN architectures [53] are given in Table 4.8. Model accuracy on the training dataset is 0.96, on the validation dataset is 0.76, and

on the test dataset is 0.76. This model also has high recall and low specificity values, meaning that this model performs better in identifying positive samples.

Table 4.8: Evaluation metrics of DeepInterface Balci et al. model.

	Accuracy	Loss	Recall	Specificity	Precision	NPV
Training Dataset	0.960	0.107	0.955	0.965	0.965	0.956
Validation Dataset	0.757	1.157	0.821	0.694	0.728	0.795
Test Dataset	0.760	1.154	0.825	0.694	0.730	0.799

As we already mentioned to find the best performing model that minimizes the training running time maximizes the accuracy achieved on validation dataset, systematic architecture search was performed. The evaluation metrics of the best model with architecture resembling deep residual neural network ResNet [57] are given in Table 4.9. Model accuracy on the training dataset is 0.94, on the validation dataset is 0.76, and on the test dataset is 0.77. This is the highest validation accuracy we managed to obtain by trying different residual network architectures.

Table 4.9: Evaluation metrics of DeepInterface ResNet model.

	Accuracy	Loss	Recall	Specificity	Precision	NPV
Training Dataset	0.945	0.142	0.942	0.948	0.948	0.942
Validation Dataset	0.760	0.827	0.838	0.683	0.725	0.808
Test Dataset	0.770	0.830	0.846	0.694	0.734	0.819

The evaluation metrics of the best model with architecture resembling deep inception network Google Net [56] are given in Table 4.10. Model accuracy on the training dataset is 0.92, on the validation dataset is 0.76, and on the test dataset is 0.76. This is the highest validation accuracy we managed to obtain by trying different inception network architectures.

Table 4.10: Evaluation metrics of DeepInterface GoogleNet model.

	Accuracy	Loss	Recall	Specificity	Precision	NPV
Training Dataset	0.917	0.332	0.912	0.922	0.921	0.913
Validation Dataset	0.757	0.810	0.877	0.637	0.707	0.838
Test Dataset	0.765	0.820	0.883	0.646	0.714	0.847

4.2 *Output Scores*

For each interface complex our DeepInterface model outputs two scores. One score is a positive class score and second score is a negative class score. Decision variable is calculated as the difference between these two scores and is called diff-score (diff-score = positive score – negative score). In case that the diff-score is greater than 0 it is said that the input is a native-like protein interface.

To illustrate how a single diff-score is assigned to a prediction Figure 4.1 and Figure 4.2 are provided. Positive and negative class scores are shown in red and blue color respectively, together with corresponding diff-scores in yellow color.

Figure 4.1 shows scores of 92 incorrect docking models and 1 native-like protein-protein complex of 3c9aAB interface.

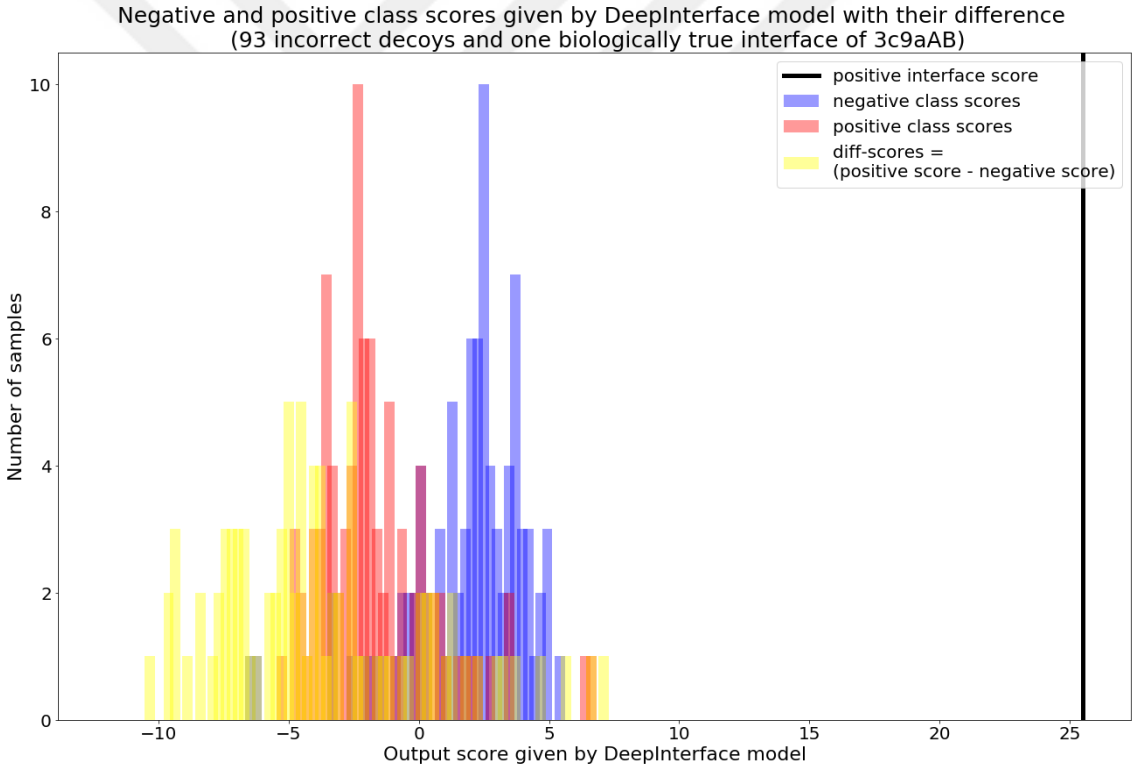

Figure 4.1: Distribution of DeepInterface scores for 1 positive and 92 negative structures of 3c9aAB interface.

Since most of the structures in the Figure 4.1 are actually negative samples it is not a surprise to see that diff-scores (yellow) are mostly located below zero meaning that the model correctly identified them as negative (i.e. true negatives). Although there is only one near-native structure among 93 interfaces for which scores are shown, there are still few diff-scores greater than 0 in the Figure 4.1 (i.e., false positives). The diff-score

of the native-like structure is colored in black. This structure which is actually a positive sample is correctly identified as positive (i.e. true positive) and has the highest diff-score meaning that the native-like structure was ranked by our model as 1st among the total 93 interfaces shown.

Figure 4.2 shows scores of 94 incorrect docking models and 1 native-like protein-protein complex of 2pukAB interface. Again, diff-scores (yellow) are mostly located below zero and black vertical line corresponding to native-like structure`s diff-score has the highest value meaning that the native-like structure was ranked by our model as 1st among the total 95 interfaces shown.

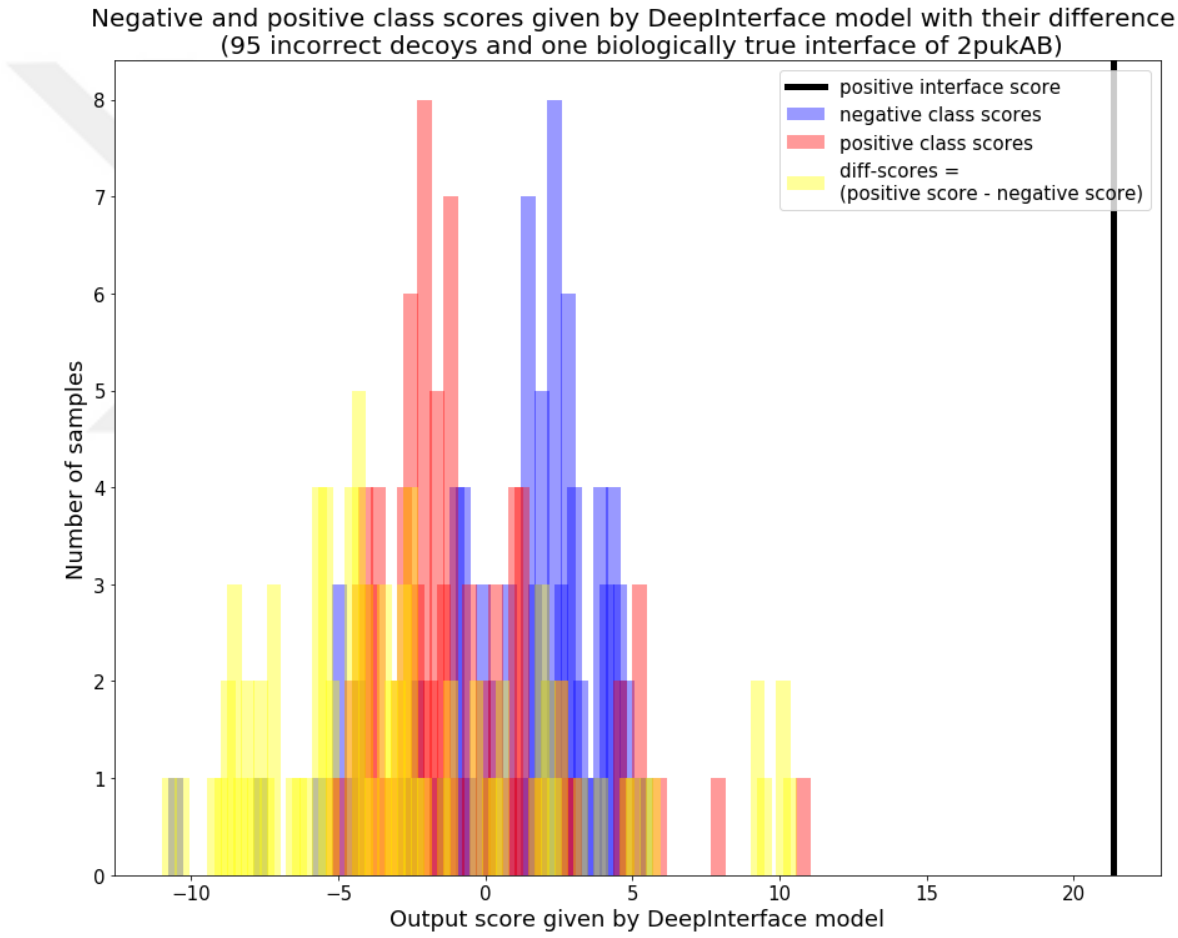

Figure 4.2: Distribution of DeepInterface scores for 1 positive and 94 negative structures of 2pukAB interface.

4.3 *Performance Analysis on Complexes from Test Dataset*

We performed a further evaluation of our DeepInterface models by determining the rank of a single native interface (i.e., positive sample) of a protein-protein complex put within the set of approximately 100 incorrect docking models (i.e., negative samples).

In this analysis we compared DeepInterface models` performance with docking models` scoring function ZRANK [44] and 3D CNN based models called DOVE [38] used for docking models` ranking.

DeepInterface models that are included in this comparison are: our highest validation accuracy DeepInterface model having VGG16 resembling architecture [54] (will be further named as VGG16 DeepInterface), our highest validation accuracy model with architecture resembling deep residual neural network ResNet [57] (will be further named as ResNet DeepInterface), our highest validation accuracy model with architecture resembling deep inception network Google Net [56] (will be further named as GoogleNet DeepInterface), and the DeepInterface [41] model created by Balci et al. with architecture resembling classical Lenet-like CNN architectures [53] (will be further named as Balci et al. DeepInterface).

ZRANK [44], developed by Pierce and Weng, is a scoring function which includes van der Waals, desolvation, and detailed electrostatics. DOVE, developed by Wang et al., is a 3D CNNs based model to predict if the docking model has an acceptable quality or not, based on the CAPRI criteria [32]. As input features atom coordinates and distance-dependent contact potentials, GOAP [71] and ITscore [34] are used. Input is represented as grid of 20x20x20 dimensions with voxel size of 1A resulting in 20 $Å^3$ cube (models called DOVE-ATOM20) or a voxel size of 2 Å resulting in 40 $Å^3$ cube (models called DOVE-ATOM40) input representation. Based on the feature channels used this group provides 8 models to users: DOVE-ATOM20 which uses only atom coordinates as features and input protein-protein interfaces are placed into 20 $Å^3$ grid, DOVE-ATOM40 which uses only atom coordinates as features and input protein-protein interfaces are placed into 40 $Å^3$ grid, DOVE-GOAP which only uses GOAP [71] distance-dependent potential as feature, DOVE-ITscore which only uses ITscore [34] distance-dependent potential as feature, DOVE-ATOM40-GOAP which uses which uses atom coordinates and GOAP [71] distance-dependent potential as features and input protein-protein interfaces are placed into 40 $Å^3$ grid, DOVE-ATOM40-ITscore which uses which uses atom coordinates and ITscore [34] distance-dependent potential as features and input protein-protein interfaces are placed into 40 $Å^3$ grid, DOVE-GOAP-ITscore which uses distance-dependent potentials GOAP [71] and ITscore [34] as features, and DOVE-ATOM40-GOAP-ITscore which uses atom coordinates and distance-dependent potentials GOAP [71] and ITscore [34] as features and input protein-protein interfaces

are placed into 40 $Å^3$ grid. According to their published performance analysis [38], two best performing models are DOVE-ATOM20 and DOVE-ATOM40. Therefore, we included these two models in this comparison analysis.

For this analysis we randomly selected 100 positive protein-protein complexes from test dataset. These complexes are native protein-protein complexes deposited to PDB after 2012. For each of 100 complexes we produced 300 decoys using GRAMM-X [81] protein docking tool. Out of 300 produced decoys per complex we selected 100 incorrect decoys per complex according to CAPRI criteria. This resulted in 100 incorrect decoys and one native structure per complex. We then scored and ranked them using DeepInterface models, ZRANK, and DOVE. We looked where did these scoring tools place native pdb structure (positive sample) among 100 incorrect decoys (negative samples).

When comparing DeepInterface models with ZRANK all 100 cases are used. However, when DOVE is included in comparison only 79 cases are used because DOVE data preparation was not successful for other 21 cases and DOVE models could not provide any score for them. Complexes that are used in this analysis can be found in Table B.1 in Appendices.

To score the decoy protein-protein models we processed them separately for each scoring tool used in the comparison. DeepInterface models require data pre-processing explained in Section 3.1. Briefly, first protein-protein interfaces were extracted from all decoy models using PRISM [2] criteria, second they were all translated and rotated to obtain similar poses for all, and finally they were placed into 40 cubes of 50x50x50 A3 dimensions which is the input format used by DeepInterface models. Afterwards, 100 decoys and one native structure per complex were scored and ranked according to their diff-score explained in Section 4.2. To use ZRANK [44] on the decoy protein-protein models we added polar hydrogens to the structures. Afterwards, 100 decoys and one native structure per complex were scored and ranked using ZRANK [44] scoring function. To use DOVE models, we named receptor chains as “A” and ligand chains as “B” in all structures. Afterwards, 100 decoys and one native structure per complex were scored and ranked using DOVE-ATOM20 and DOVE-ATOM40 3D CNNs based models.

