PLOS ONE

RESEARCH ARTICLE

Predicting human protein function with multi-task deep neural networks

Rui Fa[1,2], Domenico Cozzetto[1,2], Cen Wan[1,2], David T. Jones[1,2]*

1 The Francis Crick Institute, London, United Kingdom, **2** Computer Science Department, University College London, London, United Kingdom

* d.t.jones@ucl.ac.uk

OPEN ACCESS

Citation: Fa R, Cozzetto D, Wan C, Jones DT (2018) Predicting human protein function with multi-task deep neural networks. PLoS ONE 13(6): e0198216. https://doi.org/10.1371/journal.pone.0198216

Editor: Lukasz Kurgan, University of Alberta, CANADA

Received: February 1, 2018

Accepted: May 15, 2018

Published: June 11, 2018

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Funding: RF received funding from Elsevier; DC and CW are supported by the UK Biotechnology and Biological Sciences Research Council (references: BB/L020505/1 and BB/L002817/1). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: We have no competing financial, professional or personal interests that

Abstract

Machine learning methods for protein function prediction are urgently needed, especially now that a substantial fraction of known sequences remains unannotated despite the extensive use of functional assignments based on sequence similarity. One major bottleneck supervised learning faces in protein function prediction is the structured, multi-label nature of the problem, because biological roles are represented by lists of terms from hierarchically organised controlled vocabularies such as the Gene Ontology. In this work, we build on recent developments in the area of deep learning and investigate the usefulness of multi-task deep neural networks (MTDNN), which consist of upstream shared layers upon which are stacked in parallel as many independent modules (additional hidden layers with their own output units) as the number of output GO terms (the tasks).

MTDNN learns individual tasks partially using shared representations and partially from task-specific characteristics. When no close homologues with experimentally validated functions can be identified, MTDNN gives more accurate predictions than baseline methods based on annotation frequencies in public databases or homology transfers. More importantly, the results show that MTDNN binary classification accuracy is higher than alternative machine learning-based methods that do not exploit commonalities and differences among prediction tasks. Interestingly, compared with a single-task predictor, the performance improvement is not linearly correlated with the number of tasks in MTDNN, but medium size models provide more improvement in our case. One of advantages of MTDNN is that given a set of features, there is no requirement for MTDNN to have a bootstrap feature selection procedure as what traditional machine learning algorithms do. Overall, the results indicate that the proposed MTDNN algorithm improves the performance of protein function prediction. On the other hand, there is still large room for deep learning techniques to further enhance prediction ability.

Background

The biological roles of the vast majority of known amino acid sequences remain partly or completely unknown: the UniProtKB database [1] currently stores more than 60 million

might have influenced the performance or presentation of the work described in this manuscript. The fact that one of our founders, Elsevier, is a commercial company does not alter our adherence to PLOS ONE policies on sharing data and materials.

sequences, but UniProt-GOA [2] lists only about 600 thousand experimentally-supported functional annotations in the form of Gene Ontology (GO) terms [3]. This information is far from uniformly spread across protein sequences, so elucidating their molecular activities, their whereabouts, their biological partners, and the environmental conditions enabling them has been increasingly dependent on computational methods that mostly perform annotation transfers from sequence [4]. Because naive or iterative application of these methods can generate uncontrolled error propagation in databases [5], curators nowadays rely on complementing transfers from orthologous proteins and from domain family assignments with mappings between controlled vocabularies and GO [2]. Notwithstanding, a substantial fraction of deposited sequences still has no annotations at all, many more lack information for at least one GO domain, and the hypotheses generated are often too generic to suggest a limited number of specific validation assays.

Machine learning represents an attractive avenue to help fill in this gap, by modelling the relationship between protein function and the features extracted from individual or multiple biological data sources. When informative patterns can be detected, this approach can overcome the limitations of homology-based transfers due to the lack of similar sequences with known function, or to misleading alignment results. Many research groups have tested this hypothesis with success by examining heterogeneous data sources such as protein sequences [6–9], genomic information [10,11], gene expression profiles [12], and functional association networks [13], using neural networks (NN) [14], support vector machines (SVM) [7–9] and random forests [15].

Further efforts are going into integrative approaches, that try to leverage the strengths of individual methods and data types and to lessen the effects of their intrinsic limitations [16–19]. For instance, protein sequence and structure analysis can predict molecular function GO terms much better than biological process terms; in turn, the latter are more confidently inferred from genome-wide datasets. The evaluation results of the community-wide Critical Assessment of Function Annotation (CAFA) experiments confirmed these observations, but also highlighted that predicting protein function accurately still remains an open problem [20,21].

One of the major challenges supervised learning methods face in protein function prediction is the structured and multi-label nature of the problem, because the biological roles are described by sets of terms from the hierarchically organised GO domains. Given the complexity of this challenge and the tools at hand, most previous studies benefitted from handling many more tractable binary classification tasks [7–9] - one for each GO term. So far, very few groups have tried to build one classifier able to predict all relevant labels at once [14], but recent developments in the field of machine learning now make this approach feasible. Deep learning is a fast-evolving area of research, which tries to address regression or classification problems by extracting informative internal representations of the input data (aka feature representations) at different levels of abstraction. This is usually achieved with artificial deep neural networks (DNN), which include multiple hidden layers with hundreds of units each aimed at capturing high-level feature representations. The concept of DNN appeared a few decades ago, but remained impractical until Hinton and colleagues showed that layer-wise pretraining techniques allow deep networks to learn better feature representations and leading to improved classification performance[22]. Its growing power and popularity depend on several theoretical and technical advances that speed up training and reduce the risk of overfitting. Rectified activation functions force the model to learn sparse representations and ease vanishing-gradient issues that typically affect networks with many layers and sigmoid activation functions [23]. The dropout regularization technique—which consists in randomly omitting a different subset of the model parameters during training—greatly helps perform model

averaging and reduce the risk of overfitting to known observations [24]. Finally, batch-normalization transforms each neuron input values from a randomly chosen subset of the training data (aka mini-batch) so that their distributions do not change dramatically during training [25]. The increasing availability of general-purpose graphical processing units (GPUs) and application programming interfaces (APIs) have also made a substantial contribution to the widespread application of deep learning techniques within both academia and industry.

Deep learning has been applied to a wide range of problems in sequence and -omics data analysis, biomedical imaging and biomedical signal processing [26–29]. Multi-task deep neural networks are a particular type of architectures which consist of initial shared layers followed by as many independent modules (made up of individual output neurons possibly downstream of additional hidden layers) as the number of target labels (aka tasks). This design is meant to exploit commonalities and differences among tasks through the combination of shared and task-specific representations, and thus has the potential to compensate for the limited number of observations available for some tasks. This modelling approach has been previously applied to virtual screening [30], toxicity prediction [31] and prediction of protein biophysical features including secondary structure, solvent accessibility, transmembrane segments and signal peptides [32].

In this work, we investigate the usefulness of multi-task DNN (MTDNN) to tackle protein function prediction, an area which is expected to benefit from learning the dependencies among functional classes. We build a two-stage MTDNN structure, in which a set of feedforward layers shared by all tasks are in the first stage and as many task-specific feedforward layers as the number of tasks are parallel stacked upon the shared layers. This structure leverages both the shared representations of all tasks and specific characteristics of individual tasks. The effectiveness of the proposed approach is gauged against a naive multi-label DNN (MLDNN) —a feedforward structure with as many output units as the number of GO terms and several shared hidden layers—as well as three baseline methods. The experimental results show that MTDNN achieve higher F_1 scores than the other methods tested; interestingly, the performance improvement over a single-task predictor is not linearly correlated with the number of tasks in the MTDNN model: medium size models with 20–50 GO terms appear to be more effective in our case. Another advantage is that there is no requirement for MTDNN to have a bootstrap feature selection procedure when given a set of features, in contrast to many traditional machine learning methods. This flexibility may lead to our future work to improve the performance of protein function prediction further by connecting MTDNN with many different biological data sources, such as graph embedding features from protein-protein interaction networks, or gene expression data *etc*.

