# Hierarchical Multi-Label Classification with Chained Neural Networks

Jônatas Wehrmann, Rodrigo C. Barros, Silvia N. das Dôres
Pontifícia Universidade Católica do RS
Av. Ipiranga, 6681, Porto Alegre-RS, Brazil
rodrigo.barros@pucrs.br

Ricardo Cerri
Universidade Federal de São Carlos
Rodovia Washington Luís, Km 235
São Carlos-SP, Brazil
cerri@dc.ufscar.br

## ABSTRACT

In classification tasks, an object usually belongs to one class within a set of disjoint classes. In more complex tasks, an object can belong to more than one class, in what is conventionally termed *multi-label classification*. Moreover, there are cases in which the set of classes are organised in a hierarchical fashion, and an object must be associated to a single path in this hierarchy, defining the so-called *hierarchical classification*. Finally, in even more complex scenarios, the classes are organised in a hierarchical structure and the object can be associated to multiple paths of this hierarchy, defining the problem investigated in this article: *hierarchical multi-label classification* (HMC). We address a typical problem of HMC, which is protein function prediction, and for that we propose an approach that chains multiple neural networks, performing both local and global optimisation in order to provide the final prediction: one or multiple paths in the hierarchy of classes. We experiment with four variations of this chaining process, and we compare these strategies with the state-of-the-art HMC algorithms for protein function prediction, showing that our novel approach significantly outperforms these methods.

## CCS Concepts

•**Computing methodologies** → **Neural networks;**

## Keywords

hierarchical multi-label classification, neural networks, protein function prediction

## 1. INTRODUCTION

There is a niche of classification tasks in which classes are not disjoint but organised in a hierarchical structure, namely *hierarchical classification* (HC) problems. In these cases, objects are associated with a given superclass and its subclasses. The hierarchical structure that organises the set of classes can assume the form of a tree or of a Directed Acyclic Graph (DAG). There is a subset of complex HC problems in which each object can be associated to several different paths of the class hierarchy, namely *hierarchical multi-label classification* (HMC). Typical HMC problems are text classification [12, 5, 10] and bioinformatics tasks such as protein function prediction [13, 11, 16].

Algorithms that perform HMC must be capable of labeling objects according to one or multiple paths in the class hierarchy by optimising a loss function either locally or globally [14]. Algorithms that perform local learning attempt to discover the specificities that are present in regions of the class hierarchy, later combining the predictions to provide the final classification. Global approaches for HMC, on the other hand, usually consist of a single classifier capable of associating objects with their corresponding classes in the hierarchy at once [17, 3, 4, 1].

There is a trade-off regarding the use of either global or local approaches. Global approaches are usually cheaper than local approaches, and they do not suffer from the error-propagation problem. However, global approaches are usually less likely to capture local information from the hierarchy, eventually suffering from underfitting. Local approaches, in turn, are often more computationally expensive since they rely on multiple classifiers, but they are often capable of extracting local information from regions of the hierarchy. Nevertheless, the use of many classifiers per level may result in the loss of label dependency during the training process [14]. To combine the advantages of both local and global approaches, we argue that it is possible to use a hybrid strategy capable of optimising both local and global loss functions. Our novel approach is based on a chain of neural networks hereafter called **H**ierarchical **M**ulti-label **C**lassification with **C**hained **N**eural **N**etworks (HMC-CNN).

In HMC-CNN, a multi-layer perceptron (MLP) is trained per hierarchical level, and each MLP is responsible for capturing local information from the classes of its respective level. The system is chained, and its goal is to propagate local information by plugging the output of the MLP responsible for level $l$ in the input of the MLP responsible for level $l + 1$. It could be considered a local HMC approach, since each MLP is responsible for predicting the classes of a given level. Nevertheless, the chain of neural networks are totally



*SAC 2017,April 03-07, 2017, Marrakech, Morocco*



http://dx.doi.org/10.1145/3019612.3019664

connected both for the forward and backward phases, which means the sub-networks benefit from both local information (local gradients from the respective level) and global information (global gradients from the last layer of the chained system). We show that HMC-CNN comfortably establishes the new state-of-the-art in protein function prediction.

## 2. RELATED WORK

This section discusses recent machine learning based HMC algorithms for protein function prediction. In [17], the authors proposed three algorithms based on the concept of Predictive Clustering Trees (PCT): Clus-HMC (global method), Clus-SC (local method), and Clus-HSC (local method), and Clus-HMC provided the best results.