As already explained, since DOVE data preparation was unsuccessful for 21 cases, we separate this analysis into two parts. In the first part we compare DeepInterface models with ZRANK using all 100 cases. And in the second part, we compare

DeepInterface models with ZRANK and DOVE models using 79 cases that DOVE managed to process.

Results of the first part of analysis can be seen in Table 4.11 and Figure 4.3, which show the top 1%, 10%, and 20% success rates of DeepInterface models and ZRANK in ranking the near-native interface among its corresponding decoys for all 100 cases selected from the test dataset. The top 1% for example, means that the near-native interface is placed among the top 1% of its corresponding ranked list.

In Figure 4.3, it can be seen that DeepInterface models perform better than ZRANK in all categories, top 1%, top5%, top 10%, and top 20%. Also, while performance of ZRANK does not improve much between top-x percentages, DeepInterface models` performance nearly doubles. Also, in top 20% almost all of the positives are found using DeepInterface models.

Table 4.11: Ranking of the positive interface among its incorrect decoys.

Ranking Tool	positive sample in top 1%	positive sample in top 5%	positive sample in top 10%	positive sample in top 20%
ZRANK	22%	26%	30%	34%
VGG16 DeepInterface	40%	59%	68%	78%
GoogleNet DeepInterface	43%	67%	74%	79%
ResNet DeepInterface	48%	66%	76%	83%
VGG19 DeepInterface	37%	58%	70%	80%
DeepInterface Balci et al.	42%	62%	70%	79%

In this analysis the best performing DeepInterface model is ResNet DeepInterface, which has 48% success rate in top 1%, and 83% success rate in top 20%. Other DeepInterface models are competitive as well, and as the second best performing model, we can select GoogleNet DeepInterface which has 43% success rate in top 1%, and 79% success rate in top 20%.

To conclude, DeepInterface models perform much better than ZRANK according to the first part of this analysis.

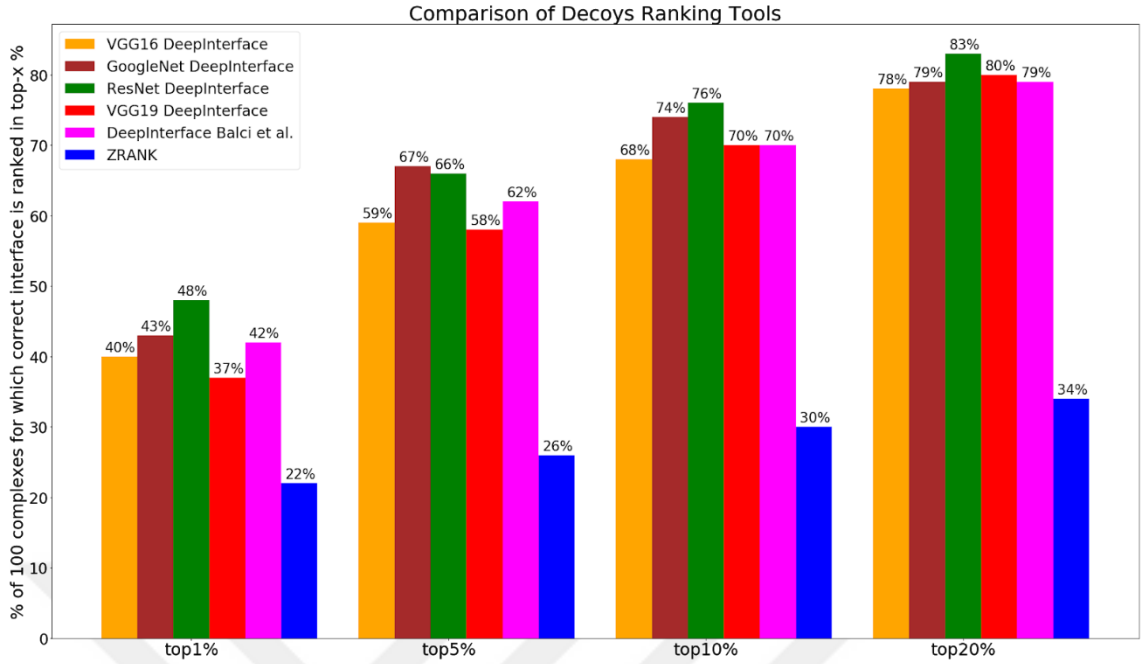

Figure 4.3: Comparison of decoys ranking tools on complexes from test dataset.

Table 4.12 and Figure 4.4 show the top 1%, 10%, and 20% success rates of DeepInterface models, ZRANK, DOVE-ATOM20 and DOVE-ATOM40 in ranking the near-native interface among its corresponding decoys for 79 cases selected from the 100 cases used in the previous analysis that DOVE managed to process.

In Figure 4.3, it can be seen that DeepInterface models perform better than ZRANK in all categories, top 1%, top5%, top 10%, and top 20%.

In Figure 4.4 it can be seen that DeepInterface models perform better than ZRANK, DOVE-ATOM20, and DOVE-ATOM40 in all categories, top 1%, top5%, top 10%, and top 20%. Again, in top 20% almost all of the positives are found using DeepInterface models.

Table 4.12: Ranking of the positive interface among its incorrect decoys.

Ranking Tool	positive sample in top 1%	positive sample in top 5%	positive sample in top 10%	positive sample in top 20%
ZRANK	23%	28%	33%	37%
VGG16 DeepInterface	46%	61%	70%	81%
GoogleNet DeepInterface	49%	71%	78%	82%
ResNet DeepInterface	51%	70%	80%	85%
VGG19 DeepInterface	42%	62%	75%	82%

DeepInterface Balci et al.	47%	67%	73%	80%
DOVE-ATOM20	3%	9%	13%	20%
DOVE-ATOM40	1%	6%	9%	20%

In this analysis the best performing DeepInterface model is ResNet DeepInterface which has 51% success rate in top 1%, and 85% success rate in top 20%. Other DeepInterface models are competitive as well, and again as the second best performing model, we can select GoogleNet DeepInterface which has 49% success rate in top 1%, and 82% success rate in top 20%.

To conclude, DeepInterface models perform much better than ZRANK, DOVE-ATOM20, and DOVE-ATOM40 according to the second part of this analysis.

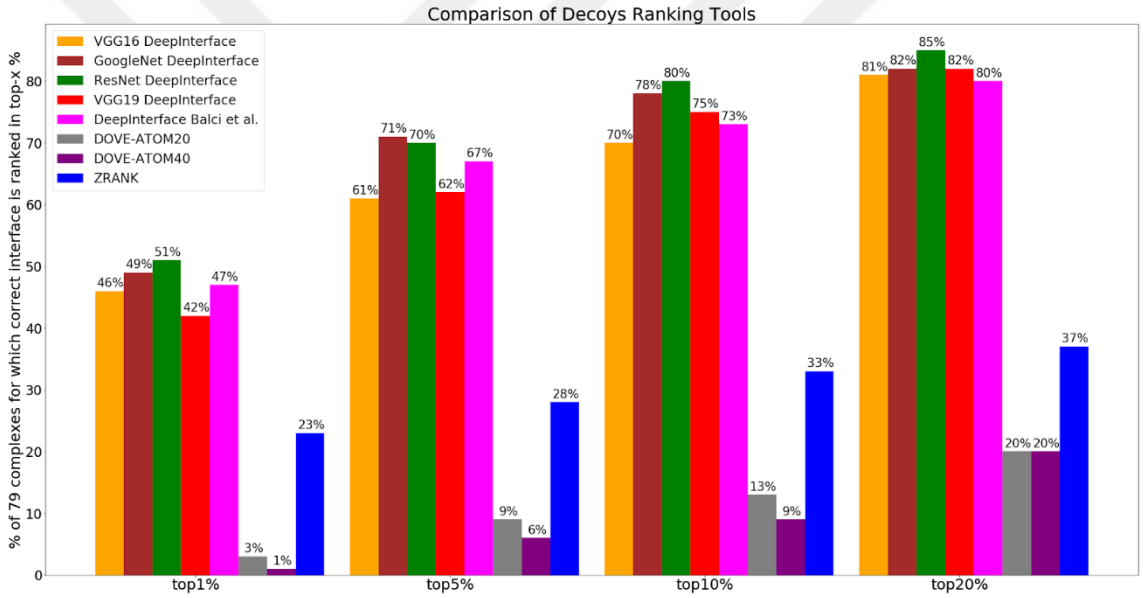

Figure 4.4: Comparison of decoys ranking tools on complexes from test dataset.

Ranks of all cases used in this analysis can be found in Table B.2 in Appendices.

4.4 Performance Analysis on ZDOCK Benchmark 4.0

For further analysis of DeepInterface models we compared their performance with docking models` scoring function IRAD [47]. Dataset used for this further performance analysis is widely used ZDOCK benchmark dataset version 4.0 [42].

IRAD [47], developed by Vreven et al., is a scoring function which combines the atom-based ZRANK [45] terms, van der Waals term separated into attractive and repulsive components, and electrostatics (short range and long range), with atom-based

statistical potential IFACE [46], and a variant of Zhang, Kolinski, and Skolnick [50] residue-based potential.

ZDOCK benchmark dataset version 4.0 [42] dataset contains 176 protein complexes each having on average 54,000 decoys produced with the ZDOCK 3.0.2 docking tool [43]. These complexes are divided into rigid-body, medium, and difficulty categories (123 rigid-body, 29 medium, 24 difficult). Produced decoys of complexes belonging to medium and rigid-body categories (total 152 complexes) are scored and ranked by IRAD scoring function [47]. There is no IRAD score provided for the decoys of complexes coming from difficulty category (24 complexes). Since IRAD scores for complexes belonging to the difficulty category are not provided we did not include these complexes in this analysis. Because DeepInterface models accept two chain protein-protein interfaces as input, only complexes having two chains were taken into consideration in this analysis; this resulted in 97 complexes left for the analysis. There are on average 54,000 decoys produced for each complex in the benchmark. Each decoy is provided with a Root-Mean-Square Deviation (RMSD) value. According to performance analyses of ZDOCK 3.0.2, decoys with RMSD value < 2.5 Å are called a "hit". We decided to use the same criteria for the "hits", and in our analysis search for the hits placed in the top x % of the ranked list produced by scoring algorithms we are comparing. Since we are searching for hits in this analysis, complexes that did not have any hit in their 54,000 decoys were discarded which resulted in 84 complexes left for the analysis. To prevent any bias in terms of similar protein-protein structures between the complexes used in this analysis and the training dataset we used for our DeepInterface models the redundant samples were discarded. A redundant sample is identified as a complex within 84 complexes from the ZDOCK benchmark dataset version 4.0 [42] to be used in this analysis that has more than 40% sequence similarity when compared to one or more samples coming from the training set. There was one such complex, and this complex was also discarded leaving us with total 83 complexes to use in this analysis. Complexes that are used in this analysis together with the rest of the complexes from the ZDOCK benchmark dataset version 4.0 [42] can be found in Table C.1 in Appendices.

For each of 83 complexes that are used in this analysis, 54,000 decoys were produced using ZDOCK 3.0.2 To save time and memory, not all 54,000 decoys were scored by DeepInterface models. We randomly selected 1,000 decoys per complex out of

54,000 possible decoys per complexes. Decoys were randomly sampled until at least one hit was present in the selected 1,000 decoys per complex.

To score the decoys with DeepInterface models we first processed them. DeepInterface models require data pre-processing explained in Section 4.3. DeepInterface models that are included in this comparison are: VGG16 DeepInterface, ResNet DeepInterface, GoogleNet DeepInterface, and the Balci et al. DeepInterface.

Since decoy protein-protein models were already scored and ranked by IRAD [47] in the ZDOCK benchmark dataset version 4.0 [42] that we used, we did not do any extra data processing or ranking to include IRAD in the comparison since scores by IRAD were already provided with the decoy structures.

This analysis results are shown in Figure 4.5 and Table 4.13. We compare all previously listed DeepInterface models with IRAD scoring algorithms using 83 complexes which extraction from the ZDOCK benchmark 4.0 was previously explained.

Table 4.13: Ranking of "hits" of the complexes from ZDOCK benchmark 4.0.

Ranking Tool	At least one hit in top 1%	At least one hit in top 5%	At least one hit in top 10%	At least one hit in top 20%
IRAD	5%	11%	13%	16%
VGG16 DeepInterface	4%	16%	24%	45%
GoogleNet DeepInterface	2%	12%	25%	49%
ResNet DeepInterface	1%	13%	25%	46%
VGG19 DeepInterface	2%	11%	20%	30%
DeepInterface Balci et al.	2%	13%	27%	46%

According to the success rates shown in Figure 4.5 we can see that in the top 1%, IRAD performs the best by placing at least one hit into top 1% structures for 5% of 83 complexes. At the second place is the VGG16 DeepInterface model with 4% success rate which is very close to 5% achieved by IRAD. On the other hand, when we look at the top 5%, top 10% and top 20% we can see that DeepInterface models perform better than IRAD in all cases. The best performing DeepInterface model in top 5% is the VGG16 DeepInterface model with 16% success rate. IRAD success rate in top 5% is much lower with 11%. In the top 10%, the best performing DeepInterface model is DeepInterface

Balci et al. with 27% success rate. When we look at IRAD we can see that its success rate did not increase much when compared to the top 5% and it equals 13%. And finally, in the top 20%, the best performing DeepInterface model is GoogleNet DeepInterface with 49% success rate. Again, when we look at IRAD we can see that its success rate did not increase much when compared to the top 5% or top 10% and it equals 16%.

To conclude, DeepInterface models are competitive when compared with IRAD in top 1%, and perform much better than IRAD in top 5%, top 10%, and top 20%.

