Hierarchical Learning of Dependent Concepts for Human Activity Recognition

Aomar Osmani[1], Massinissa Hamidi[1](✉), and Pegah Alizadeh[2]

[1] LIPN-UMR CNRS 7030, Univ. Sorbonne Paris Nord, Villetaneuse, France
[2] Léonard de Vinci Pôle Universitaire, Research Center, 92 916 Paris, La Défense
{ao,hamidi}@lipn.univ-paris13.fr, pegah.alizadeh@devinci.fr

Abstract. In multi-class classification tasks, like human activity recognition, it is often assumed that classes are separable. In real applications, this assumption becomes strong and generates inconsistencies. Besides, the most commonly used approach is to learn classes one-by-one against the others. This computational simplification principle introduces strong inductive biases on the learned theories. In fact, the natural connections among some classes, and not others, deserve to be taken into account. In this paper, we show that the organization of overlapping classes (multiple inheritances) into hierarchies considerably improves classification performances. This is particularly true in the case of activity recognition tasks featured in the SHL dataset. After theoretically showing the exponential complexity of possible class hierarchies, we propose an approach based on transfer affinity among the classes to determine an optimal hierarchy for the learning process. Extensive experiments show improved performances and a reduction in the number of examples needed to learn.

Keywords: Activity recognition · Dependent concepts · Meta-modeling.

1 Introduction

Many real-world applications considered in machine learning exhibit dependencies among the various to-be-learned concepts (or classes) [6,17]. This is particularly the case in human activity recognition from wearable sensor deployments which constitutes the main focus of our paper. This problem is two-folds: the high volume of accumulated data and the criteria selection optimization. For instance, are the criteria used to distinguish between the activities (concepts) *running* and *walking* the same as those used to distinguish between *driving a car* and *being in a bus*? what about distinguishing each individual activity against the remaining ones taken as a whole? Similarly, during the annotation process, when should someone consider that *walking* at a higher pace corresponds actually to *running*? These questions naturally arise in the case of the SHL dataset [7] which exhibits such dependencies. The considered activities in this dataset are difficult to separate due to the existence of many overlaps among certain activities. Some of the important causes for these overlaps are: (1) the on-body sensors deployments featured by this dataset, due to sensors coverage overlaps, tend to

capture movements that are not necessarily related to a unique activity. Authors in [8], for example, have exhibited such overlaps; (2) The difficulty of data annotation during data collection conducted in real-world conditions. For instance, the annotation issues can include the time-shift of a label with respect to the activity [19], as well as wrong or missing labels [13]. Similarly, long lines of research in computer vision [20] and time-series analysis [19,13] raised these issues which hinder the development and large-scale adoption of these applications.

To solve these problems, we propose an original approach for structuring the considered concepts into hierarchies in a way that very similar concepts are grouped together and tackled by specialized classifiers. The idea is that classifications at different levels of the hierarchy may rely on different features, or different combinations of the same features [27]. Indeed, many real-world classification problems are naturally cast as hierarchical classification problems [1,24,25,27]. A work on the semantic relationships among the categories in a hierarchical structure shows that they are usually of the type *generalization-specialization* [27]. In other words, the lower-level categories are supposed to have the same general properties as the higher-level categories plus additional more specific properties. The problem at hand is twice difficult as we have to, first, find the most appropriate hierarchical structure and, second, find optimal learners assigned to the nodes of the hierarchical structure.

We propose a data-driven approach to structure the considered concepts in a bottom-up approach. We start by computing the affinities and dependencies that exist among the concepts and fuse hierarchically the closest concepts with each other. We leverage for this a powerful technique based on transfer which showed interesting empirical properties in various domains [26,14]. Taking a bottom-up approach allows us to leverage learning the complete hierarchy (including the classifiers assigned to each non-leaf node) incrementally by reusing what was learned on the way. Our contributions are as follows: (1) we propose a theoretical calculation for computing the total number of tree hierarchical combinations (the search space for the optimal solution) based on the given number of concepts; (2) we propose an approach based on transfer affinity to determine an optimal organization of the concepts that improves both learning performances and accelerates the learning process; (3) extensive experiments show the effectiveness of organizing the learning process. We noticeably get a substantial improvement of recognition performances over a baseline which uses a flat classification setting; (4) we perform a comprehensive comparative analysis of the various stages of our approach which raises interesting questions about concept dependencies and the required amount of supervision.

2 Problem Statement

In this section, we briefly review the problem of hierarchical structuring of the concepts in terms of formulation and background. We then provide a complexity analysis of the problem size and its search space.