Methods

Data collection

The protein sequences and feature data, the GO term vocabulary and the functional annotations used in this study are identical to those used to develop FFPred3 [9]. Annotations for human proteins were obtained from the Gene Ontology Annotation (GOA) [2] database released on 2015-02-02. The Gene Ontology (GO) OBO file released on 2015-02-03 was used for term definitions and semantic relations [33]. All GO terms in the biological process (BP), molecular function (MF) or cellular component (CC) domain with at least 150 annotations and 500 putative negative examples were selected. The negative examples of one GO term are the proteins that are (i) not labelled with the term under consideration, its descendants and its ancestors; (ii) nonetheless bear at least 2 MF terms and 2 BP terms with evidence code other than IC, NAS, TAS, and IEA. In total, 868 GO terms across the three domains were selected.

Amino acid sequences were retrieved from UniProtKB version 2015_03 [1], and encoded through 258 features covering 14 different functional and structural aspects, including protein secondary structure, intrinsically disordered regions, transmembrane segments, signal peptides, post-translational modification sites, coiled-coil regions, and other sequence motifs [9]. These biophysical attributes can be easily calculated or predicted from amino acid sequences or their evolutionary profiles as reported before [8].

For benchmarking purposes, the set of human proteins that received GO term assignments supported by evidence code EXP, IDA, IMP, IGI, IEP, TAS or IC exclusively between 2015–02 and 2017–02 was collated from GOA database. Annotations to the term "*protein binding*" (GO:0005515) were discarded because they convey limited functional information. This test set was made up of 1754 annotations for 707 proteins in total—349 MF annotations for 236 proteins, 556 BP annotations for 259 proteins, and 849 CC annotations for 492 proteins. Considering backpropagation, we have 2196 MF annotations covering 527 GO terms, 7353 BP annotations covering 1712 GO terms, and 4906 CC annotations covering 256 GO terms.

Multi-task deep neural networks (MTDNN)

Overview. MTDNN implements a multi-task architecture, with a set of feedforward layers shared by all tasks, upon which as many task-specific feedforward subnets as GO terms under investigation are parallel stacked—see Fig 1(A). This layout is meant to help the network learn individual tasks partially using a shared representation and the rest from task-specific characteristics. The network architecture is implemented using *Lasagne* and *Theano* [34]; each hidden layer is fully connected to the previous one, has batch normalised input values and has dropout applied in the course of training. The neurons in the hidden layers are activated by rectified linear functions, while output units make use of softmax functions with two outputs. The confidence score of each GO term corresponds to the value associated with the positive class from the relevant neuron.

Branching. We experienced difficulties in training one MTDNN network for each GO domain, because the high demands for memory exceeded the amount available on the GPUs. Therefore, we grouped the GO terms based on the "*is_a*" relationships in GO, and trained one separate model for each *branch*—one of the subgraphs rooted at each level-1 nodes—to predict

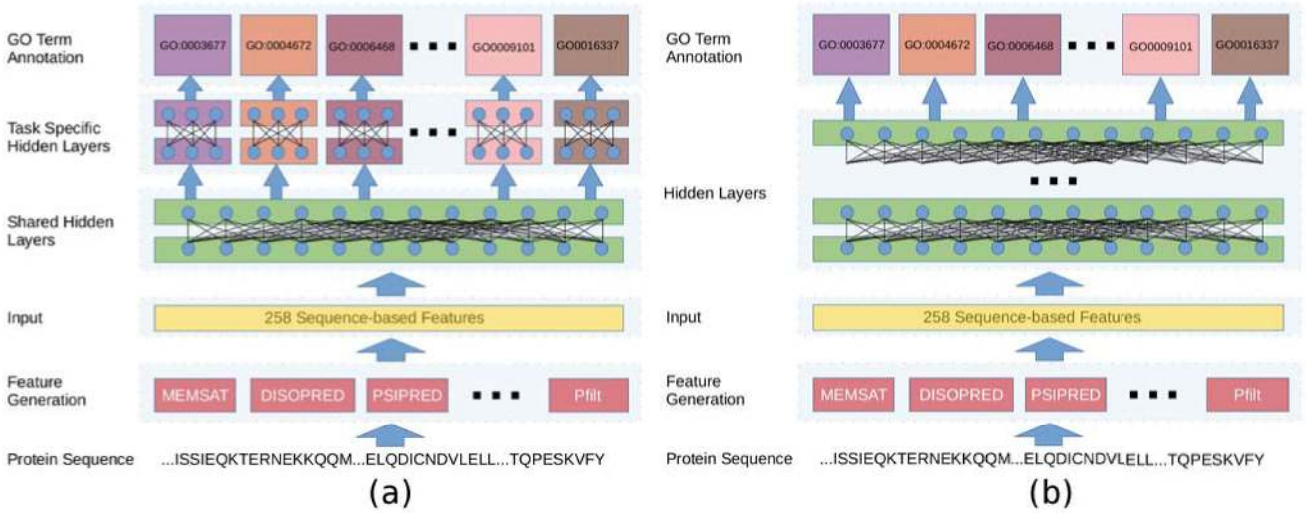

Fig 1. Schematic diagrams of MTDNN (a), and MLDNN (b).

https://doi.org/10.1371/journal.pone.0198216.g001

the subset of its descendants included in our vocabulary. For example, our vocabulary lists 17 descendants of the level-1 term *immune system process* (GO:0002376), and the corresponding branch has 18 tasks, inclusive of itself. This procedure led to 38 branches in total: 18 within the biological process (BP) domain, 11 for molecular function (MF), and 9 for cellular component (CC). The summary of all branches is provided in supplementary material S1 Table. The number of GO terms in each branch varies a lot branch by branch. The largest branch is *biological regulation* (GO:0065007) with 218 output units for as many GO terms, while the smallest ones have only 2 output nodes. Given the semantic relationships in GO, some terms were included in different branches: *cell morphogenesis involved in neuron differentiation* (GO:0048667) appears in four branches, namely *cellular process* (GO:0009987), *single-organism process* (GO:0044699), *cellular component organization or biogenesis* (GO:0071840), *developmental process* (GO:0032502) for example. There are 290 out of 606 GO terms in BP, 71 out of 104 GO terms in CC, and 8 out of 158 GO terms in MF appearing in multiple branches, which generate as many predictions. For such GO terms, MTDNN returns one combined result by calculating the average value from the different branches.

Training and optimisation. All tasks in MTDNN were trained individually while the shared layers were updated for all tasks. To this end, the protein sequences were initially clustered at 50% sequence identity with kClust [35], and for each task the resulting clusters were assigned to either the training or test set. Approximately 70–80% were used for training and the remaining 20–30% for testing—making sure there are at least 35 positive test examples for each GO term.

