Supplementary Material

deepNF: Deep network fusion for protein function prediction

Vladimir Gligorijević [*1], Meet Barot [†1], and Richard Bonneau [‡1,2]

[1]Center for Computational Biology, Flatiron Institute, Simons Foundation, New York, NY 10010, USA
[2]Department of Biology, Center for Genomics and Systems Biology, New York University, New York, NY 10003, USA

[*]vgligorijevic@flatironinstitute.org
[†]mbarot@flatironinstitute.org
[‡]rb133@nyu.edu

1 Structural properties of individual STRING networks

	Yeast					Human				
network type	ρ	$\langle k \rangle$	R	D	$\langle C \rangle$	ρ	$\langle k \rangle$	R	D	$\langle C \rangle$
neighborhood	0.019	41.99	4	8	0.23	0.009	30.04	5	9	0.22
fusion	0.011	3.49	8	14	0.05	0.017	4.17	7	13	0.05
cooccurence	0.016	6.94	8	15	0.45	0.005	13.01	11	21	0.45
coexpression	0.019	109.01	6	11	0.42	0.006	102.25	7	14	0.38
experimental	0.011	71.34	3	6	0.16	0.002	37.28	5	9	0.19
database	0.011	27.11	10	19	0.62	0.005	41.46	6	11	0.59

Table S1: **Basic measures of individual STRING networks:** network density $\rho = \frac{2L}{N(N-1)}$; average node degree $\langle k \rangle$; network radius (minimum eccentricity) R; network diameter (maximum eccentricity) D; average clustering coefficient $\langle C \rangle$. All measures are computed on a network's largest connected component.

2 Annotation data for temporal holdout

	Yeast		Human	
	# (train, test, valid)	GO terms	# (train, test, valid)	GO terms
MF	(3436, 202, 966)	20	(8633, 1131, 3596)	74
BP	(3293, 170, 1561)	43	(6818, 1272, 5107)	331
CC	(3424, 246, 879)	11	(7656, 1254, 4843)	54

Table S2: **Train, test and validation annotations used in the temporal holdout validation of our method.**

deepNF Algorithm

Input: Adjacency matrices $\{\mathbf{A}^{(1)}, \dots \mathbf{A}^{(N)}\}$,
annotation matrix $\mathbf{Y} \in \mathbb{R}^{n \times f}$
Output: Predicted function score matrix $\hat{\mathbf{Y}} \in \mathbb{R}^{|\mathbf{r}| \times f}$

---

for $j \in \{1, \dots N\}$ **do**
  $\mathbf{P}^{(j)} = \text{RWR}(\mathbf{A}^{(j)})$; (Eq. 1)
  $\mathbf{X}^{(j)} = \text{PPMI}(\mathbf{P}^{(j)})$; (Eq. 2)
end
$\theta = \text{MDATrain}(\{\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(N)}\})$; (Eq. 3)
$\mathbf{H}_c = \text{ExtractMDAFeatures}(\{\mathbf{X}^{(1)}, \dots \mathbf{X}^{N}\})$;
for $h \in \{1, \dots f\}$ **do**
  $\beta_h = \text{SVMTrain}(\mathbf{H}_c\{\mathbf{q}\}, \mathbf{Y}_{:h}\{\mathbf{q}\})$;
end
for $h \in \{1, \dots f\}$ **do**
  $\hat{\mathbf{Y}}_{:h}\{\mathbf{r}\} = \text{SVMPredict}(\beta_h, \mathbf{H}_c\{\mathbf{r}\})$;
end

Algorithm 1: deepNF protocol. N is the number of protein networks, n is the number of proteins in each network, and f is the number of function columns in $\mathbf{Y}$. The training and test protein indices are given by $\mathbf{q}$ and $\mathbf{r}$, respectively. θ is the set of the MDA parameters trained to reconstruct the input PPMI matrices. β_h is the set of SVM parameters trained for function h.

3 Model configuration

We implemented our MDA model in `Keras` [1] with the `TensorFlow` [2] backend and trained it using the standard backpropagation algorithm. Each layer in the MDA is implemented as a `Dense` layer with the `sigmoid` activation function. Our MDA consists of multiple hidden layers that map multiple inputs, represented by 6 PPMI matrices of STRING networks, to multiple outputs (see Fig. 1 in the main manuscript). After mapping the input layers to 6 hidden layers, we `concatenated` them to a single layer that is further used as an input to next hidden layer. Different MDA encoding architectures that we use in our method, are provided in Table S3.