In [11], the authors proposed a global method using Ant Colony Optimization (ACO). The method discovers classification rules in the format `if-then`, where an ACO algorithm is employed to optimize the antecedents of the rules. Basically, a sequential object covering procedure is applied to create classification rules that cover most training objects. The method is initialized with an empty set of rules, and a new rule is added to the set while the number of objects not covered by any rule is higher than a given threshold.

Bi and Kwok [2] proposed a problem transformation approach for tree and DAG hierarchies that can be used with any learner. It employs a kernel dependency estimation (KDE) approach to reduce the number of labels to a manageable number of single-label learning problems. To preserve the hierarchy information among labels, they developed a generalized Condensing Sort and Select Algorithm (CSSA), which finds an optimal approximation subtree in a tree. They project the labels to a 50-dimensional space and then use ridge regression in the learning step.

Protein function prediction problems with incomplete hierarchical labels were investigated in [18]. Hierarchical and flat (non-hierarchical) similarities between functions were considered, and combined similarity between the labels was defined. The known labels and this similarity are used to estimate missing functions in the hierarchy. Information regarding protein-protein interactions are also used. Situations in which labels are missing were simulated by randomly masking leaf functions of a protein.

In the experiments we perform in this work, we compare our methods with the 2 methods that show the best results in the literature: Clus-HMC [17] and CSSA [2]. All methods have been applied to the same datasets (and exact same training/validation/test partitions) that are used in our experiments. In addition, these methods produce the same type of output provided by HMC-CNN, and hence we use the same evaluation methodology for analyzing the results.

## 3. HIERARCHICAL MULTI-LABEL CLASSIFICATION WITH CHAINED NEURAL NETWORKS

The rationale behind HMC-CNN is to perform both local and global optimisation. It comprises a chain of neural networks where each sub-network is responsible for the predictions of a given hierarchical level. Therefore, each sub-network locally optimises a loss function, and then it propagates local information from one level to the next. Since the sub-network responsible for a given level is connected to the sub-network responsible for the subsequent level, the chained system ends when it reaches the deepest level of the class hierarchy. When that occurs, HMC-CNN backpropagates the gradients to the entire chained system, reinforcing the local gradients with global information.

We present next the details of three versions of HMC-CNN, namely HMC-CNN-P (which only propagates **p**redictions in the forward pass), HMC-CNN-A (which **a**ugments the predictions with the training features), and HMC-CNN-U (an **u**nchained model that does not directly connect each per-level network). Moreover, we perform a detailed complexity analysis of HMC-CNN in the end of the section.

### 3.1 HMC-CNN-P

Figure 1 presents the architecture of HMC-CNN-P. In the figure, $\mathbf{x}$ is a training object and $h_{1l}$, $h_{2l}$ and $O_l$ are, respectively, the hidden and output layers of the sub-network associated with level $l$. The matrices $\mathbf{W}_{1l}$, $\mathbf{W}_{2l}$, and $\mathbf{W}_{3l}$ represent, respectively, the weights connecting the input attributes and the neurons in the first hidden layer; the activations of the first hidden layer with the neurons in the second hidden layer; and the activations in the second hidden layer with the neurons in the output layer of the sub-network associated with level $l$.

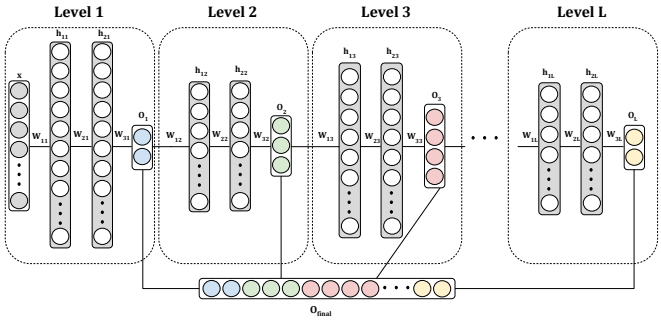


**Figure 1: HMC-CNN-P architecture. Each sub-network of the chained system predicts classes from its corresponding hierarchical level. During the backward phase, gradients flow from both the merged output class layer and the end of the chained system (deepest hierarchical level).**

In HMC-CNN, each hierarchical level is associated with a sub-network with two hidden layers, which means the number of neurons of its output layer corresponds to the number of classes of the corresponding hierarchical level. The mean squared error loss function is used so the gradients are backpropagated through each sub-network. Note that the output layer of a given sub-network is used as input layer to the subsequent sub-network, and hence the chained system ends when the final hierarchical level is reached. The output layer of the chained system is the result of merging the output layers of each sub-network, thus generating two sources of gradient flow in the network: locally per-sub-network and

globally from the output of the last sub-network.

