AWX: An Integrated Approach to Hierarchical-Multilabel Classification

Luca Masera[0000−0001−9734−4202] and Enrico Blanzieri[0000−0001−6524−0601]

University of Trento, Trento 38123, Italy *

Abstract. The recent outbreak of works on artificial neural networks (ANNs) has reshaped the machine learning scenario. Despite the vast literature, there is still a lack of methods able to tackle the hierarchical multilabel classification (HMC) task exploiting entirely ANNs. Here we propose AWX, a novel approach that aims to fill this gap. AWX is a versatile component that can be used as output layer of any ANN, whenever a fixed structured output is required, as in the case of HMC. AWX exploits the prior knowledge on the output domain embedding the hierarchical structure directly in the network topology. The information flows from the leaf terms to the inner ones allowing a jointly optimization of the predictions. Different options to combine the signals received from the leaves are proposed and discussed. Moreover, we propose a generalization of the true path rule to the continuous domain and we demonstrate that AWX's predictions are guaranteed to be consistent with respect to it. Finally, the proposed method is evaluated on 10 benchmark datasets and shows a significant increase in the performance over plain ANN, HMC-LMLP, and the state-of-the-art method CLUS-HMC.

Keywords: Hierarchical Multilabel Classification · Structured Prediction · Artificial Neural Networks

1 Introduction

The task of multilabel classification is an extension of binary classification, where more then one label may be assigned to each example [17]. However, if the labels are independent, the task can be reduced without loss of generality to multiple binary tasks. Of greater interest is the case where there is an underlying structure that forces relations through the labels. These relations define a notion of consistency in the annotations, that can be exploited in the learning process to improve the prediction quality. This task goes under the name of hierarchical multilabel classification (HMC) and can be informally defined as the task of assigning a subset of consistent labels to each example in a dataset [21].

* L. Masera and E. Blanzieri are with the Department of Computer Science and Information Engineering, University of Trento, Via Sommarive, 9, 38123 Trento, Italy corresponding email: luca.masera@unitn.it

Knowledge is organized in hierarchies in a wide spectrum of applications, ranging from content-categorization [16,19] to medicine [14] and biology [13,4,10]. Hierarchies can be described by trees or direct acyclic graphs (DAG), where the nodes are the labels (we will refer to them as terms in the rest of the paper) and the edges represent *is_a* relations that occurs between a child node and its parents. These relations can be seen as a logical implication, because if a term is true then also its parents must be true. In other words, *"the pathway from a child term to its top-level parent(s) must always be true"* [4]. This concept was introduced by "The Gene Ontology Consortium" (GO) under the name of "true path rule" (TPR) to guarantee the consistency of the ontology with respect to the annotations, such that, whenever a gene product is found to break the rule, the hierarchy is remodelled consequently. Besides guaranteeing the consistency of the annotation space, the TPR can be forced also on the predictions. Inconsistencies in the predictions have been shown to be confusing for the final user, who will likely not trust and reject them [11]. Even though there are circumstances where inconsistencies are accepted, we will focus on the strict case, where the TPR should hold for predictions as well.

HMC has a natural application in bioinformatics, where ontologies are widely used as annotation space in predictive tasks. The critical assessment of functional annotation (CAFA) [7,12], for example, is the reference challenge for the protein function prediction community and uses the GO terms as annotations. The ontology comprises thousands of terms organized in three DAGs and the concepts expressed by some of those terms are so specific that just few proteins have been experimentally found belonging to them. Therefore, even though a perfect multilabel classification on the leaf nodes would solve the problem, the lack of examples forces the participants to exploit the hierarchical structure, by learning a single model [15] or by correcting the output of multiple models *a posteriori* [5].

HMC methods can be characterized in terms of local (top-down) or global (one-shot)approaches. The former [2,1,5] rely on traditional classifiers, training multiple models for each or subset of the labels, and applying strategies for selecting training examples or correcting the final predictions. Global methods [21,15], on the other hand, are specific classification algorithms that learn a single global model for the whole hierarchy. Vens *et al.* [21] compare the two approaches, and propose a global method called CLUS-HMC, which trains one decision-tree to cope with the entire classification problem. The proposed method is then compared with its naïve version CLUS-SC, which trains a decision-tree for each class of the hierarchy, ignoring the relationships between classes, and with CLUS-HSC, which explores the hierarchical relationships between the classes to induce a decision-tree for each class. The authors performed the experiments using biological datasets, and showed that the global method was superior both in the predictive performance and size of the induced decision-tree. CLUS-HMC has been shown to have state-of-the-art performance, as reported in the study by Triguero *et al.* [20].

More recently, the introduction of powerful GPU architectures brought artificial neural networks (ANNs) back to the limelight [9,6,18]. The possibility to scale the learning process with highly-parallel computing frameworks allowed the community to tackle completely new problems or old problems with completely new and complex ANNs' topologies. However, ANN-based methods that account for HMC have not yet evolved consequently. Attempts to integrate ANNs and HMC have been conducted by Cerri *et al.* [2,1]. They propose HMC-LMLP, a local model where for each term in the hierarchy is trained an ANN, that is fed with both the original input and with the output of models built for the parent terms. The performance are comparable with CLUS-HMC, however, because of the many models trained, the proposed approach is not scalable with deep learning architectures that requires a considerable amount of time for training. To the best of our knowledge there are no better model that exploits ANNs in the training process.

