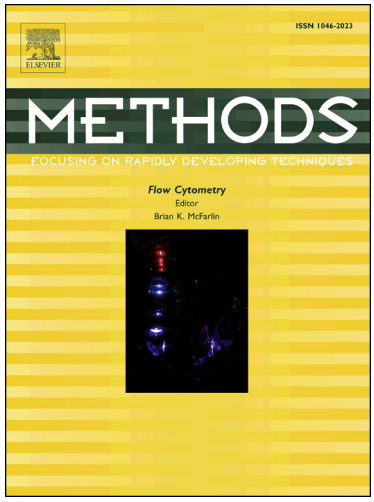

Near Perfect Protein Multi-Label Classification with Deep Neural Networks

Balázs Szalkai, Vince Grolmusz

PII: S1046-2023(17)30035-X
DOI: http://dx.doi.org/10.1016/j.ymeth.2017.06.034
Reference: YMETH 4264

To appear in: *Methods*

Please cite this article as: B. Szalkai, V. Grolmusz, Near Perfect Protein Multi-Label Classification with Deep Neural Networks, *Methods* (2017), doi: http://dx.doi.org/10.1016/j.ymeth.2017.06.034

Near Perfect Protein Multi-Label Classification with Deep Neural Networks

Balázs Szalkai[a,*], Vince Grolmusz[a,b,*]

[a] *PIT Bioinformatics Group, Eötvös University, H-1117 Budapest, Hungary*
[b] *Uratim Ltd., H-1118 Budapest, Hungary*

Abstract

Biological sequences can be considered as data items of high-, non-fixed dimensions, corresponding to the length of those sequences. The comparison and the classification of biological sequences in their relations to large databases are important areas of research today. Artificial neural networks (ANNs) have gained a well-deserved popularity among machine learning tools upon their recent successful applications in image- and sound processing and classification problems. ANNs have also been applied for predicting the family or function of a protein, knowing its residue sequence. Here we present two new ANNs with multi-label classification ability, showing impressive accuracy when classifying protein sequences into 698 UniProt families (AUC=99.99%) and 983 Gene Ontology classes (AUC=99.45%).

Introduction

Proteins are widely studied by numerous highly sophisticated tools in life science laboratories and by computational approaches. One important problem is the functional annotation or classification of proteins, using only structural information of these molecules.

There are several levels of protein structure characterization [1]: primary, secondary, tertiary and quaternary structures. The primary structure describes the residue (i.e., amino acid) sequence; the secondary structure characterizes the regions of local, highly regular substructures, like α-helices and β-sheets; the tertiary structure is the three-dimensional geometry of the folded substructures, while the quaternary structure describes the multi-subunit assembly of proteins, where each subunit consists of a single poly-peptide chain.

Therefore, the most basic protein structure is the primary, while the most complex is the quaternary. It is well-known that hundreds of proteins with

*Joint corresponding authors
Email addresses: szalkai@pitgroup.org (Balázs Szalkai), grolmusz@pitgroup.org (Vince Grolmusz)

known quaternary structures publicly deposited in the Protein Data Bank [2] still lack satisfying functional annotation [3, 4].

With the knowledge of tertiary or quaternary protein structures of non-annotated proteins, sometimes it is possible to find small, characteristic parts of the molecules that may help their functional annotation. In enzymes, the chemical details of the active site [5] can be characteristic, as it was shown e.g., in the case of ASP-HIS-SER catalytic triads [6], where the position of just four spatial points described the function of the enzyme well.

When only the primary structure, i.e., the residue sequence of the protein is known, it is more challenging to assign proper functions to these macromolecules. This description can be viewed as a high-dimensional representation of the molecule, where the dimension corresponds to the length of the residue sequence of the protein.

One possible approach is the sequence alignment-based similarity search between the input residue sequence x and a properly chosen and functionally annotated reference sequence database D. For the sequence alignment one may use the exact Smith-Waterman algorithm [7, 8], or the popular BLAST or its clones [9], or a more advanced, hidden Markov-model based HMMER search [10, 11, 12]. Suppose that the similarity search for input x returns the functionally annotated $y \in D$ as the most similar sequence from D. Then we may assign the function of y to x. In other words, the input is assigned the function of the most similar sequence in a reference database.

One deep problem with this simple sequence alignment approach is that the protein sequences have more and less conservative subsequences, and a similarity in the latter has less relevance than in the former. Another related problem is that the three-dimensional structure of the proteins are much more conserved in evolution than the primary structure. By some measures, defined in [13], they are three-to-ten times more conserved. Therefore, there could be big differences in the primary structures of two proteins with almost the same three-dimensional shape and with the very same function.

Consequently, more sophisticated classification methods are needed than the simple sequence alignment approach.

Neural networks for protein classification

A fast developing area of research is the application of artificial neural networks (ANNs) for protein classification. Artificial neural networks are perhaps the most widely used artificial intelligence tools today, frequently applied for classification and – nearly real-time – image- and sound processing for numerous applications, e.g., [14, 15, 16, 17]. ANNs contain artificial neurons or perceptrons [18] as basic building blocks, each of which computes a non-linear function of the weighted sum of its inputs. This non-linear function is termed the activation function. Then the output of the neuron may be fed to another neurons as input. The neurons in the first layer (called the input layer) work on the input of the network. The output of the network is computed by the output layer. When the problem to solve is a classification task, each class is assigned a

different neuron in the output layer, which is activated if the input is classified into the corresponding class.