For performing parameter updating, HMC-CNN makes use of AdaGrad [8] instead of stochastic gradient descent (SGD). In order to show the gains that are obtained by using AdaGrad instead of SGD, we show in Figure 2 both the area under the precision-recall curve and the training loss per epoch of HMC-CNN-P when using SGD and when using AdaGrad for the parameter updating step. Note that AdaGrad converges faster than SGD, which is what is argued by the authors in [8].

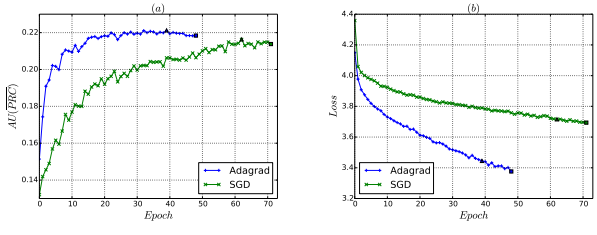


**Figure 2: Comparison of SGD and AdaGrad optimisation within HMC-CNN-P in the cellcycle dataset. (a) Values of $AU(\overline{PRC})$ per epoch. (b) Values of training loss per epoch. Triangle (△) depicts the best model whereas the square (□) indicates the end of the training phase.**

Finally, HMC-CNN follows the recent trend in neural networks (more specifically, deep learning) in which one should substantially increase the number of neurons in the hidden layers in order to enlarge the network capacity. Since a high-capacity network is extremely prone to overfitting, we perform dropout regularisation [15], which stochastically disables a percentage of neurons in each hidden layer. Besides the natural regularisation effect, dropping out neurons during training has the extra special property of emulating an ensemble of sub-networks within each network, naturally improving the predictive performance of the entire system.

## 3.2 HMC-CNN-A

The training process of HMC-CNN-A follows a different procedure than in HMC-CNN-P. The objects (feature vectors) are now used as input to all sub-networks, and not just to the first one as it was in HMC-CNN-P. Therefore, instead of simply propagating the predictions from one sub-network to the next, we augment the vector of predictions with the input feature vectors of the training objects. In Figure 3, note that the output of each sub-network is now concatenated with the training input feature vectors in order to form the input of the sub-networks.

As in HMC-CNN-P, HMC-CNN-A is also trained with AdaGrad, a large number of neurons in the hidden layers and dropout regularisation.

## 3.3 HMC-CNN-U

In HMC-CNN-U (**u**nchained), an individual sub-network is independently trained for each hierarchical level, and the sub-networks are not chained as in HMC-CNN-P and HMC-CNN-A. The only connection of the sub-networks is regarding the final merged output class layer. In Figure 4 we can

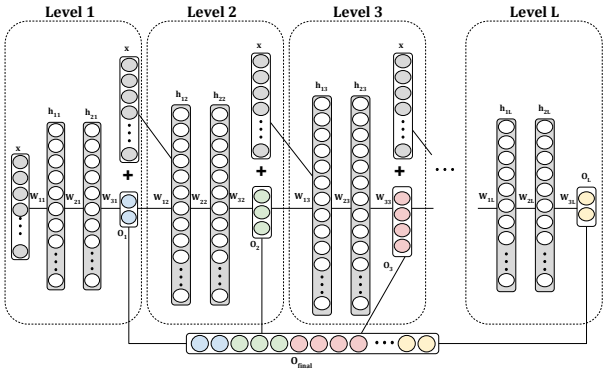


**Figure 3: HMC-CNN-A architecture. Each sub-network predicts classes from its corresponding hierarchical level, but the predictions are augmented with the training objects in order to be used as input to the subsequent sub-network. Once again gradients flow from both the merged output layer and the end of the chained system during the backward phase.**

see that now the gradients flow from a single source (final output layer) to each independent sub-network, in an end-to-end fashion.