2.1 Problem Formulation and Background

Let $\mathcal{X} \subset \mathbb{R}^n$ be the inputs vector [1] and let $\mathcal{C}$ be the set of atomic concepts (or labels) to learn. The main idea of this paper comes from the fact that the concepts to be learned are not totally independent, thus grouping some concepts to learn them against the others using implicit biases considerably improves the quality of learning for each concept. The main problem is to find the best structure of concepts groups to be learned in order to optimize the learning of each atomic concept. For this we follow the three dimensions setting defined in [10], and we consider: (1) single-label classification as opposed to multi-label classification; (2) the type of hierarchy (or structure) to be trees as opposed to directed acyclic graphs; (3) instances that have to be classified into leafs, i.e. mandatory leaf node prediction [17], as opposed to the setting where instances can be classified into any node of the hierarchy (early stopping).

A tree hierarchy organizes the class labels into a tree-like structure to represent a kind of "IS-A" relationship between labels. Specifically, [10] points out that the properties of the "IS-A" relationship can be described as asymmetry, anti-reflexivity and transitivity [17]. We define a tree as a pair $(\mathcal{C}, \prec)$, where $\mathcal{C}$ is the set of class labels and "$\prec$" denotes the "IS-A" relationship.

Let $\{(x_1, c_1), \ldots, (x_N, c_N)\} \overset{\text{i.i.d.}}{\sim} X, C$ be a set of training examples, where X and C are two random variables taking values in $\mathcal{X} \times \mathcal{C}$, respectively. Each $x_k \in \mathcal{X}$ and each $c_k \in \mathcal{C}$. Our goal is to learn a classification function $f : \mathcal{X} \to \mathcal{C}$ that attains a small classification error. In this paper, we associate each node i with a classifier $\mathcal{M}_i$, and focus on classifiers $f(x)$ that are parameterized by $\mathcal{M}_1, \ldots, \mathcal{M}_m$ through the following recursive procedure [27] (check Fig. 2):

$$f(x) = \begin{cases} \textbf{initialize } i := 0 \\ \textbf{while } (Child(i) \text{ is not empty}) \quad i := \operatorname{argmax}_{j \in Child(i)} \mathcal{M}_j(x) \\ \textbf{return } i \quad \%Child(i) \text{ is the set of children for the node } i \end{cases} \tag{1}$$

In the case of the SHL dataset, for instance, learning *train* and *subway* or *car* and *bus* before learning each concept alone gives better results. As an advantage, considering these classes paired together as opposed to the flat classification setting leads to significant degradation of recognition performances as demonstrated in some works around the SHL dataset [23]. In contrast, organizing the various concepts into a tree-like structure, inspired by domain expertise, demonstrated significant gains in terms of recognition performances in the context of the SHL challenge [12] and activity recognition in general [15,16].

Designing such structures is of utmost importance but hard because it involves optimizing the structure as well as learning the weights of the classifiers attached to the nodes of that structure (see Sec. 2.2). Our goal is then to determine an optimal structure of classes that can facilitate (improve and accelerate) learning of the whole concepts.

[1] In our case, we select several body-motion modalities to be included in our experiments, among the 16 input modalities of the original dataset: *accelerometer*, *gyroscope*, etc. Segmentation and processing details are detailed in experimental part.

2.2 Search Space Size: Complexity Analysis

A naive approach is to generate the lattice structure of concepts groups and to choose the tree hierarchies which give the best accuracy of atomic concepts. In practice, this is not doable because of the exponential (in the number of leaf nodes) number of possible trees. We propose a recurrence relation involving binomial coefficients for calculating the total number of tree hierarchies for K different concepts (class labels).

Example 1. Assume we have 3 various concepts, and we are interested in counting the total number of hierarchies for classifying these concepts. We consider that we have three classes namely c_1, c_2 and c_3, there exist 4 different tree hierarchies for learning the classification problem as following: (1) $(c_1c_2c_3)$ the tree has one level and the learning process takes one step. Three concepts are learned while each concept is learned separately from the others (flat classification), (2) $((c_1c_2)c_3)$ the tree has two levels and the learning process takes two steps: at the first level, it learns two concepts (atomic c_3 and two atomics c_1 and c_2 together). At the second level it learns separately the two joined concepts c_1 and c_2 of the first level, etc and (3) $(c_1(c_2\ c_3))$ and (4) $((c_1c_3)c_2)$.