While the building blocks of neural nets are the artificial neurons (or perceptrons), they can be viewed as a set of neuron layers. The output of a layer becomes the input of the next layer or the output of the whole network (for the last layer). A layer is a parametric function with learnable parameters. The network is the composition of these functions, and itself can be thought of as a parametric function f_η with η as the weight-parameter vector that assigns the weights of the inputs of each artificial neuron in the network. Today's neural networks are mostly deep neural networks, meaning that they have a much larger number of layers than earlier variations, resulting in a vastly increased learning capacity.

If one specifies the non-linear activation functions of the neurons, and the architecture of the network, then the neural network can be trained to perform its classification task. This learning capability is the most appealing property of neural nets. The weights of the neuron inputs, i.e., the vector η is improved step-by-step, as described below.

Neural networks are usually trained in a supervised fashion by inputting specific x values into it and backpropagating the error $\hat{y} - f_\eta(x)$. This means that a loss function is applied on the network output $y = f_\eta(x)$ and the desired output $\hat{y}$, and the parameters of the network are updated with gradient descent. The network input x can be modeled as a random variable, and $\hat{y} = g(x)$ is a function of x that needs to be approximated by the network. Let $\varepsilon(y, \hat{y})$ denote a loss function (e.g., $\varepsilon(y, \hat{y}) = (y - \hat{y})^2$ is a possible choice, named the L2 loss function). Then the expected loss of the network can be written as

$$\ell(\eta) = \mathbf{E}_x \varepsilon(f_\eta(x), g(x)),$$

i.e., as a function of the network parameter vector η.

This formula can then be approximated by substituting the expected value for the mean over a given set of possible inputs mirroring the actual input distribution. If $x_1, ...x_n$ denote a random sample of inputs, then an approximation of the above formula is

$$\tilde{\ell}(\eta) = \frac{1}{n} \sum_{i=1}^{n} \varepsilon(f_\eta(x_i), g(x_i)).$$

Updating the network weights can be done with stochastic gradient descent (SGD) [19] using the update step $\eta_{k+1} := \eta_k - \lambda \frac{\partial \tilde{\ell}}{\partial \eta}(\eta_k)$. The initial value η_0 is initialized randomly. If the learning rate λ is sufficiently small, η_k will then converge to a place of local minimum.

Stochastic gradient descent is susceptible to stalling near saddle points in the error surface, causing slow convergence. Furthermore, the size of the update steps (the differences in η) can be too large if λ is too big, and this may result in divergence. Therefore SGD has since been improved by introducing momentum or using the statistics of the gradients to normalize the update steps and thus

yielding the more modern methods such as RMSProp, Adagrad or Adam [20, 21, 22].

Classification problems with the inputs possibly corresponding to multiple classes are called multi-label classification problems. When we intend to classify proteins by using their amino-acid sequences as inputs and different functional or structural classes as outputs, we also need multi-label classification procedures, as a protein may be assigned one or more functional or structural classes.

Previous work

In the overview of the previous work, for each publication, we give an overall, short description, and in most cases, we explicitly describe the dataset used, the filtering made, the validation method, and the accuracy attained.

The NNPDB program described in [23] applied an n-gram model for classification into a small number of classes. Dataset: Oxidoreductases (334 sequences in 106 classes) and Electron transfer proteins (360 sequences in 26 classes). Filtering: none. Validation: holdout set method, cca. 33% in test set and remaining in training set. Accuracy: reported 99% on oxidoreductases, but a class was considered correctly identified if at least one of the 17 encoding modules identified it.

In the work of [24] the protein sequences were stored in 20 x 20 bi-peptide matrices, and the neural network was trained in an unsupervised manner. Dataset: All human proteins SwissProt release 19.0, 8/91. Filtering: length greater than 50. Number of sequences after filtering: 1758. Validation: no train/test set (unsupervised clustering). The accuracy of the method was not reported explicitly.

In [25] the ProCANS tool was constructed for the classification of the members of the PIR database [26], by using the n-gram model with SVD (singular value decomposition) and MPL (multi-layer perceptrons). The proteins were encoded by the ae12 system into length-462 vectors. In the performance evaluation the authors of [25] counted the classification as "exact" if the correct superfamily was present in the first 5 suggestions, i.e. in a quite "tolerant" way. Even with this definition, the family classification accuracy is 97.02%, somewhat worse than our precision and recall values (where we say that a classification is successful when it returns the exact family of the protein). Additionally, in [25], for the computation of the singular value decomposition (SVD) the authors applied 659 training and also the 235 test proteins, and not only the 659 training proteins. The target classification classes in this work are pairwise disjoint, i.e., the authors did not solve a multi-label classification task (while we do in the present contribution). Dataset: part of PIR (894 sequences for SVD study). Filtering: none. Validation: holdout set method.

The neural networks, built in [27], applied the n-gram model with SVD and MPL and attained a 90% sensitivity when classifying unknown sequences into 3311 PIR superfamilies and families [26]. In [28] a hybrid neural network – sequence alignment approach was applied for gene family identifications. Our approach does not use sequence alignment, just pure ANN tools in the classification.

The authors of [29] classified proteins in transmembrane and not-transmembrane groups with ANNs. In a subsequent work, [30] subdivided the non-transmembrane proteins into further three classes, i.e., four classes in total. Our contribution classifies the whole UniProt into 698 classes, and not just 4.