- **Model encoding architecture**:

Yeast	Human
$[6 \times n, 600]$	$[6 \times n, 1200]$
$[6 \times n, 6 \times 2000, 600]$	$[6 \times n, 6 \times 2500, 1200]$
$[6 \times n, 6 \times 2000, 6000, 600]$	$[6 \times n, 6 \times 2500, 9000, 1200]$
$[6 \times n, 6 \times 2000, 6000, 3000, 600]$	$[6 \times n, 6 \times 2500, 9000, 6000, 600]$
$[6 \times n, 6 \times 2000, 7200, 4800, 600]$	$[6 \times n, 6 \times 2500, 12000, 9000, 6000, 1200]$
$[6 \times n, 6 \times 2000, 7200, 4800, 2400, 1200]$	$[6 \times n, 6 \times 2500, 12000, 9000, 6000, 1200]$

Table S3: **MDA architectures for Yeast and Human STRING networks. See the Architecture Example in Fig. S1 for the expansion of the shorthand notation.**

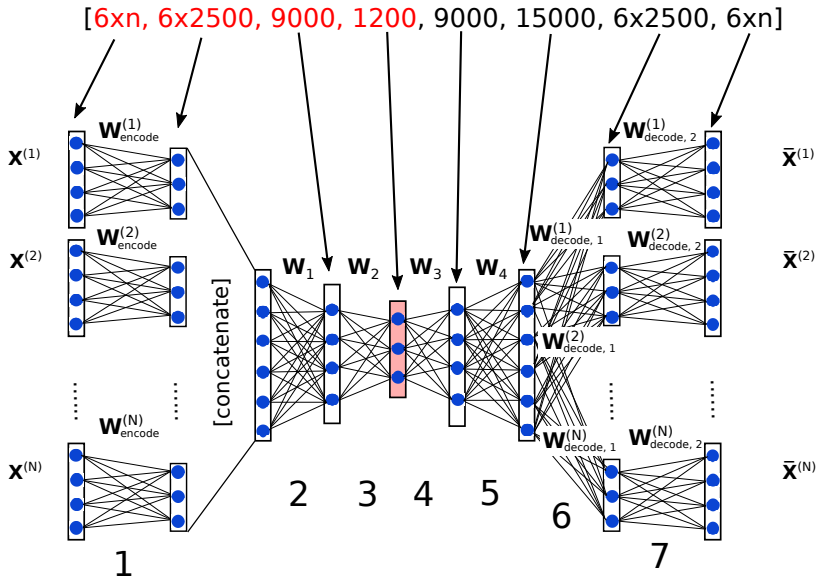

Figure S1: **Architecture Example.**

- **Model hyperparameters**:

We optimize the cost function (Eq. 4 in the main manuscript) using stochastic gradient descent (SGD) `optimizer` with `momentum`. We have tried a variety of activation functions

hyperparameters	Yeast	Human
`batch_size`	128	256
`epochs`	10	20
`lr`	0.2	0.2
`momentum`	0.95	0.95

Table S4: **Model hyperparameters.**

including tanh and ReLU, and have found empirically that the sigmoid activation performed the best. Optimizers such as `Adam` and `Adagrad` seemed not to have much of an effect on final performance of our models, so for simplicity we chose to train our models using stochastic gradient descent with momentum. The model is trained with different batch sizes and number of epochs. To prevent overfitting, we also implemented `early stopping` by training our model on 90% of the data and computing the validation loss of reconstruction on the remaining 10% of the data. We monitor the validation loss at the end of every epoch and stop the training when validation does not decrease for 2 consecutive epochs. The values of hyperparameters are provided in Table S4. The implementation of our method is freely available online [3].

Regarding the SVM step of our method, we used the exact same grid search procedure as in the Mashup paper for choosing the optimal hyperparameters:

- RBF kernel bandwidth: $\gamma \in \{0.001, 0.01, 0.1, 1.0\}$
- regularization parameter of the SVM: $C \in \{0.1, 1.0, 10.0, 100.0\}$

We use this grid for both *Mashup* and *deepNF*. We re-run *Mashup* for both cross-validation and temporal-holdout validation using its MATLAB implementation downloaded from `http://cb.csail.mit.edu/cb/mashup/`.

Similar to *Mashup*, we also run *GeneMANIA* for both cross-validation and temporal-holdout validation using its MATLAB implementation downloaded from: `http://morrislab.med.utoronto.ca/sara/SW`. We use Platt calibration [4] to convert *GeneMANIA*s scores into probability score before evaluation, as suggested in the Mashup paper [5].