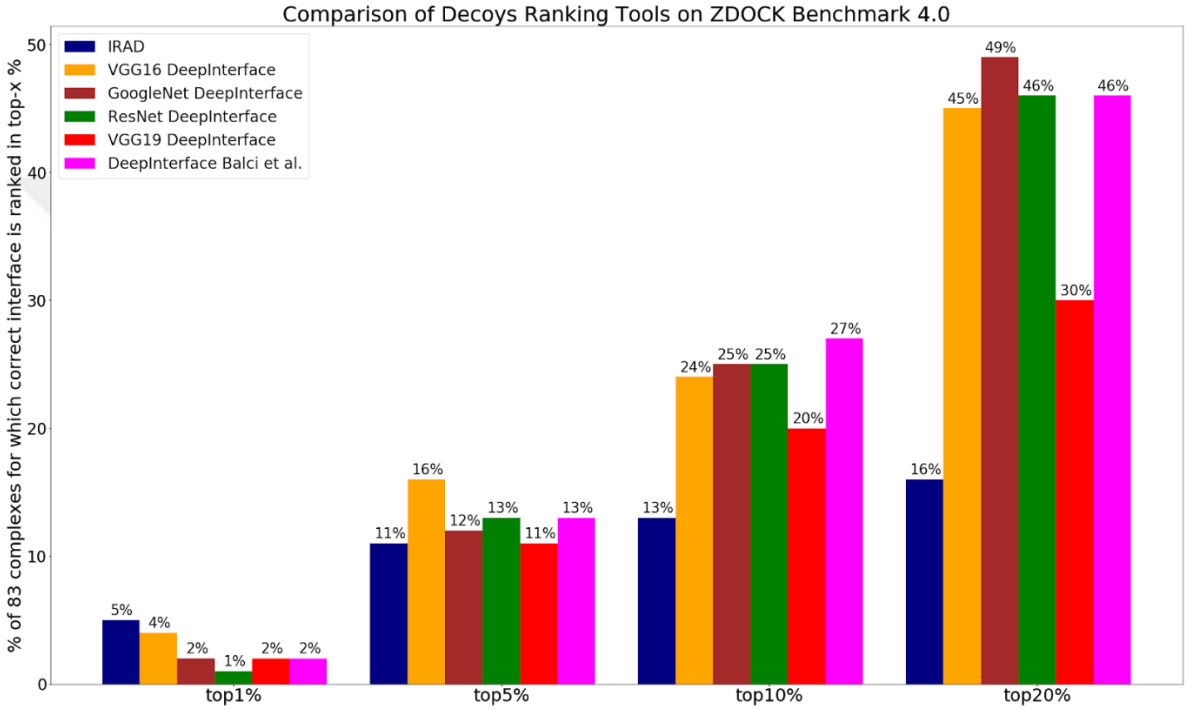

Figure 4.5: Comparison of decoys` scoring algorithms on ZDOCK Benchmark 4.0.

To summarize, DeepInterface models seem competitive in all top-x % comparisons, especially in top 5%, top 10% and top 20% where they keep improving much more than IRAD.

Chapter 5:
CONCLUSION

Protein-protein complex structures are crucial to understand mechanisms of protein binding and identify protein-protein interfaces. Knowledge of protein-protein interfaces aids characterization of cellular signaling pathways, and structure-based drug design. Therefore, it is important to be able to determine protein-protein interfaces easily and reliably. Since doing this experimentally takes time and effort, it is promising to see that computational docking tools can provide thousands of possible complex structures, decoys, in a short time. Multiple methods have been developed for scoring and ranking decoys to identify the reliable near-native complexes that can be used in further biological research. In this study, we improved a 3D convolutional neural network-based decoys` scoring and ranking approach called DeepInterface. As an input, DeepInterface uses atomic coordinates and amino acid types of the residues located at the decoy`s protein interface region to score the decoy and validate if it resembles protein-protein complexes deposited in the PDB repository.

We compared multiple CNN architectures and searched the hyperparameters` space to find the best performing model on the Validation dataset. Results show that the VGG16 resembling architecture with the first maxpool layer missing and filter size of 3x3x3 in the convolutional layers has highest accuracy on the test dataset used for models` testing.

Our results prove that the DeepInterface model can discriminate positive interfaces from the incorrect ones with 81% accuracy achieved on the test dataset. Furthermore, when we compared DeepInterface models with scoring/ranking tool called ZRANK and 3D CNN based model for decoys` ranking called DOVE on complexes from test dataset we showed that our models are far better than both ZRANK and DOVE in top 1%, top 5%, top 10%, and top 20%. We also compared the DeepInterface models with another scoring/ranking tool called IRAD on the ZDOCK benchmark 4.0 and showed that our models are competitive and perform nearly same with IRAD in top 1%, while they perform much better than IRAD in the top 5%, top 10% and top 20%.

Performance of the machine learning models greatly depends on the datasets used. Having big and complete space descriptive datasets largely improves the quality of the predictions made. As more complexes are being deposited to the PDB database and more

decoys are made available by docking servers, datasets that can be used by machine learning based models like DeepInterface will continue to expand which will eventually lead to better accuracy and higher reliability.

BIBLIOGRAPHY

[1] Keskin, O., Gursoy, A., Ma, B., & Nussinov, R. (2008). Principles of Protein−Protein Interactions: What are the Preferred Ways For Proteins To Interact? *Chemical Reviews, 108*(4), 1225-1244. doi:10.1021/cr040409x

[2] Tuncbag, N., Gursoy, A., Nussinov, R., & Keskin, O. (2011). Predicting protein-protein interactions on a proteome scale by matching evolutionary and structural similarities at interfaces using PRISM. *Nature protocols, 6*(9), 1341-1354.

[3] Ozlem, K., Nurcan, T., & Attila, G. (2008). Characterization and Prediction of Protein Interfaces to Infer Protein-Protein Interaction Networks. *Current Pharmaceutical Biotechnology, 9*(2), 67-76.

[4] Winter, C., Henschel, A., Tuukkanen, A., & Schroeder, M. (2012). Protein interactions in 3D: From interface evolution to drug discovery. *Journal of Structural Biology, 179*(3), 347-358.

[5] Jones, S., & Thornton, J. M. (1997). Analysis of protein-protein interaction sites using surface patches. *J Mol Biol, 272*(1), 121-132. doi:10.1006/jmbi.1997.1234

[6] Li, X., Keskin, O., Ma, B., Nussinov, R., & Liang, J. (2004). Protein-protein interactions: hot spots and structurally conserved residues often locate in complemented pockets that pre-organized in the unbound states: implications for docking. *J Mol Biol, 344*(3), 781-795. doi:10.1016/j.jmb.2004.09.051

[7] Tuncbag, N., Gursoy, A., Guney, E., Nussinov, R., & Keskin, O. (2008). Architectures and functional coverage of protein-protein interfaces. *J Mol Biol, 381*(3), 785-802. doi:10.1016/j.jmb.2008.04.071

[8] Baspinar, A., Cukuroglu, E., Nussinov, R., Keskin, O., & Gursoy, A. (2014). PRISM: a web server and repository for prediction of protein-protein interactions and modeling their 3D complexes. *Nucleic Acids Res, 42*(Web Server issue), W285-289. doi:10.1093/nar/gku397

[9] Aytuna, A. S., Gursoy, A., & Keskin, O. (2005). Prediction of protein-protein interactions by combining structure and sequence conservation in protein interfaces. *Bioinformatics, 21*(12), 2850-2855. doi:10.1093/bioinformatics/bti443

[10] Ogmen, U., Keskin, O., Aytuna, A. S., Nussinov, R., & Gursoy, A. (2005). PRISM: protein interactions by structural matching. *Nucleic Acids Res, 33*(Web Server issue), W331-336. doi:10.1093/nar/gki585

[11] Liang, S., Meroueh, S. O., Wang, G., Qiu, C., & Zhou, Y. (2009). Consensus scoring for enriching near-native structures from protein-protein docking decoys. *Proteins, 75*(2), 397-403.

[12] Nadalin, F., & Carbone, A. (2018). Protein-protein interaction specificity is captured by contact preferences and interface composition. *Bioinformatics, 34*(3), 459-468. doi:10.1093/bioinformatics/btx584

[13] Lensink, M. F., Velankar, S., Kryshtafovych, A., Huang, S. Y., Schneidman-Duhovny, D., Sali, A., . . . Wodak, S. J. (2016). Prediction of homoprotein and heteroprotein complexes by protein docking and template-based modeling: A CASP-CAPRI experiment. *Proteins, 84 Suppl 1*(Suppl Suppl 1), 323-348. doi:10.1002/prot.25007

[14] Anishchenko, I., Kundrotas, P. J., Tuzikov, A. V., & Vakser, I. A. (2015). Structural templates for comparative protein docking. *Proteins, 83*(9), 1563-1570. doi:10.1002/prot.24736

[15] Pierce, B. G., Hourai, Y., & Weng, Z. (2011). Accelerating Protein Docking in ZDOCK Using an Advanced 3D Convolution Library. *PLOS ONE, 6*(9), e24657. doi:10.1371/journal.pone.0024657

[16] Venkatraman, V., Yang, Y. D., Sael, L., & Kihara, D. (2009). Protein-protein docking using region-based 3D Zernike descriptors. *BMC Bioinformatics, 10*, 407. doi:10.1186/1471-2105-10-407

[17] Katchalski-Katzir, E., Shariv, I., Eisenstein, M., Friesem, A. A., Aflalo, C., & Vakser, I. A. (1992). Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proc Natl Acad Sci U S A, 89*(6), 2195-2199. doi:10.1073/pnas.89.6.2195

[18] Padhorny, D., Kazennov, A., Zerbe, B. S., Porter, K. A., Xia, B., Mottarella, S. E., . . . Kozakov, D. (2016). Protein–protein docking by fast generalized Fourier transforms on 5D rotational manifolds. *Proceedings of the National Academy of Sciences, 113*(30), E4286. doi:10.1073/pnas.1603929113

[19] Moal, I. H., & Bates, P. A. (2010). SwarmDock and the use of normal modes in protein-protein docking. *International journal of molecular sciences, 11*(10), 3623-3648.

[20] Fischer, D., Lin, S. L., Wolfson, H. L., & Nussinov, R. (1995). A geometry-based suite of moleculardocking processes. *J Mol Biol, 248*(2), 459-477.

[21] Gray, J. J., Moughon, S., Wang, C., Schueler-Furman, O., Kuhlman, B., Rohl, C. A., & Baker, D. (2003). Protein-protein docking with simultaneous optimization of rigid-body

displacement and side-chain conformations. *J Mol Biol, 331*(1), 281-299. doi:10.1016/s0022-2836(03)00670-3

[22] Oliwa, T., & Shen, Y. (2015). cNMA: a framework of encounter complex-based normal mode analysis to model conformational changes in protein interactions. *Bioinformatics, 31*(12), i151-i160.

[23] Alam, N., Goldstein, O., Xia, B., Porter, K. A., Kozakov, D., & Schueler-Furman, O. (2017). High-resolution global peptide-protein docking using fragments-based PIPER-FlexPepDock. *PLoS Comput Biol, 13*(12), e1005905. doi:10.1371/journal.pcbi.1005905

[24] Kurcinski, M., Jamroz, M., Blaszczyk, M., Kolinski, A., & Kmiecik, S. (2015). CABS-dock web server for the flexible docking of peptides to proteins without prior knowledge of the binding site. *Nucleic Acids Res, 43*(W1), W419-424. doi:10.1093/nar/gkv456

[25] Esquivel-Rodríguez, J., & Kihara, D. (2012). Fitting multimeric protein complexes into electron microscopy maps using 3D Zernike descriptors. *J Phys Chem B, 116*(23), 6854-6861. doi:10.1021/jp212612t

[26] Ritchie, D. W., & Grudinin, S. (2016). Spherical polar Fourier assembly of protein complexes with arbitrary point group symmetry. *Journal of Applied Crystallography, 49*(1), 158-167. doi:10.1107/S1600576715022931

[27] Schneidman-Duhovny, D., Inbar, Y., Nussinov, R., & Wolfson, H. J. (2005). Geometry-based flexible and symmetric protein docking. *Proteins, 60*(2), 224-231. doi:10.1002/prot.20562

[28] Peterson, L. X., Shin, W. H., Kim, H., & Kihara, D. (2018). Improved performance in CAPRI round 37 using LZerD docking and template-based modeling with combined scoring functions. *Proteins, 86 Suppl 1*(Suppl 1), 311-320. doi:10.1002/prot.25376

[29] Peterson, L. X., Togawa, Y., Esquivel-Rodriguez, J., Terashi, G., Christoffer, C., Roy, A., . . . Kihara, D. (2018). Modeling the assembly order of multimeric heteroprotein complexes. *PLoS Comput Biol, 14*(1), e1005937. doi:10.1371/journal.pcbi.1005937

[30] Peterson, L. X., Roy, A., Christoffer, C., Terashi, G., & Kihara, D. (2017). Modeling disordered protein interactions from biophysical principles. *PLoS Comput Biol, 13*(4), e1005485. doi:10.1371/journal.pcbi.1005485

[31] van Zundert, G. C. P., Melquiond, A. S. J., & Bonvin, A. (2015). Integrative Modeling of Biomolecular Complexes: HADDOCKing with Cryo-Electron Microscopy Data. *Structure, 23*(5), 949-960. doi:10.1016/j.str.2015.03.014

[32] Lensink, M. F., Velankar, S., Baek, M., Heo, L., Seok, C., & Wodak, S. J. (2018). The challenge of modeling protein assemblies: the CASP12-CAPRI experiment. *Proteins, 86 Suppl 1*, 257-273. doi:10.1002/prot.25419

[33] Kingsley, L. J., Esquivel-Rodríguez, J., Yang, Y., Kihara, D., & Lill, M. A. (2016). Ranking protein-protein docking results using steered molecular dynamics and potential of mean force calculations. *J Comput Chem, 37*(20), 1861-1865. doi:10.1002/jcc.24412

[34] Huang, S. Y., & Zou, X. (2008). An iterative knowledge-based scoring function for protein-protein recognition. *Proteins, 72*(2), 557-579. doi:10.1002/prot.21949

[35] Lu, H., Lu, L., & Skolnick, J. (2003). Development of unified statistical potentials describing protein-protein interactions. *Biophys J, 84*(3), 1895-1901. doi:10.1016/s0006-3495(03)74997-2

[36] Nadaradjane, A. A., Guerois, R., & Andreani, J. (2018). Protein-Protein Docking Using Evolutionary Information. *Methods Mol Biol, 1764*, 429-447. doi:10.1007/978-1-4939-7759-8_28

[37] Fink, F., Hochrein, J., Wolowski, V., Merkl, R., & Gronwald, W. (2011). PROCOS: computational analysis of protein-protein complexes. *J Comput Chem, 32*(12), 2575-2586. doi:10.1002/jcc.21837

[38] Wang, X., Terashi, G., Christoffer, C. W., Zhu, M., & Kihara, D. (2019). Protein docking model evaluation by 3D deep convolutional neural networks. *Bioinformatics, 36*(7), 2113-2118. doi:10.1093/bioinformatics/btz870

[39] Chen, H., & Zhou, H. X. (2005). Prediction of interface residues in protein-protein complexes by a consensus neural network method: test against NMR data. *Proteins, 61*(1), 21-35. doi:10.1002/prot.20514

[40] Popov, P., & Grudinin, S. (2015). Knowledge of Native Protein–Protein Interfaces Is Sufficient To Construct Predictive Models for the Selection of Binding Candidates. *Journal of Chemical Information and Modeling, 55*(10), 2242-2255. doi:10.1021/acs.jcim.5b00372

[41] T. Balcı, "Fast Screening Algorithms for Protein Interface Structural Alignments," MSc, Computer Science and Engineering, Koç University, 2018.