In the publication [31], protein sequences were classified into four superfamilies only. The article [32] constructed a neural network for classifying 10 superfamilies from the PIR database [26]. The neural network constructed in [33] performed a yes-no classification into 2 classes: globin or non-globin.

Choosing input from the Protein Data Bank [2], [34] constructed and trained multiple fully-connected multilayer perceptrons (MLPs) for function prediction. The accuracy rate of the prediction was 75%. The works of [35, 36] apply a hybrid motif-search & neural network approach for classifying into a maximum of 7 protein classes.

Using a small network and two models: a simple 2-gram model (lsa2) and a more involved hydrophobicity-based (hyd2) model, the authors of [37] performed protein classification into five functional classes and four families. Further protein sequence classification results were reviewed in [38] up to the year 2013.

More recently, in [39] proteins from the PIR database were classified into 10 superfamilies with a maximal accuracy of 93.69%.

The work [40] described an ANN-based Gene Ontology functional classification solution that yielded less than 90% AUC for one class, and 80% AUC for two further classes. Dataset: 30k sequences for Bos taurus and 15k sequences for Gallus gallus. Validation: holdout set method (10%). We note that our Gene Ontology classification results have an AUC of 90.69%.

In the article [41] Gene Ontology [42] classification was done with AUC values of around 0.5; our AUC values in the present contribution are around 0.9. We classify into 983 classes, while the authors of [41] into 2849 classes. Dataset: 10 datasets (about yeast proteins, link in the article is dead). Filtering: not mentioned. Validation: holdout set method.

In [43], by applying convolution networks, DNA sequences were classified into a small number of classes (less than 10); our classifications use much more classes with high accuracy classification (in the case of UniProt, 698 classes).

In the FFPred 3 tool, the authors of [44] trained separate SVM's for several hundred classes and attained F1 values under 43%. Dataset: UniProt, with Gene Ontology. Validation method: 3- and 5-fold cross-validation. In comparison, our F1 values are around 86% in Gene Ontology and 98% in UniProt classification.

The work of [45] performed UniProt classification with different methods (SVM, LSTM, GRU, CNN) and with 589 target classes. Their best F1 value is 94.8%, while we classify into 698 classes with a better F1 value 98.63% in the present work.

In the publication [46], the authors trained the neural network with 80% of the sequences of the SwissProt subset of UniProt, and test the performance on the remaining 20% of the sequences from SwissProt. The authors attain nearly 100% accuracy, but they only classified into 4 classes, while here we show a neural network that classifies SwissProt into 698 classes with a near-

100% accuracy. Dataset of [46]: SwissProt, trained on only 4 protein functions. Filtering: sequence length between 10 and 1000 or 10 and 2000 for different classes. Number of sequences after filtering: unknown. Validation method: 5-fold cross-validation.

Methods and Materials

We have applied the SwissProt subset of the UniProt protein database [47], acquired from `http://uniprot.org` as starting point (using the query "goa:(*) AND reviewed:yes"), containing 526,526 sequences having Gene Ontology IDs [42] at the date of download of 15 February 2017. The sequences were downloaded along with their assigned UniProt families. This set was shuffled and then divided into training and test sets using the bash commands `head -5000` and `tail -n +5001`. Since the data had headers, the test set contained 4,999 protein sequences, and the training set had the rest (521,527 sequences).

We trained two models on this dataset: one for Gene Ontology functional classification [42] and one for UniProt family classification [47]. Each of these had to solve a *multilabel classification task*, as a sequence could have been assigned more than one functions or families. There was a logical relationship among the attributes (functions/families) in both cases, describable using a directed acyclic graph (DAG), where each edge signifies an implication: for each edge $A \to B$, if an entry belongs to the class (has attribute) A, then it will also belong to the class B. Classes that do not have exiting edges are termed the *roots* of this graph, and the *level* of each class is a non-negative integer corresponding to the length of the shortest path from that class to a root. This means that roots have level 0, and each non-root node (class/attribute) has a level greater than 0. (c.f., Figure 1).

In Gene Ontology, the `is_a` relation defines this directed acyclic graph on the functional attributes, e.g. if a protein has the function "thyroid hormone generation" (GO:0006590), then this implies that the protein falls into the category "thyroid hormone metabolic process" (GO:0042403), which in turn implies the functions "phenol-containing compound metabolic process" (GO:0018958), "cellular modified amino acid metabolic process" (GO:0006575) and "hormone metabolic process" (GO:0042445), because if a compound is a "thyroid hormone", then it is also a "phenol-containing compound", "cellular modified amino acid" and "hormone".

On the other hand, UniProt families exist on 4 levels: superfamily, family, subfamily and sub-subfamily. Here each class can belong to zero to one parent class, and may or may not have child classes, so the relationships of the classes can be represented by a forest of directed trees (arborescences). For example, "HOG1 sub-subfamily" belongs to "MAP kinase subfamily", which belongs to "Ser/Thr protein kinase family", which is a child of "Protein kinase superfamily". But not all roots of the directed forest are superfamilies: for example, "Mimivirus L114/R131 family" has no parent despite being a family and not a superfamily.