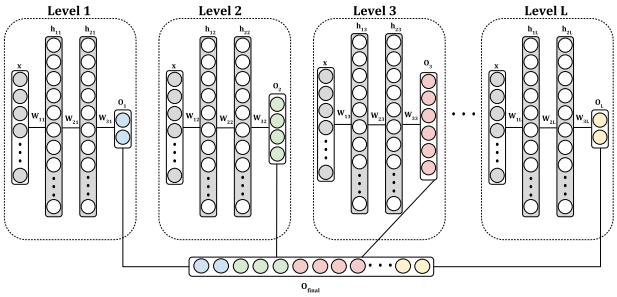


**Figure 4: HMC-CNN-U architecture. Networks are not chained but regarding the final merged output class layer. Gradients now flow from a single source (output layer), with only local information being used to train each sub-network.**

## 3.4 HMC-CNN-$\mu$: An Ensemble of HMC-CNN Models

Our last proposed version of HMC-CNN is an ensemble of the three previous versions: HMC-CNN-P, HMC-CNN-A, and HMC-CNN-U. In this approach, namely HMC-CNN-$\mu$, we simply average the real-valued class vector that holds the prediction for each version, generating a final combined prediction.

## 3.5 Generating global predictions

To generate the final global prediction for a test object, thresholds are applied to the output prediction values from each of the sub-networks to define the predictions for each level. If the output of a given class neuron $j$ is equal to or

larger than a given threshold, the object is assigned to class $c_j$. The final classification for HMC-CNN is given by a binary vector $\mathbf{v}$ of size $|C|$, where $C$ is the set of all classes in the hierarchy. If the output value of neuron $j$ is equal to or larger than a given threshold, the value 1 is assigned to position $\mathbf{v}_j$. Otherwise, the position is set to 0.

Different threshold values result in different predicted classes, and since the activation function used in all neurons is the logistic sigmoid function (values ranging from 0 to 1), the threshold values range within [0,1]. The larger the threshold value, the lower the number of predicted classes. Conversely, the lower the threshold value, the larger the number of predicted classes. Nevertheless, note that during the classification process, the output values that are passed from sub-network to sub-network are not the values obtained after the application of a threshold, but the regular activation values within [0,1]. The application of the threshold is only performed to generate the final global predictions.

After HMC-CNN has generated the final predictions, a post-processing phase is necessary to eventually correct inconsistencies. For instance, it is possible that a subclass is predicted by HMC-CNN but its superclass is not. The post-processing phase guarantees that only consistent predictions are made. For such, this phase removes those predicted classes whose superclasses were not predicted.

### 3.6 Computational complexity

Considering the computational cost, each sub-network used in HMC-CNN-A has a complexity of $\mathcal{O}(W_l)$, with $W_l$ being the number of weights and biases of the sub-network associated with level $l$. Assume that $A$ is the number of attributes in the dataset, $H_{1l}$ and $H_{2l}$ are the number of neurons in the first and second hidden layers of the sub-network associated with level $l$, and $O_l$ is the number of output neurons of the sub-network associated with level $l$. We can then define $W_1$ as $(A+1)\times H_{11}+(H_{11}+1)\times H_{21}+(H_{21}+1)\times O_1$. From the second level onwards, $W_l$ is defined as $(O_{l-1}+A+1)\times H_{1l}+(H_{1l}+1)\times H_{2l}+(H_{2l}+1)\times O_l$. The overall training cost of each sub-network associated with level $l$ in HMC-CNN is then $\mathcal{O}(W_l \times m \times n)$, with $m$ being the number of training objects and $n$ the number of training epochs.

In HMC-CNN-U, the computational cost is lower, since the predictions are not passed along the chained system. For HMC-CNN-P, the computational cost in the first level considers the number of features of the objects. From the second level onwards, only the number of classes is considered, since the classes are the unique input of the sub-networks.

## 4. EXPERIMENTAL METHODOLOGY

In this section, we present the full experimental methodology that is employed during experimentation in order to allow for reproducibility. We first present the HMC algorithms that are the current state-of-the-art in protein function prediction. We also describe the datasets that are used and the evaluation criteria for assessing the predictive performance of each algorithm. Additionally, we present the parameter setting employed by HMC-CNN. The source code of all HMC-CNN versions are available for download[1].