[42] Hwang, H., Vreven, T., Janin, J., & Weng, Z. (2010). Protein-protein docking benchmark version 4.0. *Proteins, 78*(15), 3111-3114. doi:10.1002/prot.22830

[43] Chen, R., Li, L., & Weng, Z. (2003). ZDOCK: an initial-stage protein-docking algorithm. *Proteins, 52*(1), 80-87. doi:10.1002/prot.10389

[44] Pierce, B., & Weng, Z. (2007). ZRANK: reranking protein docking predictions with an optimized energy function. *Proteins, 67*(4), 1078-1086. doi:10.1002/prot.21373

[45] Pierce, B., & Weng, Z. (2008). A combination of rescoring and refinement significantly improves protein docking performance. *Proteins, 72*(1), 270-279. doi:10.1002/prot.21920

[46] Mintseris, J., Pierce, B., Wiehe, K., Anderson, R., Chen, R., & Weng, Z. (2007). Integrating statistical pair potentials into protein complex prediction. *Proteins, 69*(3), 511-520. doi:10.1002/prot.21502

[47] Vreven, T., Hwang, H., & Weng, Z. (2011). Integrating atom-based and residue-based scoring functions for protein-protein docking. *Protein Sci, 20*(9), 1576-1586. doi:10.1002/pro.687

[48] Tobi, D., & Bahar, I. (2006). Optimal design of protein docking potentials: Efficiency and limitations. *Proteins: Structure, Function, and Bioinformatics, 62*(4), 970-981. doi:10.1002/prot.20859

[49] Glaser, F., Steinberg, D. M., Vakser, I. A., & Ben-Tal, N. (2001). Residue frequencies and pairing preferences at protein-protein interfaces. *Proteins, 43*(2), 89-102.

[50] Zhang, Y., Kolinski, A., & Skolnick, J. (2003). TOUCHSTONE II: a new approach to ab initio protein structure prediction. *Biophys J, 85*(2), 1145-1164. doi:10.1016/s0006-3495(03)74551-2

[51] Chakrabarti, P., & Janin, J. (2002). Dissecting protein-protein recognition sites. *Proteins, 47*(3), 334-343. doi:10.1002/prot.10085

[52] Hwang, H., Pierce, B., Mintseris, J., Janin, J., & Weng, Z. (2008). Protein-protein docking benchmark version 3.0. *Proteins, 73*(3), 705-709. doi:10.1002/prot.22106

[53] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation, 1*(4), 541-551. doi:10.1162/neco.1989.1.4.541

[54] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv 1409.1556.*

[55] Nair, V., & Hinton, G. (2010). *Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair* (Vol. 27).

[56] Szegedy, C., Wei, L., Yangqing, J., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015, 7-12 June 2015). *Going deeper with convolutions.* Paper presented at the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[57] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. ArXiv:1512.03385 [Cs].

[58] Xu, Q., & Dunbrack, R. L., Jr. (2011). The protein common interface database (ProtCID)--a comprehensive database of interactions of homologous proteins in multiple crystal forms. *Nucleic Acids Res, 39*(Database issue), D761-770. doi:10.1093/nar/gkq1059

[59] Jordan, R. A., El-Manzalawy, Y., Dobbs, D., & Honavar, V. (2012). Predicting protein-protein interface residues using local surface structural similarity. *BMC Bioinformatics, 13*, 41. doi:10.1186/1471-2105-13-41

[60] Cukuroglu, E., Gursoy, A., Nussinov, R., & Keskin, O. (2014). Non-redundant unique interface structures as templates for modeling protein interactions. *PLOS ONE, 9*(1), e86738. doi:10.1371/journal.pone.0086738

[61] Adhikari, B., Hou, J., & Cheng, J. (2018). DNCON2: improved protein contact prediction using two-level deep convolutional neural networks. *Bioinformatics, 34*(9), 1466-1472. doi:10.1093/bioinformatics/btx781

[62] Liu, Y., Palmedo, P., Ye, Q., Berger, B., & Peng, J. (2018). Enhancing Evolutionary Couplings with Deep Convolutional Neural Networks. *Cell Syst, 6*(1), 65-74.e63. doi:10.1016/j.cels.2017.11.014

[63] Liu, Y., Ye, Q., Wang, L., & Peng, J. (2018). Learning structural motif representations for efficient protein structure search. *Bioinformatics, 34*(17), i773-i780. doi:10.1093/bioinformatics/bty585

[64] Gainza, P., Sverrisson, F., Monti, F., Rodolà, E., Boscaini, D., Bronstein, M. M., & Correia, B. E. (2020). Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods, 17*(2), 184-192. doi:10.1038/s41592-019-0666-6

[65] Torng, W., & Altman, R. B. (2017). 3D deep convolutional neural networks for amino acid environment similarity analysis. *BMC Bioinformatics, 18*(1), 302. doi:10.1186/s12859-017-1702-0

[66] Maddhuri Venkata Subramaniya, S. R., Terashi, G., & Kihara, D. (2019). Protein secondary structure detection in intermediate-resolution cryo-EM maps using deep learning. *Nat Methods, 16*(9), 911-917. doi:10.1038/s41592-019-0500-1

[67] Ragoza, M., Hochuli, J., Idrobo, E., Sunseri, J., & Koes, D. R. (2017). Protein–Ligand Scoring with Convolutional Neural Networks. *Journal of Chemical Information and Modeling, 57*(4), 942-957. doi:10.1021/acs.jcim.6b00740

[68] Derevyanko, G., Grudinin, S., Bengio, Y., & Lamoureux, G. (2018). Deep convolutional networks for quality assessment of protein folds. *Bioinformatics, 34*(23), 4046-4053. doi:10.1093/bioinformatics/bty494

[69] Pagès, G., Charmettant, B., & Grudinin, S. (2019). Protein model quality assessment using 3D oriented convolutional neural networks. *Bioinformatics, 35*(18), 3313-3319. doi:10.1093/bioinformatics/btz122

[70] Jiménez, J., Doerr, S., Martínez-Rosell, G., Rose, A. S., & De Fabritiis, G. (2017). DeepSite: protein-binding site predictor using 3D-convolutional neural networks. *Bioinformatics, 33*(19), 3036-3042. doi:10.1093/bioinformatics/btx350

[71] Zhou, H., & Skolnick, J. (2011). GOAP: a generalized orientation-dependent, all-atom statistical potential for protein structure prediction. *Biophys J, 101*(8), 2043-2052. doi:10.1016/j.bpj.2011.09.012

[72] Yu, J., & Guerois, R. (2016). PPI4DOCK: large scale assessment of the use of homology models in free docking over more than 1000 realistic targets. *Bioinformatics, 32*(24), 3760-3767. doi:10.1093/bioinformatics/btw533

[73] Liu, S., Gao, Y., & Vakser, I. A. (2008). DOCKGROUND protein-protein docking decoy set. *Bioinformatics, 24*(22), 2634-2635. doi:10.1093/bioinformatics/btn497

[74] Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., . . . Bourne, P. E. (2000). The Protein Data Bank. *Nucleic Acids Res, 28*(1), 235-242. doi:10.1093/nar/28.1.235

[75] Faure, G., Andreani, J., & Guerois, R. (2012). InterEvol database: exploring the structure and evolution of protein complex interfaces. *Nucleic Acids Res, 40*(Database issue), D847-856. doi:10.1093/nar/gkr845

[76] Zhu, H., Domingues, F. S., Sommer, I., & Lengauer, T. (2006). NOXclass: prediction of protein-protein interaction types. *BMC Bioinformatics, 7*, 27. doi:10.1186/1471-2105-7-27

[77] Söding, J. (2005). Protein homology detection by HMM-HMM comparison. *Bioinformatics, 21*(7), 951-960. doi:10.1093/bioinformatics/bti125

[78] Zhang, Y., & Skolnick, J. (2005). TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Res, 33*, 2302-2309. doi:10.1093/nar/gki524

[79] Gao, M., & Skolnick, J. (2010). iAlign: a method for the structural comparison of protein-protein interfaces. *Bioinformatics, 26*(18), 2259-2265. doi:10.1093/bioinformatics/btq404

[80] Song, Y., DiMaio, F., Wang, R. Y., Kim, D., Miles, C., Brunette, T., . . . Baker, D. (2013). High-resolution comparative modeling with RosettaCM. *Structure, 21*(10), 1735-1742. doi:10.1016/j.str.2013.08.005

[81] Tovchigrechko, A., & Vakser, I. A. (2006). GRAMM-X public web server for protein-protein docking. *Nucleic Acids Res, 34*(Web Server issue), W310-314. doi:10.1093/nar/gkl206

[82] Hubbard SJ, Thornton JM: NACCESS. *Computer Program, Department of Biochemistry and Molecular Biology*, University College London,1993.

[83] Shatsky, M., Nussinov, R., & Wolfson, H. J. (2004). A method for simultaneous alignment of multiple protein structures. *Proteins, 56*(1), 143-156. doi:10.1002/prot.10628

[84] Girvan, M., Callaway, D. S., Newman, M. E., & Strogatz, S. H. (2002). Simple model of epidemics with pathogen mutation. *Phys Rev E Stat Nonlin Soft Matter Phys, 65*(3 Pt 1), 031915. doi:10.1103/PhysRevE.65.031915

[85] Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. (2015). Julia: A Fresh Approach to Numerical Computing. doi:10.1137/141000671

[86] D. Yuret, "Knet: beginning deep learning with 100 lines of julia,", Machine Learning Systems Workshop at NIPS, 2016, vol. 2016, p. 5

[87] Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

[88] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research, 15*, 1929-1958.

[89] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., . . . Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*.

[90] Kingma, D., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.

[91] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *IEEE International Conference on Computer Vision (ICCV 2015), 1502*. doi:10.1109/ICCV.2015.123

[92] Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track, 9*, 249-256.

[93] Nesterov, Y. A method of solving a convex programming problem with convergence rate O(1/k^2). *Soviet Mathematics Doklady*. 1983; ***27***(2): pp. 372-376.

APPENDIX A

EVALUATION METRICS

Table A.1: Evaluation metrics of VGG resembling DeepInterface models that achieved >=81% accuracy on the validation dataset.

Model name	Accuracy	Loss	Recall	Specificity	Precision	NPV
Layers combination category						
vgg19bn_nomaxpool_1_3_model						
Training	0.948	0.139	0.948	0.948	0.948	0.948
Validation	0.807	0.855	0.898	0.715	0.759	0.875
Test	0.816	0.906	0.910	0.722	0.766	0.889
vgg19bn_nomaxpool_5_model						
Training	0.909	0.229	0.913	0.906	0.906	0.913
Validation	0.810	0.865	0.885	0.735	0.770	0.864
Test	0.814	0.887	0.885	0.742	0.775	0.866
vgg11bn_nomaxpool_1_model						
Training	0.936	0.169	0.935	0.937	0.937	0.936
Validation	0.803	1.079	0.891	0.716	0.758	0.868
Test	0.809	1.363	0.895	0.723	0.764	0.873
vgg16bn_nomaxpool_3_model						
Training	0.942	0.155	0.942	0.943	0.943	0.942
Validation	0.803	1.585	0.923	0.683	0.744	0.899
Test	0.808	1.665	0.923	0.692	0.750	0.900
vgg13bn_nomaxpool_2_model						
Training	0.948	0.141	0.948	0.947	0.947	0.948
Validation	0.803	1.047	0.916	0.689	0.747	0.892
Test	0.809	1.139	0.917	0.701	0.754	0.894
vgg16bn_nomaxpool_1_3_5_model						
Training	0.947	0.141	0.948	0.947	0.947	0.948
Validation	0.788	0.906	0.871	0.705	0.747	0.845

Test	0.791	0.945	0.877	0.705	0.748	0.851
vgg16bn_nomaxpool_2_model						
Training	0.932	0.179	0.931	0.933	0.932	0.932
Validation	0.798	1.112	0.830	0.767	0.781	0.818
Test	0.797	1.216	0.829	0.765	0.779	0.817
vgg13bn_nomaxpool_1_model						
Training	0.932	0.179	0.931	0.933	0.933	0.931
Validation	0.807	0.788	0.877	0.737	0.769	0.857
Test	0.809	0.881	0.879	0.739	0.771	0.859
vgg16bn_nomaxpool_1_model						
Training	0.946	0.144	0.945	0.947	0.947	0.945
Validation	0.806	1.132	0.928	0.683	0.745	0.905
Test	0.808	1.324	0.926	0.690	0.749	0.903
vgg19bn_nomaxpool_4_model						
Training	0.943	0.151	0.943	0.944	0.944	0.943
Validation	0.804	0.856	0.895	0.712	0.757	0.872
Test	0.808	0.891	0.898	0.719	0.761	0.876
Filter 5x5x5 category						
vgg13bn_nomaxpool_1_3_filter5_model						
Training	0.936	0.170	0.937	0.935	0.935	0.937
Validation	0.807	1.095	0.944	0.670	0.741	0.923
Test	0.810	1.247	0.945	0.676	0.744	0.924
vgg13bn_nomaxpool_2_filter5_model						
Training	0.950	0.135	0.949	0.951	0.951	0.949
Validation	0.810	1.106	0.932	0.689	0.750	0.910
Test	0.814	1.199	0.932	0.696	0.754	0.910
L2 regularization category						
vgg19bn_nomaxpool_5_l2_reg_1e-06_model						
Training	0.910	0.231	0.910	0.911	0.911	0.910
Validation	0.799	2.570	0.899	0.699	0.749	0.874