4 Study of different MDA architectures: cross-validation and temporal holdout validation performance

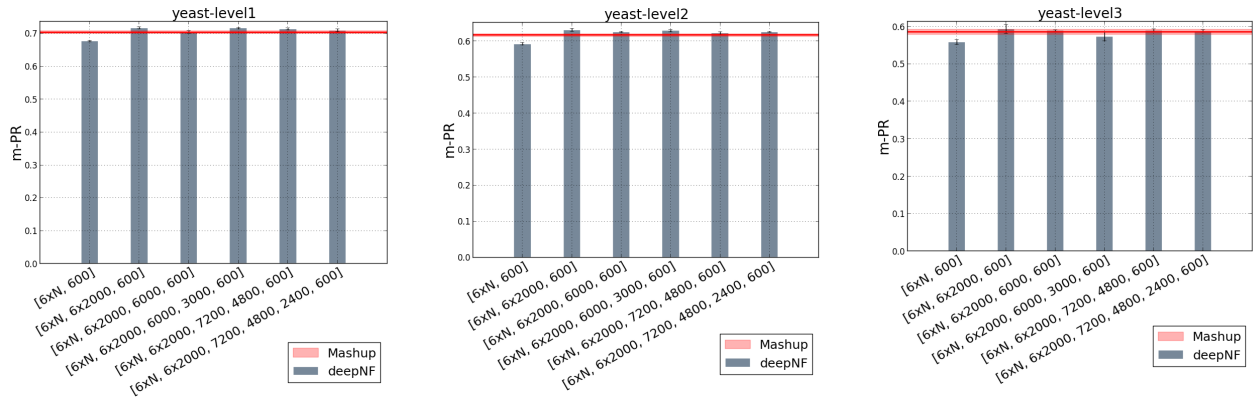

Figure S2: Comparison of *deepNF's* 5-fold **cross validation** performance in **yeast**, measured by macro-averaged AUPR (M-AUPR) and computed on the low-dimensional features extracted from the MDA with different architectures and hidden layer sizes shown on the x-axis. We show the performance on different hierarchical levels of MIPS annotations, i.e., level 1 (left), level 2 (middle) and level 3 (right). Average performance of *Mashup* on the same annotations along with error bars is shown in red. Results are summarized over ten CV trials.

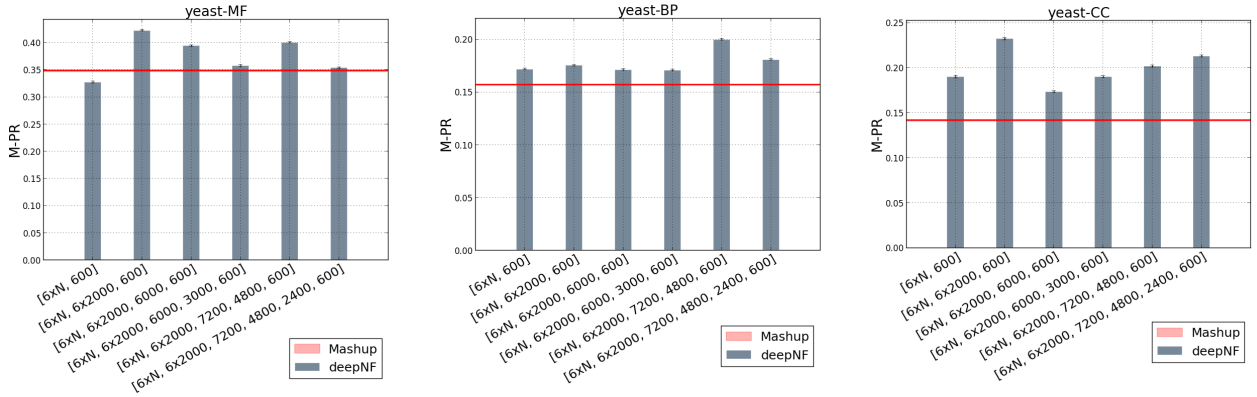

Figure S3: Comparison of *deepNF's* **temporal holdout** performance in **yeast**, measured by macro-averaged AUPR (M-AUPR) and computed on the low-dimensional features extracted from the MDA with different architectures and hidden layer sizes shown on the x-axis. We show the performance on different GO ontologies, i.e., MF (left), BP (middle) and CC (right). The performance of *Mashup* on the same annotations is shown in red.