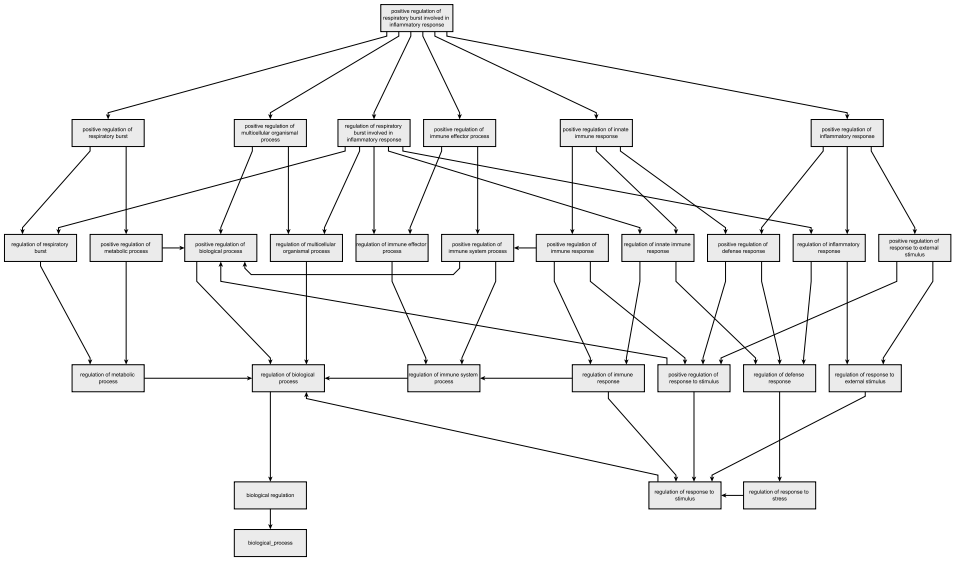

Figure 1: A small example from Gene Ontology [42]. Each edge corresponds to an implication: for each edge $A \to B$, if an entry belongs to the class (has attribute) A, then it will also belong to the class B. Classes that do not have exiting edges are termed the *roots* of this graph, and the *level* of each class is a non-negative integer corresponding to the length of the shortest path from that class to a root.

When training the network, each training protein sequence was fed as the input (x) of the network, along with the classification target ($\hat{y}$) of that sequence encoded as a 0-1 vector. Each class was represented as a coordinate in the target vector. If the sequence belonged to a specific class, then all the classes reachable from that class in the DAG were included in the classification target, encoded as ones in the target vector $\hat{y}$. If the sequence did not belong to a specific class (or any of its subclasses), then the corresponding component of the target vector was zero.

The input sequences were encoded as two arrays: one 3-dimensional array `inputSeq` with dimensions `[batch_size, max_length, dims]` and another array `inputSeqLen` encoding the length of the individual sequences with dimension `[batch_size]`. Here `batch_size` means the number of sequences in a minibatch and was set to 32. `max_length` was the maximum allowed length of a sequence: sequences longer than this were omitted in the training phase and cropped to the first `max_length` amino acids in the testing phase. This parameter was set to 2000 in our case. These parameters were largely determined by the available video memory on our GPU (4GB RAM). Parameter `dims` (=26) was the length of the vectors encoding the individual amino acids. We encoded each amino acid as a 26-dimensional vector, where the first 20 components comprised a one-hot vector (all components zero except the one uniquely identifying the amino acid in question), while the other 6 components encoded various properties of the

amino acids: charge (± 1 or 0.1 in the case of Histidine which is positive about 10% of the time and neutral 90% of the time), hydrophobicity, and the binary attributes `isPolar`, `isAromatic`, `hasHydroxyl` and `hasSulfur`. Apart from this straightforward encoding scheme, we did not use any other information about the biological properties of the sequences or their amino acids, including the secondary structure or the presence of pre-selected motifs. This means that the neural network had to work with the amino acid sequence alone, without any further help from other machine learning methods.

At training time we confined the set of sequences to those between 162 (this was the minimum length for the neural network, so the output of the last pooling layer was at least one amino acid long) and 2000 (a practical limit because of available memory). The starting "M" (Methionine) character was removed from all the sequences. We also excluded those classes from the attribute graph which had fewer than 200 or 150 sequences in the training set in the case of Gene Ontology and UniProt family classification, respectively. The classes were considered up until level 3 in both graphs (in the case of UniProt families, this was not a real restriction, as the graph already had only 4 levels ranging from level 0 to level 3 in that case). All UniProt sub-subfamilies had too few members, so in fact, all the UniProt sub-subfamilies were dropped, leaving a total of 3 levels (0..2) in that case. In the end, 983 classes were considered in the Gene Ontology task and 698 classes in the UniProt family classification task.

The deep neural network had a primarily convolutional architecture with 1D spatial pyramid pooling [15] and fully connected layers at the end. The architecture is shown in Table 1. The network had 6 one-dimensional convolution layers with kernel sizes `[6,6,5,5,5,5]` and depths (filter counts) `[128,128,256,256,512,512]`, with PReLU (parametric rectified linear unit) activation [16]. We used max pooling with kernel size and stride 2 after each convolutional layer, except the first one. Max pooling was omitted after the first layer so that the network can conserve details about the fine structure of the protein. Each max pooling layer was followed by a batch normalization layer to help normalize the statistics of the heatmaps.