Test	0.807	2.727	0.908	0.706	0.755	0.885
vgg16bn_ nomaxpool_1_l2_reg_0.0001_model						
Training	0.923	0.201	0.922	0.924	0.924	0.922
Validation	0.807	0.690	0.916	0.697	0.752	0.892
Test	0.810	0.695	0.914	0.706	0.756	0.892
vgg19bn_nomaxpool_5_l2_reg_0.0001_model						
Training	0.937	0.168	0.934	0.940	0.940	0.934
Validation	0.803	1.091	0.951	0.654	0.733	0.930
Test	0.805	1.117	0.950	0.660	0.737	0.930
vgg19bn_nomaxpool_5_l2_reg_1e-05_model						
Training	0.934	0.179	0.933	0.934	0.934	0.933
Validation	0.793	1.138	0.949	0.637	0.723	0.925
Test	0.800	1.239	0.952	0.647	0.730	0.931
Dropout category						
vgg13bn_nomaxpool_2_dropout_0.2_model						
Training	0.920	0.210	0.923	0.916	0.917	0.923
Validation	0.804	0.750	0.945	0.664	0.738	0.923
Test	0.812	0.771	0.945	0.678	0.746	0.925
vgg16bn_nomaxpool_1_dropout_0.2_model						
Training	0.902	0.253	0.907	0.897	0.898	0.906
Validation	0.799	0.591	0.911	0.686	0.744	0.885
Test	0.806	0.589	0.908	0.704	0.754	0.884
vgg13bn_nomaxpool_1_3_5_dropout_0.2_model						
Training	0.918	0.211	0.923	0.914	0.915	0.922
Validation	0.807	0.552	0.906	0.707	0.756	0.883
Test	0.809	0.543	0.902	0.715	0.760	0.880
vgg19bn_nomaxpool_1_3_dropout_0.2_model						
Training	0.914	0.222	0.919	0.909	0.910	0.919
Validation	0.800	0.554	0.930	0.671	0.739	0.905
Test	0.810	0.550	0.931	0.689	0.750	0.909
vgg11bn_nomaxpool_1_dropout_0.2_model						
Training	0.920	0.208	0.923	0.916	0.917	0.923

Validation	0.807	0.733	0.918	0.697	0.752	0.895
Test	0.808	0.809	0.912	0.704	0.755	0.889
vgg16bn_nomaxpool_2_dropout_0.5_model						
Training	0.875	0.315	0.876	0.874	0.874	0.876
Validation	0.770	0.560	0.945	0.594	0.700	0.916
Test	0.780	0.546	0.944	0.616	0.711	0.916
Learning rate decay category						
vgg19bn_ nomaxpool_5_lr_decay_0.3_model						
Training	0.967	0.095	0.965	0.969	0.969	0.965
Validation	0.808	1.711	0.944	0.671	0.742	0.923
Test	0.815	1.747	0.949	0.680	0.748	0.931
vgg16bn_nomaxpool_1_lr_decay_0.2_model						
Training	0.966	0.094	0.964	0.968	0.968	0.964
Validation	0.807	1.393	0.948	0.665	0.739	0.927
Test	0.810	1.564	0.950	0.671	0.743	0.931
vgg16bn_nomaxpool_1_lr_decay_0.1_model						
Training	0.963	0.100	0.962	0.964	0.964	0.962
Validation	0.806	1.197	0.941	0.670	0.740	0.919
Test	0.813	1.297	0.945	0.681	0.747	0.925
vgg19bn_nomaxpool_5_lr_decay_0.4_model						
Training	0.969	0.088	0.965	0.972	0.972	0.966
Validation	0.812	1.703	0.941	0.684	0.748	0.921
Test	0.816	1.796	0.941	0.691	0.753	0.921
vgg16bn_nomaxpool_1_lr_decay_0.6_model						
Training	0.931	0.181	0.931	0.932	0.932	0.931
Validation	0.815	0.903	0.918	0.713	0.762	0.897
Test	0.817	0.983	0.918	0.717	0.765	0.897
vgg19bn_nomaxpool_5_lr_decay_0.6_model						
Training	0.960	0.111	0.958	0.961	0.961	0.958
Validation	0.805	2.020	0.936	0.674	0.742	0.914
Test	0.812	2.034	0.943	0.681	0.747	0.923

vgg19bn_nomaxpool_5_lr_decay_0.5_model						
Training	0.947	0.143	0.947	0.947	0.947	0.947
Validation	0.810	3.019	0.928	0.692	0.751	0.906
Test	0.814	3.636	0.932	0.697	0.754	0.911
vgg16bn_nomaxpool_1_lr_decay_0.5_model						
Training	0.969	0.087	0.968	0.970	0.970	0.968
Validation	0.803	2.630	0.939	0.666	0.738	0.916
Test	0.807	2.871	0.940	0.674	0.742	0.918
vgg16bn_nomaxpool_1_lr_decay_0.8_model						
Training	0.977	0.066	0.975	0.978	0.978	0.975
Validation	0.804	4.456	0.949	0.658	0.735	0.929
Test	0.811	4.712	0.954	0.668	0.742	0.936
vgg16bn_nomaxpool_1_lr_decay_0.9_model						
Training	0.962	0.103	0.961	0.964	0.964	0.961
Validation	0.812	0.981	0.917	0.707	0.758	0.895
Test	0.816	1.095	0.917	0.715	0.763	0.896
vgg16bn_nomaxpool_1_lr_decay_0.3_model						
Training	0.968	0.087	0.967	0.970	0.969	0.967
Validation	0.804	1.208	0.939	0.668	0.739	0.916
Test	0.809	1.260	0.940	0.679	0.745	0.919
vgg19bn_nomaxpool_5_lr_decay_0.2_model						
Training	0.957	0.119	0.954	0.960	0.960	0.954
Validation	0.800	0.857	0.927	0.673	0.739	0.902
Test	0.806	0.879	0.930	0.682	0.746	0.907
vgg19bn_nomaxpool_5_lr_decay_0.8_model						
Training	0.955	0.123	0.954	0.957	0.957	0.954
Validation	0.808	1.467	0.901	0.714	0.759	0.878
Test	0.810	1.516	0.902	0.718	0.762	0.880
vgg16bn_nomaxpool_1_lr_decay_0.4_model						
Training	0.953	0.127	0.950	0.955	0.955	0.951
Validation	0.810	1.216	0.937	0.683	0.747	0.915
Test	0.813	1.322	0.935	0.691	0.752	0.915

Batch size category						
vgg16bn_nomaxpool_1_32_batch_model						
Training	0.923	0.201	0.922	0.924	0.924	0.922
Validation	0.807	0.690	0.916	0.697	0.752	0.892
Test	0.810	0.695	0.914	0.706	0.756	0.892
vgg19bn_nomaxpool_5_32_batch_model						
Training	0.937	0.168	0.934	0.940	0.940	0.934
Validation	0.803	1.091	0.951	0.654	0.733	0.930
Test	0.805	1.117	0.950	0.660	0.737	0.930
vgg19bn_nomaxpool_5_128_batch_model						
Training	0.931	0.178	0.934	0.928	0.928	0.934
Validation	0.803	2.427	0.880	0.726	0.763	0.858
Test	0.806	2.539	0.881	0.731	0.766	0.860

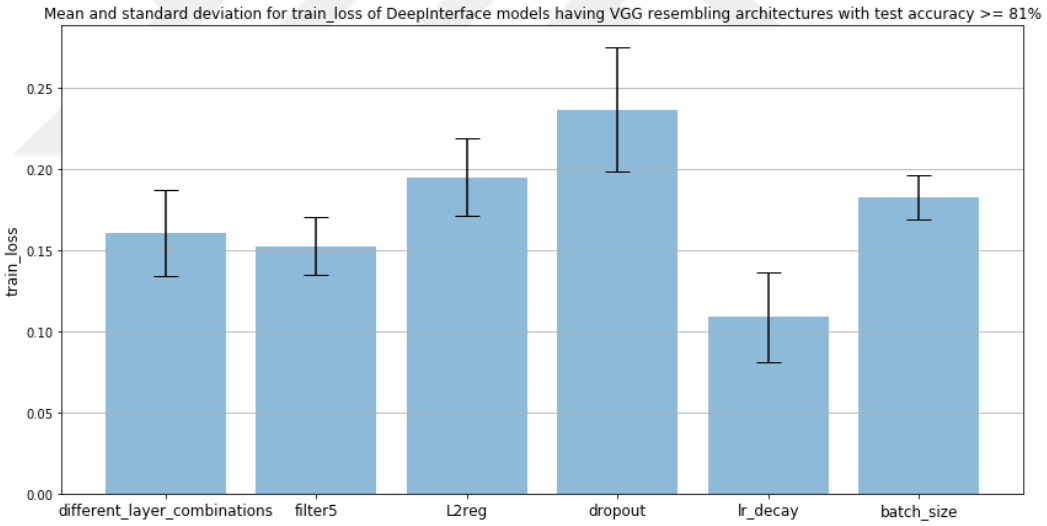

Figure A.1: Mean and standard deviation for loss metric of Training dataset.

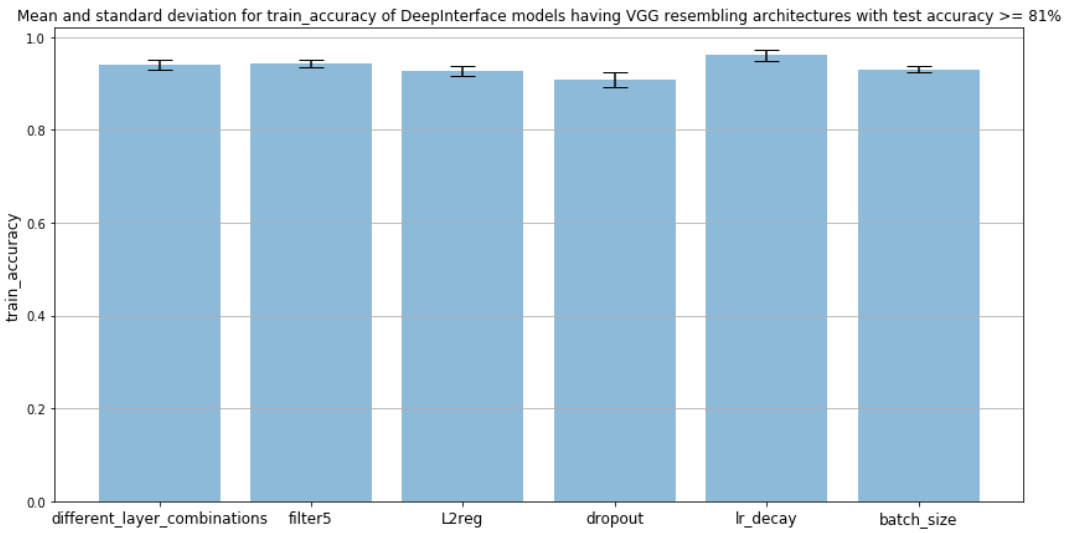

Figure A.2: Mean and standard deviation for accuracy metric of Training dataset.

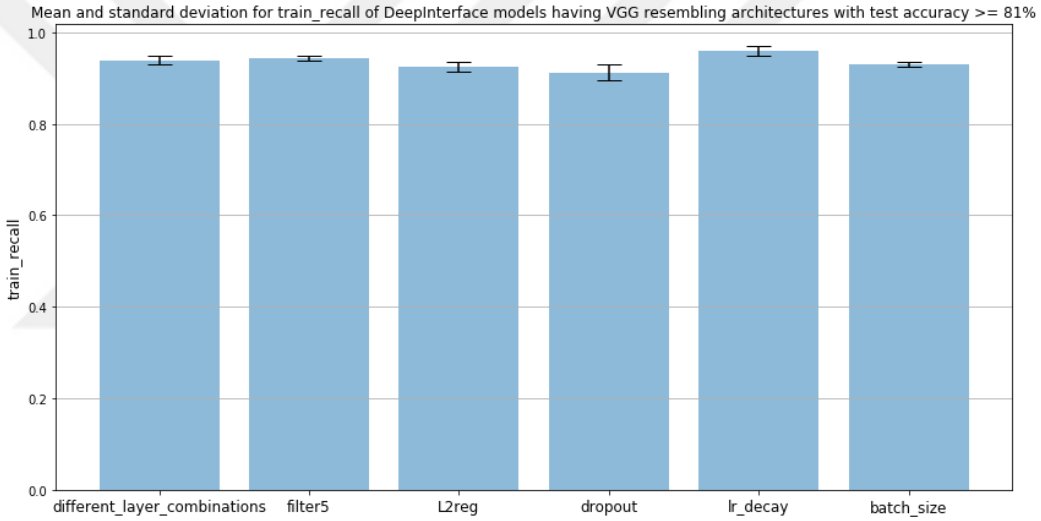

Figure A.3: Mean and standard deviation for recall metric of Training dataset.

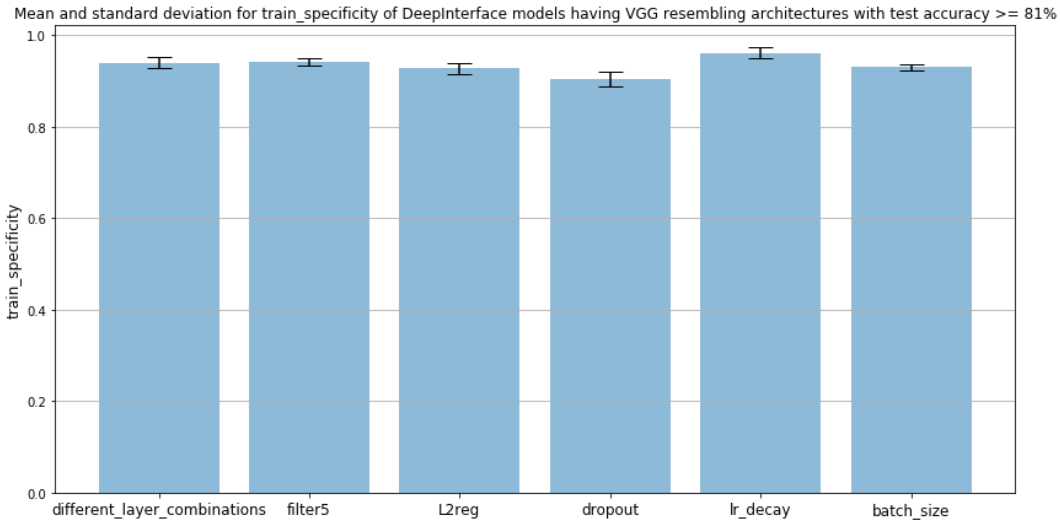

Figure A.4: Mean and standard deviation for specificity metric of Training dataset.