We compare HMC-CNN with two HMC algorithms, which are considered the state-of-the-art for protein function prediction: PCT-based method Clus-HMC [17], and CSSA [2]. Clus-HMC is a global-based method that builds a single decision tree to cope with all classes simultaneously. CSSA is a kernel dependency estimation approach that reduces the number of labels by projecting them into a lower dimensional space and then allowing the application of any given classifier (here, we use its original version that employs ridge regression for learning).

We experiment over 10 freely available[2] datasets related to protein function prediction. These datasets are related to issues like phenotype data and gene expression levels, and are structured as trees. Table 1 presents the main characteristics of the training, validation, and test partitions from each dataset. The reader is referred to [17] for the full description of each dataset. As in [17], 2/3 of each dataset were used for inducing the classification models and 1/3 for test.

As discussed in Section 3, the outputs of HMC-CNN for each class are probability values, and the same is true for the baseline algorithms. Hence, the final predictions (binary vector indicating the presence or absence of each class) are generated after thresholding these probability values. The choice of optimal threshold is difficult and often subjective. Therefore, we follow the trend of HMC research in which we avoid choosing thresholds by employing precision-recall curves (PR-curves) as the evaluation criterion for comparing the different approaches. For generating a PR-curve for a given classification method, one must select a predefined number of different thresholds within [0,1] to be applied over the outputs of each method, finally generating several precision and recall points in the PR plane. The interpolation of these points [6] constitute a PR-curve, and the quantitative criterion one analyses is the area under such a curve ($AU(\overline{PRC})$).

For defining the hyper parameters of HMC-CNN, we performed non-exhaustive tests that optimized the validation accuracy regarding the cellcycle dataset. We use learning rate of 0.05, $\epsilon = 10^{-6}$, dropout of 50%, logistic sigmoid neurons, and weight initialization as described in [9].

The number of neurons in each layer is defined by $\log_{10} O_l \times 128$. E.g., a sub-network with 18 classes in its corresponding hierarchical level will have a total of $(\log_{10} 18 \times 128) = (1.26\times128) \approx 160$ neurons in each hidden layer. Our chained neural networks are trained by optimizing the mean-squared error.

In order to provide some reassurance about the validity and non-randomness of the obtained results, we present the results of statistical tests by following the approach proposed by 7]. This approach seeks to compare multiple algorithms on multiple data sets, and it is based on the use of the Friedman test with a corresponding post-hoc test. If the null hypothesis of similar performances is rejected, then we proceed with the Nemenyi post-hoc test for pairwise comparisons.

[1] https://goo.gl/AalVHL

[2] http://www.cs.kuleuven.be/~dtai/clus/hmcdatasets.html

**Table 1: Summary of datasets: number of attributes ($|A|$), number of classes ($|C|$), number of classes per level, total number of instances (Total), and number of multi-label instances (Multi).**

| Dataset | $\|A\|$ | $\|C\|$ | Classes per level | Training | | Valid | | Test | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Total | Multi | Total | Multi | Total | Multi |
| Cellcycle | 77 | 499 | 18/80/178/142/77/4 | 1628 | 1323 | 848 | 673 | 1281 | 1059 |
| Church | 27 | 499 | 18/80/178/142/77/4 | 1630 | 1322 | 844 | 670 | 1281 | 1057 |
| Derisi | 63 | 499 | 18/80/178/142/77/4 | 1608 | 1309 | 842 | 671 | 1275 | 1055 |
| Eisen | 79 | 461 | 18/76/165/131/67/4 | 1058 | 900 | 529 | 441 | 837 | 719 |
| Expr | 551 | 499 | 18/80/178/142/77/4 | 1639 | 1328 | 849 | 674 | 1291 | 1064 |
| Gasch1 | 173 | 499 | 18/80/178/142/77/4 | 1634 | 1325 | 846 | 672 | 1284 | 1059 |
| Gasch2 | 52 | 499 | 18/80/178/142/77/4 | 1639 | 1328 | 849 | 674 | 1291 | 1064 |
| Pheno | 69 | 455 | 18/74/165/129/65/4 | 656 | 537 | 353 | 283 | 582 | 480 |
| Seq | 478 | 499 | 18/80/178/142/77/4 | 1701 | 1344 | 879 | 679 | 1339 | 1079 |
| Spo | 80 | 499 | 18/80/178/142/77/4 | 1600 | 1301 | 837 | 666 | 1266 | 1047 |

# 5. EXPERIMENTS AND DISCUSSION

In this section, we present the experiments that were carried out to compare the predictive performance of the four HMC-CNN versions with the state-of-the-art HMC algorithms. Table 2 presents the $AU(\overline{PRC})$ values for tested algorithms, namely HMC-CNN-P, HMC-CNN-A, HMC-CNNP-U, HMC-CNN-$\mu$, Clus-HMC, and CSSA. We highlight in bold the best absolute values that were obtained per dataset.