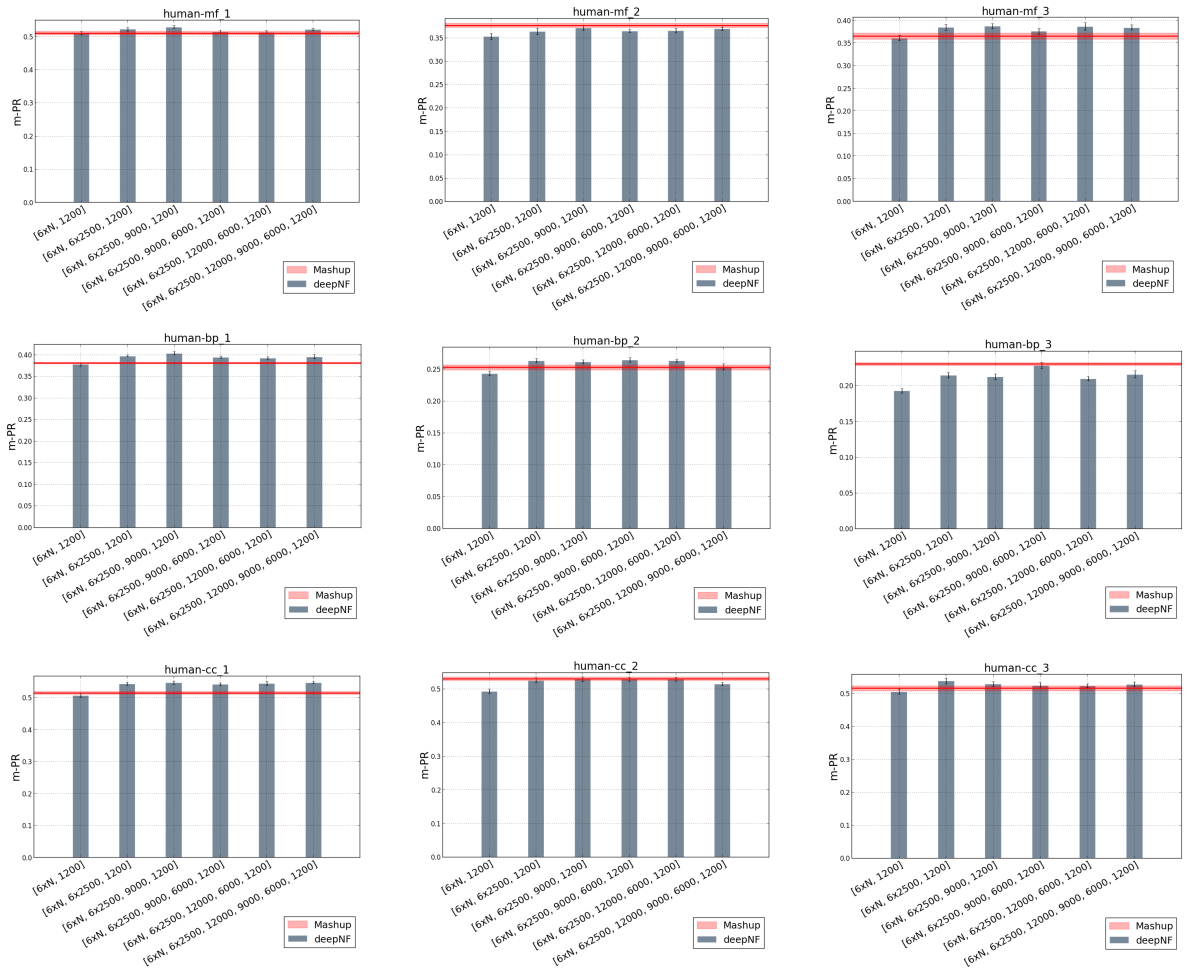

Figure S4: Comparison of *deepNF's* 5-fold **cross validation** performance in **human**, measured by macro-averaged AUPR (M-AUPR) and computed on the low-dimensional features extracted from the MDA with different architectures and hidden layer sizes shown on the x-axis. We show the performance on GO annotations with different levels of frequencies, for every ontology, i.e. MF (top), BP (middle) and CC (bottom). Average performance of *Mashup* on the same annotations along with error bars is shown in red. Results are summarized over ten CV trials.

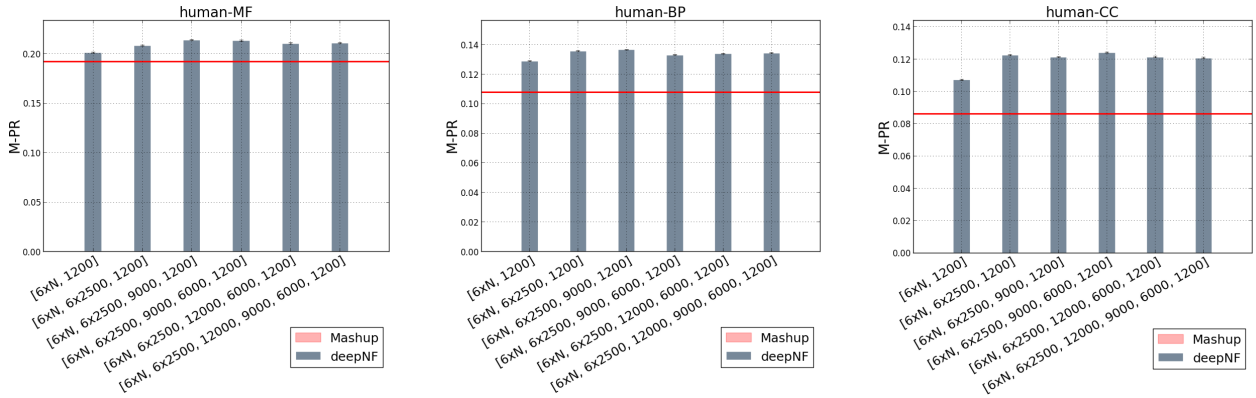

Figure S5: Comparison of *deepNF's* **temporal holdout** performance in **human**, measured by macro-averaged AUPR (M-AUPR) and computed on the low-dimensional features extracted from the MDA with different architectures and hidden layer sizes shown on the x-axis. We show the performance on the three GO branches, i.e., MF (top), BP (middle) and CC (bottom). The performance of *Mashup* on the same annotations is shown in red.