In this work we present AWX (Adjacency Wrapping matriX), a novel ANN output component. We aim at filling the gap between HMC and ANNs left open in the last years, enabling HMC tasks to be tackled with the power of deep learning approaches. The proposed method incorporates the knowledge on the output-domain directly in the learning process, in form of a matrix that propagates the signals coming from the previous layers. The information flows from the leaves, up to the root, allowing a jointly optimization of the predictions. We propose and discuss two approaches to combine the incoming signals, the first is based on the `max` function, while the second on ℓ-norms. AWX can be incorporated on top of any ANN, guaranteeing the consistency of the results with respect to the TPR. Moreover, we provide formal description of the HMC task, and propose a generalization of the TPR to the continuous case. Finally AWX is evaluated on ten benchmark datasets and compared against CLUS-HMC, that is the state-of-the-art, HMC-LMLP and the simple multi-layer perceptron MLP.

2 The HMC task

This section formalizes the task of hierarchical multilabel classification (HMC) and introduces the notation used in the paper. Consider the hierarchy involved in the classification task described by a DAG $H = (\mathcal{T}, E)$, where $\mathcal{T} = \{t_1, \ldots, t_m\}$ is a set of m terms and $E = \{\mathcal{T} \times \mathcal{T}\}$ is a set of directed edges. In particular the edges in E represent "*is_a*" relations, i.e. given a sample x and $\langle t_u, t_v \rangle \in E$, t_u *is_a* t_v means that t_u implies t_v, $t_u(x) \implies t_v(x)$ for all x.

The following box collects relevant definitions.

child t_u is a child of t_v iff $\langle t_u, t_v \rangle \in E$, $children(t_v)$ returns the children of t_v;
parent t_v is a parent of t_u iff $\langle t_u, t_v \rangle \in E$, $parents(t_u)$ returns the parents of t_u;
root a term t_v such that $parents(t_v) = \varnothing$;
leaf a term t_u such that $children(t_u) = \varnothing$, $\mathcal{F} = \{t_u | child(t_u) = \varnothing\}$ is the set of leaves;
ancestors the set of terms belonging to all the paths starting from a term to the root, $ancestors(t_v)$ returns the set of ancestors of t_v;
descendants the set of terms belonging to the paths in the transposed graph H^T starting from a term to the leaves.

Let $\mathbf{X}$ be a set of i.i.d. samples in $\mathbb{R}^d$ drawn from an unknown distribution, and $\mathbf{Y}$ the set of the assignments $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ of an unknown labelling function $y : \mathbf{X} \to \mathcal{P}(\mathcal{T})$[1], namely $\mathbf{y}_i = y(\mathbf{x}_i)$. The function y is assumed to be consistent with the TPR (formalized in the next paragraph). Let $\mathcal{D}$ be the dataset $\mathcal{D} = \{\langle \mathbf{x}_1, \mathbf{y}_1 \rangle, \ldots, \langle \mathbf{x}_n, \mathbf{y}_n \rangle\}$ where $\mathbf{x}_i \in \mathbf{X}$, and $\mathbf{y}_i \in \mathbf{Y}$. For convenience the labels $\mathbf{y}_i$ assigned to the sample $\mathbf{x}_i$ are expressed as a vector in $\{0,1\}^m$ such that the j-th element of $\mathbf{y}$ is 1 iff $t_j \in y(\mathbf{x}_i)$. The hierarchical multilabel classification can be defined as the task of finding an estimator $\hat{y} : \mathbf{X} \to \{0,1\}^m$ of the unknown labelling function. The quality of the estimator can be assessed with a loss function $L : \mathcal{P}(\mathcal{T}) \times \mathcal{P}(\mathcal{T}) \to \mathbb{R}$, whose minimization is often the objective in the learning process.

2.1 True path rule

The TPR plays a crucial role in the hierarchical classification task, imposing a consistency over the predictions. The definition introduced in Section 1 can now be formalized within our framework. The *ancestors* function, that returns the terms belonging to all the paths starting from a node up to the root, can be computed by

$$ancestors(t_u) = \begin{cases} \left(\bigcup_{t_k \in par(t_u)} anc(t_k) \right) \cup par(t_u) & \text{if } par(t_u) \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases} \quad (1)$$

where *anc* and *par* are shorthand abbreviations for the *parents* and *ancestors* functions.

Definition 1. *The labelling function y observes the TPR iff*

$$\forall t_u \in \mathcal{T}, t_u \in y(\mathbf{x}_i) \implies ancestors(t_u) \subset y(\mathbf{x}_i).$$

[1] $\mathcal{P}(\cdot)$ is the power set of a given set.