Theorem 1. *Let $L(K)$ be the total number of trees for the given K number of concepts. The total number of trees for $K+1$ concepts satisfies the following recurrence relation: $L(K+1) = \binom{K}{K-1}L(K)L(1) + 2\sum_{i=0}^{K-2}\binom{K}{i}L(i+1)L(K-i)$. (See Appendix A in the supplementary material for complete proof).*

3 Proposed Approach

Our goals are to: (i) organize the considered concepts into hierarchies such that the learning process accounts for the dependencies existed among these concepts; (ii) characterize optimal classifiers that are associated to each non-leaf node of the hierarchies. Structuring the concepts can be performed using two different approaches: a **top-down** approach where we seek to decompose the learning process; and a **bottom-up** approach where the specialized models are grouped together based on their affinities. Our approach takes the latter direction and constructs hierarchies based on the similarities between concepts. This is because, an hierarchical approach as a bottom-up method is efficient in the case of high volume SHL data-sets. In this section, we detail the different parts of our approach which are illustrated in Fig. 1. In the rest of this section, we introduce the three stages of our approach in detail: *Concept similarity analysis*, *Hierarchy derivation*, and *Hierarchy refinement*.

3.1 Concept Similarity (Affinity) Analysis

In our **bottom-up** approach we leverage transferability and dependency among concepts as a measure of similarity. Besides the nice empirical properties of this measure (explained in the *Properties* paragraph below), the argument behind it

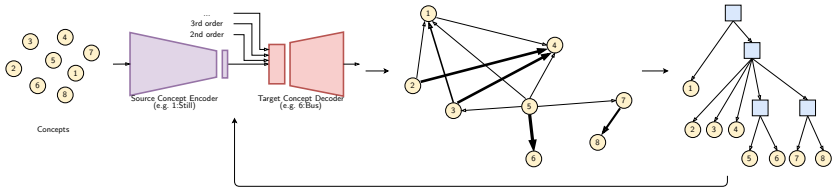


Fig. 1: Our solution involves several repetitions of 3 main steps: (1) Concept similarity analysis: encoders are trained to output, for each source concept, an appropriate representation which is then fine-tuned to serve target concepts. Affinity scores are depicted by the arrows between concepts (the thicker the arrow, the higher the affinity score). (2) Hierarchy derivation: based on the obtained affinity scores, a hierarchy is derived using an agglomerate approach. (3) Hierarchy refinement: each non-leaf node of the derived hierarchy is assigned with a model that encompasses the appropriate representation as well as an ERM which is optimized to separate the considered concepts.

is to reuse what has been learned so far at the lower levels of the hierarchies. Indeed, we leverage the models that we learned during this step and use them with few additional adjustments in the final hierarchical learning setting.

Transfer-based affinity. Given the set of concepts $\mathcal{C}$, we compute during this step an affinity matrix that captures the notion of transferability and similarity, among the concepts. For this, we first compute for each concept $c_i \in \mathcal{C}$ an encoder $f_\theta^{c_i}$ (parameterized by θ) that learns to map the c_i labeled inputs, to $\mathcal{Z}_{c_i}$. Learning the encoder's parameters consists in minimizing the reconstruction error, satisfying the following optimization [22]: $\text{argmin}_{\theta,\theta'} \mathbb{E}_{x,c\sim X,C|c=c_i} \mathcal{L}(g_{\theta'}^{c_i}(f_\theta^{c_i}(x)), x)$, where $g_{\theta'}^{c_i}$ is a decoder (parameterized by θ') which maps back the learned representation into the original inputs space. We propose to leverage the learned encoder, for a given concept c_i, to compute affinities with other concepts via fine-tuning of the learned representation. Precisely, we fine-tune the encoder $f_{c_i}^{\theta}$ to account for a target concept $c_j \in \mathcal{C}$. This process consists, similarly, in minimizing the reconstruction error, however rather than using the decoder $g_{\theta'}^{c_i}$ learned above, we design a genuine decoder $g_{\theta'}^{c_j}$ that we learn from the scratch. The corresponding objective function is $\text{argmin}_{\theta,\theta'} \mathbb{E}_{x,c\sim X,C|c=c_j} \mathcal{L}(g_{\theta'}^{c_j}(f_\theta^{c_i}(x)), x)$. We use the performance of this step as a *similarity score* from c_i to c_j which we denote by $p_{c_i \to c_j} \in [0, 1]$. We refer to the number of examples belonging to the concept c_j used during fine-tuning as the *supervision budget*, denoted as b, which is used to index a given measure of similarity. It allows us to have an additional indicator as to the similarity between the considered concepts. The final similarity score is computed as $\frac{\alpha \cdot p_{c_i \to c_j} + \beta \cdot b}{\alpha + \beta}$. We set α and β to be equal to $\frac{1}{2}$.

Properties. In many applications, e.g. computer-vision [26] and natural language processing [14], several variants of the transfer-based similarity measure have been shown empirically to improve (i) the **quality** of transferred models (wins

against fully supervised models), (ii) the **gains**, i.e. win rate against a network trained from scratch using the same training data as transfer networks', and more importantly (iii) the **universality** of the resulting structure. Indeed, the affinities based on transferability are stable despite the variations of a big corpus of hyperparameters. We provide empirical evidence (Sec. 4.2) of the appropriateness of the transfer-based affinity measure for the separability of the similar concepts and the difficulty to separate concepts that exhibit low similarity scores.