One of the most severe issues of protein function prediction using deep neural networks is the imbalance training problem, i.e., the numbers of positive examples in some GO terms are much fewer than the numbers of their negative examples. To deal with the issue, we investigated two strategies. The first one is that we aggregated weights of classes into the loss function to impose an additional cost on the model for making mistakes on the minority class during training. In this strategy, each batch composes of examples randomly sampled from the training set. The second strategy is that we oversampled the minority class, kept each batch with balanced positive and negative examples during the training, and in the prediction phase, penalized the minority class with a weight. The weight is defined as the ratio of the number of negatives to the number of positives in the training set. To determine the best strategy to deal with imbalanced training sets, we did a quick experiment only on MF terms because MF has a relatively small group of GO terms. The F_{max} performance of the strategy one is *0.244* and its F_1 score at a threshold equal to 0.5 is 0.219, while the second strategy has obtained the F_{max} score *0.311* and *0.292* for its F_1 score at threshold equal to 0.5. The results indicate that the strategy using balanced batch training and penalized inferring produced better results. Therefore, the results reported in the Results section were obtained using the balanced training strategy.

The training procedure consisted of 100 epochs and was divided into two stages: during the first 50 epochs both the shared and the specific layers were updated, while only the specific layers were modified during the last 50 epochs. This measure was aimed at reducing the degrees of freedom when task-specific layers were trained and at exploiting general feature representations of biological function to learn finer details. During each epoch, the GO terms were trained consecutively, and their order was randomly shuffled every time to minimise the risk of biasing the shared parameters towards more recently observed training examples.

Each MTDNN branch was independently optimised by searching a set of hyperparameters, including the number of shared layers, the number of hidden units in each shared layer, the number of specific layers, the number of hidden units in each shared layer, dropout rate, and learning rate—see supplementary material S2 Table for more details. We employed the HYPEROPT package [36] to search the hyperparameter space randomly in 100 trials. The final

models are taken from the parameters that maximize the average F_1 score with threshold equal to 0.5 on the holdout test set.

Multi-label deep neural networks (MLDNN)

MLDNN implements a straightforward solution to multi-label problems and consists of a feed-forward multi-layer architecture with 258 input nodes (one for each sequence-derived feature), followed by several fully connected layers that are shared by one output layer—see Fig 1(B). Each hidden layer has batch-normalized inputs combined through rectified linear units and is subject to dropout during training. The output layer is fully connected to the previous one, and consists of as many output neurons as the GO terms in the selected vocabulary activated by sigmoid functions, the output of which is returned as a confidence score.

To train the MLDNN method, the protein sequences were first clustered at 50% sequence identity with kClust [35], and for each GO domain such clusters were assigned to the training or test set. Approximately 80% of the data were used for training and the remaining 20% for testing.

To optimise the MLDNN method, the set of hyperparameters in supplementary material S3 Table was randomly sampled 100 times with the HYPEROPT package. For each trial, 100 epochs of training were carried out. The final models are taken from the parameters that maximize the average F_1 score with threshold equal to 0.5 on the holdout test set.

Single-task deep neural networks (STDNN)

STDNN is a traditional divide-and-conquer solution of the multi-label learning problem, which employs a single fully connected feedforward deep neural network for an individual GO term, therefore, there are totally 868 binary DNNs optimised and trained independently. Like multi-task and multi-label implementations, the protein sequences were initially clustered at 50% sequence identity with kClust [35], and for each task the resulting clusters were assigned to either the training or test set. Approximately 70–80% were used for training and the remaining 20–30% for testing–depending on the number of positive annotations in each GO term and making sure there are at least 35 positive test examples for each GO term.

To optimise the STDNN, the set of hyperparameters in S3 Table was randomly sampled 100 times with the HYPEROPT package. For each trial, 100 epochs of training were carried out. The final models are taken from the parameters that maximize the average F_1 score with threshold equal to 0.5 on the holdout test set.

Baseline methods

FFPred3 [9] was used to predict GO terms for the sequences in the benchmark set starting from the same 258 input features fed to both the MTDNN and MLDNN. Unlike the multi-task and multi-label deep learning approaches, however, FFPred3 examines the input values through a library of 868 Support Vector Machines independently trained to classify as many functional categories.

Naive predictions were generated based on the frequency of the GO term annotations for human sequences in UniProt-GOA released on 2015-02-02. The initial counts were obtained for all GO terms supported by the evidence codes EXP, IDA, IPI, IMP, IGI, IEP, IC and TAS. The data were then propagated following "*is a*" links in the GO released on 2015-02-03, and scaled between 0 and 1 for each domain separately, by dividing the final counts by the number of occurrences of the root node.

BLAST predictions were obtained by collecting all BLAST hits in the UniRef90 sequence database released on 2015–02 with an E-value greater than 1e-03. Then the annotations in UniProtKB released on 2015-02-03 supported by evidence codes EXP, IDA, IPI, IMP, IGI,

IEP, IC and TAS were transferred to the target sequences. The confidence scores of GO terms were calculated by dividing the local alignment sequence identity by 100. When multiple BLAST hits were annotated with the same GO term, the highest score was retained.

Performance evaluation

Prediction accuracy was measured by protein-centric precision-recall analysis separately for each GO domain. For each protein $\boldsymbol{x}$ in the benchmark set and decision threshold t, the set of predicted GO terms $\boldsymbol{G_{x,t}}$ was built by collecting all terms with confidence scores greater than or equal to t, and their ancestors in GO linked by "*is_a*" relationships and different from the root. The precision $p_{x,t}$ and recall $r_{x,t}$ can be respectively written as

$$p_{x,t} = \frac{\mathrm{TP}_{x,t}}{\mathrm{TP}_{x,t} + \mathrm{FP}_{x.t}} \tag{1}$$

$$r_{x,t} = \frac{\mathrm{TP}_{x,t}}{\mathrm{TP}_{x,t} + \mathrm{FN}_{x.t}} \tag{2}$$

where $\mathrm{TP}_{x,t}$ is the number of true positives, $\mathrm{FP}_{x,t}$ is the number of false positives, and $\mathrm{FN}_{x,t}$ is the number of false negatives for the benchmark protein x at threshold t. Then the average across the test set are taken as

$$p_t = \frac{1}{n_t}\sum_x p_{x,t} \tag{3}$$

$$r_t = \frac{1}{n_0}\sum_x r_{x,t} \tag{4}$$

where n_t is the number of target proteins with at least one prediction scoring above threshold t, and n_0 is the number of target proteins in the GO domain in the benchmark set. Therefore, the average F_1 for the threshold t and $F_{\max}$ were calculated as

$$F_1(t) = 2 \cdot \frac{p_t r_t}{p_t + r_t} \tag{5}$$

$$F_{\max} = \max_t F_1(t) \tag{6}$$

We also employed term-centric evaluation to measure F_1 scores for individual GO terms covered by the benchmark set.

Results and discussions

We are interested in knowing whether, how and why multitask deep neural networks improve the performance of protein function prediction. In this section, we will show both holdout and benchmark evaluation results of our optimised MTDNN models, and discuss the lesson we learned from our experiments.

Optimised models

As detailed in Methods section, the hyperparameter optimisation procedure was carried out by using the Python HYPEROPT package, which randomly samples from pre-defined hyper-parameter space. Each branch has a set of optimised hyperparameters including the network architectures like the depths of shared layers and specific layers, the numbers of hidden units

in each layer *etc*., and the other non-architecture hyperparameters like the learning rates and regularisation options. The question that interests us most is what network architecture each branch chose in the optimisation procedure. We show the depths of both the shared layers and the specific layers for all branches of three domains as stacked bar charts in Fig 2. The branches are presented in a descending order from the top to the bottom. The red number on the right-hand side of each bar is the number of GO terms in the branch. We notice that the larger branch tends to choose deeper models in general, however, we are unable to observe any correlation between the depths and the branch sizes.

