ABSTRACT

Motivation: Protein forms specific molecular complexes and specificity of its interaction is highly essential for discovering and analyzing functional roles. Protein complexes are common in vital processes such as catalysis, immunity and cellular metabolic regulations. Protein-Protein interactions are found to form homo or hetero complexes. There are numerous methods to study protein-protein interactions, includes *in vitro, in vivo* and computational approaches. *In vitro* and *in vivo* methods promise higher reliability and accuracy on results but lag in terms of speed, time and high throughput. Computational approaches are far better than *in vitro* and *in vivo* methods in terms of cost, time, and efficiency. Recently there is huge acceleration in terms of uncovering novel protein-protein interaction due to the development of large-scale high-throughput experiments. In computational approaches, the domain-domain interaction based methods majorly emphasize on protein domain features which can more effectively predict PPIs. In this thesis, an integrative domain-based method is proposed for predicting PPIs using Deep Neural Network (DNN).

Results: Various relevant features of proteins were exploited from the Database of Interacting Proteins (DIP) and Kansas University Proteomics Service (KUPS). In this article, domain-frequency based method is proposed by using DNN. When performed on the PPI data of ***Saccharomyces cerevisiae***, the method achieved a very promising prediction result. An independent data set of 14,200 yeast PPIs was used to evaluate this prediction model and the **prediction accuracy, precision, sensitivity** and **specificity** obtained was **92.67%, 98.31%, 86.85%** and **98.51%** respectively. Similar method was applied to PPI datasets of different size of ***Escherichia coli, Drosophila melanogaster, Homo sapiens*** and ***Caenorhabditis elegans,*** where accuracy was found to be **97.01%, 90.85%, 94.47%** and **88.91%** respectively. The performance of this method is superior to those of the existing domain-based machine learning methods, so this approach can be useful for predicting novel protein-protein interactions with high sensitivity and specificity. A web-based server was implemented on various organisms such as *Saccharomyces cerevisiae, Caenorhabditis elegans, Drosophila melanogaster, Homo sapiens* and *Escherichia coli*. The information related to datasets, prediction software, and supplementary files used in this paper are available on server interface - **http://bioserver.iiita.ac.in/deepinteract**.

TABLE OF CONTENTS

TABLE OF FIGURES

INTRODUCTION

Proteins are the most abundant biomolecules found in all living cells. The role of individual protein and its interaction with other proteins are crucial for several cellular processes. Clear cut understanding of Protein-Protein Interaction (PPI) of various cellular processes like signal transduction, enzymatic pathway and cellular mechanism is required to unravel the function of protein. In recent past PPI has been extensively studied in the prospect of biochemistry and metabolic networks to decipher its importance in maintaining the structural and functional integrity of living system. It has been seen that the abnormalities in the interactions among proteins may cause several abnormalities like Huntington disease, Alzheimer, cystic fibrosis (Gonzalez., *et.al*, 2012). Therefore the analysis of PPI is significant to understand the complex cellular mechanism and diseased cellular conditions of an organism and its interactome, which consists of thousands of such interactions (Sanchez C, 1999). It has been reported that the size of an organism's interactome correlates better with the biological complexity of the organism, than the genome size; which implies complexity of an organism can be estimated by considering the size of its interactome (Stumpf MP, 2008). It is further reported that *Saccharomyces cerevisae* which has about 6,000 proteins produces more than 20,000 interactions (Ganten D, 2006), which leads to the hypothesis that the human body with 2 million proteins may show more than 100,000 functional interactions. With the advent in the field of proteomics and instrumentation, various experimental and computational methods have been performed to identify and characterize PPIs. Several *in vitro* and *in vivo* methods like affinity purification, yeast 2 hybrid, tandem affinity purification, co-immuno-precipitations, pull down assays, binding reaction method, protein chips, mass spectrometry and far-Western blot analysis have been utilized to identify PPIs systematically (Uetz P, 2000; Fields S, 1989; Zhu H, 2001; Vasilescu J, 2004; Wu Y, 2007) for last few years, although these classical approaches were less cost effective and needs skilled manpower with their own associated limitations.

In search of faster and cost effective alternatives, various rule based computational methods have been developed for identifying PPI through a series of phylogenetic, structure based and physiochemical descriptors (Pazos F, 2001; Zhang QC, 2012; Block P, 2006). These methods differ in feature information and solely rely on the prior knowledge about the interacting protein pairs. Therefore in recent years, dozens of machine learning based prediction models have been proposed for the prediction of PPI. Some of the well-known machine learning techniques like Artificial Neural Network (ANN) (Chen, 2006), Support Vector Machine (Deris 2006; Rashid, 2010; Chatterjee, 2011; Bock, JR, 2001; Guo Y, 2008), Random Forest (Qi Y, 2004; Chen XW, 2005), Bayesian Network Classifier (Jansen R, 2003) etc. have been utilized to construct the computational model for prediction of the PPI.

Several earlier studies have reported the use of individual protein domain structure and its frequency of occurrences as crucial features to decide the fate of specific PPI. Domains are the basic units of protein interactions which facilitate the region for physical contact and interplay between the protein pairs. Majority of machine learning algorithm based prediction tool have used frequency of interacting protein domains as a key feature (Chen 2006; Chatterjee 2011; Guo Y, 2008; Chen XW, 2005). The use of such algorithms comes with its associated disadvantages, which often limits the accuracy and generalization of the tools. The Holy Grail machine learning algorithm like ANN is associated with vanishing gradient problem, where the use of multiple hidden layer results in almost negligible error correction. The vanishing gradient effect amalgamated with steepest descent algorithm ensures a typical local optimization of weight vector, hence limiting the accuracy of the learning. Whereas the number of features required for solution space in SVM and Random Forest technique is more often very large as compared to the deep learning techniques for a comparative similar learning accuracy.

The recently proposed Deep Neural Network (DNN) algorithm (Hinton GE & Salakhutdinov, 2006) efficiently handles non-linearity in data using fewer parameters and better hierarchical layer wise function compression. It facilitates global error correction within multiple weight layers with use of accelerated gradient learning algorithm. Further the inclusion of advance optimization algorithms such as Dropout (Wager S, Srivastava N, 2014), Adaptive Learning – ADADELTA (Zeiler MD, 2012) and Nesterov's Accelerated Gradient (Nesterov Y, 2007) were reported to enable minimal over fitting, fast error minimization and high predictive accuracy through the layers of DNN.

In this article, we have proposed Deep Neural Network (DNN) based kernel for prediction of PPIs in five species. The proposed approach which utilizes DNN, takes less number of arguments as compared to other shallower artificial intelligence techniques with better accuracy on same dataset. We have utilized protein domain frequency as the candidate feature for DNN learning with Nestrov accelerated learning to ensure a global optimization of weight parameters. The DNN based prediction has outperformed the SVM and Random Forest based tools with better accuracy and sensitivity scores. Further we have compared the result with existing prediction tools (Chen 2006; Chatterjee, 2011; Guo Y, 2008; Chen XW, 2005) by considering same dataset to infer that the proposed approach has overall better edge in prediction of PPI information.

Prominence of Protein - Protein interaction

Protein live, work and die in highly congested environment. To function they must find the cognate partner in pool of candidates. Protein-protein interactions are highly essential in cellular reproduction growth and apoptosis.

1. Holy grail and Prima Donnas

Protein-protein interaction can be considered as critical component of transcription regulation that is perhaps the true Holy Grail of molecular genetics research. As central dogma depicts by genetic transcription Information present in DNA is decoded to RNA by the cell. The critical decision made by cell, that is what information is to be decoded and when to decode is tightly managed by cell using clusters of proteins so-called as transcription assembly of cell.

To understand cell regulation is the most difficult, complex and intriguing challenge that present molecular biologist are facing nowadays. RNA polymerase (DNA-dependent RNA polymerase) is the prima donna in the world of genetic regulation, which is itself a complex assembly of proteins gathered to perform a predefined task. To understand what makes RNA polymerase so selective to choose a single gene out of entire genome and at specific time, understanding of protein-protein interaction is utmost important.

Functional RNA polymerase is made up of mainly four subunits: 1) beta (β) subunit (MW: 150,000), 2) beta prime (β′) (MW: 160,000), 3) alpha (α) (MW: 40,000), 4) sigma (σ) (MW: 70,000). All unit as a whole complex known as **RNA polymerase holoenzyme** and without σ subunit it is known as the **core enzyme.**

Transcription is followed by phase of the translation, when m-RNA so synthesized in the transcription gets converted in to peptide and then to functional protein (after post translational modification). Process of translation is carried out by a complex called as translational assembly which involves proteins like ribosome and initiating factors. Initiation factors include a group of proteins like elF-1, elF2, elF3, elF4, elF5 and elF6. Initiation factors perform vital role in initiation maintaining and in termination of the cell translation. Translation assembly is also made up of the many proteins and all component protein of these assemblies interacts due to physical and chemical interaction between them.

2. <u>Cell Cycle</u>

Animal development from zygotic stage to multicellular one requires countless rounds of cell division. During cell cycle events are initiated sequentially and repeatedly. Cell cycle consists of 4 phase 1) S phase, 2) M phase, 3) G1 Phase and 4) G_2 Phase. Cyclin-dependent kinases (CDKs) are the molecules involved driving cell cycle through various phases maintaining tight control over each phase. CDKs are small serine/threonine protein requires association with cyclin subunit for their activation. *C. elegans* genome encodes multiple members of the CDK family. Of many CDKs such as CDK-1 and CDK-4, are found to be essential for cell-cycle regulation (Boxem *et al*., 1999). The CDK subunit is inactive as a protein kinase without the cyclin unit. The binding of cyclin subunit renders the complex functional (Connell-Crowley *et al*., 1993).

3. <u>Apoptosis</u>

The process of programmed cell death in multicellular organism is known as Apoptosis. Apoptosis is finely controlled process through various cellular regulators. Various protein molecule such as poly ADP ribose polymerase (Chiarugi A, 2002), small mitochondrial-derived activator (SMAC), *inhibitor of apoptosis proteins* (IAPs), a group of cysteine proteases (caspases) (Fesik SW, 2001), Cytochrome c, mitochondrial apoptosis-induced channel (MAC) (Dejean LM, 2006), Apoptotic protease activating factor - 1 (*Apaf-1*), apoptosome, Bax, Bak, Bcl-2, Bcl-xL and Mcl-1 are found to be induction, inhibition and regulation of the apoptosis. Process of apoptosis is the classical example of the how beautifully protein –protein interaction regulates critical cellular processes. One of the causes of cancer like deadly disease is inhibition of normal pathway of apoptosis mechanism.

<u>What Makes Protein Interact??</u>

Some key factors include chemical properties, geometrical factors and Electrostatic properties of the amino acids at protein's surface are found to be key factor in protein-protein interaction. These are the key determinant which decides the nature and site (hotspot) of the protein-protein interaction on the protein surface.

1. **<u>Hydrophobic patches</u>**

   In determination of structure and function of a protein, hydrophobicity plays a very important role. Hydrophobicity of surface amino acid residues is an important factor and diving force behind folding of soluble proteins. Hydrophobicity of amino acid residues forming pockets on the surface of the protein frequently involved in determination of nature of the protein that will interact with it and in binding of the ligand molecule for therapeutic activity.

   Distribution of the Hydrophobic and Hydrophilic amino acid residues in proteins seem to be kind of organized one. Generally hydrophobic amino acid largely found at the core of the protein as presence of these residue at surface may make the protein fat soluble and it may precipitate in aqueous medium and render it unstable in body fluids, whereas hydrophilic amino acid residues are largely found to be present at the surface to facilitate solubility of the protein and transportation in aqueous body fluid.

   The energetic cost of exposing hydrophobic surface is proportional to its area. Protein-protein interfaces are, to a large extent, well packed (Lo Conte, *et. al*. 1999) and are often composed of a buried hydrophobic core surrounded by a more hydrophilic ring partly exposed to solvent (Bahadur RP, *et. al.* 2008).

2. **<u>Solvent Accessible surface area</u>**

   Solvent-accessible area was originally defined and computed by Lee and Richards (Lee B, *et al*. 1971) for core concept refer to link - http://bose.utmb.edu/poster1/page2.html. The accessible surface area of protein is central part for computing the effect of protein salvation. The free energy of protein salvation is linearly related to the ASA in a continuum approach (Eisenberg D, *et al. 2002*). Solvent accessible surface area gives an idea about the surface residue. Surface residues found by above method can be used to find most probable surface residue that will take part in interaction using other methods like hydrophobic patches and electrostatic patches. Mean Hydrophobicity value (Conte *et. al.,*1999) was calculated for all residues defined in the interface of each complex. In the entire complex the interface has an intermediate hydrophobicity between those of the interior (hydrophobic) and the exterior (hydrophilic). When hydrophobicity values of the interface are compared between homodyne and the hetero complex sites is seen that, the homo dimer rarely occur or function as monomer and their hydrophobic surfaces are permanently buried within protein-protein complexes (Susan J, *et al.* 1996).

3. **Electrostatic patches**

   Electrostatic complementarities are mediated by long range electrostatic field due to charged or partially charged atoms. Salt bridges can be considered as one of the chief long range forces. It is proved that protein-protein interactions are electrostatically guided by a long-range force bringing the interacting partners together (Ye Li, *et al*. 2013).

MATERIALS AND METHODS

Dataset

The PPI data of *Saccharomyces cerevisiae* was collected and retrieved from Database of Interacting Proteins (DIP) (Xenarios I, 2002) and Kansas University Proteomics Service (KUPS) database (Chen *et al.*2010). The PPI information stored in DIP is *experimentally validated* and *manually* curated. Whereas the KUPS combines three high quality manually curated PPI databases i.e. IntAct (Kerrien S *et al.*2012), MINT (Zanzoni A *et al.*2002) and HPRD (Prasad T *et al.* 2009) with their associated features and annotations retrieved from UniProt (Wu CH *et al.* 2006). Negative dataset was collected from KUPS which consists of non-interacting domain pair. The FASTA sequences for all such candidate proteins were extracted from UniProt Database (http://www.uniprot.org/uniprot/).

We have used Perl script for parsing the DIP and KUPS dataset, generating feature vector and for building domain frequency matrix for each screened protein pairs.Originally 15,000 interacting protein pairs (positive dataset) were extracted from DIP and 15,000 non-interacting protein pairs (negative dataset) were extracted from KUPS. Pfam (Punta M *et al.* 2011) database was used to extract domains corresponding to particular amino acid sequences of every candidate protein pairs. The protein domains frequency was used as a prime feature for training algorithm, therefore the protein pairs showing higher interacting domain frequency were selected in preprocessing stage. The reason behind selecting domain with higher frequency and discarding lower frequency domain was to avoid unnecessary expansion of feature space and to reduce computational complexity of the DNN model. The cutoff threshold of domain frequency was kept at '2' through a series of trial and error run of training algorithm. In preprocessing stage several protein pairs were discarded from the training dataset with two screening criteria (i) Protein with obsolete sequences information at UniProt, (ii) Protein pair where at least one of the interacting members shows absence of domain information.