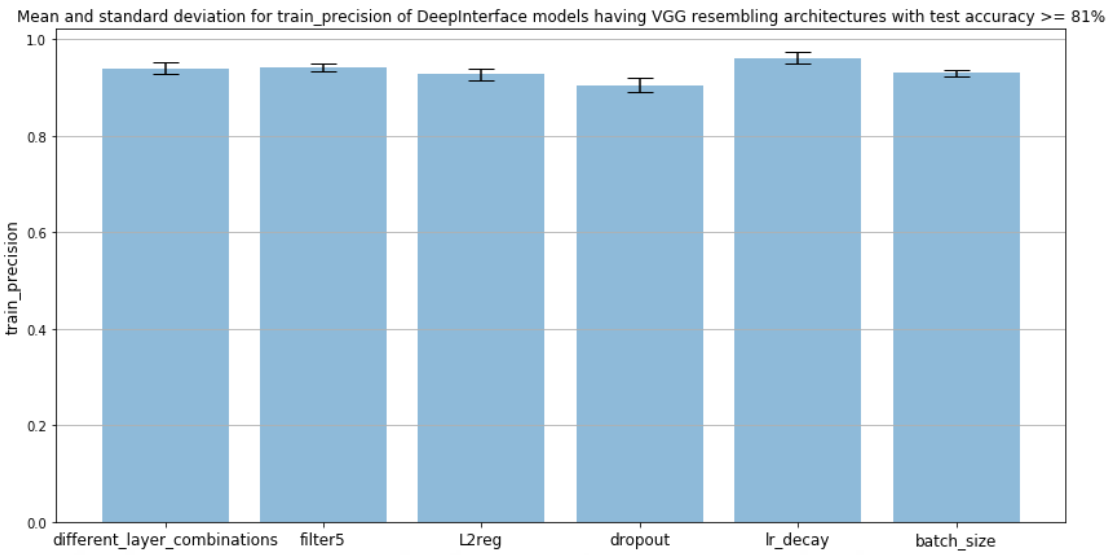

Figure A 5: Mean and standard deviation for precision metric of Training dataset.

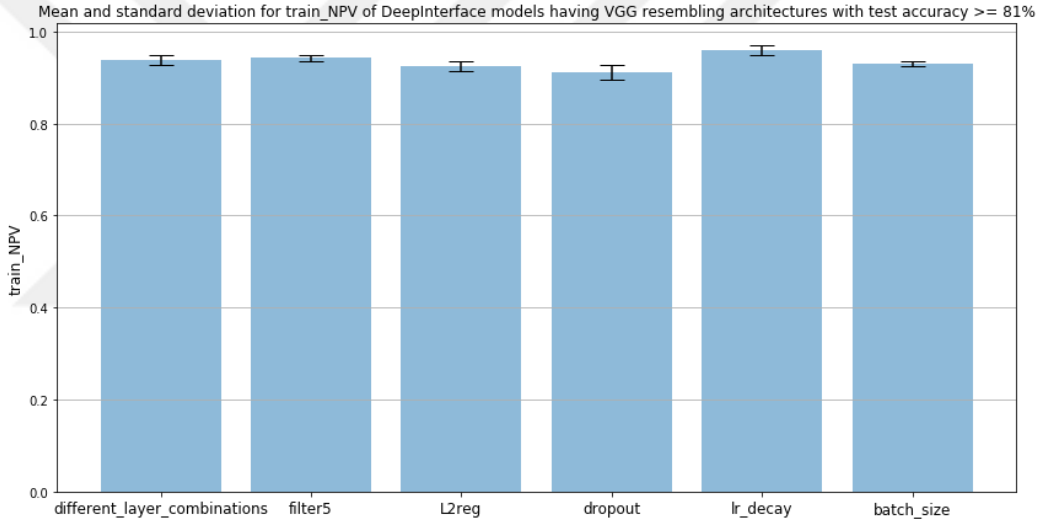

Figure A.6: Mean and standard deviation for NPV metric of Training dataset.

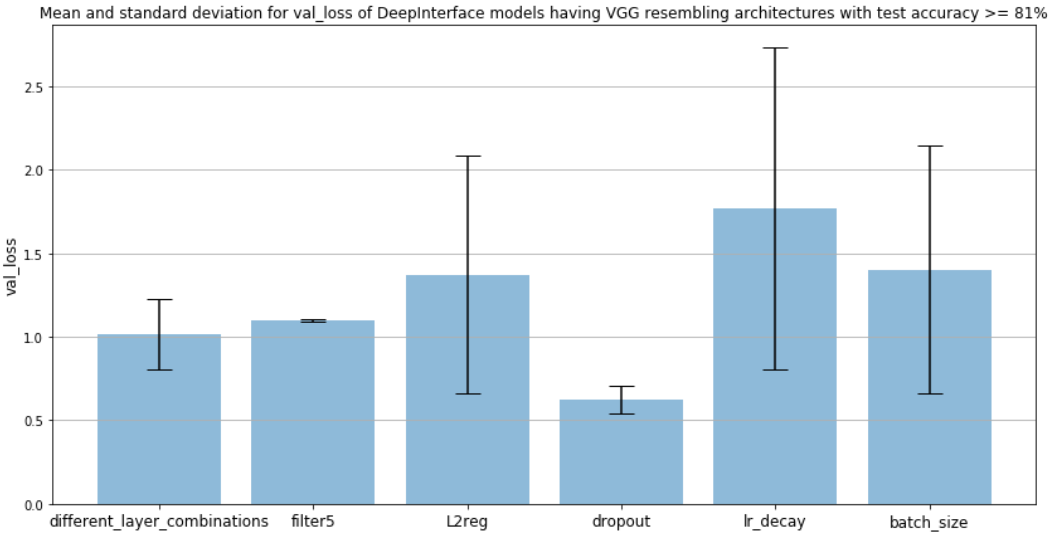

Figure A.7: Mean and standard deviation for loss metric of Validation dataset.

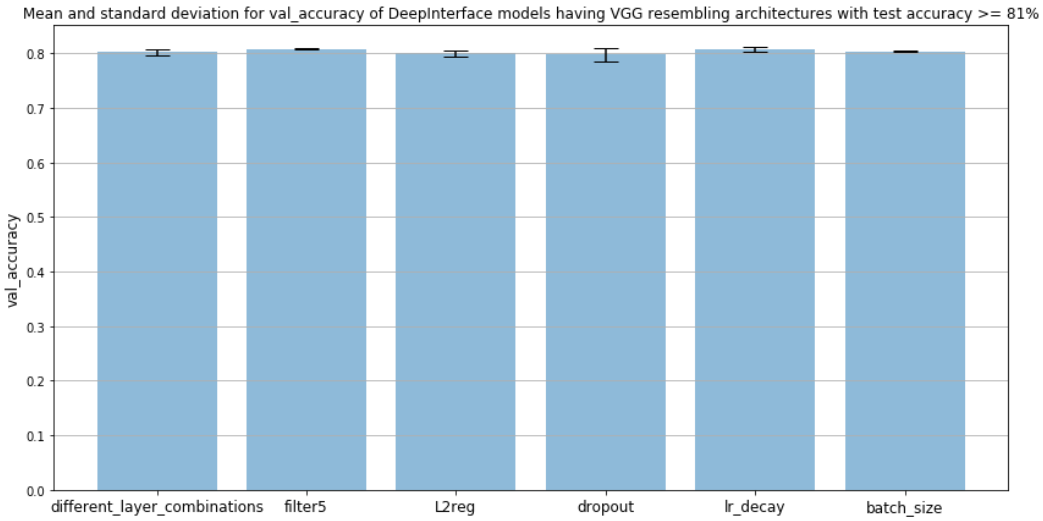

Figure A.8: Mean and standard deviation for accuracy metric of Validation dataset.

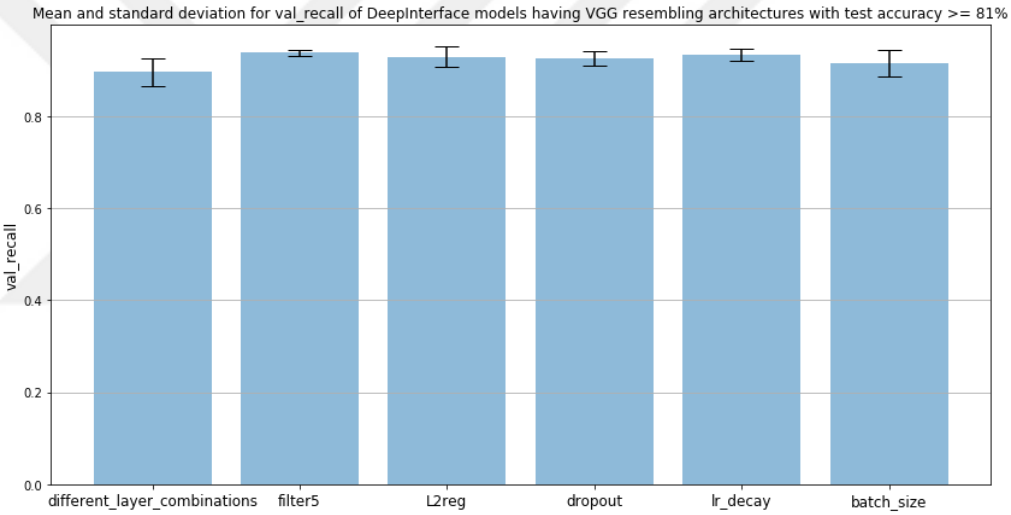

Figure A.9: Mean and standard deviation for recall metric of Validation dataset.

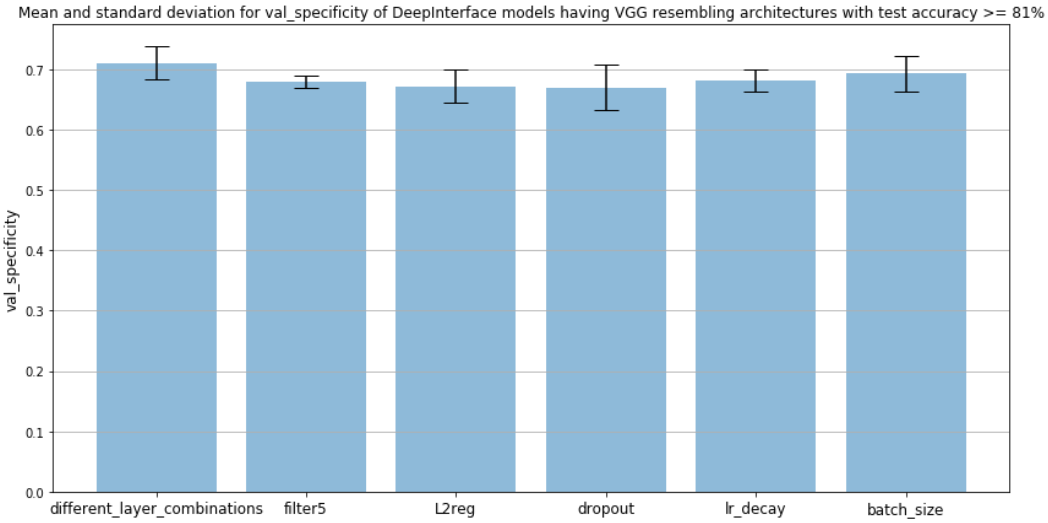

Figure A.10: Mean and standard deviation for specificity metric of Validation dataset.

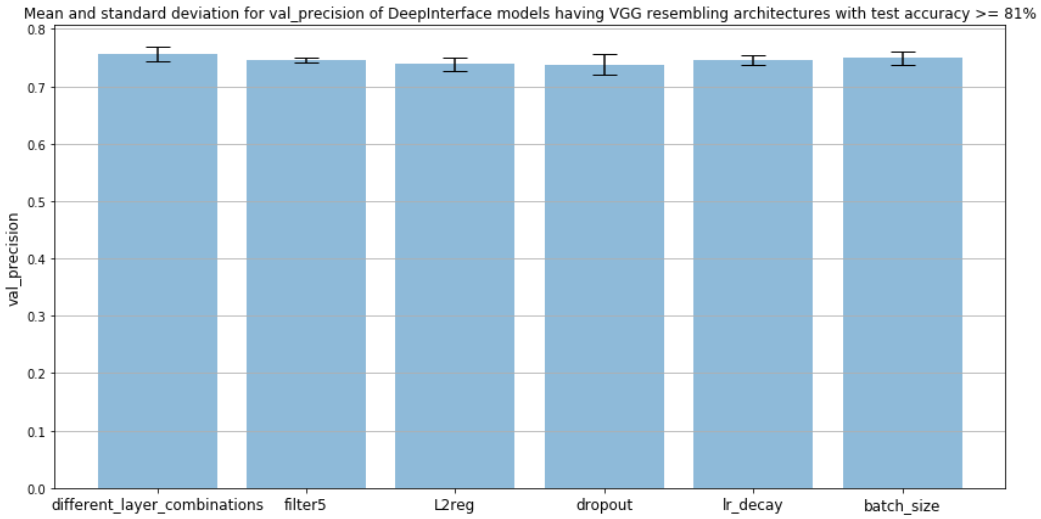

Figure A.11: Mean and standard deviation for precision metric of Validation dataset.