Generalized TPR The above definition holds for binary annotations, but, as we will see in Section 2.2, hierarchical classifiers often do not set thresholds and predictions are evaluated based only on the output scores order. We introduce here a generalized notion of TPR, namely the generalized TPR (gTPR), that expands the TPR to this setting. Intuitively it can be defined by imposing a partial order over the DAG of predictions' scores. In sthis way the TPR is respected for each global threshold.

Definition 2. *The gTPR is respected iff* $\forall \langle t_u, t_v \rangle \in E$ *is true that* $\hat{y}_v \geqslant \hat{y}_u$.

This means that for each each couple of terms in a *is_a* relation, the prediction scores for the parent term must be grater or equal to the one of the child. In the extreme case of predictions that have only binary values, the gTPR clearly coincide with the TPR.

2.2 Evaluation metrics

Multilabel classification requires a dedicated class of metrics for performance evaluation. Zang *et al.* [22] reports an exhaustive set of those metrics highlighting properties and use cases. We report here the definitions of the metrics required to evaluate AWX and compare it with the state-of-the-art.

Selecting optimal thresholds in the setting of HMC is not trivial, due to the natural unbalance of the classes. Indeed, by the TPR, classes that lay in the upper part of the hierarchy will have more annotated examples with respect to one on the leaves. Metrics that do not set thresholds, such as the area under the curve (AUC), are therefore very often used. In particular we will use the area under the precision recall curve, with three different averaging variants, each one highlighting different aspects of the methods.

The micro-averaged area under the precision recall curve ($\mathrm{AUC}(\overline{\mathrm{PR}})$) computes the area under a single curve, obtained computing the micro-averaged precision and recall of the m classes

$$\begin{aligned} \overline{Prec} &= \frac{\sum_i^m TP_i}{\sum_i^m TP_i + \sum_i^m FP_i} \\ \overline{Rec} &= \frac{\sum_i^m TP_i}{\sum_i^m TP_i + \sum_i^m FN_i} \end{aligned} \tag{2}$$

where TP_i, FP_i and FN_i are respectively the number of true positives, the false positives and the false negatives of the i-th term. It gives a global snapshot of the prediction quality but is not sensitive to the size of the classes.

To take more into account the classes with fewer examples, we use also macro-averaged ($\overline{\mathrm{AUCPR}}$) and weighted ($\overline{\mathrm{AUCPR}}_w$) area under the precision recall curve. Both compute AUCPR_i for each class $i \in \{1, \ldots, m\}$, which are then

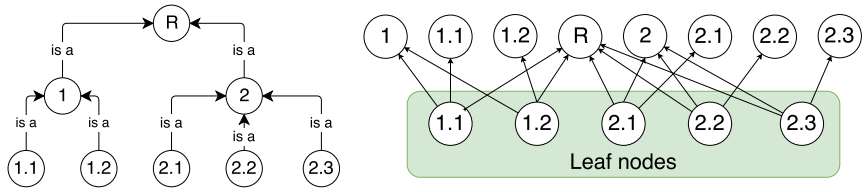

Fig. 1: **a**: Hierarchical tree structure. Sub-tree of the FunCat [13] annotation tree. **b**: Adjacency scheme described by E' starting from the adjacent tree.

averaged uniformly by the former and proportionally by the latter.

$$\begin{aligned} \overline{\text{AUCPR}} &= \frac{1}{m}\sum_{i}^{m}\text{AUCPR}_i \\ \overline{\text{AUCPR}}_w &= \sum_{i}^{m} w_i \cdot \text{AUCPR}_i \end{aligned} \tag{3}$$

where $w_i = v_i / \sum_j^m v_j$ with v_i the frequency of the i-th class in the dataset.

3 Model description

This section describes the AWX hierarchical output layer, that we propose in this paper. Consider an artificial neural network with L hidden layers and the DAG representing the hierarchy $H = (\mathcal{T}, E)$. Let

$$E' = \{\langle t_u, t_v\rangle | t_u \in \mathcal{F}, t_u = t_v \vee t_v \in \mathit{ancestors}(t_u)\}.$$

Note that for each $\langle t_u, t_v\rangle \in E'$ holds that $t_u \in \mathcal{F}$. Let $\mathbf{R}$ be a $|\mathcal{F}| \times m$ matrix that represents the information in E', where $r_{i,j} = 1$ iff $\langle t_i, t_j\rangle \in E'$ and 0 otherwise. Fig. 1b shows an example of the topology described by E'.

Now, let $\mathbf{y}^L$, $\mathbf{W}^L$ and $\mathbf{b}$ denote respectively the output, the weight matrix and the bias vector of the last hidden layer in the network and $\mathbf{r}_i$ the i-th column vector of $\mathbf{R}$. The AWX hierarchical layer is then described by the following equation

$$\begin{aligned} \mathbf{z} &= \mathbf{W}^L \cdot \mathbf{y}^L + \mathbf{b}^L \ , \\ \hat{y}_i &= \texttt{max}(\mathbf{r}_i \circ (f(z_1), \ldots, f(z_{|\mathcal{F}|}))^T) \end{aligned} \tag{4}$$

where $\circ$ is the symbol of the Hadamard product, `max` is the function returning the maximum component of a vector, and f is an activation function $f : \mathbb{R} \to [0, 1]$ (e.g. the sigmoid function). This constraint on the activation function is required to guarantee the consistency of the predicted hierarchy as we will show in Section 3.1.