After the last convolutional layer, we applied a 1D variant of SPP (spatial pyramid pooling) to convert the output of the last max pooling layer into a fixed-length representation of each (variable-length) sequence. We performed SPP on 3 levels with 1, 4 and 16 divisions, respectively. This means that the activation of the neurons was max-pooled over the whole sequence, then the sequence was divided into four almost equally sized parts and the activations were max-pooled over each of the 4 subsequences, then again the sequence was divided into 16 parts, yielding 21 values altogether for each sequence and each of the 512 filters. Consequently, after SPP, the network state could be represented as an array of shape `[batch_size, 21, 512]`.

The output of the spatial pyramid pooling layer was fed into a fully-connected layer with 1024 units and PReLU activation, followed by a dropout layer with $p = 0.5$ to avoid overfitting [14], and a batch normalization layer to normalize the mean and standard deviation. Then a second fully connected layer with sigmoid activation assigned numerical values (likelihoods) between

conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
batch_norm (scale=False)
conv (size=6, stride=1, depth=128, padding=VALID, activation=prelu)
max_pool (size=2, stride=2, padding=VALID)
batch_norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max_pool (size=2, stride=2, padding=VALID)
batch_norm (scale=False)
conv (size=5, stride=1, depth=256, padding=VALID, activation=prelu)
max_pool (size=2, stride=2, padding=VALID)
batch_norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max_pool (size=2, stride=2, padding=VALID)
batch_norm (scale=False)
conv (size=5, stride=1, depth=512, padding=VALID, activation=prelu)
max_pool (size=2, stride=2, padding=VALID)
batch_norm (scale=False)
spp (levels=3, divs_per_level=4)
fully_connected (units=1024, activation=prelu)
dropout (p=0.5)
batch_norm (scale=True)
fully_connected (units=983, activation=sigmoid)

Table 1: Gene Ontology functional classifier network

0 and 1 for each class, yielding the output array y with shape `[batch_size, n_classes]`. Note that softmax activation cannot be used because the network had to perform a multi-label classification task.

We defined the loss of the neural network as the weighted cross entropy between the predictions and the targets. If C denotes the number of classes, and ℓ the network loss, and N the minibatch size, then let

$$\ell := \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} w_j \hat{y}_{ij} (-\log y_{ij}) + (1 - \hat{y}_{ij})(-\log(1 - y_{ij})).$$

Here the class weights w_j ($1 \leq j \leq C$) are responsible for class balancing to avoid misclassification of instances belonging to infrequent classes. w_j is defined as $\max\{1, \min\{5, \mu(s)/s_j\}\}$, where s_j is the size (number of sequences) of the jth class, and $\mu(s)$ is the mean of the class sizes. Thus the error of misclassifying an instance originally belonging to an infrequent class is weighted up by a factor between 1 and 5.

We also added an L2 regularization penalty to ℓ to reduce the risk of overfitting, with $\lambda = 6 \times 10^{-11}$.

Both neural networks (the Gene Ontology and also the UniProt family classifier) were implemented in TensorFlow [48, 49, 50], and trained for 150000 iterations (minibatches), i.e. 9.2 epochs, with a fixed learning rate of 0.002. Training took 29 hours on an NVIDIA Geforce GTX 750 Ti GPU. For simplicity, no validation set was used, as overfitting was hoped to be largely addressed by the regularization methods (batch normalization, dropout, L2 regularization). We calculated various performance measures of the networks on the test set, including precision, recall and F1-value, both per class and altogether. The AUC (area under the ROC curve) was calculated using micro-averaging (the ROC curve is the true positive rate as the function of the false positive rate). The Python programming language was used in the implementation.

For each level of the class graph, the perfect prediction rate was also determined: this was defined as the number of sequences where the set of classes on the specific level of the graph was perfectly predicted by the network. As the test set had sequences shorter than the minimum length or longer than the maximum length, the networks could not be tested on all the test sequences, but only 3776 and 3744 sequences, respectively.

Results and Discussion

The performance of the two networks on the test set is summarized in Table 2 below. To our knowledge, both of our networks outperform all previously described purely neural solutions on these classification tasks, detailed in section "Previous work" in the Introduction.

In the evaluation of the results we apply the following quality measures:

Precision denotes the number of true positives divided by the number of predicted positives.

Network	# classes	Precision	Recall	F1-value	AUC
Gene Ontology	983	91.17%	81.62%	86.13%	99.45%
UniProt families	698	99.75%	97.53%	98.63%	99.99%

Table 2: The general evaluation of the performance of our neural networks. Note the very high number of classes and the near-perfect accuracy, compared to the previous work, listed in the Introduction. Precision denotes the number of true positives divided by the number of predicted positives. *Recall* (or sensitivity) denotes the number of true positives divided by the actual number of positives. The *F1 value* is the harmonic mean of precision and recall. The area under the ROC curve (*AUC*) corresponds to the probability that a model outputting a score between 0 and 1 ranks a randomly chosen positive sample higher than a randomly chosen negative sample.

Level	Classes	Precision	Recall	F1-value
0	3	93.99%	98.23%	96.07%
1	53	89.60%	85.00%	87.24%
2	236	91.01%	80.78%	85.59%
3	691	91.42%	77.16%	83.68%

Table 3: Per-level performance of the Gene Ontology network

Recall (or sensitivity) denotes the number of true positives divided by the actual number of positives.

The *F1 score* is the harmonic mean of precision and recall.

The area under the ROC curve (*AUC*) corresponds to the probability that a model outputting a score between 0 and 1 ranks a randomly chosen positive sample higher than a randomly chosen negative sample.