Similar data collection, domain frequency screening and preprocessing scheme were adopted on PPI datasets of *Escherichia coli, Homo sapiens, Drosophila melanogaster* and *Caenorhabditis elegans*. Details about the source of positive as well as negative dataset for *Escherichia coli, Homo sapiens, Drosophila melanogaster and Caenorhabditis elegans* along with *Saccharomyces cerevisiae* is summarized in Table 1.

Data Partitioning

The interacting and non-interacting protein pairs were labeled as Positive and Negative class with class tag '1' and '0' respectively. Entire dataset was sampled into training set and testing set in 7:3 ratios. Training data were utilized for DNN kernel learning where the testing data were not exposed to the kernel during the training process. The length of the feature vector for the protein pair was kept at 5582, with inclusion of all high frequency domain information. The length of the input feature vector also enhances the generalization potential of DNN.

We selected 8525 interacting protein-protein pairs and 8525 non-interacting pairs of *Saccharomyces cerevisiae* dataset. The training dataset and test dataset were created by 7:3 ratio resulted in 5968 training sample and 2557 test sample both for positive and negative class tag.

To reduce the position specific input feeding biasness, we have considered the auto-correlated input vectors. The concept of autocorrelation of input vectors for a given protein pair implies, resulting proteins$A$ and B in a protein pairwhich can be fed as two interaction vectors *i.e.* $A \otimes B$ and $B \otimes A$. So the training and testing data matrix were populated with both the instances of these auto-correlated pair by flipping the domain frequency vector. This resulted in 10,228 protein pairs of test dataset and 23,872 protein pairs of training dataset for both interacting and non-interacting pairs. Similar method of partitioning was adopted for PPI datasets of *E. coli, H. sapiens, D. melanogaster* and *C. elegans.* Details regarding size of training and testing datasets of these organisms are summarized in Table 1.

Features Extraction

An appropriate feature space representation describes that the data is essential for the performance of any supervised machine learning techniques. It has been reported in several earlier studies, that the domain composition play an important role in the prediction of PPI (Pawson T, 2003; Kim WK, 2002).The individual protein domain provides the physical space to facilitate the contact and interplay between the interacting proteins.

The above said data were processed through Perl script to identify 4,668 types of domain in positive dataset and 2,599 types of domain in negative dataset. To select the relevant features, only those domains which shows domain frequency above threshold is being selected. The cutoff threshold was fixed at '2' and it resulted in 1784 and 1460 screened domains from positive and negative dataset respectively. We have further removed out the redundant domains and took single instances which resulted in 2791 domains. The feature vector was constructed by taking 2791domains for each protein, hence every protein pairs was represented by feature vector of size 5582 as shown in Fig. 1.

Let,$D = [X_1, X_2, \ldots\ldots X_n]$ represent n training or testing samples in database D and $X_i = [\,x_1^{(i)}, x_2^{(i)}, \ldots.\, x_{2791}^{(i)}, yi]$ represents the i^{th} protein pair with 5582 features attributes and X_i belongs to class y_i . Furthermore the $y_i = 1$ refers to 'interacting pair' class whereas $y_i = 0$ indicates 'non-interacting pair' class. The feature set consists of domain frequencies -D^f . For example, if protein P_1 from a specific pair has domain frequency $D^f_{PF004\ 00}$equals to 3, then corresponding feature value for protein $P_1$for domain name "PF00400" will also be 3. If domain(s) is/are present in the corresponding protein pairs, entry will be marked as its frequency else will be marked as '0' for absence of domains information. This will create a sparse data input matrix consisting of few values and rest all zeroes, for all protein pairs.

Similar method for constructing feature vector was applied to PPI datasets of *E. coli, H. sapiens, D. melanogaster* and *C. elegans.* Details of domains considered for the feature vector construction for these four organisms is summarized in Table 1.

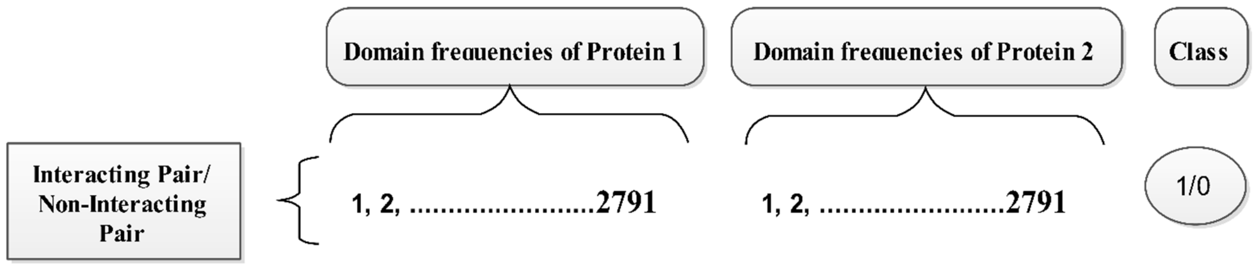

Domain freauencies of Protein 1
Domain freauencies of Protein 2
Class
Interacting Pair/ Non-Interacting Pair
1, 2, ...2791
1, 2, ...2791
1/0

Figure 1: Feature representation of Interacting and Non-Interacting Protein Pairs

Table 1. Summary of all data sources, data contents, detected and selected domains of *Caenorhabditis elegans*, *Drosophila melanogaster, Homo sapiens* and *Escherichia coli.*

Organism name	**Total domains from positive dataset**	**Used domain from positive dataset**	**Total domain from negative dataset**	**Used domain from negative dataset**	**Total positive database**	**Source of positive data**	**Total Negative Database**	**Source of negative data**	**Used Positive Database**	**Used Negative Database**	**Test** $A \otimes B + B \otimes A$	**Train** $A \otimes B + B \otimes A$	**No. of Feature (for protein pair)**
S. cerevisiae	4668	1784	2599	1460	15000	DIP	15000	KUPS	8525	8525	10228	23872	5582
E.coli	3058	1423	2225	1071	12250	DIP	9668	KUPS	9105	9105	10928	25492	4768
H. sapiens	5120	3617	2599	1306	15000	KUPS	10790	KUPS	9973	9973	11964	27928	8080
D. melanogaster	5165	2071	2867	1459	15091	KUPS	15000	KUPS	8256	8256	9904	23120	5544
C. elegans	1843	1843	314	314	3395	DIP	2889	KUPS	1498	1498	1798	4196	3702

DNN Implementation

DNN was implemented on Hadoop, utilizing H2O Oxdata framework which facilitate the general mathematical analysis with R. Hadoop is an Apache framework for Big Data analytics and H2O Oxdata module is an integral part of Hadoop for doing mathematical computation. DNN was based on a multi-layer feed-forward artificial neural network (ANN) which is trained using back-propagation algorithm. The main concept behind success of DNN is to train the individual layers of the entire network one at a time. While training DNN, we first train a network with first hidden layer, and then we go on adding next layer to previously trained network until last layer .We take the old network with $n - 1$ hidden layers at each step, and add an additional n^{th} hidden layer. n^{th} layer takes input from previous *n-1*th layer, which has already been trained. DNN can be used as supervised and non- supervised leaning methods. When all layers of the network get trained individually, resulted weight from training the layers individually then used to initialize the weight in the final/overall network. This approach is known as greedy layer-wise training.

For the proposed work we have used tanh (rescaled version of the sigmoid function) as an activation function. In case of tanh the activation value is symmetric and in range of -1 to +1 as given Eq.2.1.

$$\tanh(\gamma)=\frac{e^{\gamma}-e^{-\gamma}}{e^{\gamma}+e^{-\gamma}} \tag{2.1}$$

Similar to the back propagation algorithm the weight and bias were updated based on first-order gradient information as shown below,

$$W_{t+1}^{\ell} = W_t^{\ell} + \mathcal{E}\Delta W_t^{\ell} \tag{2.2}$$

and

$$b_{t+1}^{\ell} = b_t^{\ell} + \mathcal{E}\Delta b_t^{\ell} \tag{2.3}$$

Where, W_t^{ℓ} and b_t^{ℓ} are weight matrix and bias vector after t^th update at layer ℓ in a network with learning rate $\mathcal{E}$.

$$\boldsymbol{W}_{t+1}^{\ell} = \rho \boldsymbol{W}_t^{\ell} + (1-\rho)\frac{1}{M_b}\sum_{m=1}^{M_b} \nabla_{\boldsymbol{W}_{t+1}^{\ell} j(W,b;O^m,y^m)} \tag{2.4}$$

$$\boldsymbol{b}_{t+1}^{\ell} = \rho \boldsymbol{b}_t^{\ell} + (1-\rho)\frac{1}{M_b}\sum_{m=1}^{M_b} \nabla_{\boldsymbol{b}_{t+1}^{\ell}(W,b;O^m,y^m)} \tag{2.5}$$

The average weight matrix gradient and the average bias vector gradient at iteration $(\mathrm{t}+1)$ is given in Equation 2.4 and 2.5 respectively. Where O and y are observation and input vector respectively. $j(W,b;O^m,y^m)$ Represents loss function given $\{\boldsymbol{W},\boldsymbol{b}\}$ as model parameters for sample size $M, (0 \le m < M)$. Convergence can be reached much faster if model update is made based on all previous gradients instead of only current one. ρ is a factor called as momentum, which remains between 0.9 - 0.99.

For binomial classification as in our case, the output neuron represents a class $i \in \{1, \dots, C\}$, where C represents number of classes. The probability $P_{dnn}(i|O)$, classify the output of value V_i^L of i^{th} output neuron to class i.

Actual output at perception P at any layer ℓ having n perceptron in previous layer can be calculated as,

$$class_i = P_{dnn}(i|O), = O_P^{\ell} = tanh(\textstyle\sum_{n=1}^{n=n} y_n^{\ell-1} W_n^{\ell-1,\ell} - b_P^{\ell}) \qquad (2.6)$$

Where, $y_n^{\ell-1}$an output from previous layer ℓ-1 is, $W_n^{\ell-1,\ell}$ is a weight between all perceptrons of ℓ-1 and perceptron$P$ at layer ℓ.b_n^{ℓ} is a bias at layer ℓ for perceptron P.

The proposed DNN classifier was trained using 10-fold cross-validation to avoid over fitting of training dataset. Further the best model out of cross-validation was chosen as final model for model validation through test dataset. The proposed DNN consists of 4 hidden layers with 700, 500, 250 and 100 perceptrons in each layer respectively. The DNN architecture discussed above was selected through a series of trial and error runs. Outputs were marked as 1 or 0, where 1 indicates interacting protein-protein pair and 0 indicates non-interacting protein-protein pair. Threshold parameter of DNN learning was kept at 0.95.

Optimization algorithms

1. Dropout

To avoid over fitting and slow convergence in proposed DNN the dropout has been introduced, with input dropout rate at 0.2 whereas the dropout rate for each hidden layer was kept at 0.5. The dropout prevents brittle co-adaptation and over fitting by adding noise to the perceptron in network layers. The dropout technique uses a constant probability for omitting a unit along with its connections from the network. For multilayer feed forward network, the activity $\boldsymbol{S}$ in perceptron $\boldsymbol{i}$ of layer h can be expressed as:

$$O_i^{\mathrm{h}} = tanh(S_i^{\mathrm{h}}) = tanh(\textstyle\sum_{\ell<\mathrm{h}} \sum_j W_{ij}^{h\ell} O_j^{\ell}) \quad \text{With} \quad O_J^{\mathrm{O}} = I_j \quad (2.7)$$

Where Wdenotes weight and O_i^{h} is the output of perceptron i of layer h, and dropout can be defined as:

$$O_i^{\mathrm{h}} = tanh(S_i^{\mathrm{h}}) = tanh(\textstyle\sum_{\ell<\mathrm{h}} \sum_j W_{ij}^{h\ell} \delta_j^{\ell} O_j^{\ell}) \qquad \text{With} \qquad O_J^{\mathrm{O}} = I_j \ (2.8)$$

$$NWGM(O_i^{\mathrm{h}}) = \frac{\prod_{\mathcal{N}} {O_i^{\mathrm{h}}}^{P(\mathcal{N})}}{\prod_{\mathcal{N}} {O_i^{\mathrm{h}}}^{P(\mathcal{N})} + \prod_{\mathcal{N}} (1-O_i^{\mathrm{h}})^{P(\mathcal{N})}} \qquad (2.9)$$

Where δ_j^{ℓ} and $NWGM$ are Bernoulli selector variable, normalized weighted geometric mean respectively. $\boldsymbol{\mathcal{N}}$Spans over all possible sub-networks formed in network as a result of dropout.

$$E(O_i^{\mathrm{h}}) \approx NWGM(O_i^{\mathrm{h}}) \qquad (2.10)$$

$$NWGM(O_i^{\mathrm{h}}) = (\tanh)_i^{\mathrm{h}}[E(S_i^{\mathrm{h}})] \qquad (2.11)$$

$$E(S_i^{\mathrm{h}}) = \sum_{\ell<h}\sum_j W_{ij}^{h\ell} P_j^{\ell} E(O_j^{\ell}) \qquad (2.12)$$

Equation 2.10, 2.11 and 2.12 are fundamental equations, which represents recursive dropout ensemble in DNN. Since the numbers O_i^{h} are non-identical over possible sub networks $\boldsymbol{\mathcal{N}}$ in our DNN approach, $NWGM$ provides a good approximations.

2. <u>ADADELTA (Adaptive learning rate)</u>

Likewise with every learning algorithm, the learning rate is of prime importance to decide the convergence of system to global minima value. It has been reported that choosing slow learning rate results in delay convergence and a higher learning rate can cause system to diverge in term of objective function. To ensure a smooth learning we have adopted ADADELTA method, which prevents the continuous decay of learning rate throughout training and it automatically adjust the global learning rate (Duchi J, Hazan E, 2011).