Being $\mathbf{R}$ binary by definition, the Hadamard product in Equation 4, acts as a mask, selecting the entries of $\mathbf{z}$ corresponding to the non-zero elements of $\mathbf{r}_i$.

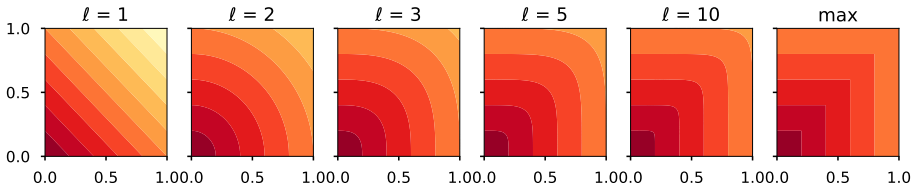

Fig. 2: Shape of the function $z = ||(x, y)^T||_\ell$ at different values of l and the comparison with the `max` function. Darker colors corresponds to lower values of z, while brighter ones to higher.

The `max` represents a straightforward way of propagating the predictions through the hierarchy, but it complicates the learning process. Indeed, the error can be back-propagated only through the maximum component of $\mathbf{r}_i$, leading to local minima. The `max` function can be approximated by the ℓ-norm of the incoming signals as follows

$$\hat{y}_i = \begin{cases} ||\mathbf{r}_i \circ f(\mathbf{z})||_\ell, & \text{if } < 1 \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

The higher ℓ, the more similar the results will be to the ones obtained by the `max`, the closer is ℓ to 1 the more each component of the vector contributes to the result. Figure 2 shows a two-dimensional example, and we can see that with $\ell = 5$ the norm is already a good approximation of the `max`. On the other hand, we can notice that, even if the input is in $[0, 1]$, the output exceeds the range and must therefore be clipped to 1. Especially with ℓ close to 1, the ℓ-norm diverges from the `max`, giving output values that can be much higher than the single components of the input vector.

3.1 The gTPR holds for AWX

In this section we prove the consistency of the AWX output layer with respect to the gTPR, introduced in Section 2.1.

We want to show that $\forall < t_u, t_v > \in E, \hat{y}_v \geqslant \hat{y}_u$ holds for $\hat{y}_v, \hat{y}_u$ in Equation 4.

Proof. Note that Eq. 4 can be rewritten as $\hat{y}_v = max(\mathcal{C}_v)$, where

$$\mathcal{C}_v = \{f(z_u) | < t_u, t_v > \in E'\}$$

is the set of the contributions to the predictions coming from the leaf terms. In the special case of leaf terms, $t_u \in \mathcal{F}$, by construction, $\mathcal{C}_u = \{f(z_u)\}$ therefore $\hat{y}_u = f(z_u)$. We can express the statement of the thesis as $\forall < t_u, t_v > \in E$,

$$\hat{y}_v = max(\mathcal{C}_v) \geqslant max(\mathcal{C}_u) = \hat{y}_u \tag{6}$$

Being the *max* function monotonic, the above inequality holds if $C_u \subseteq C_v$.

Consider the base case $\langle t_u, t_v \rangle \in E$ such that $t_u \in \mathcal{F}$. It clearly holds that $\mathcal{C}_u = \{f(z_u)\} \subseteq \mathcal{C}_v$, because if $\langle t_u, t_v \rangle \in E$ then $\langle t_u, t_v \rangle \in E'$ and therefore $f(z_u) \in \mathcal{C}_v$.

Now consider two generic terms in a "*is_a*" relation $\langle t_u, t_v \rangle \in E$ and their contributions sets $\mathcal{C}_u$ and $\mathcal{C}_v$. By design

$$\forall\ t_k \in \mathcal{F}, \langle t_k, t_u \rangle \in E' \implies \langle t_k, t_v \rangle \in E'$$

and therefore

$$\{f(z_k) | \langle t_k, t_u \rangle \in E'\} \subseteq \{f(z_k) | \langle t_k, t_v \rangle \in E'\}$$
$$\mathcal{C}_u \subseteq \mathcal{C}_v,$$

and Equation 6 holds. ■

The reasoning proceeds similarly for the estimator $\hat{y}_i$ in Equation 5, but in order to guarantee the consistency the input must be in $[0, +\infty)$ since the ℓ-norm is monotonic only in that interval.

3.2 Implementation

The model has been implemented within the Keras [3] framework and a public version of the code is available at `https://github.com/lucamasera/AWX`. The choice of Keras was driven by the will of integrating AWX into deep-learning architectures, and at the time of writing Keras represents a widely-used framework in the area.

An important aspect to consider is that AWX is independent from the underlying network, and can therefore been applied to any ANN that requires a consistent hierarchical output.