We also show the summary of the numbers of hidden units in each layer of all optimised models as pie charts in Fig 3. Less than half branches chose three shared layers and 5% models have only one shared hidden layer. 47% of the models chose 800 hidden units in their first shared layer and 79% chose 500 hidden units in their second shared layer. 79% of the models chose 500 hidden units in their first specific layers and the same percentage of models chose 300 hidden units in their second specific layers. This result tells us that larger and deeper models are favourable in many branches according to our experiments.

Holdout set evaluation

We only compare the performance of MTDNN with FFPred rather than MLDNN in the holdout evaluation as a consequence of the different architectures of the two predictors. MLDNN

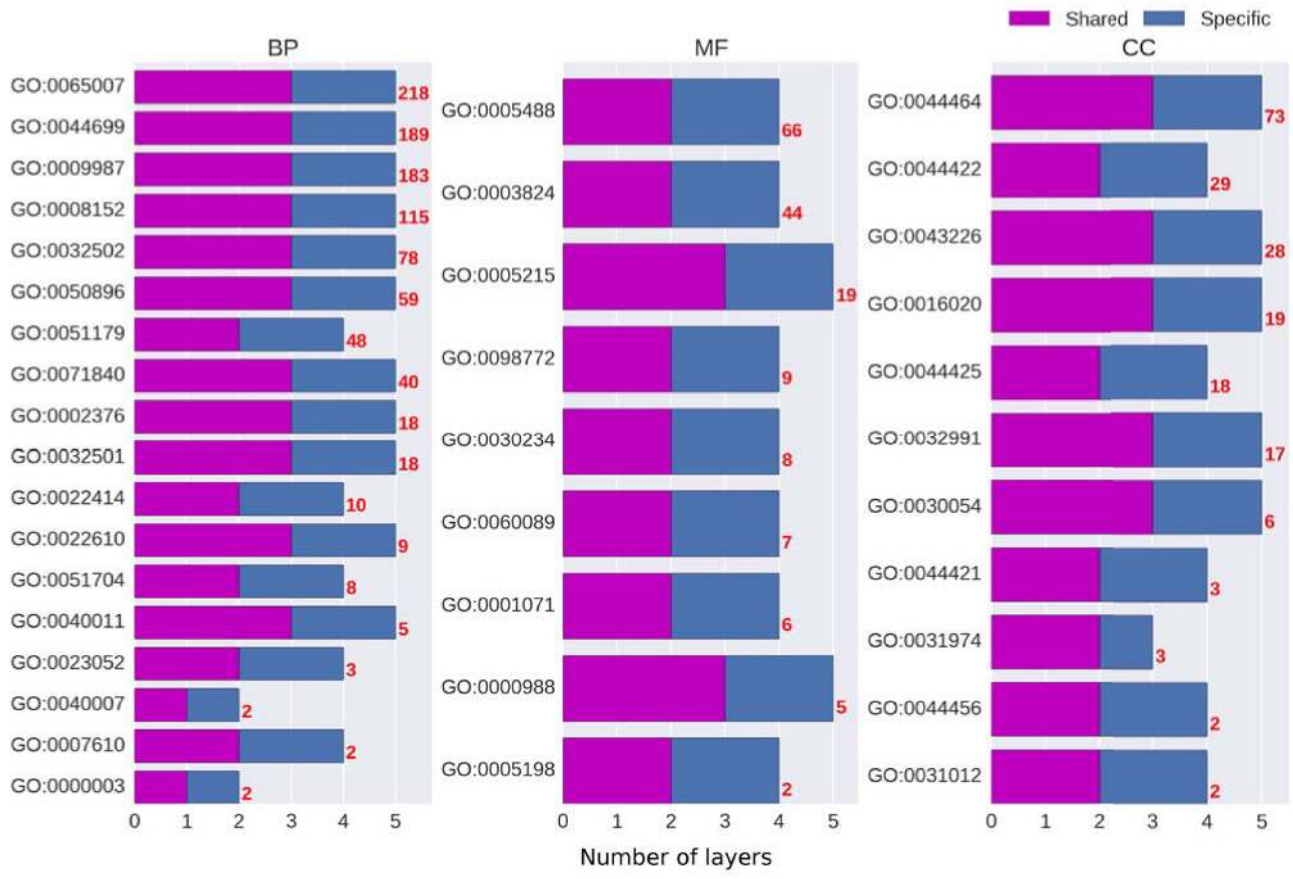

Fig 2. The numbers of layers of the optimised MTDNN models in BP, MF and CC domains. The red numbers on the right-hand side of bars represent the numbers of GO terms in individual branches.