The learning parameters x is updated over a time to optimize an objective function $f(x)$ as

$$x_{t+1} = x_t + \Delta x_t \qquad (2.13)$$

By ADADELTA method of updating Δx_t from time $t = 0$ to $t = T$.

$$\Delta x_t = -\frac{\eta}{\sqrt{\sum_{T=1}^{t} g_T^2}} g_t \qquad (2.14)$$

As in Eq. 2.14 the denominator accumulates square gradient for all previous iterations and hence is responsible for continuous decay of learning rate $\boldsymbol{\eta}$. Where g_t is a gradient of parameter at t^{th} iteration. Also instead of accumulating squared gradient from each iteration from beginning of training, ADADELTA accumulates it over a fixed size window (ω). This prevents denominator to grow infinite and ensure learning continues to make progress even after many iteration. As ADADELTA uses first order information, so it requires very less computation per iteration over Newton's method or quasi-Newton's methods used in shallow learning algorithm.

3. <u>Nesterov's Accelerated Gradient</u>

Further we have used Nesterov's Accelerated Gradient descent (Nesterov Y, 2007), which accelerated the optimization process to reach global minima using less computational complexity over time. The plain Gradient Descent algorithm has a rate of convergence as $O\left(\frac{1}{t}\right)$, while the Nesterov's Accelerated gradient has rate of convergence as $O\left(\frac{1}{t^2}\right)$.

Computational Speed and Network Architecture

The computational speed associated with DNN mainly depends on three factors, (i) number of hidden layers used, (ii) size of input feature vector and (iii) number of output nodes. Even though the proposed DNN architecture uses higher number of input nodes the sparse nature of data matrix resulted in fewer input values to be used in weight optimization computation. Also the number of hidden layers was fixed at '4' as given in Figure. 2. It took approximately 20 minutes for each fold cross validation for training sample on i7, 2.6 GHz AMD machine with 12 GB RAM. Steps to install and setup H2O - Oxdata on windows 8.1 and 64 machine are enlisted in Appendix – 1

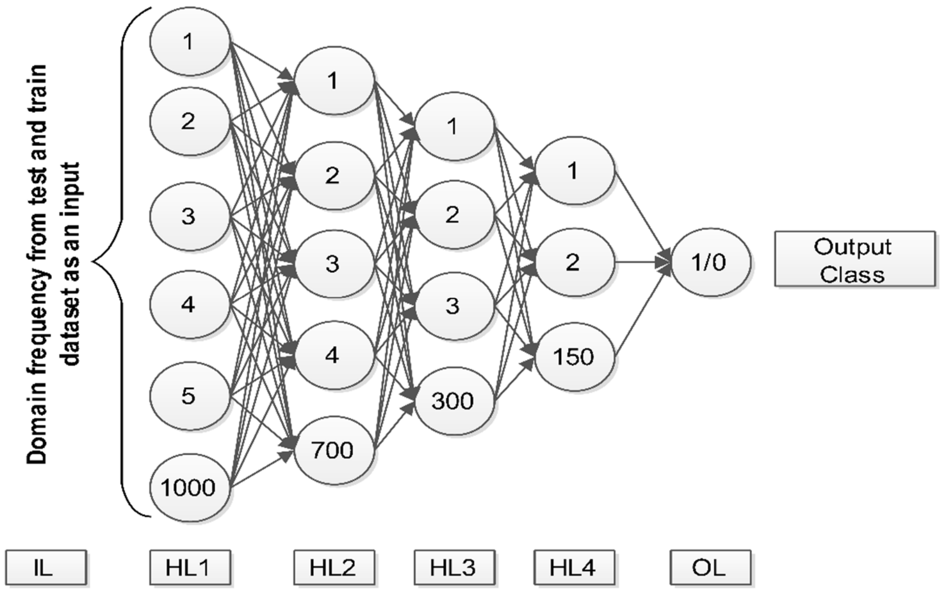


Figure 2: A Deep Neural Network used for training and testing (IL=Input layer, HL=Hidden layer, OL=Output layer).

Similar models were used for training and testing of PPI datasets of *E. coli, H. sapiens, D. melanogaster* and *C. elegans* with 4768, 8080, 5544 and 3702 features respectively.

A basic DNN implementation Program

#COMMENT START

#==#

#h2o.init-Connect to H2O and Install R Package

#system.file-for loading local files

#h2o.importFile- For generating unique hex identity for system file

#h2o.deeplearning-Performs Deep learning neural networks on an H2OParsedData object.

#override_with_best_model-If enabled, override the final model with the best model #found during training. Defaults to true.

#nfolds (Optional)- Number of folds for cross-validation. If nfolds >= 2, then validation must remain empty.

#activation-A string indicating the activation function to use. Must be either "Tanh", "TanhWithDropout", "Rectifier", "RectifierWithDropout", "Maxout" or "MaxoutWithDropout".

#hidden-Hidden layer sizes (e.g. c(100,100)

#epochs-How many times the dataset should be iterated (streamed), can be fractional

#balance_classes-Balance training data class counts via over/under-sampling (for imbalanced data)

#Input_dropout_ratio-Input layer dropout ratio (can improve generalization, try 0.1 or 0.2)

#hidden_dropout_ratios-Hidden layer dropout ratios (can improve generalization), #specify one value per hidden layer, defaults to 0.5

#h2o.predict-H2O Model Predictions

#h2o.performance-Performance Measures

# h2o.saveModel - Save a H2OModel object to a disk and can be loaded back into H2O using h2o.loadModel.

#plot-Draw scatter plot of a particular performance measure vs. thresholds for a H2O model, or the ROC curve.

#==#

#COMMENT ENDS

#Programm Starts

#make appropriate changes as per your need

```
library(h2o)
library(MASS)
localH2O = h2o.init(nthreads = -1,max_mem_size = "10g")#10gb
RAM and all CPU
irisPath = system.file("extdata", "train.csv", package = "h2o")
#train.csv- training dataset of an organism
iris.hex = h2o.importFile(localH2O, path = irisPath)
model<-h2o.deeplearning(x = 1:3542, y = 3543, data = iris.hex,
activation = "TanhWithDropout", # or 'Tanh'
              input_dropout_ratio = 0.2, # % of inputs dropout
              hidden_dropout_ratios = c(0.5,0.5,0.5,0.5), # % for
nodes dropout
             nfold=10,epoch=10,balance_classes = TRUE, hidden
             = c(1000,700,300,100),
             nesterov_accelerated_gradient=TRUE,adaptive_rate=
             TRUE)
print(model)
name<-paste('model_name_here_',i,sep="")
h2o.saveModel(object = model, dir =
"C:/Users/Decepticonn/Documents/R/win-
library/3.1/h2o/extdata/",name=name, save_cv = TRUE, force =
TRUE)
#model = h2o.loadModel(localH2O,
"C:/Users/Decepticonn/Documents/R/win-
library/3.1/h2o/extdata/cele_server") #for loading existing
model
test = system.file("extdata", "test.csv", package = "h2o")
#test.csv- testing dataset of an organism

test.hex = h2o.importFile(localH2O, path = test)
h2o_yhat_test <- h2o.predict(model, test.hex)
summary(h2o_yhat_test)
print(h2o_yhat_test)
```

```
perf = h2o.performance(h2o_yhat_test[,3], test.hex$class,
measure = "precision")
perf = h2o.performance(h2o_yhat_test[,3], test.hex$class,
measure = "precision")
plot(perf, type = "cutoffs")     # Plot precision vs. thresholds
plot(perf, type = "roc")        # Plot ROC curve
write.matrix(h2o_yhat_test,
paste("C:/Users/rashmi/Documents/R/win-
library/3.1/h2o/extdata/",name))
h2o.shutdown(localH2O, prompt = FALSE)
            #==============================================#
```

<u>RESULT</u>

The prediction performance of DNN was evaluated using standard measures including Accuracy, Sensitivity, Specificity, Precision, Matthews's Correlation Coefficient (MCC), Youden's Index and ROC Area under Curve (AUC) score. Which is described below in Equation (3.1-3.10). True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) were calculated for testing data. TP represents the Positive Protein-Protein Interaction classified correctly, TN represents the Negative Protein-Protein Interaction classified correctly, similarly FP and FN are Non-interacting protein classified as Interacting one and Positive Protein-Protein Interaction classified as Non-interacting one, respectively. 10 fold cross validation summary of *Saccharomyces cerevisiae* given in figure 3. Prediction performance of best model found during training of *S. cerevisiae* is summarized in Table 3 and Table 4.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.1}$$

$$\text{True positive rate or Sensitivity} = \frac{TP}{TP+FN} \tag{3.2}$$

$$\text{Positive prediction value or Precision} = \frac{TP}{TP+FP} \tag{3.3}$$

$$\text{Specificity} = \frac{TN}{TN+FP} \tag{3.4}$$

$$\text{False discovery rate (FDR)} = 1 - \text{Positive prediction value} \tag{3.5}$$

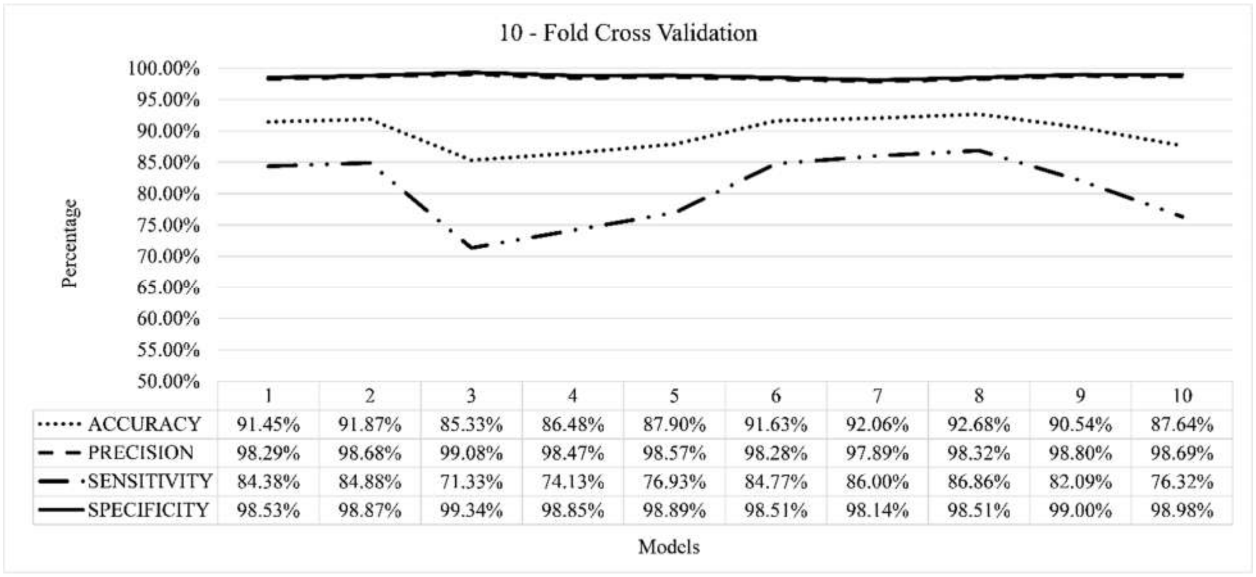
$$\text{False negative rate (FNR)} = 1 - \text{True positive rate} \tag{3.6}$$

$$\text{Fall-out or false positive rate (FPR)} = \frac{FP}{FP+TN} \tag{3.7}$$

$$\text{Negative predictive value (NPV)} = \frac{TN}{TN+FN} \tag{3.8}$$

$$\text{MCC} = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{3.9}$$

$$\text{Youden's index (J)} = \text{Sensitivity} + \text{Specificity} - 1 \tag{3.10}$$



	1	2	3	4	5	6	7	8	9	10
ACCURACY	91.45%	91.87%	85.33%	86.48%	87.90%	91.63%	92.06%	92.68%	90.54%	87.64%
PRECISION	98.29%	98.68%	99.08%	98.47%	98.57%	98.28%	97.89%	98.32%	98.80%	98.69%
SENSITIVITY	84.38%	84.88%	71.33%	74.13%	76.93%	84.77%	86.00%	86.86%	82.09%	76.32%
SPECIFICITY	98.53%	98.87%	99.34%	98.85%	98.89%	98.51%	98.14%	98.51%	99.00%	98.98%

Figure 3: Results of 10-fold cross validation applied to training dataset of *Saccharomyces cerevisiae*. Model 8 was selected as the best one on the basis of accuracy, precision, sensitivity and specificity.

Table 2 Confusion Matrix for model 8 (*Saccharomyces cerevisiae*).

	P' (Predicted)	**N' (Predicted)**
P (Actual)	4442	76
N (Actual)	672	5039

Table 3 Performance and analysis result (*Saccharomyces cerevisiae*)

Sensitivity / Recall (in %)	86.8596%
Precision (in %)	98.3178%
Specificity (in %)	98.5142%
Accuracy (in %)	92.6784%
Matthews Correlation Coefficient (MCC)	0.8596
Youden's Index (J)	0.8537