For example, our UniProt family classifier network achieved an F1-value of 98.63%, much better than the 94.85% reported by [45] (which was achieved in an easier task, involving a classification into only 589 instead of 698 classes).

We also calculated the per-level precision and recall of the two networks. Probably because of the smaller number of nodes and greater difference among protein sequences in different classes, the topmost level is the easiest to classify for the Gene Ontology network. The UniProt family network performed the best on level 2 (UniProt subfamilies that have a containing family and superfamily), but this is probably because there were only 13 nodes at that level.

From our results and previous work we can conclude that the Gene Ontology functional classification task seems to be harder for artificial neural networks than the UniProt family classification task, probably because the assignment of

Level	Classes	Precision	Recall	F1-value
0	524	99.84%	97.68%	98.75%
1	161	99.31%	96.66%	97.97%
2	13	100.00%	100.00%	100.00%

Table 4: Per-level performance of the UniProt family network

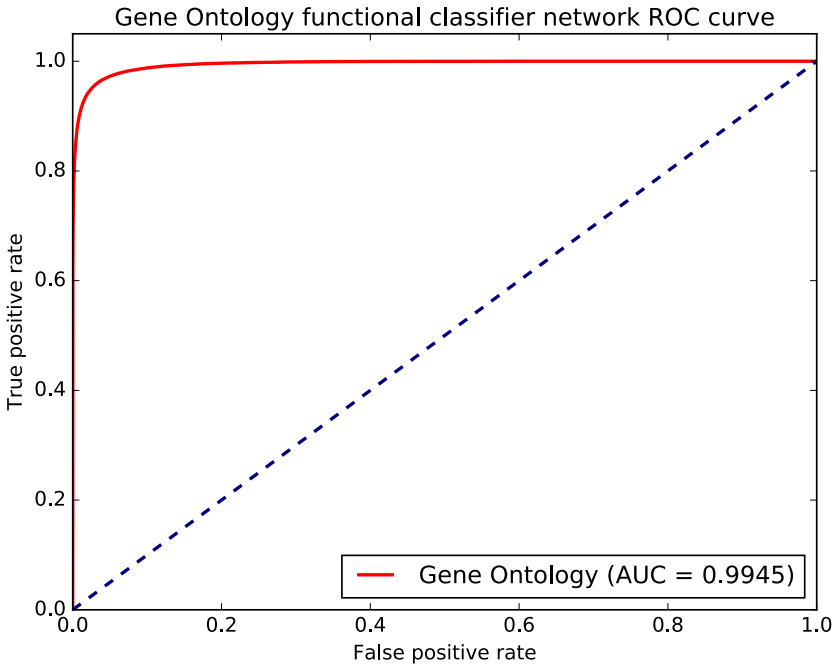

Figure 2: The ROC curve for the classifier network for the Gene Ontology task.The AUC value is 99.45%

UniProt families depends heavily on sequence similarity, and thus it is easier to classify proteins into UniProt families instead of functional classes based purely on the amino acid sequence data. Additionally, the class graph has an easier (forest) structure in the UniProt family case.

The ROC curves for the two classifier networks are shown in Figures 2 and 3. From the AUC values (99.45%, 99.99%) it is clear that the networks achieve excellent classification performance, unmatched by prior architectures, listed in the Introduction.

Conclusions

We have constructed deep artificial neural networks for protein classification into UniProt families and Gene Ontology classes. By the detailed comparison of previous work, our neural networks outperformed the existing solutions and have attained a near 100% of accuracy in multi-label, multi-family classification.

We also have conducted some experiments with the simplification of the network architecture. According to our experience, batch normalization is crucial to the performance of these networks, along with the number of layers (the overall depth of the networks): network variants without batch normalization

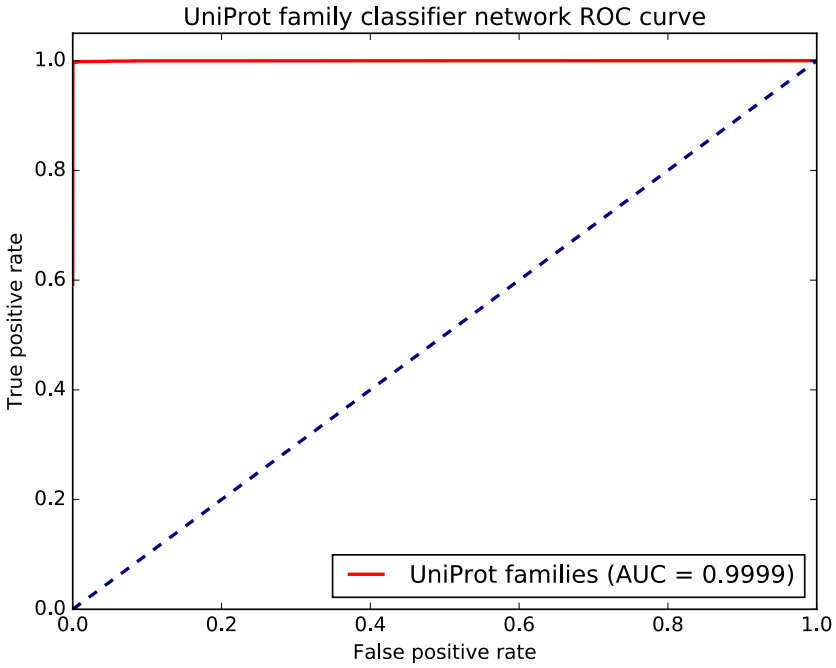

Figure 3: The ROC curve for the classifier network for the UniProt task. The AUC value is 99.99%

and 5 (instead of 6) layers showed a performance drop of several percentage points. This emphasizes that deeper neural networks with more parameters have a much larger capacity for learning good representations, and normalizing the statistics of the layers can greatly improve network performance, as others already observed in image classification tasks [17]. We hypothesize that, with more GPU RAM available, one can further improve upon the performance of our neural network by simply increasing the number of convolutional or fully connected layers, but overfitting may become a problem for such large network architectures.