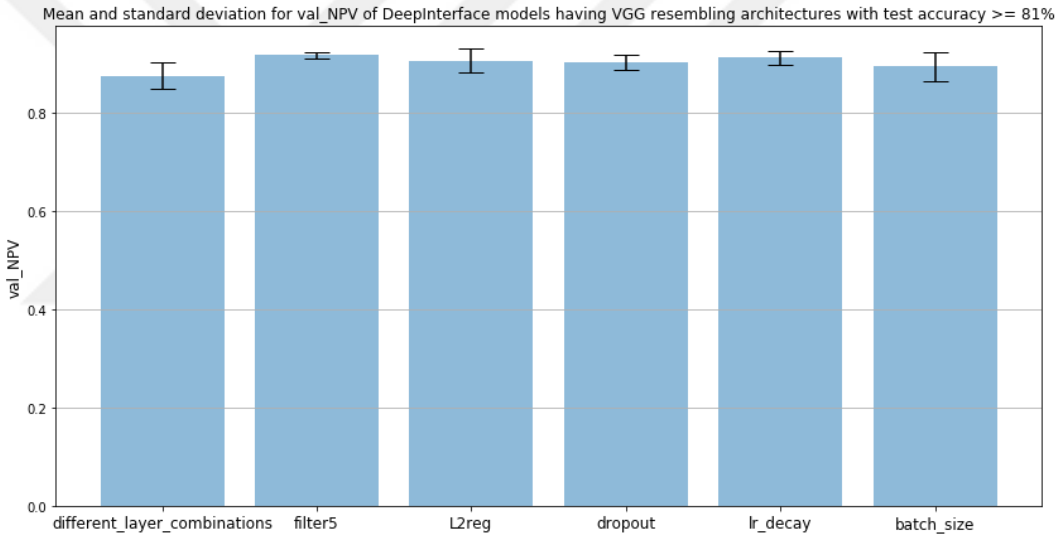

Figure A.12: Mean and standard deviation for NPV metric of Validation dataset.

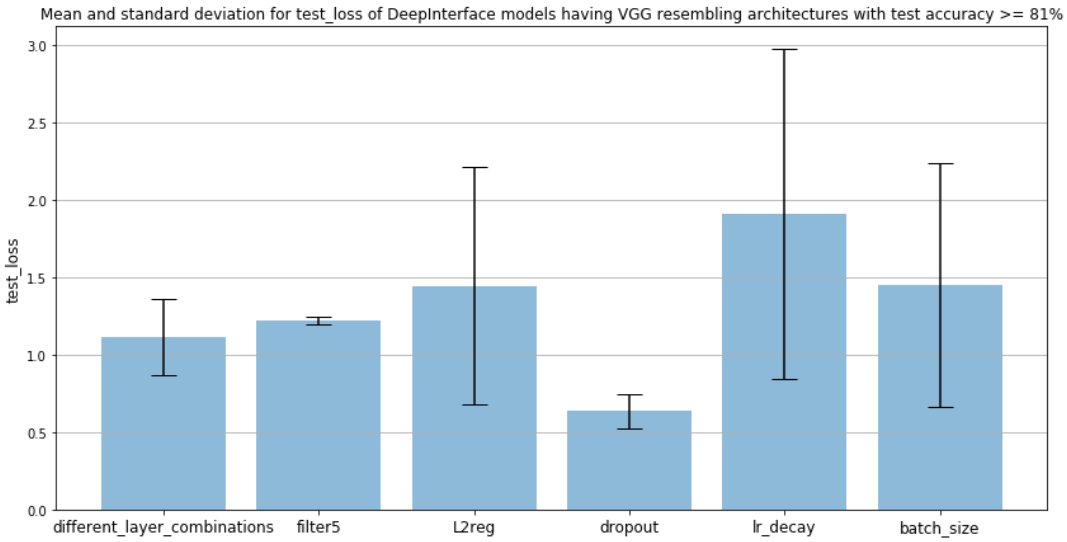

Figure A.13: Mean and standard deviation for loss metric of Test dataset.

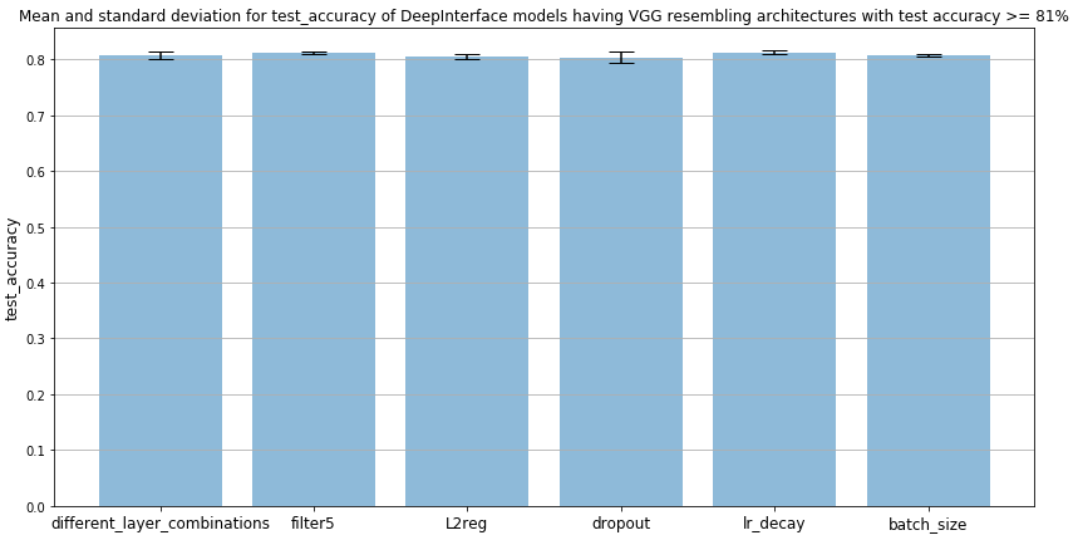

Figure A.14: Mean and standard deviation for accuracy metric of Test dataset.

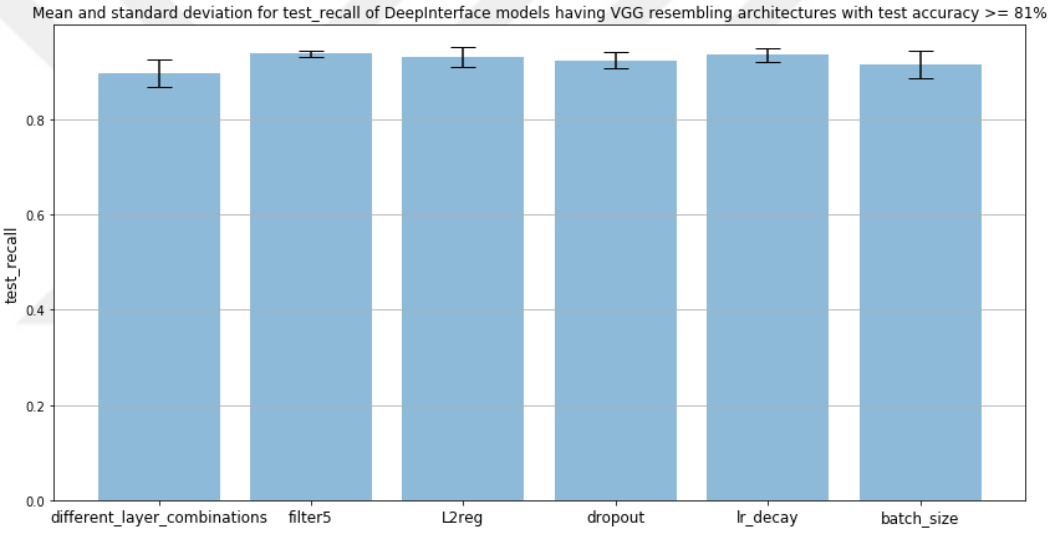

Figure A.15: Mean and standard deviation for recall metric of Test dataset.

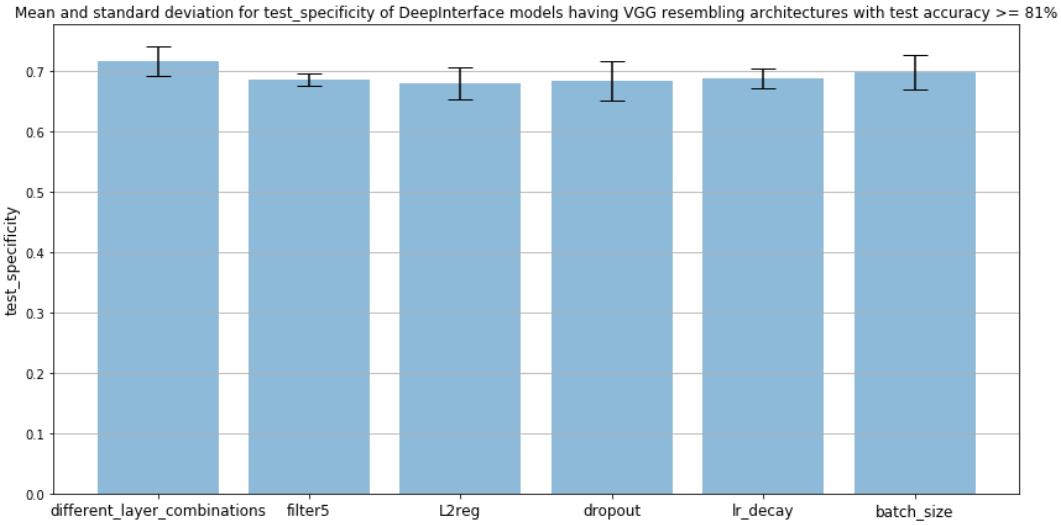

Figure A.16: Mean and standard deviation for specificity metric of Test dataset.

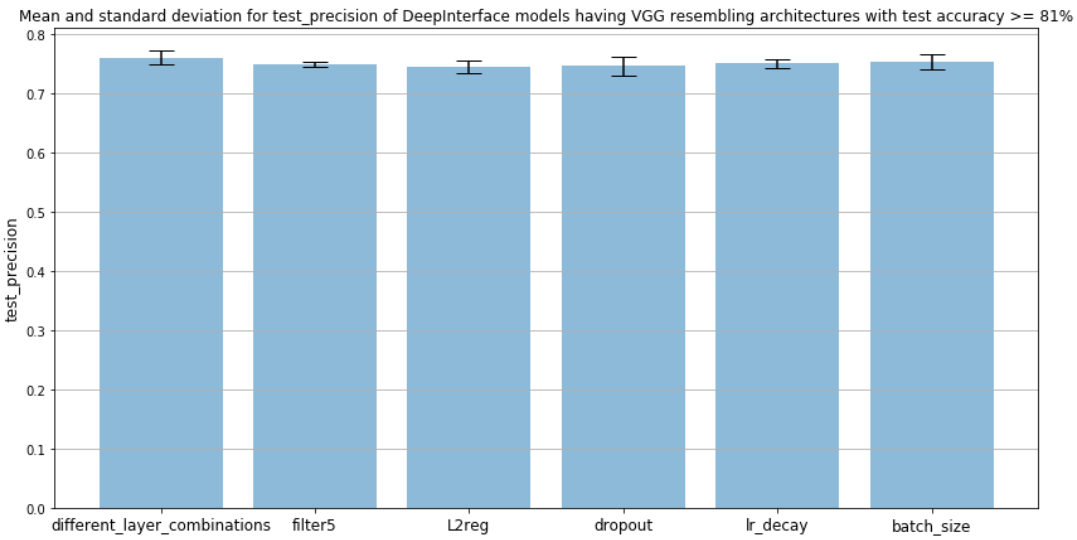

Figure A.17: Mean and standard deviation for precision metric of Test dataset.

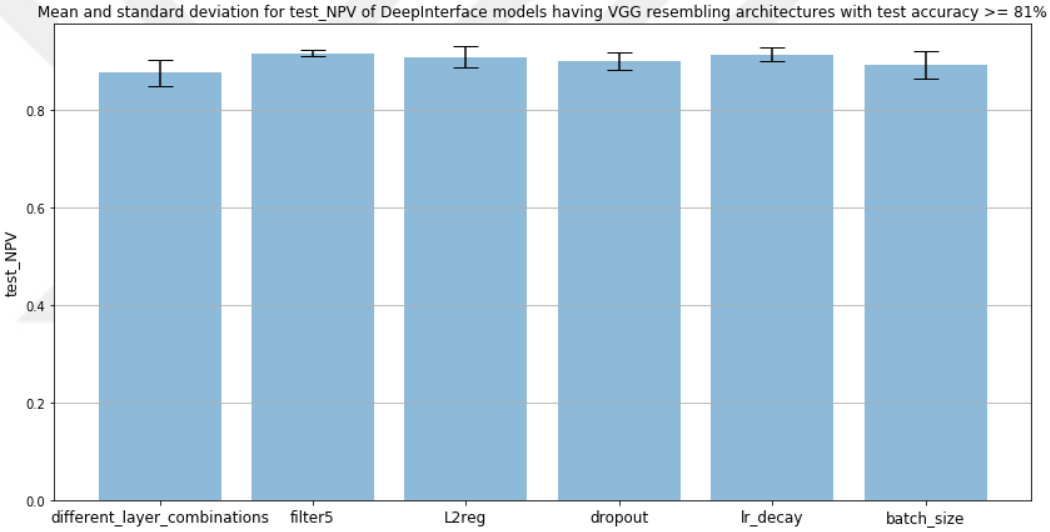

Figure A.18: Mean and standard deviation for NPV metric of Test dataset.

APPENDIX B

PERFORMANCE ANALYSIS

Table B.1: Native protein-protein complexes used in comparison analysis.