https://doi.org/10.1371/journal.pone.0198216.g002

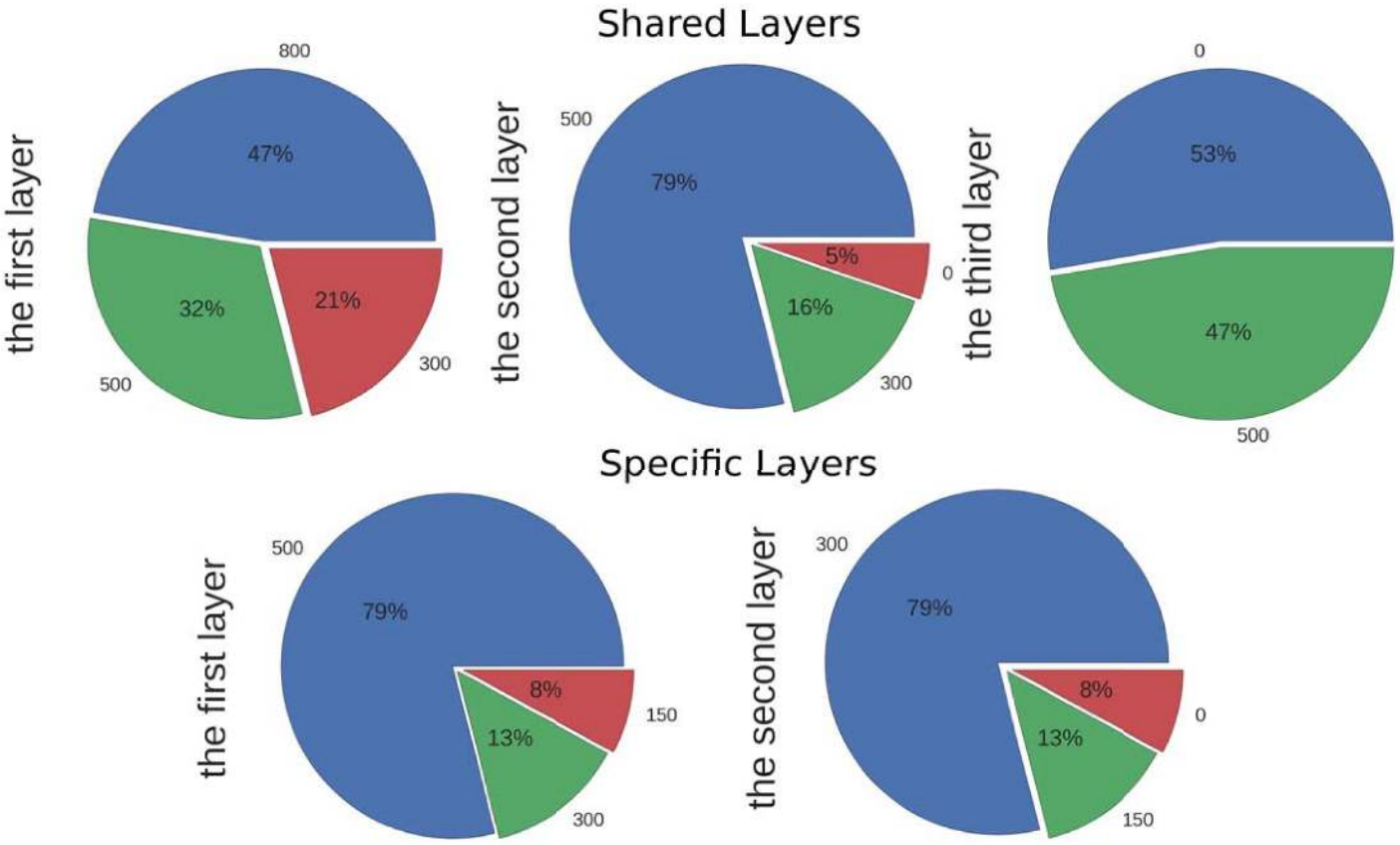

Fig 3. The summary of the numbers of hidden units in individual shared and specific layers of the optimised MTDNN models in pie charts. The upper row presents the choices of numbers of hidden units in the three shared layers; the lower row presents the choices of numbers of hidden units in the two specific layers. The colours represent numbers of hidden units which are shown around the Pie Charts.

https://doi.org/10.1371/journal.pone.0198216.g003

is made up of one shared multi-layer feedforward neural network followed by an output layer with N outputs, where N is the number of GO terms considered. In this case each protein and its known annotations can only be used once, and the make-up of the training, validation and holdout sets depends only on the sequence clustering results. MTDNN includes shared upstream layers to learn general latent representations of the data, while different classification patterns are modelled through the downstream GO term-specific layers. In this case, different GO terms will have different training, validation and holdout sets because of the different protein sets they annotate. We evaluated term-centric F_1 scores for both MTDNN and FFPred and considered the average F_1 score differences between MTDNN and FFPred. In particular, we define term-centric F_1 score the difference as:

$$\Delta F_1 = F_1^{\text{MTDNN}} - F_1^{\text{FFPred}}$$

In Fig 4, we show the average F_1 score differences for two groups of GO terms–all terms in our vocabulary and specific terms—in BP, MF, and CC domains. The definition of specific terms in our case are those terms in our vocabulary which either have not child term at all or have children terms but they are not in our vocabulary. There are mainly two observations from this figure: firstly, generally speaking, MTDNN has better performance than FFPred, especially a considerable improvement in BP; secondly, though MTDNN overall improves the performance, specific terms improved less than general terms.

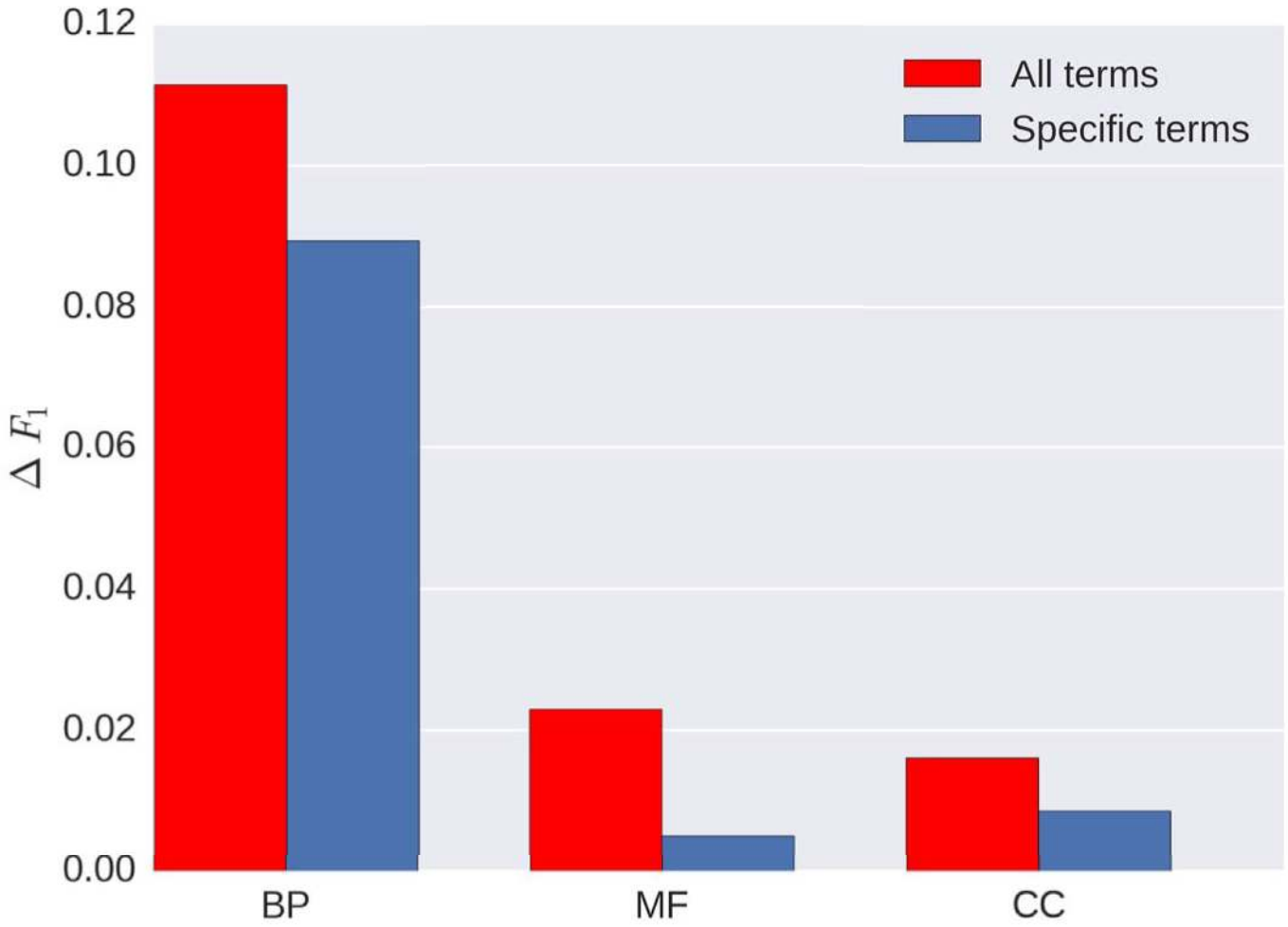

Fig 4. Average F_1 score differences between MTDNN and FFPred in BP, MF and CC domains. The red bars represent the performance of all GO terms in our vocabulary and the blue bars represent the performance of specific terms.

https://doi.org/10.1371/journal.pone.0198216.g004

Another question we should like to address is whether larger branches improve more than smaller branches, i.e. a model with more tasks gains more improvement from using MTDNN than a model with fewer tasks. This would be expected if MTDNN can effectively exploit both the shared and task-specific representations that are learnt from the different training sets. To address this question, we show the average F_1 score differences for individual branches in Fig 5. The branches are ordered ascendingly from the left to the right in terms of their sizes and the red numbers on the top of bars are the numbers of GO terms in individual branches. In general, on one hand, small branches perform poorly, on the other hand, it does not seem to be true that the larger the branch is, the better it performs. There is no clear pattern that the performance improvement correlates with the size of the branch overall. In BP, there indeed is a weak pattern that branches with larger size tend to perform better, but there is no such pattern in MF and CC. The Spearman's rank correlation coefficients between the branch size and performance improvement for three GO domains are shown in supplementary material S4 Table. We believe that it is a complex question combining other factors like the numbers of positive and negative examples in each GO term of each branch. The performance measures of all branches for the holdout set evaluation are shown in S5 Table.