Acknowledgments

BS was supported through the new national excellence program of the Ministry of Human Capacities of Hungary. The authors declare no conflicts of interest.

References

[1] Thomas E Creighton. *Proteins: structures and molecular properties*. Macmillan, 1993.

[2] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28:235–242, 2000.

[3] Nurul Nadzirin and Mohd Firdaus-Raih. Proteins of unknown function in the Protein Data Bank (PDB): an inventory of true uncharacterized proteins and computational tools for their analysis. *International Journal of Molecular Sciences*, 13:12761–12772, Oct 2012. ISSN 1422-0067. doi: 10.3390/ijms131012761.

[4] Zoltan Szabadka and Vince Grolmusz. High throughput processing of the structural information in the protein data bank. *J Mol Graph Model*, 25 (6):831–836, Mar 2007. doi: 10.1016/j.jmgm.2006.08.004. URL `http://dx.doi.org/10.1016/j.jmgm.2006.08.004`.

[5] Gabor Ivan, Zoltan Szabadka, and Vince Grolmusz. A hybrid clustering of protein binding sites. *FEBS J*, 277(6):1494–1502, Mar 2010. doi: 10.1111/j.1742-4658.2010.07578.x. URL `http://dx.doi.org/10.1111/j.1742-4658.2010.07578.x`.

[6] Gabor Ivan, Zoltan Szabadka, Rafael Ordog, Vince Grolmusz, and Gabor Naray-Szabo. Four spatial points that define enzyme families. *Biochem Biophys Res Commun*, 383(4):417–420, Jun 2009. doi: 10.1016/j.bbrc.2009.04.022. URL `http://dx.doi.org/10.1016/j.bbrc.2009.04.022`.

[7] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–197, Mar 1981.

[8] Gabor Ivan, Daniel Banky, and Vince Grolmusz. Fast and exact sequence alignment with the Smith–Waterman algorithm: The SwissAlign webserver. *Gene Reports*, 4:26–28, 2016.

[9] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, Oct 1990. doi: 10.1016/S0022-2836(05)80360-2. URL `http://dx.doi.org/10.1016/S0022-2836(05)80360-2`.

[10] Sean R Eddy. A new generation of homology search tools based on probabilistic inference. *Genome Inform*, 23(1):205–211, Oct 2009.

[11] Sean R Eddy. Accelerated profile HMM searches. *PLoS Comput Biol*, 7 (10):e1002195, Oct 2011. doi: 10.1371/journal.pcbi.1002195. URL `http://dx.doi.org/10.1371/journal.pcbi.1002195`.

[12] Balazs Szalkai, Ildiko Scheer, Kinga Nagy, Beata G Vertessy, and Vince Grolmusz. The metagenomic telescope. *PloS One*, 9:e101605, 2014. ISSN 1932-6203. doi: 10.1371/journal.pone.0101605.

[13] Kristoffer Illergard, David H Ardell, and Arne Elofsson. Structure is three to ten times more conserved than sequence–a study of structural response in protein cores. *Proteins*, 77:499–508, Nov 2009. ISSN 1097-0134. doi: 10.1002/prot.22458.

[14] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 1026–1034, December 2015. doi: 10.1109/ICCV.2015.123.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv*, 2015.

[18] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5 (4):115–133, 1943.

[19] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4):185–196, 1993.

[20] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Yann Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. In *Advances in Neural Information Processing Systems*, pages 1504–1512, 2015.

[22] YN Dauphin, H de Vries, J Chung, and Y Bengio. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *arXiv preprint arXiv:1502.04390*, 2015.

[23] Cathy H Wu and Tzx-Chung Chang. Protein classification using a neural network database system. In *Proceedings of the conference on Analysis of neural network applications*, pages 29–41. ACM, 1991.

[24] Edgardo A Ferran, Pascual Ferrara, and Bernard Pflugfelder. Protein classification using neural networks. In *ISMB*, pages 127–135, 1993.

[25] C Wu, M Berry, Y S Fung, and J McLarty. Neural networks for molecular sequence classification. *Proceedings. International Conference on Intelligent Systems for Molecular Biology*, 1:429–437, 1993. ISSN 1553-0833.

[26] W C Barker, J S Garavelli, H Huang, P B McGarvey, B C Orcutt, G Y Srinivasarao, C Xiao, L S Yeh, R S Ledley, J F Janda, F Pfeiffer, H W Mewes, A Tsugita, and C Wu. The protein information resource (pir). *Nucleic acids research*, 28:41–44, Jan 2000. ISSN 0305-1048.

[27] Cathy Wu, Michael Berry, Sailaja Shivakumar, and Jerry McLarty. Neural networks for full-scale protein sequence classification: Sequence encoding with singular value decomposition. *Machine Learning*, 21(1-2):177–193, 1995.