5 Cross-validation performance

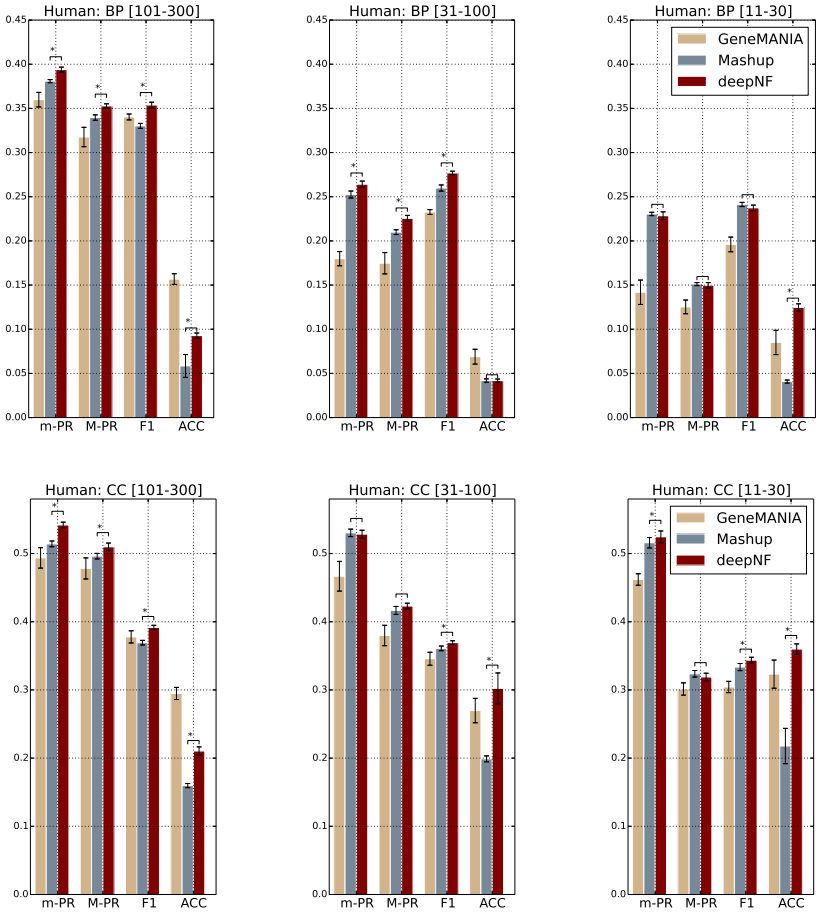

Figure S6: Performance of our method in integrating **Human** STRING networks, with the MDA architecture $[6 \times N, 6 \times 2500, 9000, 1200]$, in 5-fold cross validation in comparison to prediction performance of the state-of-the-art integration method, *Mashup*, and *GeneMANIA*. Performance is measured by the area under the precision-recall curve, summarized over **all GO terms** both under the micro-averaging (m-AUPR) and macro-averaging (M-AUPR) schemes; F1 score and accuracy (ACC). Performance of the methods is shown separately for BP (top) and CC (bottom) ontologies, where each ontology is further divided into three levels: level 1, level 2 and level 3 annotating 101-300, 31-100 and 11-30 proteins respectively. The error bars are computed based on 10 trials and asterisks indicate where the performance of *deepNF* is *significantly* higher than the performance of *Mashup* (rank-sum p-value < 0.01).

Given that the procedure for selecting the training proteins in temporal holdout validation might result in data that happens to give our method an advantage over prior methods, we performed an additional cross-validation analysis on the same set of GO terms used in the temporal holdout validation using the annotations from Uniprot-GOA released in 2017 for both yeast and human proteins. Unlike temporal holdout validation, which uses a fixed set of training proteins, the cross-validation procedure randomly selects proteins in

each trial so that final evaluation of the method is independent of the choice of training proteins. The temporal holdout-oriented cross-validation results for human and yeast are shown in Figure S6 in the Supplementary Material. Even in this way of evaluation, we show that *deepNF* outperforms *Mashup* and *GeneMANIA* in every ontology except in the cellular component branch of yeast. Since we report improvements in performance consistently in cross-validation for both the general set of GO terms (see Figs. 2, 3 in the text) and the terms selected from the temporal holdout procedure (see Fig. S6 in the Supplementary Material), we reduce the possibility of bias in splitting the data in favor of our method in temporal holdout validation. In addition, the temporal holdout validation procedure is a better simulation of the actual use of protein function prediction methods in the wild, as discussed in [6].