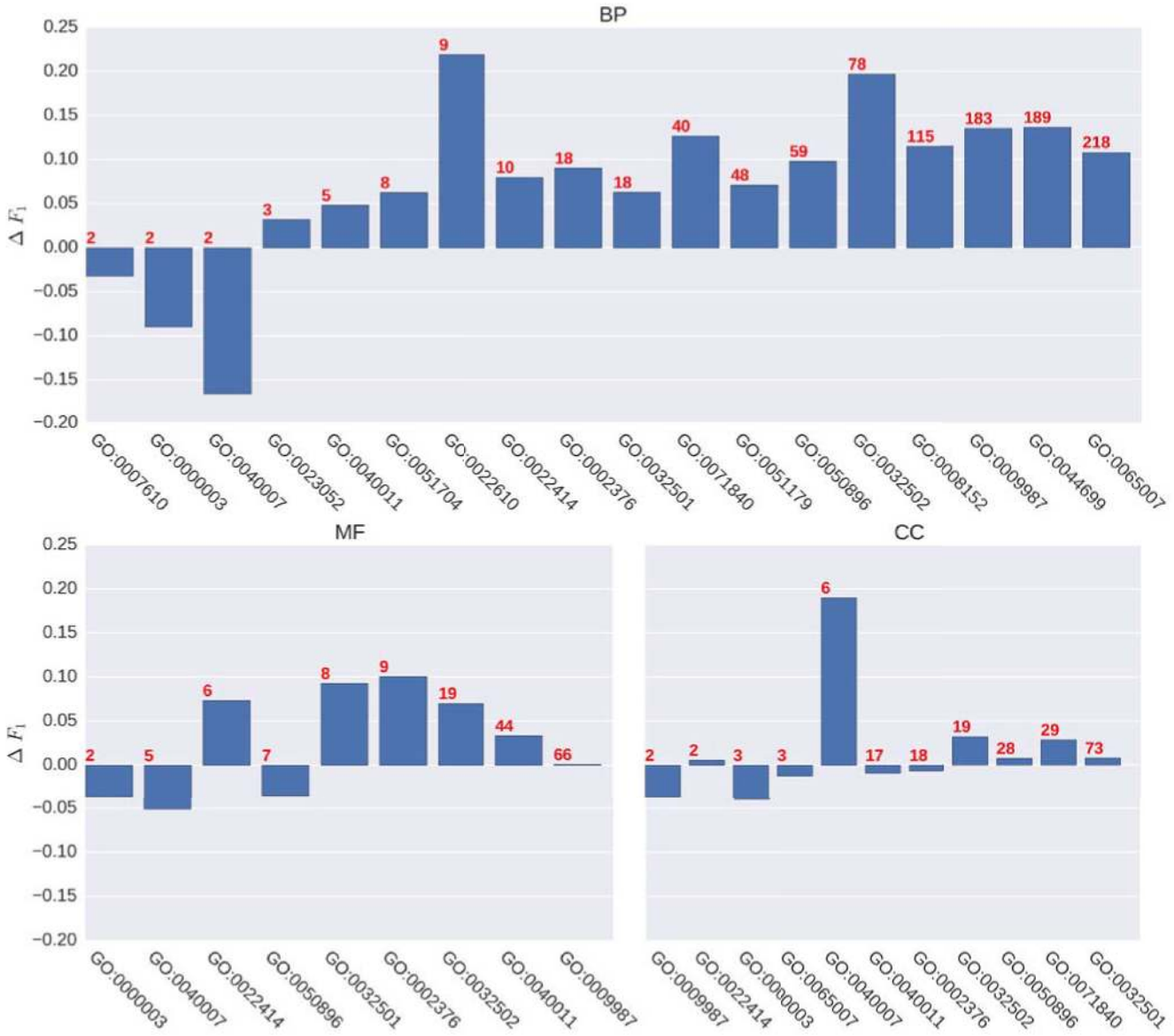

Fig 5. Average F_1 score difference between MTDNN and FFPred for all branches in BP, MF and CC domains. The red numbers on the top of bars represent the numbers of GO terms in individual branches.

https://doi.org/10.1371/journal.pone.0198216.g005

Benchmark set evaluation

Here, one important question we need to address is, whether the MTDNN method is better than other methods in the benchmark set. To answer this question, we firstly evaluated the F_{max} performance of all compared methods following the practice in CAFA [20,21]. We report the results of five methods at the decision thresholds that maximize the F_1 scores for each GO domain using the benchmark set in Table 1. The BLAST and naive methods are the two providing the worst F_{max} performance. Note that naive method has a good F_{max} performance in CC only because of the fact that the majority of proteins locate in the cytoplasm is biased towards the naive frequency counting method. More interestingly, the results indicate that the F_{max} performance of MTDNN is better than the other methods in BP and CC, but worse than

Table 1. F_{max} performance comparison between MTDNN and other prediction methods. For the five methods, the table reports F_{max} and the threshold values that produce the F_{max} in three GO domains, namely BP, MF, and CC. The best F_{max} scores in respective domains are marked in bold font.

Methods	BP		MF		CC	
	Threshold	F_{max}	*Threshold*	F_{max}	*Threshold*	F_{max}
MTDNN	0.52	**0.298**	0.58	0.311	0.54	**0.484**
MLDNN	0.26	0.287	0.30	0.343	0.23	0.449
STDNN	0.99	0.288	0.99	0.338	0.95	0.396
FFPred	0.80	0.293	0.85	**0.355**	0.86	0.477
BLAST	0.25	0.106	0.22	0.126	0.21	0.212
Naive	0.14	0.252	0.23	0.266	0.45	0.474

https://doi.org/10.1371/journal.pone.0198216.t001

FFPred, MLDNN and STDNN in MF. We notice that the thresholds at which other methods produce the F_{max} scores are farther away from 0.5 than MTDNN. Although F_{max} is commonly used to evaluate classification algorithms, it arguably is a paradox because in the reality the threshold to produce the maximum F_1 score would never be known without knowing true labels of all predictions.

A more realistic evaluation is based on standard measure of binary classification accuracy–i.e. interpreting the scores as probability estimates and therefore setting the decision threshold equal to 0.5. The results under these evaluation settings in Table 2. Three metrics, namely the average precision, average recall, and average F_1 scores are presented for the three GO domains. In terms of the F_1 score, the best performing method out of five methods is MTDNN. Its F_1 scores are better than other methods across all three domains. The results reveal that naive method and BLAST are still the two poorest methods. The results also show that FFPred and STDNN have the highest recall scores in all three domains, but with lower precision scores; on the contrary, MLDNN offers higher precision scores, but lower recall scores, in turn, lower F_1 scores. This observation reveals that FFPred and STDNN made more predictions which on the one hand increases true positives, but on the other hand also increases false positives; while MLDNN makes fewer mistakes by reducing the number of predictions. MTDNN provides a balance between precision and recall keeping higher F_1 in all three domains, i.e. MTDNN is able to reach a trade-off between predicting accurately and making more predictions.

Next, we used the benchmark annotations to assess the difference in prediction accuracy between MTDNN and FFPred, grouped the GO terms according to the size of the MTDNN branches in which they locate, and calculated the mean value of ΔF_1 of GO terms in each branch. A graphical summary of this analysis is in Fig 6. Like the observations in holdout set evaluation, the performance improvement is not always linearly correlated with the number of

Table 2. Performance comparison at threshold equal to 0.5 between MTDNN and other prediction methods. This table reports average F_1, Precision, Recall scores of five methods in three GO domains, namely BP, MF, and CC, with the threshold equal to 0.5. MLDNN* is the performance obtained at the thresholds that maximise the F_1 scores for individual GO terms in the training set. The best F_1 scores in respective domains are marked in bold font.