4 Experimental setting

In order to assess the performance of the proposed model, an extensive comparison was performed on the standard benchmark datasets[2]. The datasets cover different experiments [21] conducted on *S. cerevisiae* annotated with two ontologies, i.e. FunCat and GO. For each dataset are provided the train, the validation and the test splits of size respectively of *circa* 1600, 800, and 1200. The only exception is the Eisen dataset where, the examples per split are *circa* 1000, 500, and 800. FunCat is structured as a tree with almost 500 term, while GO is composed by three DAGs, comprising more then 4000 terms. The details of the datasets are reported in Table 1 while Figure 2 reports the distribution of terms and leaves per level. Despite having many more terms and being deeper, most of the GO terms lay above the sixth level, depicting an highly unbalance structure with just few branches reaching the deepest levels.

AWX has been tested both with the formulation in Eq. 4 (AWX_{MAX}) and with the one of Eq. 5 ($\text{AWX}_{\ell=k}$ with $k \in \{1, 2, 3\}$). The overall scheme of the

[2] `https://dtai.cs.kuleuven.be/clus/hmcdatasets/`

Dataset	d	FunCat		GO	
		$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{F}\rvert$
Cellcycle	77	499	324	4122	2041
Church	27	499	324	4122	2041
Derisi	63	499	324	4116	2037
Eisen	79	461	296	3570	1707
Expr	551	499	324	4128	2043
Gasch1	173	499	324	4122	2041
Gasch2	52	499	324	4128	2043
Hom	47034	499	324	4128	2043
Seq	478	499	324	4130	2044
Spo	80	499	296	4116	2037

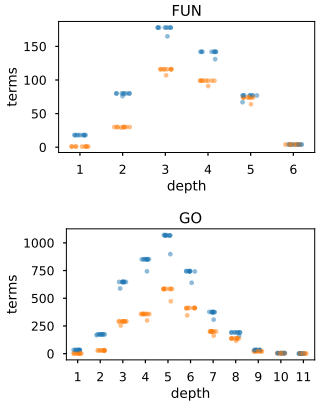

Table 1 & Fig. 2: The table reports the details of the benchmark datasets used in the experiments. d, $|\mathcal{T}|$ and $|\mathcal{F}|$ are respectively the dimensionality of the dataset, the number of terms and how many of them are leaves. The figures show the distribution of terms and leaves by level. Each dataset has a blue marker for the number of terms and an orange one for the number of leaves.

Table 2: ANN architecture used in the experiments.

`fully_connected(size=1000, activation = relu, l2 =` 10^{-3}`)`
`awx(activation = sigmoid, l2 =` 10^{-3}`)`

network employed in the experiments is reported in Table 2 and consists in just an hidden layer with 1000 units and AWX as output layer. The model has been trained with the ADAM optimizer algorithm [8] ($lr = 10^{-5}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$) and loss function

$$L = \frac{1}{N}\sum_{i}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$$

for a maximum of 1000 epochs. An early stopping criterion with zero patience has been set on the validation set, such that, as soon as the performance on the validation test degrades, the learning is stopped.

The results of the four tested variants ($\text{AWX}_{\ell=1}$, $\text{AWX}_{\ell=2}$, $\text{AWX}_{\ell=3}$, AWX_{MAX}) have been evaluated with the metrics shown in Section 2.2 and compared against three other methods.

1. **HMC-clus** [21]: state-of-the-art model based on decision trees. The results reported are taken from the original work.
2. **HMC-LMLP** [2,1]: The model is trained level by level, enriching the input space with the output of the previous level. As for HMC-clus, the reported results are taken from the original paper.

Table 3: AUC($\overline{\text{PR}}$). Bold values show the best preforming method on the dataset. The standard deviation of the computed methods is in the range $[0.001, 0.005]$ and $[0.002, 0.008]$ respectively for FunCat and GO, with the only exception of Hom, where is an order of magnitude bigger.

		CLUS-HMC	HMC-LMLP	MLP$_{leaves}$	AWX$_{\ell=1}$	AWX$_{\ell=2}$	AWX$_{\ell=3}$	AWX$_{\text{MAX}}$
FunCat	Cellcycle	0.172	0.185	0.148*	**0.205***	0.189*	0.181*	0.174
	Church	0.170	0.164	0.102*	**0.173**	0.150*	0.136*	0.127*
	Derisi	**0.175**	0.170	0.112*	**0.175**	0.152*	0.142*	0.136*
	Eisen	0.204	0.208	0.196*	**0.252***	0.243*	0.234*	0.225*
	Expr	0.210	0.196	0.201	**0.262***	0.236*	0.229*	0.223*
	Gasch1	0.205	0.196	0.182	**0.238***	0.227*	0.217*	0.209
	Gasch2	0.195	0.184	0.150*	**0.211***	0.195	0.186	0.178*
	Hom	**0.254**	0.192	0.100*	0.107*	0.109*	0.106*	0.127*
	Seq	0.211	0.195	0.188*	**0.253***	0.234*	0.227*	0.218
	Spo	**0.186**	0.172	0.117*	0.179	0.159*	0.150*	0.143*
GO	Cellcycle	0.357	0.365	0.315*	**0.441***	0.406*	0.385*	0.362
	Church	0.348	0.347	0.272*	**0.440***	0.378*	0.355*	0.329
	Derisi	0.355	0.349	0.274*	**0.424***	0.376*	0.352	0.335*
	Eisen	0.380	0.403	0.347*	**0.481***	0.449*	0.426*	0.410*
	Expr	0.368	0.384	0.357*	**0.480***	0.437*	0.418*	0.407*
	Gasch1	0.371	0.384	0.346*	**0.468***	0.437*	0.416*	0.401*
	Gasch2	0.365	0.369	0.328*	**0.454***	0.417*	0.394*	0.379
	Hom	**0.401**		0.203*	0.264*	0.256*	0.242*	0.238*
	Seq	0.386	0.384	0.347*	**0.472***	0.429*	0.412*	0.397*
	Spo	0.352	0.345	0.278*	**0.420***	0.378*	0.355	0.336