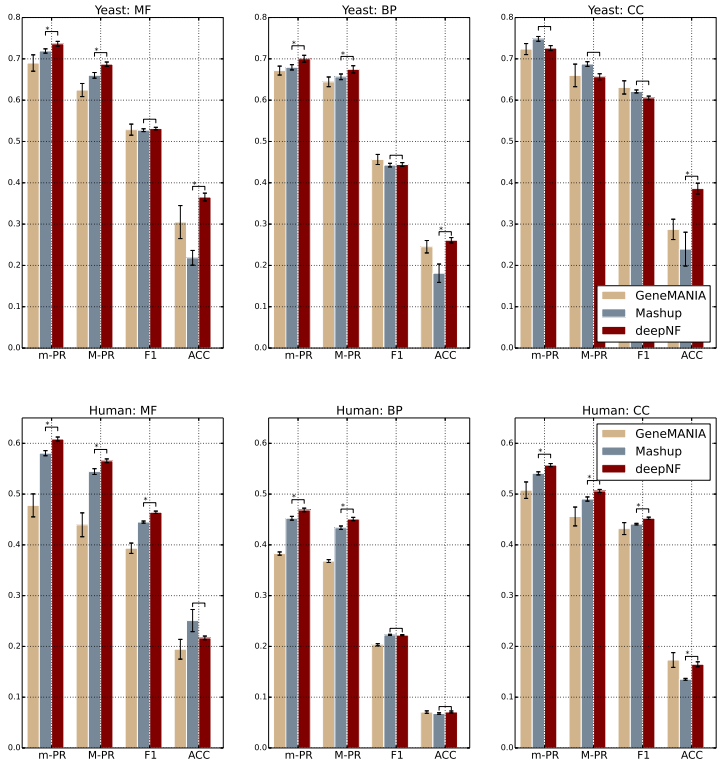

Figure S7: Cross-validation performance of our method in comparison to prediction performance of the state-of-the-art integration method, *Mashup*, as well as *GeneMANIA*. Performance is measured by the area under the precision-recall curve, summarized over **the same set of GO terms used in temporal holdout** in yeast (top) and human (bottom). Performance of the methods is shown separately for MF (left), BP (middle) and CC (right) ontologies. The error bars are computed based on 10 trials and asterisks indicate where the performance of *deepNF* is *significantly* higher than the performance of *Mashup* (rank-sum p-value < 0.01).

6 Temporal holdout validation: performance on individual GO terms

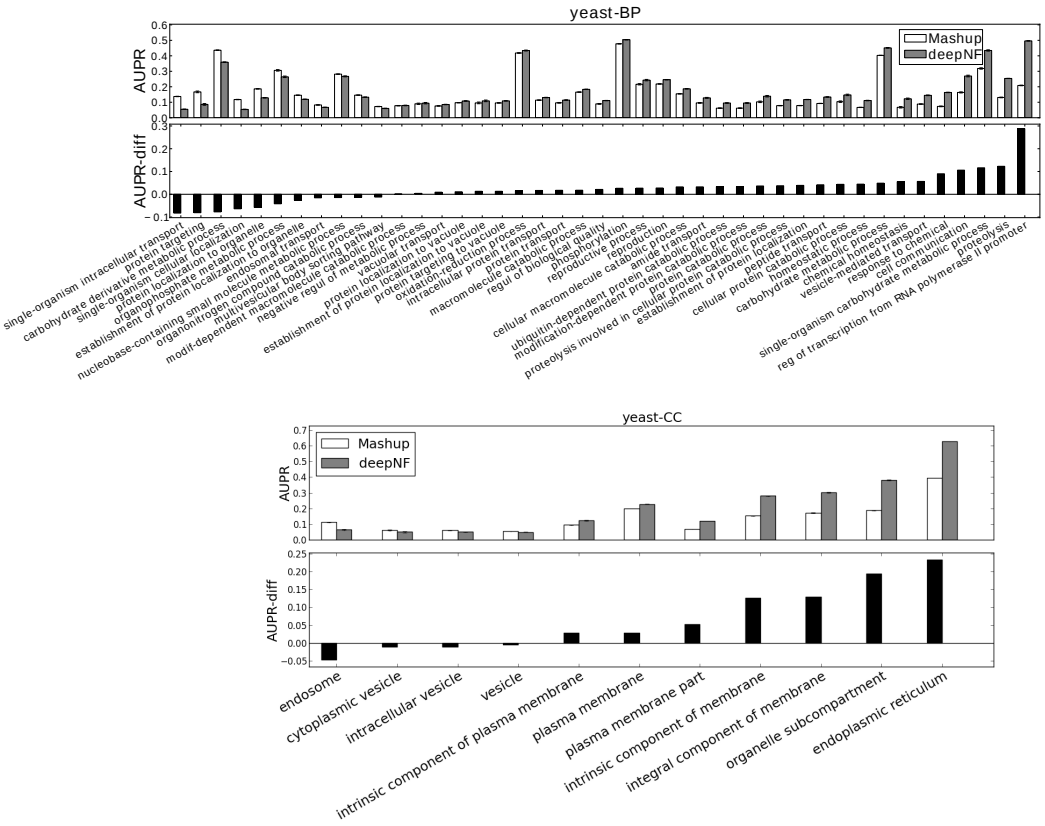

Figure S8: Temporal holdout performance of individual BP-GO terms (top) and CC-GO terms (bottom) in **yeast**, measured by AUPR. The GO term names are shown on the x-axis. We compare the performance of *deepNF* (grey) with *Mashup* (white).