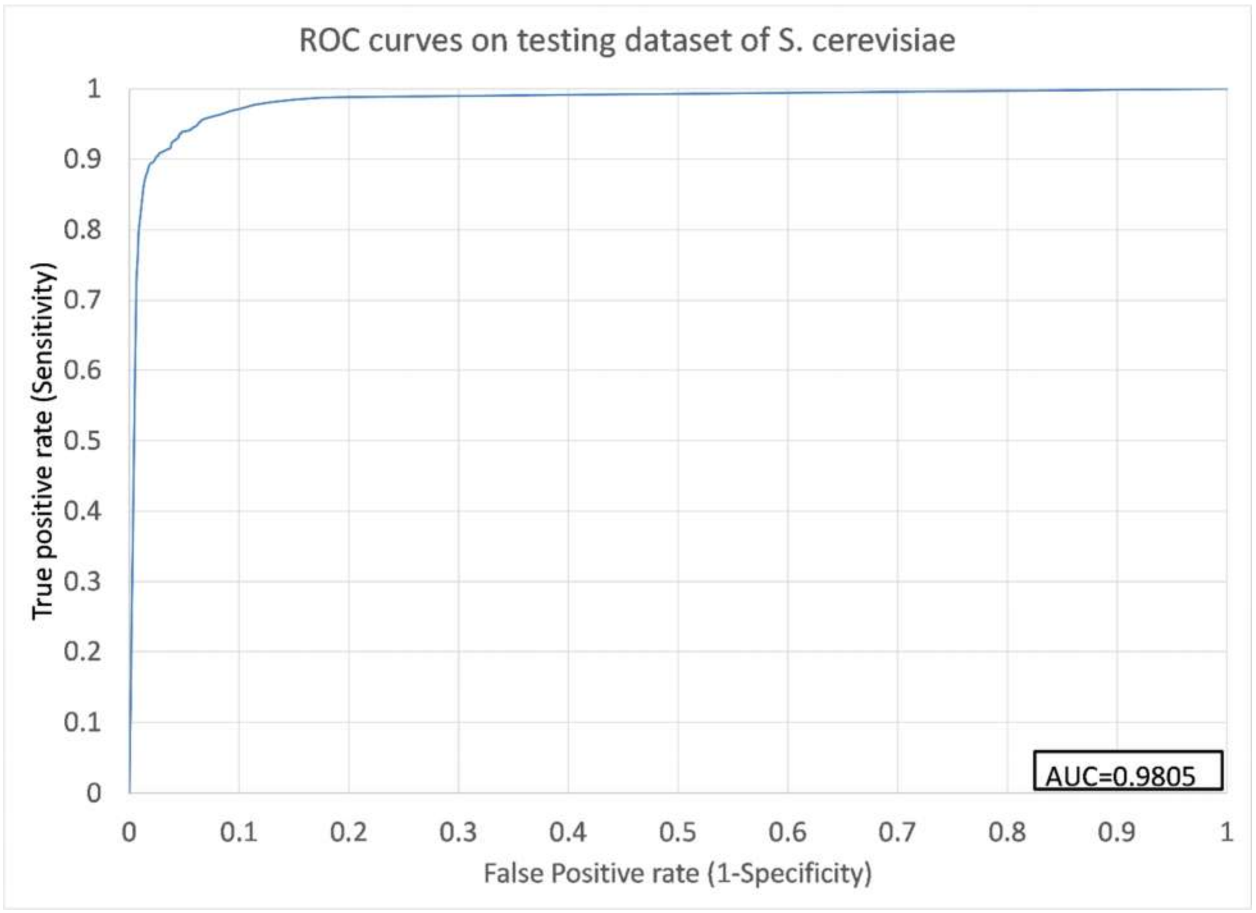
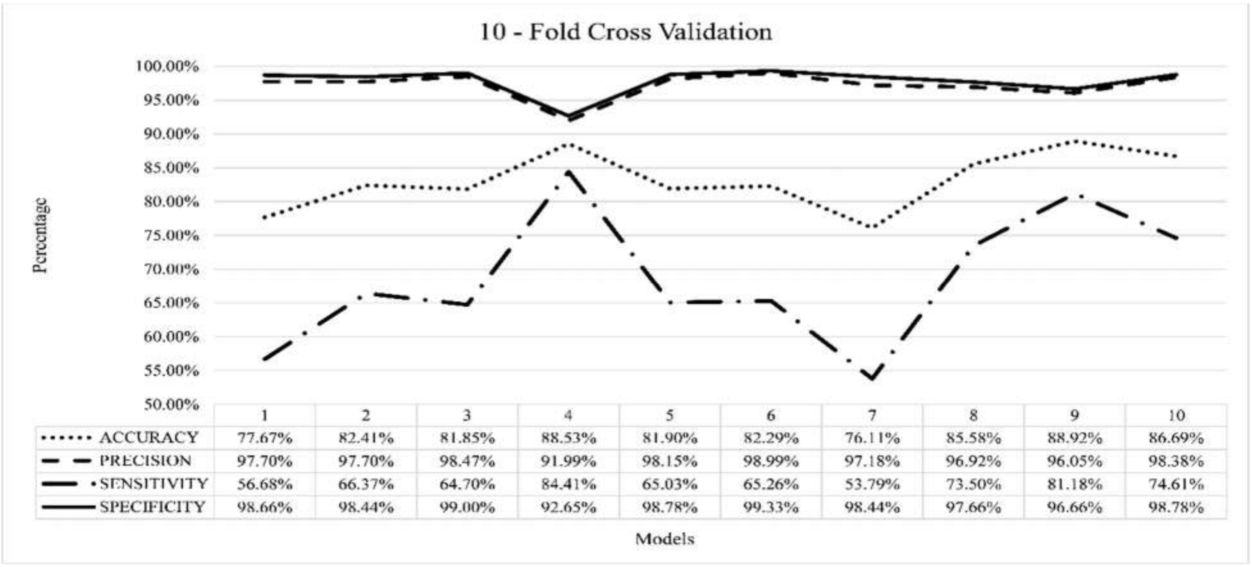


Figure 4: ROC curve for testing dataset of *Saccharomyces cerevisiae* for model 8

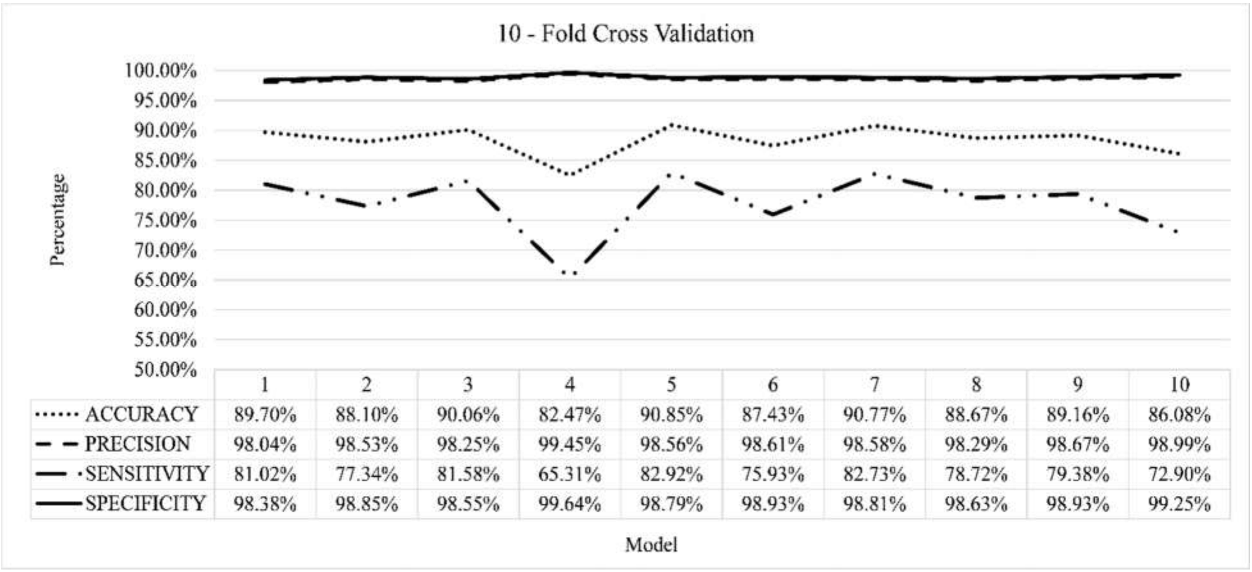
Accuracy reported was 92.67%, Matthews Correlation Coefficient was 0.8596, which indicates good prediction as it is very close to 1. Youden's Index (J) calculated was 0.8537 which indicates that there are very less false positive or false negative, i.e., the test is close to a perfect model. Prediction performance of best model found during training of *E. coli, H. sapiens, D. melanogaster* and *C. elegans* are summarized in Table 4. The 10 fold cross validation summary of the four organisms are given as line plot (Fig. 5, 6, 7, and 8)

Similar model was designed for *E. coli, H. sapiens, D. melanogaster* and *C. elegans* based on the same parameters used for designing model of *S. cerevisae.* For all organism experimental conditions including data preprocessing, data partitioning and DNN model parameters were kept constant. Details regarding source of dataset, total domains detected in positive and negative dataset, selected domains from positive and negative dataset for feature vector formation, number of testing and training sample and total elements in feature vector are summarized in Table 1. Similarly 10 fold cross validation is applied to database of *Caenorhabditis elegans*, *Drosophila melanogaster*, *Homo sapiens* and *Escherichia coli* and results are as shown below in form of the line plot.



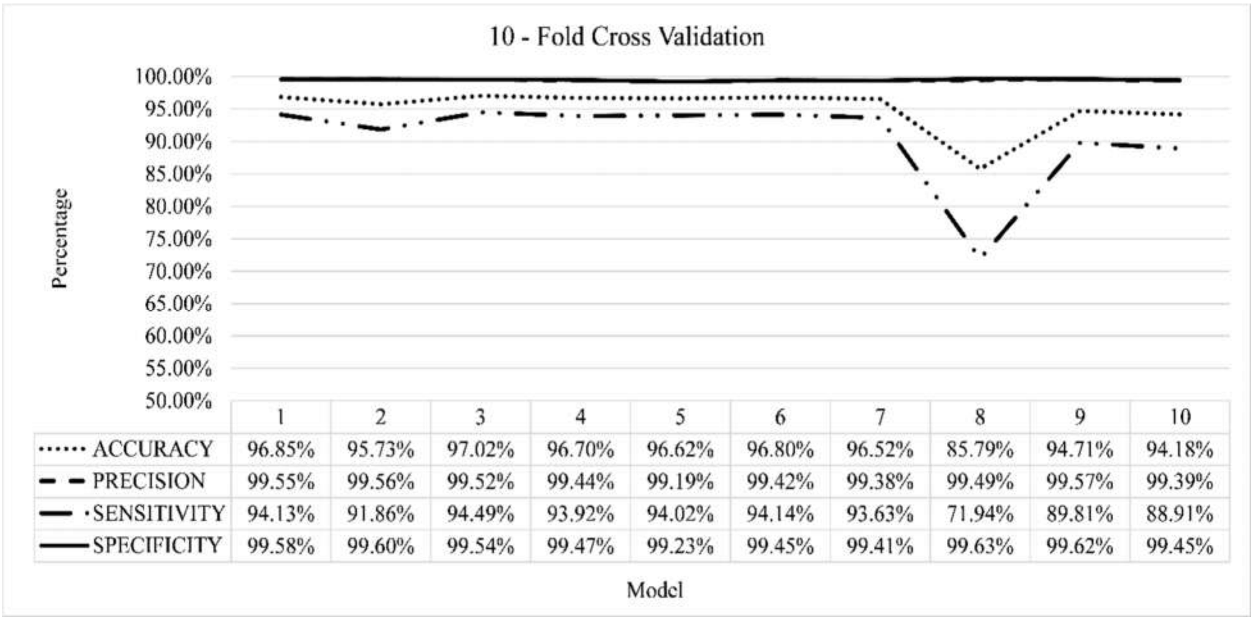
	1	2	3	4	5	6	7	8	9	10
ACCURACY	77.67%	82.41%	81.85%	88.53%	81.90%	82.29%	76.11%	85.58%	88.92%	86.69%
PRECISION	97.70%	97.70%	98.47%	91.99%	98.15%	98.99%	97.18%	96.92%	96.05%	98.38%
SENSITIVITY	56.68%	66.37%	64.70%	84.41%	65.03%	65.26%	53.79%	73.50%	81.18%	74.61%
SPECIFICITY	98.66%	98.44%	99.00%	92.65%	98.78%	99.33%	98.44%	97.66%	96.66%	98.78%

Figure 5: Results of 10-fold cross validation applied to training dataset of *C. elegans*. Model 7 was selected as the best one on the basis of accuracy, precision, sensitivity and specificity.



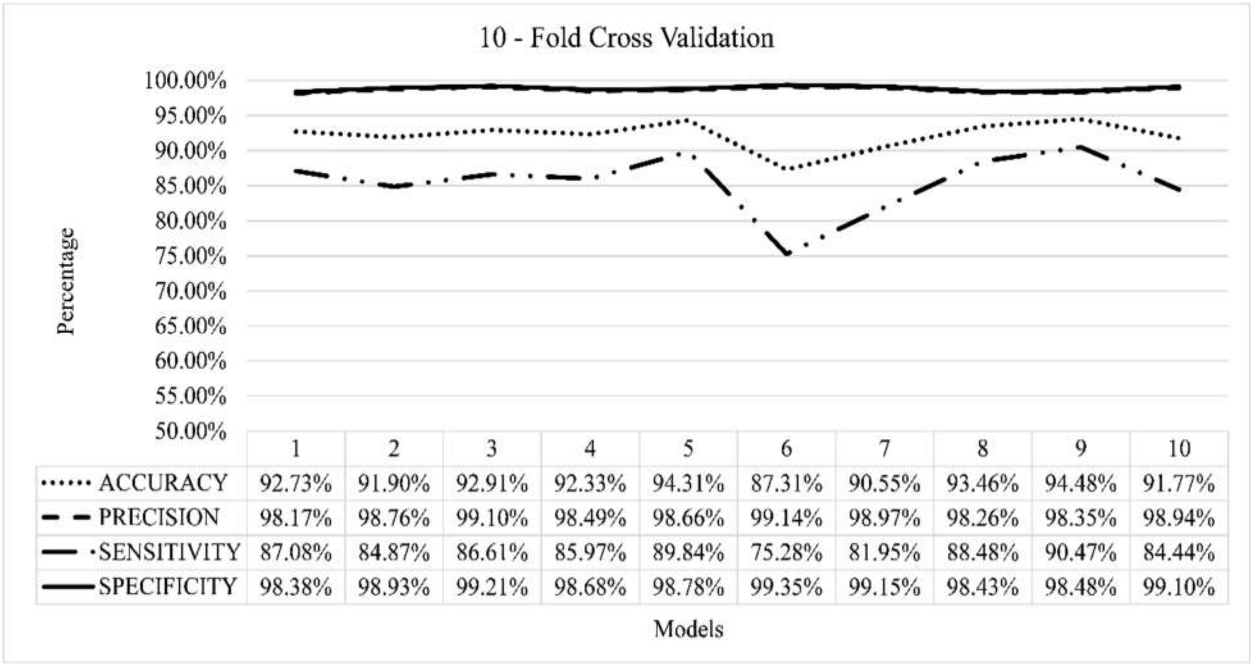
	1	2	3	4	5	6	7	8	9	10
ACCURACY	89.70%	88.10%	90.06%	82.47%	90.85%	87.43%	90.77%	88.67%	89.16%	86.08%
PRECISION	98.04%	98.53%	98.25%	99.45%	98.56%	98.61%	98.58%	98.29%	98.67%	98.99%
SENSITIVITY	81.02%	77.34%	81.58%	65.31%	82.92%	75.93%	82.73%	78.72%	79.38%	72.90%
SPECIFICITY	98.38%	98.85%	98.55%	99.64%	98.79%	98.93%	98.81%	98.63%	98.93%	99.25%

Figure 6: Results of 10-fold cross validation applied to training dataset of *D. melanogaster*. Model 5 was selected as the best one on the basis of accuracy, precision, sensitivity and specificity.