3. **MLP**$_{leaves}$: ANN trained only on the leaf terms with the same parameters of AWX. The prediction for the non-leaf terms are obtained by taking the `max` of the predictions for underlying terms in the hierarchy.

The comparison with MLP$_{leaves}$ is crucial, because it highlights the impact of jointly learning the whole hierarchy with respect to inferring the prediction after the learning phase.

Both AWX and MLP$_{leaves}$ are based on ANNs, so, in order to mitigate the effect of the random initialization of the weight matrix, the learning process has been repeated 10 times. We report the average results of the 10 iterations and the standard deviation ranges are reported in the caption of the tables. We performed a t-test with $\alpha = 0.05$ to assess the significativity of the difference with respect to the state-of-the-art and marked with * the results that passed the test.

No parameter tuning has been performed for the trained methods and the validation split has been used only for the early stopping criterion.

5 Results

In this section are reported and discussed the results obtained by the proposed method on ten benchmark datasets. Besides the comparison with the state-of-the-art, we will show the impact of AWX highlighting the differences with respect to MLP$_{leaves}$.

Table 4: $\overline{\text{AUCPR}}$. Bold values show the best preforming method on the dataset. HMC-LMLP provides no results for this metric, so it has been removed from the table. The standard deviation of the computed methods is in the range $[0.001, 0.005]$ for both FunCat and GO, with the only exception of Hom, where it is an order of magnitude bigger.

		CLUS-HMC	MLP_{leaves}	$\text{AWX}_{\ell=1}$	$\text{AWX}_{\ell=2}$	$\text{AWX}_{\ell=3}$	AWX_{MAX}
FunCat	Cellcycle	0.034	0.068*	0.075*	0.076*	0.077*	0.076*
	Church	0.029	0.040*	0.040*	**0.041***	0.040*	0.041*
	Derisi	0.033	0.047*	0.047*	0.048*	**0.049***	0.048*
	Eisen	0.052	0.095*	0.103*	0.104*	**0.106***	**0.106***
	Expr	0.052	0.114*	0.120*	**0.121***	**0.121***	0.120*
	Gasch1	0.049	0.101*	0.108*	0.110*	**0.111***	0.109*
	Gasch2	0.039	0.069*	0.080*	**0.078***	0.077*	0.078*
	Hom	0.089	0.116	0.095	0.086	0.112	**0.164**
	Seq	0.053	0.126*	0.121*	**0.126***	**0.126***	0.124*
	Spo	0.035	0.043*	0.045*	0.045*	0.045*	0.045*
GO	Cellcycle	0.021	0.057*	0.059*	0.057*	0.057*	0.057*
	Church	0.018	0.034*	0.030*	**0.032***	0.031*	0.031*
	Derisi	0.019	0.038*	0.041*	**0.040***	**0.040***	0.039*
	Eisen	0.036	0.082*	**0.091***	0.089*	0.088*	0.088*
	Expr	0.029	0.092*	**0.104***	0.102*	**0.104***	0.102*
	Gasch1	0.030	0.076*	0.086*	0.086*	0.086*	0.085*
	Gasch2	0.024	0.065*	0.067*	0.066*	0.067*	0.066*
	Hom	0.051	**0.071**	0.042	0.046	0.042	0.053
	Seq	0.036	**0.130***	**0.130***	**0.130***	0.128*	0.128*
	Spo	0.026	0.037*	0.038*	0.039*	**0.040***	0.038*

Table 3 reports the micro-averaged area under the precision recall curve ($\text{AUC}(\overline{\text{PR}})$). $\text{AWX}_{\ell=1}$ has a clear edge over the competitors, in both the ontologies. With the FunCat annotation, it is significantly better then CLUS-HMC six out of ten times and worse just in the Hom dataset, while with GO it wins nine out of ten times. $\text{AWX}_{\ell=1}$ clearly outperforms also the other AWX versions and MLP_{leaves} in all the datasets. We can notice that the performance tends to decrease for higher values of ℓ, and reaches the minimum with AWX_{MAX}. This can be explained by the distribution of the example-annotation: due to the TPR, the terms placed close to the root are more likely to be associated with more examples then the lower terms. With $\ell = 1$ each leaf, among the descendants, contributes equally to the prediction, boosting the prediction values of the upper terms.