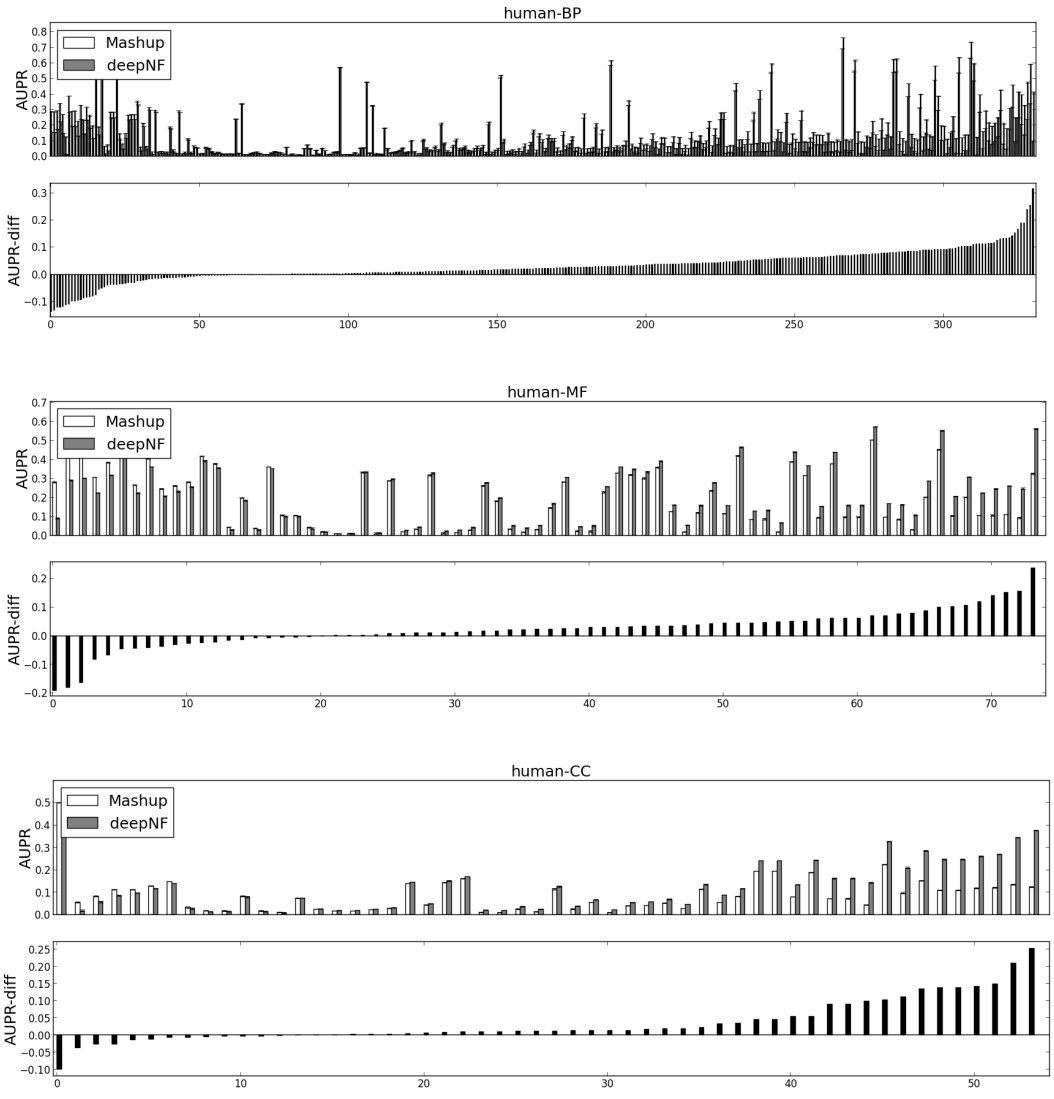

Figure S9: Temporal holdout performance of individual BP-GO (top), MF-GO terms (middle) and CC-GO terms (bottom) in **Human**, measured by AUPR. We compare the performance of *deepNF* (gray) with *Mashup* (white). Due to limited space we could not provide full GO term names on the x-axis. Please refer to the link in [3] for the lists of GO terms for this figure.

7 Baseline methods

We compare the performance of *deepNF* with three baseline methods.

- In the first method (termed *SVM-PPMI*), we train the SVM on the high-dimensional protein vectors (i.e., n-dimensional, where n is the number of nodes in the network) obtained from the PPMI matrices. We used the same grid search procedure for choosing the optimal parameters (γ of the RBF kernel, and the regularization parameter C) as described above (see Sec. 3). In Fig. S10 we show the performance of the *SVM-PPMI* on the PPMI features from a single network (i.e., PPMI matrix of the experimental STRING network) and concatenated features from all $N = 6$ networks (i.e., $N = 6$ concatenated PPMI matrices representing STRING networks).