Methods	BP			MF			CC		
	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
MTDNN	0.282	0.312	**0.296**	0.283	0.303	**0.292**	0.389	0.626	**0.48**
MLDNN	0.353	0.103	0.160	0.335	0.170	0.267	0.402	0.308	0.349
STDNN	0.070	0.676	0.127	0.112	0.606	0.189	0.168	0.858	0.281
FFPred	0.161	0.492	0.242	0.141	0.558	0.225	0.248	0.820	0.380
BLAST	0.129	0.019	0.033	0.447	0.035	0.066	0.423	0.027	0.051
Naive	0.539	0.024	0.046	0.318	0.117	0.171	0.446	0.407	0.425

https://doi.org/10.1371/journal.pone.0198216.t002

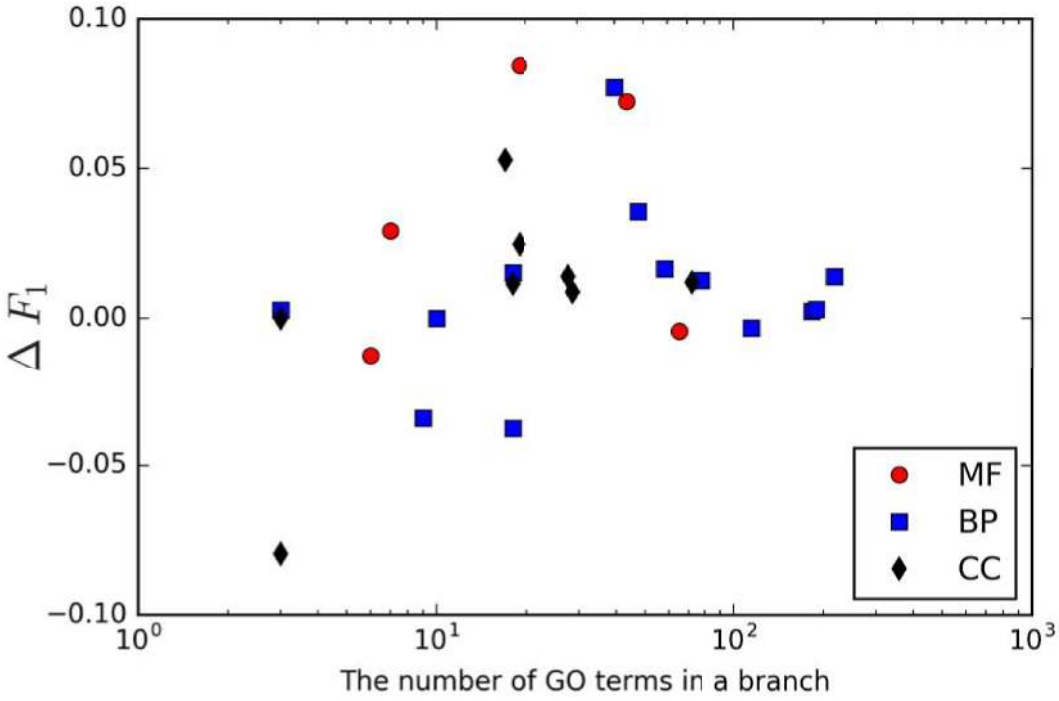

Fig 6. The scatter plot of mean values of δF_1 against the number of GO terms in a branch.

https://doi.org/10.1371/journal.pone.0198216.g006

tasks. Arguably, selecting the best choice of the size of the model is a data-dependent question and is not trivial. In our case, the range between 20 and 50 tasks in one model provides more improvement. However, breaking larger branches into smaller branches with 20–50 tasks, which sounds like a sensible solution, comes at the cost of reduced training sets, thus limiting the viability of deep learning approaches. Future work will research the best way into to develop a model using an expanded vocabulary and new datasets for our web service in the future. Additionally, all correctly predicted proteins in Benchmark set evaluation are listed in S6 Table.

Conclusions

In this paper, we developed a multi-task deep neural network (MTDNN) architecture to tackle the multi-label problem in protein function prediction. MTDNN is able to learn both a shared feature representation from all GO terms and specific feature characteristics from individual specific terms by employing two stacked multi-layer structures, one shared by all tasks and another one specific to each task on top of the shared one. Importantly, it is no requirement for MTDNN to proceed a bootstrap feature selection as what many traditional machine learning algorithms usually do. We compared MTDNN with five baseline methods, namely naive method, BLAST, FFPred, STDNN, and naive MLDNN. We then evaluated the accuracy of the proposed MTDNN using both holdout set and benchmark set. The benchmark set is a set of human proteins that had no experimentally verified annotations at that time, but received some in the following 24 months. The results show that impressively MTDNN offers considerably better performance in holdout set evaluation, and also considerably better performance on the benchmark set at the decision threshold equal to 0.5 by balancing precision and recall,

which makes MTDNN more favourable in practical use than the other methods tested. Another interesting result is that the performance improvement of MTDNN over the single-task predictor is not always linearly correlated with the number of tasks in the model. In our case, medium size models provided more improvement. Encouraged by the success in the current study, which suggests that MTDNN is a better solution to tackle the multi-label problem, we intend to improve the performance of predicting protein functions further by connecting MTDNN with various heterogeneous data sources, such as graph embedding features from protein-protein interaction networks, or gene expression data.

Software availability

All data and codes are available from http://bioinf.cs.ucl.ac.uk/downloads/mtdnn.

Supporting information

S1 Table. Summary of all branches. This table reports all branch root terms, the number of GO terms in each branch, the names of branch root terms and their domains.
(DOCX)

S2 Table. Hyperparameters and their value space for MTDNN. *Different layers may have different number of hidden units.
(DOCX)

S3 Table. Hyperparameters and their value space for MLDNN.
(DOCX)

S4 Table. The Spearman's rank correlation coefficients between the branch size and performance improvement for three GO domains.
(DOCX)

S5 Table. Holdout set evaluation, namely precision, recall and F1 scores.
(XLSX)

S6 Table. All correctly predicted proteins in Benchmark set evaluation.
(XLSX)

Acknowledgments

The authors are grateful to the members of the UCL Bioinformatics Group and to Jaqui Hodgkinson, Chris Cheadle, and Hongbao Cao from Elsevier for valuable discussions. This work made use of the high-performance computing facility of the Department of Computer Science at University College London. R.F. acknowledges funding from Elsevier; D.C and C.W. are supported by the UK Biotechnology and Biological Sciences Research Council (References: BB/L020505/1 and BB/L002817/1). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Author Contributions

Conceptualization: Rui Fa, Domenico Cozzetto, Cen Wan, David T. Jones.

Data curation: Domenico Cozzetto.

Formal analysis: Rui Fa, Domenico Cozzetto.

Funding acquisition: David T. Jones.

Investigation: Rui Fa, Domenico Cozzetto.

Methodology: Rui Fa, David T. Jones.

Project administration: David T. Jones.

Software: Rui Fa.

Supervision: David T. Jones.

Validation: Rui Fa, Domenico Cozzetto.

Visualization: Rui Fa.

Writing – original draft: Rui Fa.

Writing – review & editing: Rui Fa, Domenico Cozzetto, Cen Wan, David T. Jones.