	1	2	3	4	5	6	7	8	9	10
······ ACCURACY	96.85%	95.73%	97.02%	96.70%	96.62%	96.80%	96.52%	85.79%	94.71%	94.18%
– – PRECISION	99.55%	99.56%	99.52%	99.44%	99.19%	99.42%	99.38%	99.49%	99.57%	99.39%
– ·SENSITIVITY	94.13%	91.86%	94.49%	93.92%	94.02%	94.14%	93.63%	71.94%	89.81%	88.91%
—— SPECIFICITY	99.58%	99.60%	99.54%	99.47%	99.23%	99.45%	99.41%	99.63%	99.62%	99.45%

Figure 7: Results of 10-fold cross validation applied to training dataset of *E. coli*. Model 6 was selected as the best one on the basis of accuracy, precision, sensitivity and specificity.



	1	2	3	4	5	6	7	8	9	10
······ ACCURACY	92.73%	91.90%	92.91%	92.33%	94.31%	87.31%	90.55%	93.46%	94.48%	91.77%
– – PRECISION	98.17%	98.76%	99.10%	98.49%	98.66%	99.14%	98.97%	98.26%	98.35%	98.94%
– ·SENSITIVITY	87.08%	84.87%	86.61%	85.97%	89.84%	75.28%	81.95%	88.48%	90.47%	84.44%
—— SPECIFICITY	98.38%	98.93%	99.21%	98.68%	98.78%	99.35%	99.15%	98.43%	98.48%	99.10%

Figure 8: Results of 10-fold cross validation applied to training dataset of *H. sapiens*. Model 9 was selected as the best one on the basis of accuracy, precision, sensitivity and specificity.

Table 4 Experimental results for *Caenorhabditis elegans, Drosophila melanogaster*, *Homo sapiens* and *Escherichia coli*.

Organism name	***S. cerevisiae***	***E.coli***	***H. sapiens***	***D. melanogaster***	***C. elegans***
TP	4442	5163	5412	4106	729
TN	5039	5439	5891	4892	868
FP	672	301	570	846	169
FN	76	25	91	60	30
ACCURACY	92.6784%	97.0168%	94.4751%	90.8522%	88.9198%
PRECISION	98.3178%	99.5181%	98.3464%	98.5598%	96.0474%
SENSITIVITY	86.8596%	94.4912%	90.4714%	82.9160%	81.1804%
SPECIFICITY	98.5142%	99.5425%	98.4788%	98.7884%	96.6592%
FPR	0.1314	0.0551	0.0953	0.1708	0.1882
NPV	0.8823	0.9476	0.9118	0.8526	0.8370
F	0.9223	0.9694	0.9424	0.9006	0.8799
FDR	0.0168	0.0048	0.0165	0.0144	0.0395
FNR	0.1314	0.0551	0.0953	0.1708	0.1882
MCC	0.8596	0.8275	0.8924	0.8275	0.7879
Youden's Index	0.8537	0.8382	0.8895	0.8170	0.7784

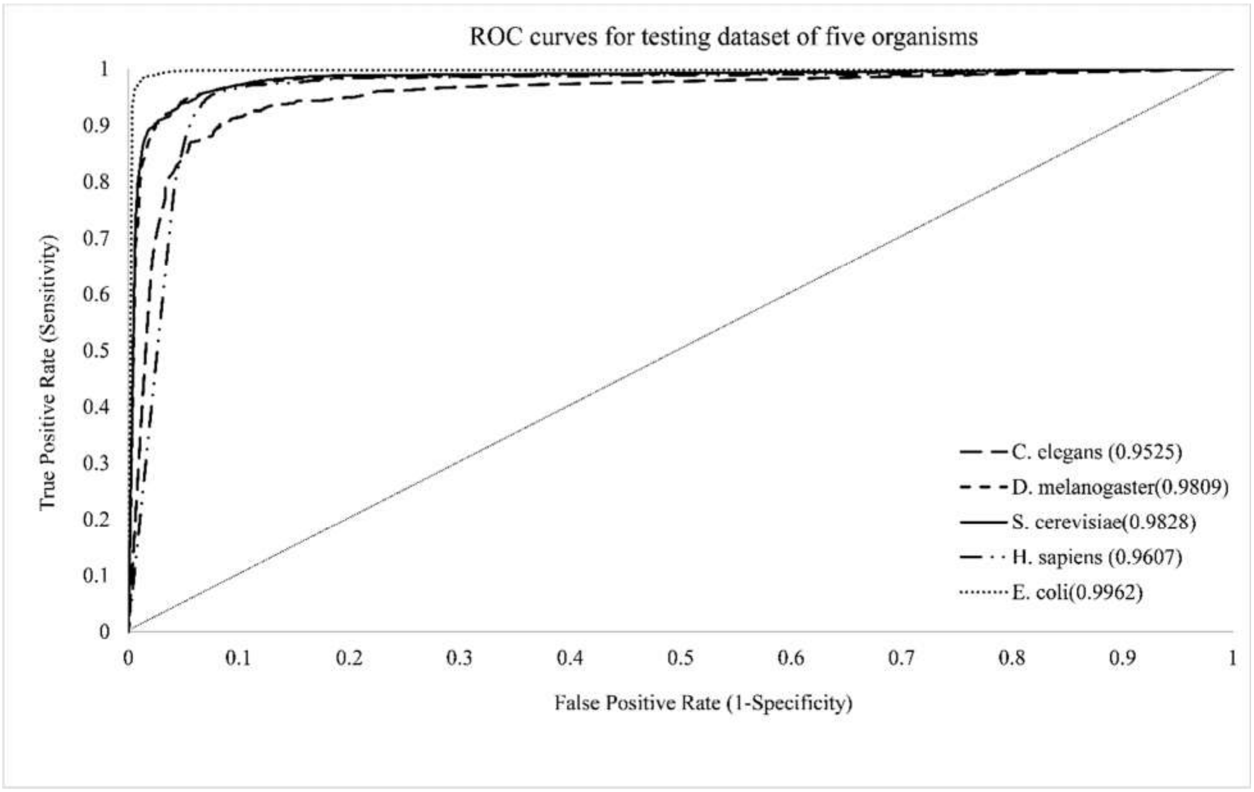


Figure 9: ROC curve for testing dataset of *Caenorhabditis elegans, Drosophila melanogaster*, *Saccharomyces cerevisiae, Homo sapiens* and *Escherichia coli* using the best model out of 10 fold cross validation.

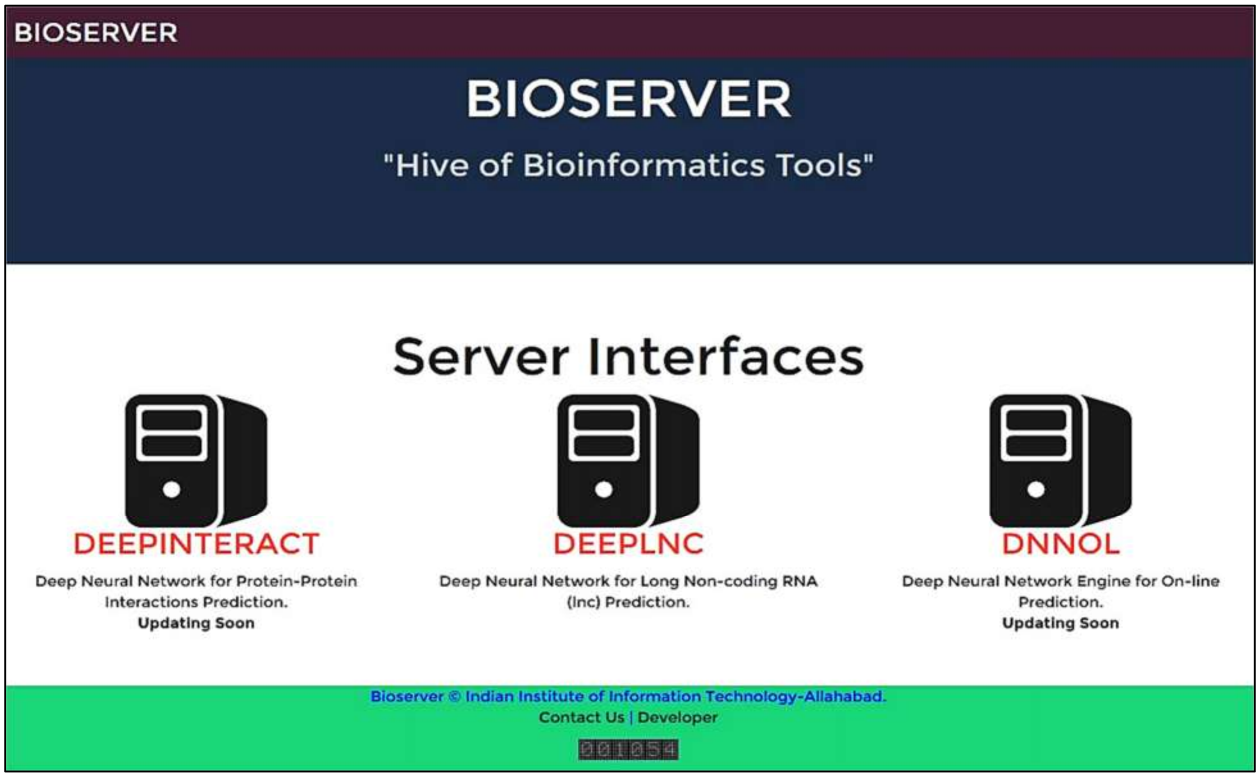
OUTREACH

Research is fruitful if it reaches to other researchers or common people who can take advantage of it or can extend it further, taking this point in consideration we implemented a web-based server “bioserver”, (available at http://bioserver.iiita.ac.in) and standalone software with all trained model which is available on Github.

Server

Bioserver is presently hosting 3 tools namely DeepInteract, DeepLNC and DNNOL. DeepLNC (http://bioserver.iiita.ac.in/deeplnc/) is the tool for prediction of lncRNAs based on Deep Neural Network. DNNOL (http://bioserver.iiita.ac.in/dnnol/) is an online Deep Neural Network engine for training and testing data online.

We have also implemented a web server named “DeepInteract” now available at http://bioserver.iiita.ac.in/deepinteract/. DeepInteract version: 1.1 is available for five organisms *Saccharomyces cerevisiae, Caenorhabditis elegans*, *Escherichia coli, Homo sapiens* and *Drosophila melanogaster.* Server uses the DNN based model for each organism in backend. HTML and CSS were used for web interface whereas Perl, R and H2O Oxdata were used to handle backend processing. Web server version: 1.1 can provide result within 1.2 seconds of job submission with server load of 10 clients’ simultaneous request. Now, DeepInteract is available in latest version 2.0. Version 2.0 (vivid) comes with enhanced, easy to understand and well formatted output in form of report.



BIOSERVER

BIOSERVER

"Hive of Bioinformatics Tools"

Server Interfaces

DEEPINTERACT

Deep Neural Network for Protein-Protein Interactions Prediction.
Updating Soon

DEEPLNC

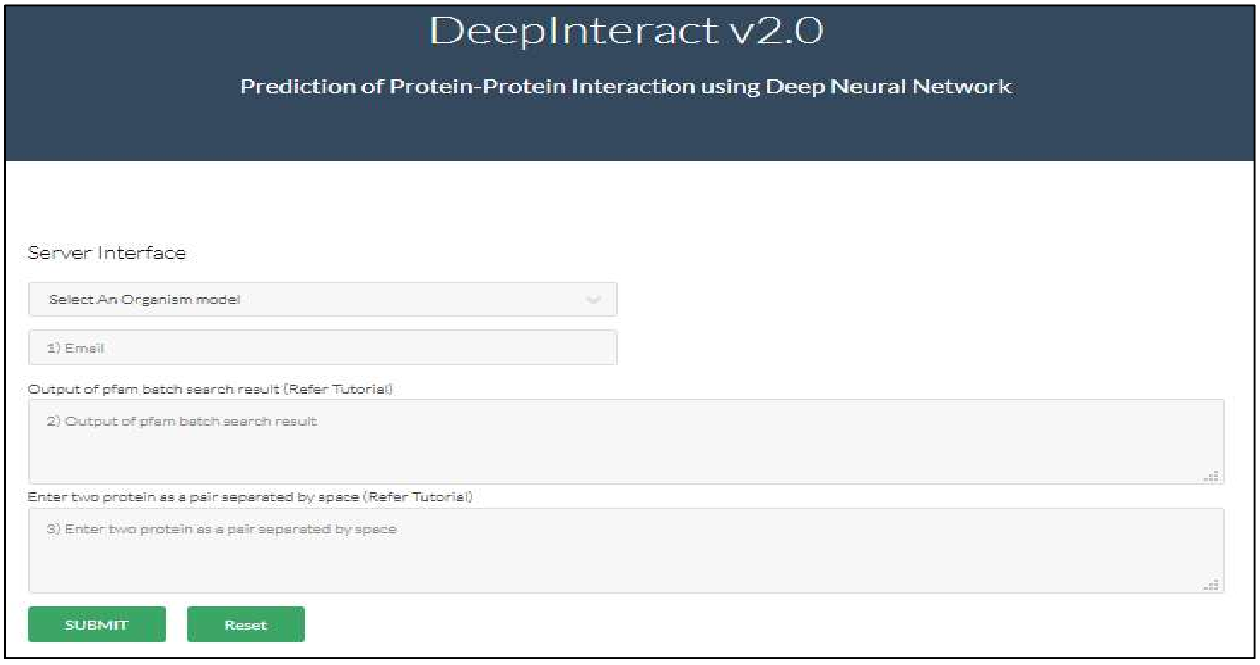
Deep Neural Network for Long Non-coding RNA (lnc) Prediction.