For the HMC-CNN versions, we calculated its $AU(\overline{PRC})$ for the validation dataset at each epoch. When the $AU(\overline{PRC})$ value reaches a plateau and does not increase for 10 epochs, we stop the training process and select the best neural network from the validation set to execute over the test objects.

The first analysis we perform by examining Table 2 is regarding HMC-CNN-$\mu$ (our most competitive version of HMC-CNN) and its performance when compared with the current state-of-the-art approaches. Observe that HMC-CNN-$\mu$ is the algorithm with the greatest number of wins and best average ranking of the experiment, comfortably surpassing the state-of-the-art approaches. We did expect HMC-CNN-$\mu$ to be the best HMC-CNN method since it averages the results of the three stand-alone versions, acting like an ensemble of HMC strategies.

We can also notice that HMC-CNN-P, HMC-CNN-A, and HMC-CNN-U outperform Clus-HMC in all 10 datasets. Moreover, the three versions of HMC-CNN outperform CSSA in 6 out of 10 datasets. Only HMC-CNN-P was not capable of reaching a lower average ranking than CSSA. Therefore, it is reasonable to argue that HMC-CNN outperforms the current state-of-the-art in protein function prediction both regarding its stand-alone versions and its ensemble form HMC-CNN-$\mu$.

We move to the statistical significance analysis. The first test to be executed was the Friedman test, which indicated the existence of significant differences with a $p = 1.06 \times 10^{-8}$. Hence, we moved to the post-hoc Nemenyi test, which is presented in Figure 5. For this particular analysis, we employ the graphical representation suggested by 7], the so-called *critical diagrams*. In such a diagram, a horizontal line represents the axis on which we plot the average rank values of the methods. When comparing all the algorithms against each other, we connect the groups of algorithms that are not significantly different through a horizontal line. We also show the critical difference given by the Nemenyi test above the graph, which was $CD = 2.38$. By analysing the Nemenyi statistical test, we can observe that HMC-CNN-$\mu$ is the only method that outperforms Clus-HMC and HMC-CNN-P with statistical significance. Whereas the test does not show a significant difference between HMC-CNN-$\mu$, HMC-CNN-U, HMC-CNN-A, and CSSA, it is easy to see that HMC-CNN-$\mu$ is the preferred method, presenting the lowest average ranking. Indeed, note that CSSA is in the limit of the critical difference regarding HMC-CNN-$\mu$, which means HMC-CNN-$\mu$ probably outperforms CSSA for looser $\alpha$ values (the test shows values for $\alpha = 0.05$ but the difference is probably significant for $\alpha = 0.1$). Clus-HMC is still considered one of the best approaches for HMC since it generates interpretable trees while CSSA and HMC-CNN are black-box approaches. Notwithstanding, HMC-CNN arguably establishes itself as the novel state-of-the-art approach for HMC of protein functions.

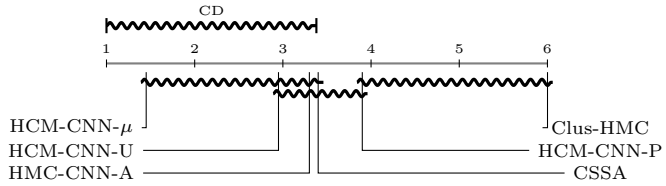


**Figure 5: Critical diagram for the Nemenyi post-hoc statistical test.**

# 6. CONCLUSIONS AND FUTURE WORK

In this study we propose a novel neural network based method for hierarchical multi-label classification in tree-structured hierarchies, namely HMC-CNN. It comprises a chain of neural networks in which sub-networks are associated with specific hierarchical class levels, and each sub-network is responsible for the predictions in its corresponding level. To the best of our knowledge, HMC-CNN is the first method in the HMC literature that benefits from both local and global information at the same time. We performed several experiments using 10 protein function prediction datasets whose classes were structured as trees. According to the empirical analysis, three different versions of HMC-CNN were capable of outperforming the state-of-the-art methods in several datasets. Moreover, an ensemble of these three versions established itself as the novel state-of-the-art in protein function prediction, comfortably outper-