References

1. The UniProt Consortium. UniProt: the universal protein knowledgebase. Nucleic Acids Res. 2017; 45: D158–D169. https://doi.org/10.1093/nar/gkw1099 PMID: 27899622
2. Huntley RP, Sawford T, Mutowo-Meullenet P, Shypitsyna A, Bonilla C, Martin MJ, et al. The GOA database: gene Ontology annotation updates for 2015. Nucleic Acids Res. 2015; 43: D1057–63. https://doi.org/10.1093/nar/gku1113 PMID: 25378336
3. The Gene Ontology Consortium. The Gene Ontology: enhancements for 2011. Nucleic Acids Res. 2011; 40: D559–D564. https://doi.org/10.1093/nar/gkr1028 PMID: 22102568
4. Cozzetto D, Jones DT. Computational Methods for Annotation Transfers from Sequence. Methods Mol Biol. 2017; 1446: 55–67. https://doi.org/10.1007/978-1-4939-3743-1_5 PMID: 27812935
5. Schnoes AM, Brown SD, Dodevski I, Babbitt PC. Annotation error in public databases: misannotation of molecular function in enzyme superfamilies. PLoS Comput Biol. 2009; 5: e1000605. https://doi.org/10.1371/journal.pcbi.1000605 PMID: 20011109
6. Gaudet P, Livstone MS, Lewis SE, Thomas PD. Phylogenetic-based propagation of functional annotations within the Gene Ontology consortium. Brief Bioinform. 2011; 12: 449–462. https://doi.org/10.1093/bib/bbr042 PMID: 21873635
7. Lobley AE, Nugent T, Orengo CA, Jones DT. FFPred: an integrated feature-based function prediction server for vertebrate proteomes. Nucleic Acids Res. 2008; 36: W297–302. https://doi.org/10.1093/nar/gkn193 PMID: 18463141
8. Minneci F, Piovesan D, Cozzetto D, Jones DT. FFPred 2.0: improved homology-independent prediction of gene ontology terms for eukaryotic protein sequences. PLoS One. 2013; 8: e63754. https://doi.org/10.1371/journal.pone.0063754 PMID: 23717476
9. Cozzetto D, Minneci F, Currant H, Jones DT. FFPred 3: feature-based function prediction for all Gene Ontology domains. Sci Rep. 2016; 6: 31865. https://doi.org/10.1038/srep31865 PMID: 27561554
10. Galperin MY, Koonin EV. Comparative Genomics Approaches to Identifying Functionally Related Genes. Lecture Notes in Computer Science. 2014. pp. 1–24.
11. Pellegrini M. Using phylogenetic profiles to predict functional relationships. Methods Mol Biol. 2012; 804: 167–177. https://doi.org/10.1007/978-1-61779-361-5_9 PMID: 22144153
12. Troyanskaya OG. Putting microarrays in a context: Integrated analysis of diverse biological data. Brief Bioinform. 2005; 6: 34–43. PMID: 15826355
13. Mostafavi S, Ray D, Warde-Farley D, Grouios C, Morris Q. GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. Genome Biol. 2008; 9: S4.
14. Clark WT, Radivojac P. Analysis of protein function and its prediction from amino acid sequence. Proteins. 2011; 79: 2086–2096. https://doi.org/10.1002/prot.23029 PMID: 21671271
15. Lee BJ, Shin MS, Oh YJ, Oh HS, Ryu KH. Identification of protein functions using a machine-learning approach based on sequence-derived properties. Proteome Sci. 2009; 7: 27. https://doi.org/10.1186/1477-5956-7-27 PMID: 19664241
16. Cozzetto D, Buchan DWA, Bryson K, Jones DT. Protein function prediction by massive integration of evolutionary analyses and multiple data sources. BMC Bioinformatics. 2013; 14 Suppl 3: S1.
17. Ma X, Chen T, Sun F. Integrative approaches for predicting protein function and prioritizing genes for complex phenotypes using protein interaction networks. Brief Bioinform. 2014; 15: 685–698. https://doi.org/10.1093/bib/bbt041 PMID: 23788799

18. Wass MN, Barton G, Sternberg MJE. CombFunc: predicting protein function using heterogeneous data sources. Nucleic Acids Res. 2012; 40: W466–70. https://doi.org/10.1093/nar/gks489 PMID: 22641853

19. Piovesan D, Giollo M, Leonardi E, Ferrari C, Tosatto SCE. INGA: protein function prediction combining interaction networks, domain assignments and sequence similarity. Nucleic Acids Res. 2015; 43: W134–40. https://doi.org/10.1093/nar/gkv523 PMID: 26019177

20. Radivojac P, Clark WT, Oron TR, Schnoes AM, Wittkop T, Sokolov A, et al. A large-scale evaluation of computational protein function prediction. Nat Methods. 2013; 10: 221–227. https://doi.org/10.1038/nmeth.2340 PMID: 23353650

21. Jiang Y, Oron TR, Clark WT, Bankapur AR, D'Andrea D, Lepore R, et al. An expanded evaluation of protein function prediction methods shows an improvement in accuracy. Genome Biol. 2016; 17: 184. https://doi.org/10.1186/s13059-016-1037-6 PMID: 27604469

22. Hinton GE, Osindero S, Teh Y-W. A Fast Learning Algorithm for Deep Belief Nets. Neural Comput. 2006; 18: 1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527 PMID: 16764513

23. Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th international conference on machine learning (ICML-10). 2010. pp. 807–814.

24. Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014; 15: 1929–1958.

25. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift [Internet]. 2015. Available: https://arxiv.org/abs/1502.03167

26. Min S, Lee B, Yoon S. Deep learning in bioinformatics. Brief Bioinform. 2016; bbw068.

27. Hanson J, Yang Y, Paliwal K, Zhou Y. Improving protein disorder prediction by deep bidirectional long short-term memory recurrent neural networks. Bioinformatics. 2017; 33: 685–692. https://doi.org/10.1093/bioinformatics/btw678 PMID: 28011771

28. Wang S, Sun S, Li Z, Zhang R, Xu J. Accurate De Novo Prediction of Protein Contact Map by Ultra-Deep Learning Model. PLoS Comput Biol. 2017; 13: e1005324. https://doi.org/10.1371/journal.pcbi.1005324 PMID: 28056090

29. Sønderby SK, Winther O. Protein Secondary Structure Prediction with Long Short Term Memory Networks [Internet]. Available: https://arxiv.org/abs/1412.7828

30. Ramsundar B, Kearnes S, Riley P, Webster D, Konerding D, Pande V. Massively multitask networks for drug discovery. 2015.

31. Mayr A, Klambauer G, Unterthiner T, Hochreiter S. DeepTox: Toxicity Prediction using Deep Learning. Front Environ Sci Eng China. 2016; 3. https://doi.org/10.3389/fenvs.2015.00080

32. Qi Y, Oja M, Weston J, Noble WS. A unified multitask architecture for predicting local protein properties. PLoS One. 2012; 7: e32235. https://doi.org/10.1371/journal.pone.0032235 PMID: 22461885

33. The Gene Ontology Consortium. Expansion of the Gene Ontology knowledgebase and resources. Nucleic Acids Res. 2016; 45: D331–D338. https://doi.org/10.1093/nar/gkw1108 PMID: 27899567

34. Ketkar N. Introduction to Theano. Deep Learning with Python. 2017. pp. 33–59.

35. Hauser M, Mayer CE, Söding J. kClust: fast and sensitive clustering of large protein sequence databases. BMC Bioinformatics. 2013; 14: 248. https://doi.org/10.1186/1471-2105-14-248 PMID: 23945046

36. Bergstra J., Yamins D., Cox D. D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. Proc. of the 30th international Conference on Machine Learning (ICML 2013).