DNNOL

Deep Neural Network Engine for On-line Prediction.
Updating Soon

Bioserver © Indian Institute of Information Technology-Allahabad.
Contact Us | Developer
001054

Figure 10: Bioserver user interface at http://bioserver.iiita.ac.in



DeepInteract v2.0

Prediction of Protein-Protein Interaction using Deep Neural Network

Server Interface

Select An Organism model

1) Email

Output of pfam batch search result (Refer Tutorial)

2) Output of pfam batch search result

Enter two protein as a pair separated by space (Refer Tutorial)

3) Enter two protein as a pair separated by space

SUBMIT | Reset

Figure 11: DeepInteract utility user interface at Bioserver (http://bioserver.iiita.ac.in/deepinteract/)

1. Architecture

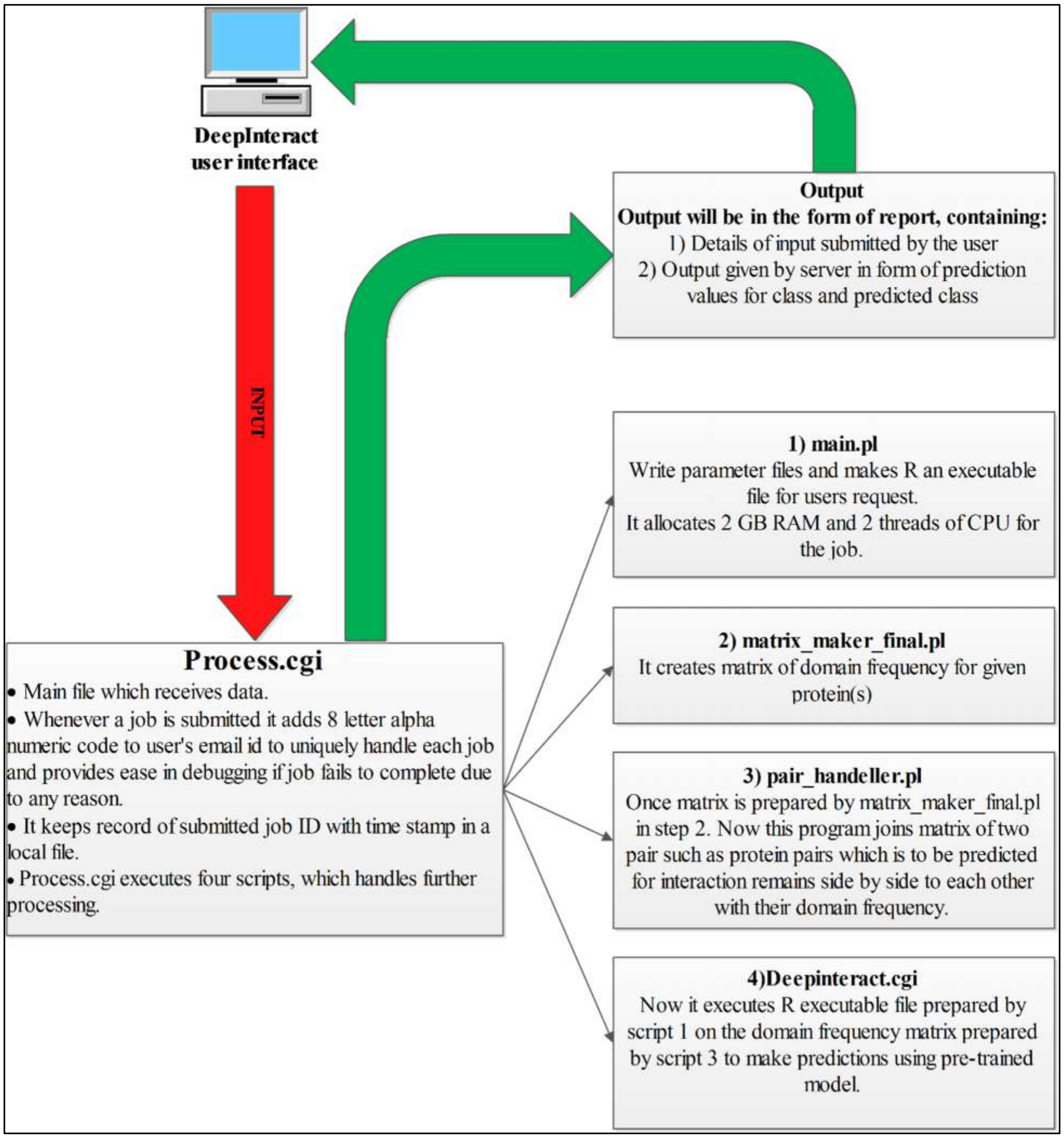
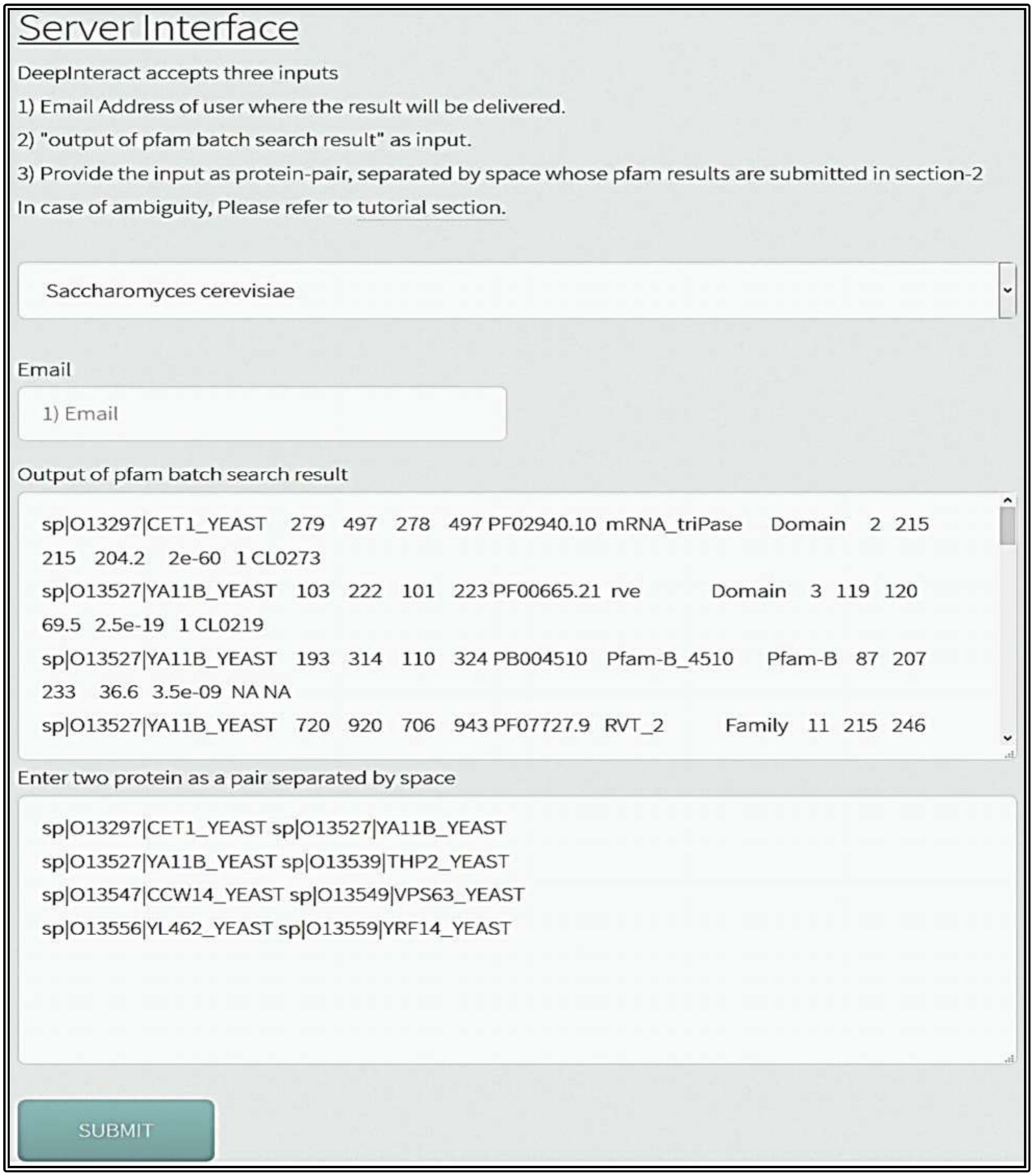

DeepInteract user interface
INPUT
Output
Output will be in the form of report, containing:
1) Details of input submitted by the user
2) Output given by server in form of prediction values for class and predicted class
1) main.pl
Write parameter files and makes R an executable file for users request.
It allocates 2 GB RAM and 2 threads of CPU for the job.
2) matrix_maker_final.pl
It creates matrix of domain frequency for given protein(s)
3) pair_handeller.pl
Once matrix is prepared by matrix_maker_final.pl in step 2. Now this program joins matrix of two pair such as protein pairs which is to be predicted for interaction remains side by side to each other with their domain frequency.
4)Deepinteract.cgi
Now it executes R executable file prepared by script 1 on the domain frequency matrix prepared by script 3 to make predictions using pre-trained model.
Process.cgi
• Main file which receives data.
• Whenever a job is submitted it adds 8 letter alpha numeric code to user's email id to uniquely handle each job and provides ease in debugging if job fails to complete due to any reason.
• It keeps record of submitted job ID with time stamp in a local file.
• Process.cgi executes four scripts, which handles further processing.

Figure 12: Depicts an internal architecture and job handling process performed by the server.

2. Input to Bioserver

DeepInteract takes two input- 1) domains corresponding to the individual protein *i.e.* the output generated by pfam batch search result. This enables user to submit as many sequence as possible at a time, 2) a protein pair which is to be tested for the PPI. User can submit *n* number of protein pairs, but domains corresponding to those proteins must be submitted in input 1.



Server Interface

DeepInteract accepts three inputs

1) Email Address of user where the result will be delivered.

2) "output of pfam batch search result" as input.

3) Provide the input as protein-pair, separated by space whose pfam results are submitted in section-2

In case of ambiguity, Please refer to tutorial section.

Saccharomyces cerevisiae

Email

1) Email

Output of pfam batch search result

```
sp|O13297|CET1_YEAST   279   497   278   497 PF02940.10  mRNA_triPase    Domain    2  215
215   204.2    2e-60   1 CL0273
sp|O13527|YA11B_YEAST   103   222   101   223 PF00665.21  rve         Domain    3  119  120
69.5  2.5e-19   1 CL0219
sp|O13527|YA11B_YEAST   193   314   110   324 PB004510   Pfam-B_4510     Pfam-B   87  207
233    36.6  3.5e-09  NA NA
sp|O13527|YA11B_YEAST   720   920   706   943 PF07727.9  RVT_2       Family   11  215  246
```

Enter two protein as a pair separated by space

```
sp|O13297|CET1_YEAST sp|O13527|YA11B_YEAST
sp|O13527|YA11B_YEAST sp|O13539|THP2_YEAST
sp|O13547|CCW14_YEAST sp|O13549|VPS63_YEAST
sp|O13556|YL462_YEAST sp|O13559|YRF14_YEAST
```

Figure 13: Illustration of data as an input accepted by DeepInteract server for predicting protein-protein Interactions

Result generated by server consists of a prediction table. In the prediction table, first column indicates whether the submitted pair is predicted as interacting one or non-interacting one. If first column is labeled as "1" then the interaction between two proteins is said to be positive otherwise it is considered as negative. Other two column indicates the probability of protein pair being interacting or non-interacting.

3. <u>Output</u>

Presently in version: 1.1, results are served in an archive file (.zip) to the user as soon as the job finishes (usually within 1.2 seconds of submission). Results contain input submitted by user, matrix generated from user's input for processing using existing model of an organism and an output file containing predicted scores for protein pair in the same order as submitted by user in pair. The conclusion about interaction can be drawn from comparing the scores for both classes with a threshold value. Output generated by the server is illustrated in APPENDIX-2.

Standalone software

A standalone version of DeepInteract Version 1.1 is available on Github at (**https://github.com/snlpatel001213/DeepInteract/**) keeping in view of researcher's need of batch processing data to predict protein-protein interaction. Standalone is made easy to install with Perl based installer and available under The MIT License (MIT) license, Copyright (c) 2015 snlpatel001213. Steps to be followed for installation of DeepInteract standalone can be found in the APPENDIX-3.

DISCUSSION

Several studies have predicted PPI using different machine learning techniques. Most of the researchers used database of *Saccharomyces cerevisiae* from DIP for the training and testing of PPI data. It is also important to mention that most of the earlier work have emphasized on protein domain information. We compared our findings with previous work using the similar data source of same organism and we found that our work using DNN is more accurate in comparison to work done by other researcher using classifiers like SVM, RF and ELM. The specificity, sensitivity and accuracy reported in the proposed work are found to be better than the previously reported results. A comparative chart summarizing these approaches, algorithms used, feature set along with accuracy are shown as Table 5.

Table 5. A comparative table summarizing the machine learning methods, features, dataset and prediction accuracy for PPI

Sl. No	**Algorithm**	**Dataset**	**Features**	**Accuracy (%)**
1	Support Vector Machine (Chatterjee, 2011)	DIP database	Frequency of Interacting Domains	86
2	Decision forest framework (Chen XW, 2005)	DIP database	Frequency of Interacting Domains	72.81
3	Support vector Machine (Das P, 2011)	DIP database	Frequency of Interacting Domains	91.22
4	Support vector Machine (Guo Y, 2008)	DIP_20070219 (negative data set of non-interacting pairs of non-co-localized proteins)	Physiochemical properties (hydrophobicity, hydrophicility, volumes of side chains of amino acids, polarity, polarizability, solvent-accessible surface area and net charge index)	86.23
5	Extreme Learning Machines (ELM), (http://www.ntu.edu.sg/home/egbhuang) and Support Vector Machine (You, Z. et al. 2013)	Dataset of Guo et al., 2008	Auto Covariance (AC), Conjoint Triad (CT), Local Descriptor (LD) and Moran autocorrelation on	87.00%

			physiological properties of amino acid	
6	Support Vector Machine (Rashid M, 2010)	Database from: Martin et al., 2005 Yellaboina et al., 2007 Pitre et al., 2006	Amino acid, dipeptide, pseudo amino acid and split amino acid	81.0

The same has been ported on web as an online tool available at http://bioserver.iiita.ac.in/deepinteract. The protein domain information are used as a feature vectors comprising of domain frequency for the DNN, the accuracy, sensitivity and precision of the models are estimated to be 92.67%, 86.85% and 98.31% for *Saccharomyces cerevisiae,* for *Escherichia coli* is 97.01%, 94.49%, 99.51%, for *Homo sapiens* is 94.47%, 90.47%, 98.34 *%,* for *Drosophila melanogaster* is 90.85%, 98.55%, 82.91% and for *Caenorhabditis elegans* is calculated to be 88.91%, 88.18%, 96.04%, respectively.

We have used minimal information (*viz*. protein domain information) for creating feature vector as far as the complexity of problem is concerned. The proposed method has very good precision with Youden's Index ranging between 0.77 to 0.88 for all models. The F rate of these models also suggest the acceptability of these models on terms of sensitivity and precision. Currently the models are limited to the five species, although the identification of protein domains which are conserved through different species can lead to a universal model for predicting PPI across species which is the subject of future research.