**Table 2: Comparison of the proposed HMC-CNN versions and the baseline methods. Values are of $AU(\overline{PRC})$.**

| Dataset | HMC-CNN-P | HMC-CNN-A | HMC-CNN-U | HMC-CNN-$\mu$ | Clus-HMC | CSSA |
|---|---|---|---|---|---|---|
| Cellcycle | 0.225 | 0.233 | 0.227 | **0.238** | 0.172 | 0.196 |
| Church | 0.174 | 0.174 | 0.175 | 0.177 | 0.170 | **0.179** |
| Derisi | 0.188 | 0.187 | 0.187 | 0.190 | 0.175 | **0.194** |
| Eisen | 0.246 | 0.274 | 0.273 | **0.279** | 0.204 | 0.220 |
| Gasch1 | 0.248 | 0.260 | 0.262 | **0.269** | 0.205 | 0.216 |
| Gasch2 | 0.229 | 0.236 | 0.234 | **0.243** | 0.195 | 0.218 |
| Pheno | 0.165 | 0.162 | 0.164 | 0.165 | 0.160 | **0.167** |
| Spo | 0.187 | 0.193 | 0.194 | 0.197 | 0.186 | **0.216** |
| Expr | 0.238 | 0.261 | 0.263 | **0.270** | 0.210 | 0.228 |
| Seq | 0.237 | 0.247 | 0.255 | **0.261** | 0.211 | 0.226 |
| Average Ranking | 3.90 | 3.30 | 2.95 | **1.45** | 6.00 | 3.40 |

forming the best methods in the literature. As future work, we intend to use protein hierarchies structured as DAGs, which are not defined in a per-level basis. Meanwhile, we are interested in applying the proposed method to other domains such as text and scene classification. Finally, it is our intention to investigate different strategies for dealing with error inconsistency.

## 7. ACKNOWLEDGMENTS

The authors would like to acknowledge the following Brazilian research agencies for funding this research: FAPESP (2015/14300-1), CAPES (23038.006924/2014-00), and CNPq (442231/2014-8).

## 8. REFERENCES

[1] R. C. Barros, R. Cerri, A. A. Freitas, and A. C. P. L. F. Carvalho. Probabilistic clustering for hierarchical multi-label classification of protein functions. In *European Conference on Machine Learning (ECML/PKDD 2013)*, pages 385–400, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[2] W. Bi and J. T. Kwok. Multi-label classification on tree- and dag-structured hierarchies. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 17–24, New York, NY, USA, 2011.

[3] R. Cerri, R. C. Barros, and A. C. P. L. F. Carvalho. A genetic algorithm for hierarchical multi-label classification. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 250–255, New York, NY, USA, 2012. ACM.

[4] R. Cerri, R. C. Barros, A. C. P. L. F. de Carvalho, and A. A. Freitas. A grammatical evolution algorithm for generation of hierarchical multi-label classification rules. In *2013 IEEE Congress on Evolutionary Computation*, pages 454–461, June 2013.

[5] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Incremental algorithms for hierarchical classification. *Machine Learning*, 7:31–54, 2006.

[6] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *International Conference on Machine Learning*, pages 233–240, 2006.

[7] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[8] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[10] A. Mayne and R. Perry. Hierarchically classifying documents with multiple labels. In *IEEE Symposium on Computational Intelligence and Data Mining*, pages 133–139, 2009.

[11] F. Otero, A. Freitas, and C. Johnson. A hierarchical multi-label classification ant colony algorithm for protein function prediction. *Memetic Computing*, 2:165–181, 2010.

[12] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7:1601–1626, 2006.

[13] L. Schietgat, C. Vens, J. Struyf, H. Blockeel, D. Kocev, and S. Dzeroski. Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, 11:2, 2010.

[14] C. Silla and A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22:31–72, 2010.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[16] G. Valentini. True path rule hierarchical ensembles for genome-wide gene function prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(3):832–847, May 2011.

[17] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73:185–214, 2008.

[18] G. Yu, H. Zhu, and C. Domeniconi. Predicting protein functions using incomplete hierarchical labels. *BMC Bioinformatics*, 16(1), 2015.