Of great interest is the comparison between MLP_{leaves} and AWX_{MAX}. We can notice that AWX_{MAX}, despite being the worst performing version of AWX, always outperforms MLP_{leaves}. Remember that the main difference between the two architectures is that with AWX_{MAX} prediction are propagated at learning time and consequently optimized, while in MLP_{leaves} the predictions of the non-leaf terms are inferred offline.

Table 4 reports the macro-averaged area under the precision recall curves ($\overline{\text{AUCPR}}$). Unfortunately HMC-LMLP does not provide the score for this metric, but AWX clearly outperforms CLUS-HMC, both with the FunCat and with the GO annotations. We can notice that the differences are significant in all the

Table 5: $\overline{\text{AUCPR}}_w$. Bold values show the best preforming method on the dataset. HMC-LMLP provides no data for the GO datasets. The standard deviation of the computed methods is in the range $[0.001, 0.005]$ for both FunCat and GO, with the only exception of Hom, where it is an order of magnitude bigger.

		CLUS-HMC	HMC-LMLP	MLP_{leaves}	$\text{AWX}_{\ell=1}$	$\text{AWX}_{\ell=2}$	$\text{AWX}_{\ell=3}$	AWX_{MAX}
FunCat	Cellcycle	0.142	0.145	0.186*	0.200*	0.204*	**0.205***	0.203*
	Church	0.129	0.118	0.132	**0.139***	**0.139***	0.139*	0.138
	Derisi	0.137	0.127	0.144*	0.150*	**0.152***	**0.152***	0.151*
	Eisen	0.183	0.163	0.229*	0.246*	**0.254***	**0.254***	0.252*
	Expr	0.179	0.167	0.239*	**0.260***	**0.260***	0.258*	0.255*
	Gasch1	0.176	0.157	0.217*	0.234*	**0.241***	0.240*	0.237*
	Gasch2	0.156	0.142	0.183*	0.200*	**0.202***	**0.202***	0.200 *
	Hom	**0.240**	0.159	0.185	0.185	0.188	0.193	0.222
	Seq	0.183	0.112	0.232*	0.260*	**0.263***	0.262*	0.258*
	Spo	0.153	0.129	0.152	0.159	**0.161***	**0.161***	0.160*
GO	Cellcycle	0.335		0.372*	0.379*	0.384*	**0.385***	0.380*
	Church	0.316		0.325*	0.328*	**0.331***	0.329*	0.327*
	Derisi	0.321		0.331*	0.334*	**0.338***	0.337*	0.336*
	Eisen	0.362		0.402*	0.418*	**0.426***	0.423*	0.418*
	Expr	0.353		0.407*	0.424*	**0.430***	**0.430***	0.427*
	Gasch1	0.353		0.397*	0.410*	**0.421***	0.419*	0.416*
	Gasch2	0.347		0.379*	0.386*	**0.394***	0.393*	0.388*
	Hom	**0.389**		0.354*	0.342*	0.349	0.345*	0.356
	Seq	0.373		0.408*	0.431*	**0.436***	0.434*	0.430*
	Spo	0.324		0.333*	0.332	**0.339***	**0.339***	0.338*

datasets, with the exception of Hom, where the variance of the computed results is above 0.04 with FunCat and 0.007 with GO. Within AWX is instead more difficult to identify a version that performs clearly better then the others, their performance are almost indistinguishable. Focusing on the differences between MLP_{leaves} and AWX_{MAX}, we can see that the former is outperformed fourteen out of twenty times by the latter. The advantage of the AWX layer in this case is not as visible as in terms of AUC($\overline{\text{PR}}$), because $\overline{\text{AUCPR}}$ gives equal weight to the curve for each class, ignoring the number of annotated examples. Within our setting, where most of the classes have few examples, this evaluation metric tends to flatten the results, and may not be a good indicator of the performance.

Table 5 reports the weighted-average of the area under the precision recall curves ($\overline{\text{AUCPR}}_w$). AWX has solid performance also considering this evaluation metric, outperforming significantly CLUS-HMC in most of the datasets. HMC-LMLP provides results only for the FunCat-annotated datasets, but appears to be not competitive with our method. Within the proposed variants of AWX, $\ell = 2$ has an edge over $\ell = 1$ and MAX, while is almost indistinguishable from $\ell = 3$. Moreover, comparing AWX_{MAX} and MLP_{leaves}, we can see that the former is systematically better then the latter.

The results reported for AWX (in all its variants) and MLP were not obtained tuning the parameters on the validation sets, but rather setting them *a priori*. The lack of parameter-tuning is clear on the Hom dataset. With all the considered evaluation metrics, this dataset significantly diverges with respect to the others. This behaviour is common also to CLUS-HMC, but unlikely the

proposed methods, it has the best performance on this dataset. An explanation of this anomaly can be found in Table 1, where we can see the difference in the dimensionality. Hom dataset features are indeed two order of magnitude more then the second biggest dataset, i.e. Expr. The sub-optimal choice of the model parameters and the dimensionality could therefore explain the deviation of this dataset from the performance trend, and these aspects should be explored in the future.