Complexes used in comparison analysis with ZRANK (100 cases)	4ov6AF, 5d6oAB, 5ir1CF, 5mdqBC, 5ensAD, 4o6oCD, 4wykBD, 4p4vAB, 5ps9AB, 5erbAC, 5wsvAD, 4ry8AC, 4ui0AC, 4wfhBG, 5djcAF, 5hxzCD, 4q3pAC, 5jynAB, 4ov6BE, 5duvAB, 5mqqCD, 4ylzBC, 5j3wCD, 5psxAB, 5fgbAF, 5f7dAC, 5b0lAF, 4r6nDF, 4uexAB, 5cvnBD, 4x96CD, 5tzfDI, 5llfAC, 5n1uAB, 5pufAB, 5ilqAC, 5mk1DK, 5adtAB, 5uwhBC, 5n8pAC, 4z0xBC, 5j1mAD, 4zrbFG, 5unmCF, 5m50BC, 4y29AB, 5b86AB, 5pynAB, 4owkEF, 5fglAC, 5xldCD, 5wqiBC, 5puiAB, 5dhvBN, 5q1bCD, 4qqiAX, 4xixEH, 5kdmBD, 4qvgCD, 5udqBC, 5vlcBE, 5vnzAC, 5hz0AB, 5jnxEH, 4ufsAB, 4r62AB, 4cihAB, 5lcnAC, 4wp2DE, 4xm4AB, 4u1yBD, 4qy4AC, 4ycgBD, 3wwbAB, 4q0zBF, 5volCF, 5dxmAC, 5v72BC, 5mk3AC, 5im3AB, 4cz2AD, 4q28BD, 4d3lAC, 4mzpEG, 5eurBC, 5pwaAB, 4y32BD, 5sviAC, 5mydCD, 4oiaAB, 5k52AD, 4qylDH, 4csyAB, 4ygxAB, 5ejcBC, 5a04CF, 5e1iAB, 4ng0AB, 5fttEF, 5aumCH
Complexes used in comparison analysis with DOVE (79 cases)	4ov6AF, 5d6oAB, 5ir1CF, 5mdqBC, 5ensAD, 4o6oCD, 5erbAC, 5wsvAD,

	4ry8AC, 4ui0AC, 4wfhBG, 5djcAF, 5hxzCD, 4q3pAC, 4ov6BE, 5duvAB, 5mqqCD, 4ylzBC, 5j3wCD, 5fgbAF, 5f7dAC, 5b0lAF, 4r6nDF, 5cvnBD, 4x96CD, 5tzfDI, 5llfAC, 5ilqAC, 5mk1DK, 5adtAB, 5n8pAC, 5j1mAD, 4zrbFG, 5unmCF, 5m50BC, 4y29AB, 5b86AB, 5fglAC, 5xldCD, 5wqiBC, 5dhvBN, 5q1bCD, 4qqiAX, 5kdmBD, 4qvgCD, 5udqBC, 5vlcBE, 5vnzAC, 5jnxEH, 4ufsAB, 4r62AB, 4cihAB, 5lcnAC, 4u1yBD, 4qy4AC, 4ycgBD, 3wwbAB, 4q0zBF, 5volCF, 5dxmAC, 5v72BC, 5mk3AC, 5im3AB, 4cz2AD, 4q28BD, 4mzpEG, 5eurBC, 4y32BD, 5sviAC, 5mydCD, 4oiaAB, 5k52AD, 4csyAB, 4ygxAB, 5ejcBC, 5a04CF, 5e1iAB, 4ng0AB, 5fttEF
Complexes that could not be processed by DOVE (21 cases)	4wykBD, 4p4vAB, 5ps9AB, 5jynAB, 5psxAB, 4uexAB, 5n1uAB, 5pufAB, 5uwhBC, 4z0xBC, 5pynAB, 4owkEF, 5puiAB, 4xixEH, 5hz0AB, 4wp2DE, 4xm4AB, 4d3lAC, 5pwaAB, 4qylDH, 5aumCH

Table B.2: Correct interface ranks by different ranking tools (DI stands for DeepInterface, ZR stands for ZRANK, D-20 stands for DOVE-ATOM20, D-40 stands for DOVE-ATOM40).

	VGG16 DI	GoogleNet DI	ResNet DI	VGG19 DI	Balci et al. DI	ZR	D-20	D-40
4ov6AF	1	1	1	1	1	100	89	100
5d6oAB	1	1	8	5	2	1	60	1

5ir1CF	62	23	25	80	31	88	6	88
5mdqBC	29	25	6	3	14	1	15	1
5ensAD	1	1	1	1	1	98	88	98
4o6oCD	1	2	6	2	3	2	66	2
5erbAC	10	1	1	1	4	98	101	98
5wsvAD	1	1	1	3	1	100	71	100
4ry8AC	42	6	4	2	3	1	39	1
4ui0AC	56	31	25	50	31	44	72	44
4wfhBG	16	24	13	10	36	98	95	98
5djcAF	1	1	1	7	1	101	18	101
5hxzCD	1	16	17	4	76	1	29	1
4q3pAC	13	1	1	4	1	101	101	101
4ov6BE	1	2	1	2	1	101	99	101
5duvAB	10	1	1	47	4	96	60	96
5mqqCD	1	1	1	1	1	10	66	10
4ylzBC	3	13	1	18	1	63	36	63
5j3wCD	2	1	1	7	1	100	13	100
5fgbAF	1	3	2	1	4	99	5	99
5f7dAC	9	5	3	1	14	1	1	1
5b0lAF	16	1	1	3	2	98	38	98
4r6nDF	2	2	2	3	1	101	71	101
5cvnBD	2	5	4	15	22	1	36	1
4x96CD	3	1	1	1	1	101	19	101
5tzfDI	1	1	1	1	1	95	28	95
5llfAC	1	4	4	21	11	12	53	12
5ilqAC	10	2	2	5	1	5	15	5
5mk1DK	8	1	17	27	4	6	51	6
5adtAB	1	10	3	1	6	1	30	1
5n8pAC	1	1	2	1	2	101	81	101
5j1mAD	30	2	53	29	1	99	36	99
4zrbFG	18	46	46	44	30	1	98	1
5unmCF	4	1	1	1	1	96	89	96

5m50BC	1	1	1	7	1	101	45	101
4y29AB	23	2	2	8	1	1	62	1
5b86AB	1	1	1	1	1	101	98	101
5fglAC	1	1	1	1	1	100	101	100
5xldCD	1	1	1	1	1	100	100	100
5wqiBC	36	10	2	60	9	1	100	1
5dhvBN	10	1	47	69	19	84	67	84
5q1bCD	13	3	5	12	3	1	3	1
4qqiAX	1	1	1	7	4	3	101	3
5kdmBD	1	1	1	1	1	100	17	100
4qvgCD	1	1	1	1	1	101	101	101
5udqBC	1	1	1	3	1	71	96	71
5vlcBE	1	1	1	1	1	100	66	100
5vnzAC	1	1	1	1	1	95	50	95
5jnxEH	35	1	1	43	8	101	39	101
4ufsAB	5	2	2	3	3	11	72	11
4r62AB	1	2	1	2	1	101	86	101
4cihAB	43	7	1	13	28	36	9	36
5lcnAC	1	3	4	1	1	1	25	1
4u1yBD	1	1	1	1	2	101	69	101
4qy4AC	1	1	1	1	1	100	79	100
4ycgBD	6	59	56	18	55	1	5	1
3wwbAB	37	76	17	17	56	9	42	9
4q0zBF	19	28	9	7	25	25	100	25
5volCF	42	20	27	54	91	3	101	3
5dxmAC	1	1	10	1	3	1	3	1
5v72BC	1	1	1	1	1	98	74	98
5mk3AC	5	1	1	1	1	92	61	92
5im3AB	2	2	4	1	4	10	98	10
4cz2AD	82	70	70	96	100	16	83	16
4q28BD	1	1	1	1	1	101	56	101
4mzpEG	1	1	1	1	1	101	62	101

5eurBC	17	53	51	51	52	1	15	1
4y32BD	3	29	68	1	75	49	30	49
5sviAC	81	64	70	7	39	1	101	1
5mydCD	32	42	7	78	15	81	26	81
4oiaAB	4	1	1	1	1	100	81	100
5k52AD	1	1	1	1	2	96	100	96
4csyAB	1	1	1	1	1	101	97	101
4ygxAB	12	10	9	6	7	1	12	1
5ejcBC	3	1	1	1	1	80	45	80
5a04CF	1	5	6	2	1	1	35	1
5e1iAB	16	9	1	1	8	99	61	99
4ng0AB	1	2	1	2	1	100	17	100
5fttEF	28	33	25	8	27	58	73	58
5aumCH	29	23	8	27	57	1	N/A	N/A
4qylDH	32	55	72	17	76	20	N/A	N/A
5pwaAB	5	19	20	6	9	100	N/A	N/A
4d3lAC	1	1	1	1	1	99	N/A	N/A
4wp2DE	1	2	1	24	5	97	N/A	N/A
4xm4AB	4	3	3	2	4	1	N/A	N/A
4xixEH	1	1	1	1	1	101	N/A	N/A
5hz0AB	50	5	39	68	64	1	N/A	N/A
4uexAB	94	8	33	86	30	24	N/A	N/A
5n1uAB	6	4	3	10	4	23	N/A	N/A
5pufAB	5	31	8	4	10	101	N/A	N/A
5puiAB	4	23	12	2	16	96	N/A	N/A
5pynAB	3	2	1	18	1	66	N/A	N/A
4owkEF	59	22	1	68	16	97	N/A	N/A
4z0xBC	2	19	30	27	84	23	N/A	N/A
5uwhBC	1	1	1	1	1	93	N/A	N/A
5psxAB	9	37	28	3	16	100	N/A	N/A
5jynAB	18	3	1	2	1	97	N/A	N/A
4wykBD	27	1	1	13	7	100	N/A	N/A

4p4vAB	28	5	5	16	4	1	N/A	N/A
5ps9AB	2	24	14	1	19	100	N/A	N/A

APPENDIX C

ZDOCK BENCHMARK 4.0

Table C.1: ZDOCK benchmark 4.0 complexes used in comparison analysis.

all complexes of ZDOCK benchmark 4.0 (176 cases)	1SBB, 1JPS, 2HMI, 1GHQ, 1KTZ, 1K74, 1D6R, 2SIC, 1SYX, 1XD3, 1EAW, 1VFB, 4CPA, 7CEI, 1XU1, 1E4K, 2O3B, 1KLU, 1OC0, 1H1V, 2PCC, 1H9D, 2Z0E, 1FQ1, 2IDO, 2HLE, 1FQJ, 1S1Q, 2NZ8, 1RV6, 1UDI, 2OOR, 1MQ8, 1GL1, 1WQ1, 1CGI, 1ATN, 1GCQ, 1GP2, 1FAK, 1NW9, 1I9R, 1GRN, 1GXD, 1WDW, 1LFD, 2OUL, 1AZS, 1JMO, 2AYO, 1PXV, 1EWY, 1RLB, 1US7, 1DQJ, 3SGQ, 1FFW, 2BTF, 2I25, 1I2M, 1BUH, 3BP8, 1BGX, 2HQS, 1EFN, 1DFJ, 1Y64, 2UUY, 2VDB, 1PVH, BOYV, 1JWH, 1MAH, 1BVK, 1BVN, 1OFU, 1JK9, 1ZM4, 2FJU, 2A5T, 1EER, 1MLC, 1NSN, 1AK4, 2ABZ, 1A2K, 1ZLI, 1QFW, 2H7V, 2B4J, 1FLE, 2AJF, 1KAC, 1YVB, 1J2J, 1CLV, 1QA9, 1AHW, 2OT3, 2FD6, 1T6B, 1K4C, 1HCF, 1OYV, 2OZA, 1NCA, 2J0T, 1OPH, 1XQS, 1B6C, 1PPE, 2O8V, 1HIA, 1Z0K, 1R0R, 1F6M, 1WEJ, 1ACB, 1KXP, 1KXQ, 1R8S, 1IRA, 1N2C, 2QFW, 1F51, 2B42, 1ML0, 1AKJ, 2JEL, 2J7P, 1GLA, 1KKL, 1FC2, 1E96, 1N8O, 1ZHH, 2MTA, 2VIS, 1IB1, 1E6J, 1JIW, 3CPH, 1Z5Y, 1EZU, 1I4D, 1TMQ, 1JTG, 2C0L, 1GPW, 1E6E, 3D5S, 1IQD, 1FCC, 1ZHI, 1M10, 2A9K, 2OOB, 1AY7, 1HE8, 1IJK, 2HRK, 1HE1, 1R6Q, 1FSK, 1F34, 2SNI, 1BJ1, 2CFH, 1BKD, 2G77, 2I9B, 1DE4, 1IBR, 1JZD, 1K5D, 1AVX
discarded difficult complexes (24 cases)	1E4K, 2HMI, 1F6M, 1FQ1, 1PXV, 1ZLI, 2O3B, 1ATN, 1BKD, 1DE4, 1EER, 1FAK, 1H1V, 1IBR, 1IRA, 1JK9, 1JMO, 1JZD, 1R8S, 1Y64, 2C0L, 2I9B, 2IDO, 2OT3

discarded complexes with more than two chains (55 cases)	1AHW, 1BVK, 1DQJ, 1E6J, 1JPS, 1MLC, 1VFB, 1WEJ, 2FD6, 2VIS, 1BJ1, 1FSK, 1I9R, 1IQD, 1K4C, 1NCA, 1NSN, 1QFW, 2QFW, 2JEL, 1EZU, 1HIA, 2MTA, 1A2K, 1AKJ, 1AZS, 1F51, 1FC2, 1FCC, 1HCF, 1I4D, 1JWH, 1K74, 1KLU, 1KXP, 1ML0, 1OFU, 1RLB, 1RV6, 1WDW, 1XU1, 2B4J, 2BTF, 2OOR, 3BP8, 1BGX, 1IJK, 1KKL, 1GP2, 1GRN, 1IB1, 1K5D, 1N2C, 1XQS, 2H7V
discarded complexes with no hit in the 54,000 produced decoys (13 cases)	2Z0E, 1ZHH, BOYV, 2A5T, 2OZA, 2J7P, 1JIW, 3CPH, 1GHQ, 1D6R, 1ACB, 1M10, 1HE8
discarded complex with >40% sequence similarity with the training dataset (1 case)	1EWY
complexes included in comparison of DeepInterface models with IRAD (83 cases)	1SYX, 1XD3, 4CPA, 7CEI, 1MQ8, 1OC0, 1H9D, 2HLE, 1FQJ, 2NZ8, 1GL1, 1GXD, 1NW9, 1GPW, 1LFD, 2OUL, 2AYO, 1US7, 1YVB, 1FFW, 2I25, 1I2M, 2HQS, 1EFN, 2UUY, 2VDB, 1PVH, 1ZM4, 2FJU, 2ABZ, 1GLA, 1FLE, 2AJF, 1KAC, 1JTG, 1J2J, 1CLV, 1T6B, 1OYV, 2J0T, 1OPH, 1N8O, 1B6C, 2O8V, 1Z0K, 1R0R, 1GCQ, 2B42, 2G77, 1S1Q, 1Z5Y, 1E6E, 3D5S, 1ZHI, 2A9K, 2OOB, 1AY7, 2HRK, 1R6Q, 3SGQ, 2SNI, 2CFH, 1AVX, 1KTZ, 1EAW, 1UDI, 1WQ1, 1CGI, 1BUH, 1DFJ, 1MAH, 1AK4, 1QA9, 1PPE, 1KXQ, 1E96, 1TMQ, 1HE1, 1F34, 1SBB, 2SIC, 2PCC, 1BVN