REFERENCES

1) Bahadur, R. P., and M. Zacharias. "The interface of protein-protein complexes: analysis of contacts and prediction of interactions." Cellular and Molecular Life Sciences 65.7-8 (2008): 1059-1072.
2) Block, P., Paern, J., Huellermeier, E., Sanschagrin, P., Sotriffer, C. A., & Klebe, G. (2006). Physicochemical descriptors to discriminate protein–protein interactions in permanent and transient complexes selected by means of machine learning algorithms. PROTEINS: Structure, Function, and Bioinformatics, 65(3), 607-622.
3) Bock, J. R., & Gough, D. A. (2001). Predicting protein–protein interactions from primary structure. Bioinformatics, 17(5), 455-460.
4) Boxem, M., Srinivasan, D. G., & van den Heuvel, S. (1999). The Caenorhabditis elegans gene ncc-1 encodes a cdc2-related kinase required for M phase in meiotic and mitotic cell divisions, but not for S phase. Development, 126(10), 2227-2239.
5) Chatterjee, P., Basu, S., Kundu, M., Nasipuri, M., & Plewczynski, D. (2011). PPI_SVM: Prediction of protein-protein interactions using machine learning, domain-domain affinities and frequency tables. Cellular & molecular biology letters, 16(2), 264-278.
6) Chen, X. W., & Liu, M. (2005). Prediction of protein–protein interactions using random decision forest framework. Bioinformatics, 21(24), 4394-4400.
7) Chen, Xue-Wen, and Mei Liu. "Domain-based predictive models for protein-protein interaction prediction." EURASIP Journal on Applied Signal Processing 2006 (2006): 55-55.
8) Chen, Xue-wen, Jong Cheol Jeong, and Patrick Dermyer. "KUPS: constructing datasets of interacting and non-interacting protein pairs with associated attributions." Nucleic acids research (2010): gkq943.
9) Chiarugi, A., & Moskowitz, M. A. (2002). PARP-1—a perpetrator of apoptotic cell death?. gene, 9, 11.
10) Connell-Crowley, L., Solomon, M. J., Wei, N., & Harper, J. W. (1993). Phosphorylation independent activation of human cyclin-dependent kinase 2 by cyclin A in vitro. Molecular biology of the cell, 4(1), 79-92.
11) Conte, Loredana Lo, Cyrus Chothia, and Joël Janin. "The atomic structure of protein-protein recognition sites." Journal of molecular biology 285.5 (1999): 2177-2198.
12) Dejean, L. M., Martinez-Caballero, S., & Kinnally, K. W. (2006). Is MAC the knife that cuts cytochrome c from mitochondria during apoptosis?. Cell Death & Differentiation, 13(8), 1387-1395.
13) Deris, Safaai, H. Alashwal, and M. R. Othman. "One-class support vector machines for protein-protein interactions prediction." International Journal of Biological and Medical Sciences 1.2 (2006): 120-127.

14) Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12, 2121-2159.
15) Fesik, S. W., & Shi, Y. (2001). Controlling the caspases. Science, 294(5546), 1477-1478.
16) Fields, S., & Song, O. (1989). A novel genetic system to detect protein-protein interactions. Nature, 340(6230), 245-246.
17) Ganten, D. (2006). Encyclopedic reference of genomics and proteomics in molecular medicine (Vol. 2). W. Birchmeier, J. T. Epplen, K. Genser, M. Gossen, B. Kersten, H. Lehrach, ... & C. Nolte (Eds.). Springer.
18) Gonzalez, M. W., & Kann, M. G. (2012). Protein interactions and disease. PLoS computational biology, 8(12), e1002819.
19) Guo, Y., Yu, L., Wen, Z., & Li, M. (2008). Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences. Nucleic acids research, 36(9), 3025-3030.
20) Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.
21) Hinton, G., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural computation, 18(7), 1527-1554.
22) Jansen, R., Yu, H., Greenbaum, D., Kluger, Y., Krogan, N. J., Chung, S., ... & Gerstein, M. (2003). A Bayesian networks approach for predicting protein-protein interactions from genomic data. Science, 302(5644), 449-453.
23) Jones, Susan, and Janet M. Thornton. "Principles of protein-protein interactions." Proceedings of the National Academy of Sciences 93.1 (1996): 13-20.
24) Kerrien, S., Aranda, B., Breuza, L., Bridge, A., Broackes-Carter, F., Chen, C., ... & Hermjakob, H. (2011). The IntAct molecular interaction database in 2012. Nucleic acids research, gkr1088.
25) Kerrien, S., Aranda, B., Breuza, L., Bridge, A., Broackes-Carter, F., Chen, C., ... & Hermjakob, H. (2011). The IntAct molecular interaction database in 2012. Nucleic acids research, gkr1088.
26) Kim, W. K., Park, J., & Suh, J. K. (2002). Large scale statistical prediction of protein-protein
27) Kim, W. K., Park, J., & Suh, J. K. (2002). Large scale statistical prediction of protein-protein
28) Lee, Byungkook, and Frederic M. Richards. "The interpretation of protein structures: estimation of static accessibility." Journal of molecular biology 55.3 (1971): 379-IN4.
29) Li, Ye, Xianren Zhang, and Dapeng Cao. "The Role of Shape Complementarity in the Protein-Protein Interactions." Scientific reports 3 (2013).
30) Martin,S. Roe,D. Faulon,JL. (2005). Predicting protein–protein interactions using signature products. Bioinformatics, 2*1*(2), 218-226.
31) Nesterov, Y. (2007). Gradient methods for minimizing composite objective function.

32) Pawson, T., & Nash, P. (2003). Assembly of cell regulatory systems through protein interaction domains. Science, 300(5618), 445-452.

33) Pawson, T., & Nash, P. (2003). Assembly of cell regulatory systems through protein interaction domains. Science, 300(5618), 445-452.

34) Pazos, F., & Valencia, A. (2001). Similarity of phylogenetic trees as indicator of protein–protein interaction. Protein engineering, 14(9), 609-614.

35) Pitre,S. Dehne,F. Chan,A. Cheetham,J. Duong,A. Emili,A. Golshani,A. (2006). PIPE: a protein-protein interaction prediction engine based on the re-occurring short polypeptide sequences between known interacting protein pairs. BMC bioinformatics, 7(1), 365.

36) Prasad, T. K., Goel, R., Kandasamy, K., Keerthikumar, S., Kumar, S., Mathivanan, S., ... & Pandey, A. (2009). Human protein reference database—2009 update. Nucleic acids research, 37(suppl 1), D767-D772.

37) Prasad, T. K., Goel, R., Kandasamy, K., Keerthikumar, S., Kumar, S., Mathivanan, S., ... & Pandey, A. (2009). Human protein reference database—2009 update. Nucleic acids research, 37(suppl 1), D767-D772.

38) Punta, M., Coggill, P. C., Eberhardt, R. Y., Mistry, J., Tate, J., Boursnell, C., ... & Finn, R. D. (2011). The Pfam protein families database. Nucleic acids research, gkr1065.

39) Punta, M., Coggill, P. C., Eberhardt, R. Y., Mistry, J., Tate, J., Boursnell, C., ... & Finn, R. D. (2011). The Pfam protein families database. Nucleic acids research, gkr1065.

40) Qi, Y., Klein-Seetharaman, J., & Bar-Joseph, Z. (2004, December). Random forest similarity for protein-protein interaction prediction from multiple sources. In Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing (pp. 531-542).

41) Rashid, M., Ramasamy, S., & PS Raghava, G. (2010). A simple approach for predicting protein-protein interactions. Current Protein and Peptide Science, 11(7), 589-600.

42) Sanchez C; Lachaize C; Janody F *et al.* (January 1999). "Grasping at molecular interactions and genetic networks in Drosophila melanogaster using FlyNets, an Internet database". Nucleic Acids Res. 27 (1): 89–94. doi:10.1093/nar/27.1.89. PMC 148104. PMID 9847149.

43) Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.

44) Stumpf MP; Thorne T; de Silva E *et al.* (May 2008). "Estimating the size of the human interactome". Proc. Natl. Acad. Sci. U.S.A. 105 (19): 6959–64. doi:10.1073/pnas.0708078105. PMC 2383957. PMID 18474861.

45) Uetz, P., Giot, L., Cagney, G., Mansfield, T. A., Judson, R. S., Knight, J. R., & Rothberg, J. M. (2000). A comprehensive analysis of protein–protein interactions in Saccharomyces cerevisiae. Nature, 403(6770), 623-627.

46) Vasilescu, J., Guo, X., & Kast, J. (2004). Identification of protein-protein interactions using in vivo cross-linking and mass spectrometry. Proteomics, 4(12), 3845-3854.

47) Wager, S., Wang, S., & Liang, P. (2013). Dropout training as adaptive regularization. In Advances in Neural Information Processing Systems (pp. 351-359)

48) Wu, C. H., Apweiler, R., Bairoch, A., Natale, D. A., Barker, W. C., Boeckmann, B., ... & Suzek, B. (2006). The Universal Protein Resource (UniProt): an expanding universe of protein information. Nucleic acids research, 34(suppl 1), D187-D191.
49) Wu, C. H., Apweiler, R., Bairoch, A., Natale, D. A., Barker, W. C., Boeckmann, B., ... & Suzek, B. (2006). The Universal Protein Resource (UniProt): an expanding universe of protein information. Nucleic acids research, 34(suppl 1), D187-D191.
50) Wu, Y., Li, Q., & Chen, X. Z. (2007). Detecting protein–protein interactions by far western blotting. Nature protocols, 2(12), 3278-3284.
51) Xenarios, I., Salwinski, L., Duan, X. J., Higney, P., Kim, S. M., & Eisenberg, D. (2002). DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. Nucleic acids research, 30(1), 303-305.
52) Xenarios, I., Salwinski, L., Duan, X. J., Higney, P., Kim, S. M., & Eisenberg, D. (2002). DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. Nucleic acids research, 30(1), 303-305.
53) Yellaboina,S. Goyal,K. Mande,SC. (2007). Inferring genome-wide functional linkages in *E. coli* by combining improved genome context methods: comparison with high-throughput experimental data. Genome research, 17(4), 527-535.
54) You,Z. et al. (2013) Prediction of protein-protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis. BMC bioinformatics, 14(8), S10.
55) Zanzoni, A., Montecchi-Palazzi, L., Quondam, M., Ausiello, G., Helmer-Citterich, M., & Cesareni, G. (2002). MINT: a Molecular INTeraction database. FEBS letters, 513(1), 135-140.
56) Zanzoni, A., Montecchi-Palazzi, L., Quondam, M., Ausiello, G., Helmer-Citterich, M., & Cesareni, G. (2002). MINT: a Molecular INTeraction database. FEBS letters, 513(1), 135-140.
57) Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. arXiv preprint arXiv:1212.5701.
58) Zhang, Q. C., Petrey, D., Deng, L., Qiang, L., Shi, Y., Thu, C. A., & Honig, B. (2012). Structure-based prediction of protein-protein interactions on a genome-wide scale. Nature, 490(7421), 556-560.
59) Zhu, H., Bilgin, M., Bangham, R., Hall, D., Casamayor, A., Bertone, P., Lan, N., *et al.*. (2001). Global analysis of protein activities using proteome chips. Science (New York, N.Y.), 293(5537), 2101-2105.

APPENDIX -1

Steps to install and setup OXDATA H20 on windows 8.1 amd64 machine

Installation Instructions

It consists of the following steps:

STEP 1. Installation of java jdk environment

- Goto http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Download 1.8.0_25 version of java jdk
- Run the installation on your machine

STEP 2. Installation of R programming language

- Goto http://cran.r-project.org/bin/windows/base/
- Download R 3.1.2 for Windows (54 megabytes, 32/64 bit)

STEP 3. Installing H20 Oxdata framework

- Goto http://0xdata.com/download/
- Download version 2.9.0.15XX
    - Unzip the content to the desktop
    - Open command prompt in Administrator privileges
    - Reach to the folder of H20 on desktop which has just been extracted
    - At that location type following commands:

```
cd ~/Desktop/
cd h2o-2.9.0.1586
java -jar h2o.jar
```

    - Point your browser to http://localhost:54321 (make sure your proxy set to allow local host)

STEP 4. Installing Interface for R

- Rstudio (it is an IDE for better programming environment for R)

- H20 installation depends on the following dependencies (obtain all packages from CRAN and run)
    I. RCurl
    II. rjson
    III. statmode
    IV. bitops
    V. manipulate

- Go to the folder of H20 which has already been extracted on the Desktop
- Goto folder named R in it
- Install H20 package for R from there