- We have also implemented a linear-SVM trained on the random projections of the input PPMI vectors (termed *SVM-RND_proj*). We adopt Gaussian random projection method ([7]) to reduce the dimensionality of the input vectors by projecting the original input space (PPMI matrices) on a randomly generated matrix with components drawn from the Gaussian distribution, $N(0, \frac{1}{n_{dim}})$, where $n_{dim} = 600$ (for Yeast) and $n_{dim} = 1200$ (for Human). In Fig. S10 we show the performance of *SVM-RND_proj* on the random projections obtained from a single PPMI matrix (of the experimental STRING network) and on the random projection obtained from the $N = 6$ concatenated PPMI matrices representing the STRING networks.

- To show the difference between our pre-processing step and the *Mashup*s pre-processing step, we run our method on the RWR diffusion states (with restart probability, $p_r = 0.5$) generated by *Mashup*s pre-processing step (*deepNF-mRWR*).

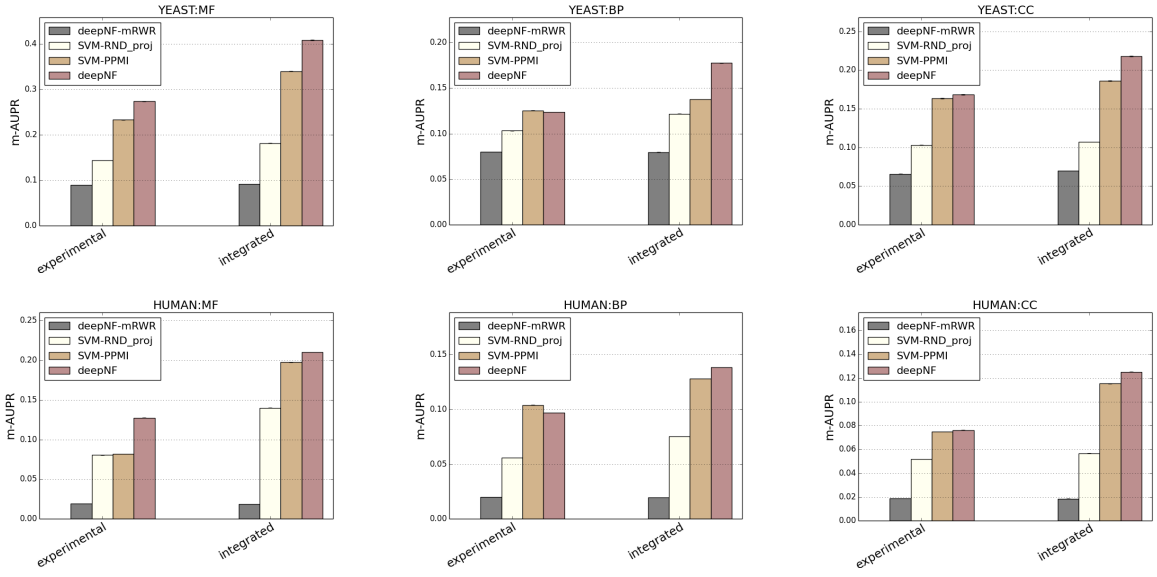

Figure S10: Temporal holdout validation performance of our method, with the MDA architecture $[6 \times n, 6 \times 2000, 600]$ for yeast and $[6 \times n, 6 \times 2500, 9000, 1200]$ for human, in comparison to prediction performance of baseline methods.

References

[1] F. Chollet *et al.*, "Keras: Deep learning for python," 2015. [Online]. Available: https://github.com/fchollet/keras

[2] M. Abadi, A. Agarwal, P. Barham *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2015. [Online]. Available: http://download.tensorflow.org/paper/whitepaper2015.pdf

[3] V. Gligorijević, M. Barot, and R. Bonneau, "deepnf: Deep network fusion for protein function prediction," 2017. [Online]. Available: https://github.com/VGligorijevic/deepNF

[4] H.-T. Lin, C.-J. Lin, and R. C. Weng, "A note on platt's probabilistic outputs for support vector machines," *Machine Learning*, vol. 68, no. 3, pp. 267–276, Oct 2007.

[5] H. Cho, B. Berger, and J. Peng, "Compact Integration of Multi-Network Topology for Functional Analysis of Genes," *Cell Systems*, vol. 3, no. 6, pp. 540–548.e5, 2016.

[6] P. Radivojac *et al.*, "A large-scale evaluation of computational protein function prediction," *Nature Methods*, vol. 10, pp. 221–227, 2013.

[7] N. Ailon and B. Chazelle, "The Fast JohnsonLindenstrauss Transform and Approximate Nearest Neighbors," *SIAM Journal on Computing*, vol. 39, no. 1, pp. 302–322, 2009.