[28] Cathy H Wu and Sailaja Shivakumar. Gene family identification network design. In *Intelligence and Systems, 1998. Proceedings., IEEE International Joint Symposia on*, pages 103–110. IEEE, 1998.

[29] Claude Pasquier and SJ Hamodrakas. An hierarchical artificial neural network system for the classification of transmembrane proteins. *Protein Engineering*, 12(8):631–634, 1999.

[30] C Pasquier, V J Promponas, and S J Hamodrakas. Pred-class: cascading neural networks for generalized protein classification and genome-wide applications. *Proteins*, 44:361–369, Aug 2001. ISSN 0887-3585.

[31] Jason TL Wang, Qicheng Ma, Dennis Shasha, and Cathy H Wu. Application of neural networks to biological data mining: a case study in protein sequence classification. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 305–309. ACM, 2000.

[32] Dianhui Wang, Nung Kion Lee, Tharam S Dillon, and Nicholas J Hoogenraad. Protein sequences classification using modular RBF neural networks. In *Australian Joint Conference on Artificial Intelligence*, pages 477–486. Springer, 2002.

[33] Jinmiao Chen and Narendra S Chaudhari. Protein family classification using second-order recurrent neural networks. *GENOME INFORMATICS SERIES*, pages 520–521, 2003.

[34] Wagner Rodrigo Weinert and Heitor Silvério Lopes. Neural networks for protein classification. *Applied Bioinformatics*, 3(1):41–48, 2004.

[35] Konstantinos Blekas, Dimitrios I Fotiadis, and Aristidis Likas. Protein sequence classification using probabilistic motifs and neural networks. In *Artificial Neural Networks and Neural Information Processing-ICANN/ICONIP 2003*, pages 702–709. Springer, 2003.

[36] Konstantinos Blekas, Dimitrios I Fotiadis, and Aristidis Likas. Motif-based protein sequence classification using neural networks. *Journal of Computational Biology*, 12(1):64–82, 2005.

[37] Andre Luis Debiaso Rossi. Protein classification using artificial neural networks with different protein encoding methods. In *Intelligent Systems Design and Applications, 2007. ISDA 2007. Seventh International Conference on*, pages 169–176. IEEE, 2007.

[38] Suprativ Saha and Rituparna Chaki. A brief review of data mining application involving protein sequence classification. In *Advances in Computing and Information Technology*, pages 469–477. Springer, 2013.

[39] Jiuwen Cao and Lianglin Xiong. Protein sequence classification with improved extreme learning machine algorithms. *BioMed Research International*, 2014, 2014.

[40] Davide Chicco, Peter Sadowski, and Pierre Baldi. Deep autoencoder neural networks for gene ontology annotation predictions. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 533–540. ACM, 2014. URL `http://dl.acm.org/citation.cfm?id=2649442`.

[41] R. Cerri, R. C. Barros, and A. C. P. L. F. de Carvalho. Hierarchical classification of gene ontology-based protein functions with neural networks. In *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, pages 1–8, July 2015. doi: 10.1109/IJCNN.2015.7280474. URL `http://ieeexplore.ieee.org/document/7280474/`.

[42] Gene Ontology Consortium. Gene Ontology Consortium: going forward. *Nucleic Acids Research*, 43:D1049–D1056, Jan 2015. ISSN 1362-4962. doi: 10.1093/nar/gku1179.

[43] N.G. Nguyen, V.A. Tran, D.L. Ngo, D. Phan, F.R. Lumbanraja, M.R. Faisal, B. Abapihi, M. Kubo, and K. Satou. Dna sequence classification by convolutional neural network. *J. Biomedical Science and Engineering*, 9: 280–286, 2016.

[44] Domenico Cozzetto, Federico Minneci, Hannah Currant, and David T Jones. Ffpred 3: feature-based function prediction for all gene ontology domains. *Scientific reports*, 6:31865, Aug 2016. ISSN 2045-2322. doi: 10.1038/srep31865. URL `http://www.nature.com/articles/srep31865`.

[45] Timothy K. Lee and Tuan Nguyen. Protein family classification with neural networks, 2016. URL `https://cs224d.stanford.edu/reports/LeeNguyen.pdf`.

[46] Xueliang Liu. Deep recurrent neural network for protein function prediction from sequence. *arXiv preprint arXiv:1701.08318*, 2017.

[47] UniProt Consortium. The Universal Protein Resource (UniProt) 2009. *Nucleic Acids Res*, 37(Database issue):D169–D174, Jan 2009. doi: 10.1093/nar/gkn664. URL `http://dx.doi.org/10.1093/nar/gkn664`.

[48] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, pages 265–283, 2016.

[49] Martín Abadi. Tensorflow: learning functions at scale. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, pages 1–1. ACM, 2016.

[50] Ladislav Rampasek and Anna Goldenberg. TensorFlow: Biology's gateway to deep learning? *Cell Systems*, 2:12–14, Jan 2016. ISSN 2405-4712. doi: 10.1016/j.cels.2016.01.009.

- New, deep, multi-label neural networks are constructed for protein classification;
- Neural network classification has numerous appealing properties ;
- In classifying UniProt inputs into 698 classes, the AUC accuracy is 99.99%;
- In classifying into 983 classes of Gene Ontology, our AUC accuracy is 99.45%;
- These accuracy values are one of the bests in the literature today.