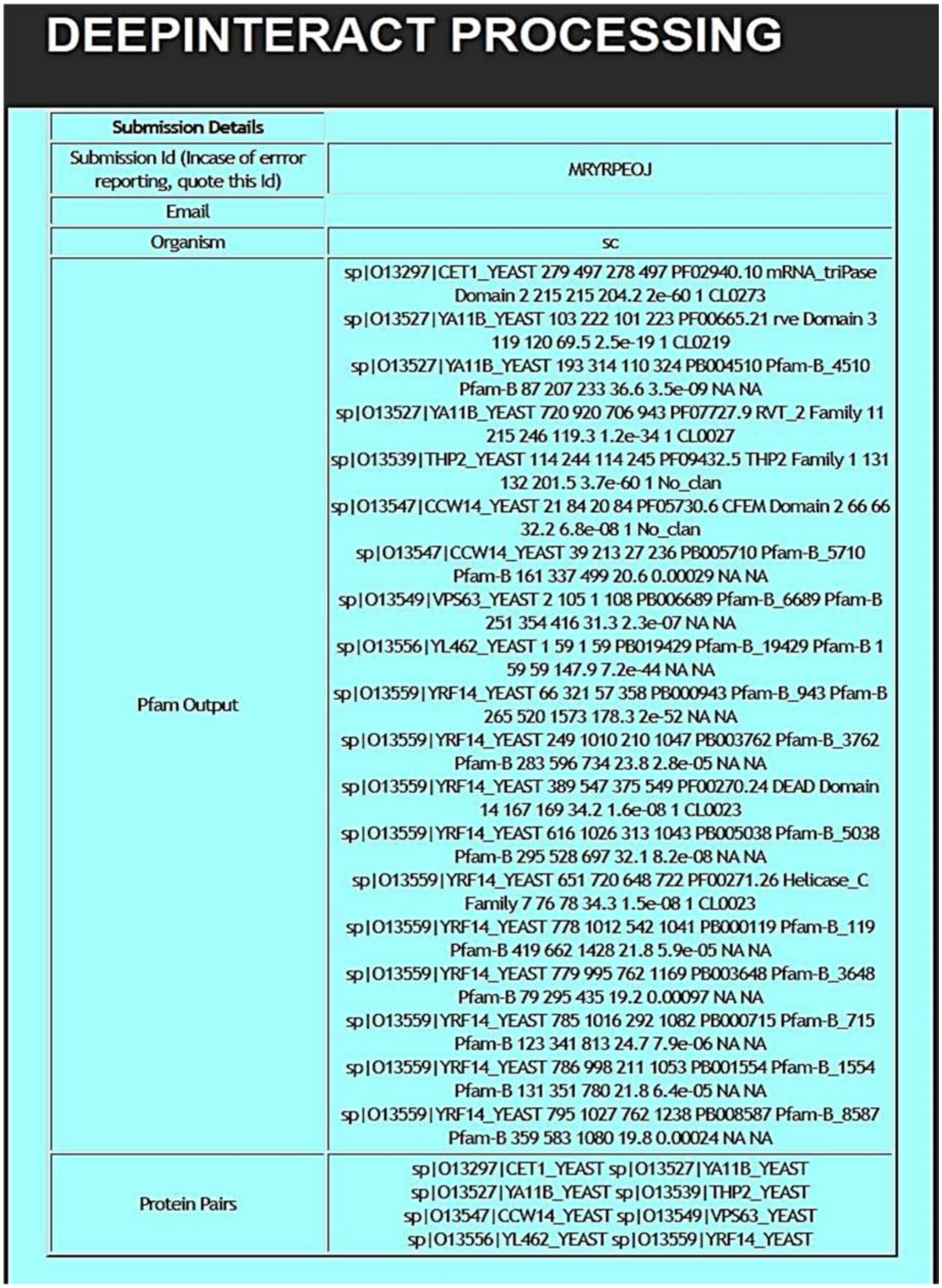
OR

After STEP-2, one can run this script in R directly and it will setup everything for you automatically. Practically it is found that manual configuration always remain more successful than automatic one

```
# The following two commands remove any previously installed H2O packages for R.
if ("package:h2o" %in% search()) { detach("package:h2o", unload=TRUE) }
if ("h2o" %in% rownames(installed.packages())) { remove.packages("h2o") }

# Next, we download packages that H2O depends on.
if (! ("methods" %in% rownames(installed.packages()))) { install.packages("methods") }
if (! ("statmod" %in% rownames(installed.packages()))) { install.packages("statmod") }
if (! ("stats" %in% rownames(installed.packages()))) { install.packages("stats") }
if (! ("graphics" %in% rownames(installed.packages()))) { install.packages("graphics") }
if (! ("RCurl" %in% rownames(installed.packages()))) { install.packages("RCurl") }
if (! ("rjson" %in% rownames(installed.packages()))) { install.packages("rjson") }
if (! ("tools" %in% rownames(installed.packages()))) { install.packages("tools") }
if (! ("utils" %in% rownames(installed.packages()))) { install.packages("utils") }

# Next, we download, install and initialize the H2O package for R.
```

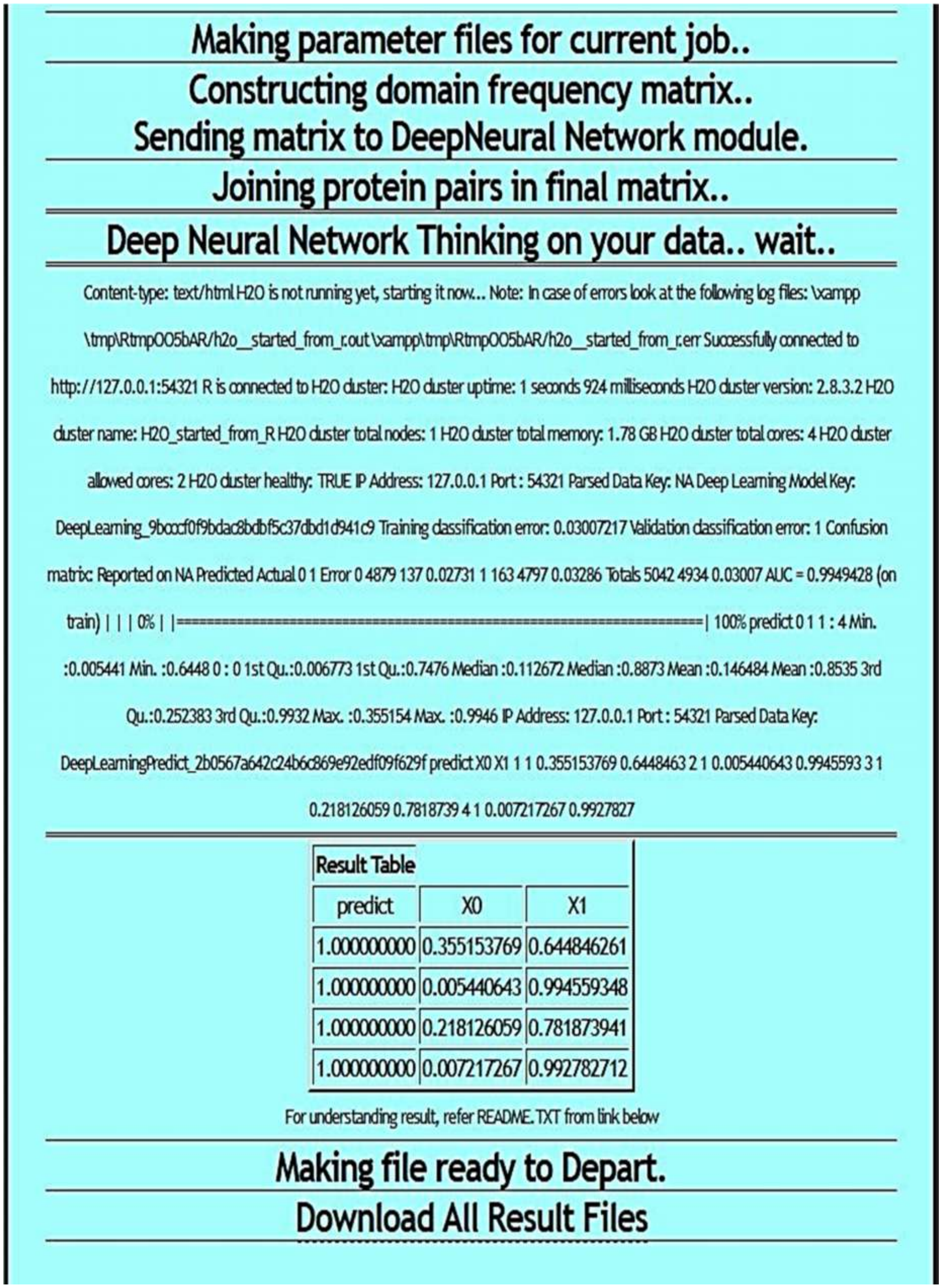
```
install.packages("h2o", repos=(c("http://s3.amazonaws.com/h2o-release/h2o/master/1757/R",
getOption("repos"))))
library(h2o)
localH2O = h2o.init()= h2o.init()

# Finally, let's run a demo to see H₂O at work.
demo(h2o.glm)
```

APPENDIX-2



DEEPINTERACT PROCESSING

Submission Details	
Submission Id (Incase of errror reporting, quote this Id)	MRYRPEOJ
Email	
Organism	sc
Pfam Output	sp|O13297|CET1_YEAST 279 497 278 497 PF02940.10 mRNA_triPase Domain 2 215 215 204.2 2e-60 1 CL0273 sp|O13527|YA11B_YEAST 103 222 101 223 PF00665.21 rve Domain 3 119 120 69.5 2.5e-19 1 CL0219 sp|O13527|YA11B_YEAST 193 314 110 324 PB004510 Pfam-B_4510 Pfam-B 87 207 233 36.6 3.5e-09 NA NA sp|O13527|YA11B_YEAST 720 920 706 943 PF07727.9 RVT_2 Family 11 215 246 119.3 1.2e-34 1 CL0027 sp|O13539|THP2_YEAST 114 244 114 245 PF09432.5 THP2 Family 1 131 132 201.5 3.7e-60 1 No_clan sp|O13547|CCW14_YEAST 21 84 20 84 PF05730.6 CFEM Domain 2 66 66 32.2 6.8e-08 1 No_clan sp|O13547|CCW14_YEAST 39 213 27 236 PB005710 Pfam-B_5710 Pfam-B 161 337 499 20.6 0.00029 NA NA sp|O13549|VPS63_YEAST 2 105 1 108 PB006689 Pfam-B_6689 Pfam-B 251 354 416 31.3 2.3e-07 NA NA sp|O13556|YL462_YEAST 1 59 1 59 PB019429 Pfam-B_19429 Pfam-B 1 59 59 147.9 7.2e-44 NA NA sp|O13559|YRF14_YEAST 66 321 57 358 PB000943 Pfam-B_943 Pfam-B 265 520 1573 178.3 2e-52 NA NA sp|O13559|YRF14_YEAST 249 1010 210 1047 PB003762 Pfam-B_3762 Pfam-B 283 596 734 23.8 2.8e-05 NA NA sp|O13559|YRF14_YEAST 389 547 375 549 PF00270.24 DEAD Domain 14 167 169 34.2 1.6e-08 1 CL0023 sp|O13559|YRF14_YEAST 616 1026 313 1043 PB005038 Pfam-B_5038 Pfam-B 295 528 697 32.1 8.2e-08 NA NA sp|O13559|YRF14_YEAST 651 720 648 722 PF00271.26 Helicase_C Family 7 76 78 34.3 1.5e-08 1 CL0023 sp|O13559|YRF14_YEAST 778 1012 542 1041 PB000119 Pfam-B_119 Pfam-B 419 662 1428 21.8 5.9e-05 NA NA sp|O13559|YRF14_YEAST 779 995 762 1169 PB003648 Pfam-B_3648 Pfam-B 79 295 435 19.2 0.00097 NA NA sp|O13559|YRF14_YEAST 785 1016 292 1082 PB000715 Pfam-B_715 Pfam-B 123 341 813 24.7 7.9e-06 NA NA sp|O13559|YRF14_YEAST 786 998 211 1053 PB001554 Pfam-B_1554 Pfam-B 131 351 780 21.8 6.4e-05 NA NA sp|O13559|YRF14_YEAST 795 1027 762 1238 PB008587 Pfam-B_8587 Pfam-B 359 583 1080 19.8 0.00024 NA NA
Protein Pairs	sp|O13297|CET1_YEAST sp|O13527|YA11B_YEAST sp|O13527|YA11B_YEAST sp|O13539|THP2_YEAST sp|O13547|CCW14_YEAST sp|O13549|VPS63_YEAST sp|O13556|YL462_YEAST sp|O13559|YRF14_YEAST



Making parameter files for current job..

Constructing domain frequency matrix..
Sending matrix to DeepNeural Network module.

Joining protein pairs in final matrix..

Deep Neural Network Thinking on your data.. wait..

```
Content-type: text/html H2O is not running yet, starting it now... Note: In case of errors look at the following log files: \xampp
\tmp\RtmpOO5bAR/h2o__started_from_r.out \xampp\tmp\RtmpOO5bAR/h2o__started_from_r.err Successfully connected to
http://127.0.0.1:54321 R is connected to H2O cluster: H2O cluster uptime: 1 seconds 924 milliseconds H2O cluster version: 2.8.3.2 H2O
cluster name: H2O_started_from_R H2O cluster total nodes: 1 H2O cluster total memory: 1.78 GB H2O cluster total cores: 4 H2O cluster
allowed cores: 2 H2O cluster healthy: TRUE IP Address: 127.0.0.1 Port : 54321 Parsed Data Key: NA Deep Learning Model Key:
DeepLearning_9bcccf0f9bdac8bdbf5c37dbd1d941c9 Training classification error: 0.03007217 Validation classification error: 1 Confusion
matrix: Reported on NA Predicted Actual 0 1 Error 0 4879 137 0.02731 1 163 4797 0.03286 Totals 5042 4934 0.03007 AUC = 0.9949428 (on
train) | | | 0% | |=====================================================================| 100% predict 0 1 1 : 4 Min.
:0.005441 Min. :0.6448 0 : 0 1st Qu.:0.006773 1st Qu.:0.7476 Median :0.112672 Median :0.8873 Mean :0.146484 Mean :0.8535 3rd
Qu.:0.252383 3rd Qu.:0.9932 Max. :0.355154 Max. :0.9946 IP Address: 127.0.0.1 Port : 54321 Parsed Data Key:
DeepLearningPredict_2b0567a642c24b6c869e92edf09f629f predict X0 X1 1 1 0.355153769 0.6448463 2 1 0.005440643 0.9945593 3 1
0.218126059 0.7818739 4 1 0.007217267 0.9927827
```

Result Table

predict	X0	X1
1.000000000	0.355153769	0.644846261
1.000000000	0.005440643	0.994559348
1.000000000	0.218126059	0.781873941
1.000000000	0.007217267	0.992782712

For understanding result, refer README.TXT from link below

Making file ready to Depart.

Download All Result Files

Figure 14: Illustration of result generated by DeepInteract server after query submission.

APPENDIX-3

DeepInteract - A Deep Neural Network for Protein Protein Interaction Prediction (neural network of more than 5 million connections)

1) **You must have Perl version 5.14 installed.**
2) **Make sure you already have installed Xampp server (version 3.1.0 or higher).**
3) **Installation of java jdk environment.**
    - Goto http://www.oracle.com/technetwork/java/javase/downloads/index.html
    - Download 1.8.0_25 version of java jdk .
    - Run the installation on your machine.
4) **Make sure you already have installed version of R (3.1.x) and H20 oxdata program.**
    - if not......then follow, following steps.
    - **STEP 1**. Installation of R programming language.
        I. Goto http://cran.r-project.org/bin/windows/base/
        II. Download R 3.1.2 for Windows (54 megabytes, 32/64 bit).

      STEP 2. Installing H20 Oxdata framework
        I. Goto http://0xdata.com/download/
        II. Download version 2.9.0.15XX.

    ✓ Unzip the content to the desktop.
    ✓ Open command prompt in Administrator privileges.
    ✓ Reach to the folder of H20 on desktop which has just been extracted.
    ✓ At that location type following commands:

```
cd ~/Desktop/                                   cd h2o-
2.9.0.1586                          java -jar h2o.jar
```

    - Point your browser to http://localhost:54321 (make sure your proxy set to allow local host).
5) **Installing Interface for R**.
    - Rstudio (it is an IDE for better programming environment for R).
    - H20 installation depends on the following dependencies (obtain all packages from CRAN and run).
        a. RCurl

b. rjson
c. statmode
d. bitops
e. manipulate

- Go to the folder of H20 which has already been extracted on the Desktop
- Goto folder named R in it
- Install H20 package for R from there

6) Installion of DeepInteract

run install.perl from command line or from ide. This will do all dirty work for you