AWX has solid overall performance. AUC($\overline{\text{PR}}$) is the most challenging evaluation metric, where the improvement over the state-of-the-art is smaller, while with the last two metrics, AWX appears to have a clear advantage. The choice of the value for ℓ depends on the metric we want to optimize, indeed AWX performs the best with $\ell = 1$ considering AUC($\overline{\text{PR}}$), while if we consider $\overline{\text{AUCPR}}$ or $\overline{\text{AUCPR}}_w$ values of $\ell = 2$ or $\ell = 3$ have an edge over the others. The direct comparison between MLP_{leaves} and AWX_{MAX} is clearly in favour of the latter, which wins almost on all the datasets. This highlights the clear impact of the proposed approach, that allows a jointly optimization of all the classes in the datasets.

6 Conclusion

In this work we have proposed a generalization to the continuous domain of the true path rule and presented a novel ANN layer, named AWX. The aim of this component is to allow the user to compute consistent hierarchical predictions on top of any deep learning architecture. Despite the simplicity of the proposed approach, it appears clear that AWX has an edge over the state-of-the-art method CLUS-HMC. Significant improvements can be seen almost on all the datasets for the macro and weighted averaged area under the precision recall curves evaluation metric, while the advantage in terms of AUC($\overline{\text{PR}}$) is significant in six out of ten datasets.

Part of improvement could be attributed to the power and flexibility of ANN over decision trees, but we have shown that AWX systematically outperforms also HMC-LMLP, based on ANN, and MLP_{leaves}, that has exactly the same architecture as AWX, but with sigmoid output layer just for the leaf terms.

Further work will be focused to test the proposed methods with real-world challenging datasets, integrating AWX in deep learning architectures. Another interesting aspect will be to investigate the performance of AWX with semantic-based metrics, both globally or considering only the leaf terms.

References

1. Ricardo Cerri, Rodrigo C. Barros, and Andre C. P. L. F. de Carvalho. Hierarchical classification of Gene Ontology-based protein functions with neural networks. *2015 Int. Jt. Conf. Neural Networks*, pages 1–8, 2015.
2. Ricardo Cerri, Rodrigo C. Barros, and André C.P.L.F. de Carvalho. Hierarchical multi-label classification using local neural networks. *J. Comput. Syst. Sci.*, 80(1):39–56, 2014.

3. François Chollet et al. Keras, 2015.
4. Gene Ontology Consortium. Creating the gene ontology resource: design and implementation. *Genome Res.*, 11(8):1425–33, 2001.
5. Qingtian Gong, Wei Ning, and Weidong Tian. GoFDR: A Sequence Alignment Based Method for Predicting Protein Functions. *Methods*, 93:3–14, 2015.
6. Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.
7. Yuxiang Jiang et al. An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome biology*, pages 1–17, 2016.
8. Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
9. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
10. Alexey G Murzin, Steven E Brenner, Tim Hubbard, and Cyrus Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536–540, 1995.
11. Guillaume Obozinski, Gert Lanckriet, Charles Grant, Michael I Jordan, and William Stafford Noble. Consisten probabilistic outputs for protein function prediction. *Genome biology*, 9(65):1–19, 2008.
12. Predrag Radivojac, Wyatt T Clark, Tal Ronnen Oron, Alexandra M Schnoes, Tobias Wittkop, Artem Sokolov, Kiley Graim, Christopher Funk, Karin Verspoor, Asa Ben-Hur, Gaurav Pandey, and Jeffrey M Yunes. A large-scale evaluation of computational protein function prediction. *Nature Methods*, 10(3):221–227, 2013.
13. Andreas Ruepp, Alfred Zollner, Dieter Maier, Kaj Albermann, Jean Hani, Martin Mokrejs, Igor Tetko, Ulrich Güldener, Gertrud Mannhaupt, Martin Münsterkötter, and H. Werner Mewes. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research*, 32(18):5539–5545, 2004.
14. Lynn Marie Schriml, Cesar Arze, Suvarna Nadendla, Yu-Wei Wayne Chang, Mark Mazaitis, Victor Felix, Gang Feng, and Warren Alden Kibbe. Disease ontology: a backbone for disease semantic integration. *Nucleic acids research*, 40(D1):D940–D946, 2011.
15. Artem Sokolov and Asa Ben-Hur. Hierarchical classification of gene ontology terms using the gostruct method. *Journal of bioinformatics and computational biology*, 8(02):357–376, 2010.
16. Radu Soricut and Daniel Marcu. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics, 2003.
17. Mohammad S Sorower. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 18, 2010.
18. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

19. Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 521–528. IEEE, 2001.
20. Isaac Triguero and Celine Vens. Labelling strategies for hierarchical multi-label classification techniques. *Pattern Recognit.*, pages 1–14, 2015.
21. Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Mach. Learn.*, 73(2):185–214, 2008.
22. Min Ling Zhang and Zhi Hua Zhou. A review on multi-label learning algorithms. *IEEE Trans. Knowl. Data Eng.*, 26(8):1819–1837, 2014.