3.2 Hierarchy Derivation

Given the set of *affinity scores* obtained previously, we derive the most appropriate hierarchy, following an agglomerative clustering method combined with some additional constraints. The agglomerative clustering method proceeds by a series of successive fusions of the concepts into groups and results in a structure represented by a two-dimensional diagram known as a dendrogram. It works by (1) forming groups of concepts that are close enough and (2) updating the affinity scores based on the newly formed groups. This process is defined by the recurrence formula proposed by [11]. If defines a distance between a group of concepts (k) and a group formed by fusing i and j groups (ij) as $d_{k(ij)} = \alpha_i d_{ki} + \alpha_j d_{kj} + \beta d_{ij} + \gamma |d_{ki} - d_{kj}|$, where d_{ij} is the distance between two groups i and j. By varying the parameter values $\alpha_i, \alpha_j, \beta$, and γ, we expect to get clustering schemes with various characteristics.

In addition to the above updating process, we propose additional constraints to refine further the hierarchy derivation stage. Given the dendrogram produced by the agglomerative method above, we define an *affinity threshold* τ such that if the distance at a given node is $d_{ij} \geq \tau$, then we merge the nodes to form a unique subtree. In addition, as we keep track of the quantities of data used to train and fine-tune the encoders during the transfer-based affinity analysis stage, this indicator is exploited to inform us as to which nodes to merge. Let $\mathcal{T}$ be the derived hierarchy (tree) and let t indexes the non-leaf or internal nodes. The leafs of the hierarchy correspond to the considered concepts. For any non-leaf node t, we associate a model $\mathcal{M}_t$ that encompasses (1) an encoder (denoted in the following simply by $\mathcal{Z}_t$ in order to focus on the representation) that maps inputs X to representations $\mathcal{Z}_t$ and (2) an ERM (Empirical Risk Minimizer) [21] f_t (such as support vector machines SVMs) that outputs decision boundaries based on the representations produced by the encoder.

3.3 Hierarchy Refinement

After explaining the hierarchy derivation process, we will discuss: (1) which representations are used in each individual model; and (2) how each individual model (including the representation and the ERM weights) is adjusted to account for both the local errors and also those of the hierarchy as a whole.

Which representations to use? The question discussed here is related to the encoders to be used in each non-leaf node. For any non-leaf node t we distinguish

two cases: (i) all its children are leafs; (ii) it has at least one non-leaf node. In the first case, the final considered ERM representation, associated with the non-leaf node, is the representation learned in the concept affinity analysis step (first-order transfer-based affinity). In the second case, we can either fuse the nodes (for example, in a case of classification between 3 concepts, we get all 3 together rather than, first {1} vs. {2,3}, then {2} vs. {3}), or keep them as they are and leverage the affinities based on higher-order transfer where, rather than accounting for a unique target concept, the representation is then fine-tuned. Fig. 2 illustrates how transfers are performed between non-leaf nodes models. We index the models with the encoder $\mathcal{M}_{[\mathcal{Z}_i]}$. In the case of higher-order transfer, the models are indexed using all concepts involved in the transfer, i.e. $\mathcal{M}_{[\mathcal{Z}_{i,j,\ldots}]}$.

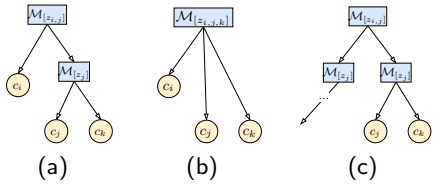


Fig. 2: Transfers are performed between non-leaf nodes models. The hierarchy in (a) can be kept as they are merged to form the hierarchy in (b). (b): a high-order transfer between the concepts c_i, c_j, and c_k is performed. (c): no transfers can be made.

Adjusting models weights. Classifiers are trained to output a hypothesis based on the most appropriate representations learned earlier. Given the encoder (representation) assigned to any non-leaf node t, we select a classifier $\hat{f} := \text{argmin}_{f \in \mathcal{H}} \hat{R}(f, \mathcal{Z}_t)$ where $\hat{R}(f, \mathcal{Z}_t) := \frac{1}{M} \sum_{x,c \sim X,C | c \in Child(t)} \mathbb{E}_{z \sim \mathcal{Z}_t | x}[\mathcal{L}(c, f(z))]$ and $\mathcal{H}$ is the hypothesis space. Models are adjusted to account for local errors as well as for global errors related to the hierarchy as a whole. In the first case, the loss is defined as the traditional hinge loss used in SVMs which is intended to adjust the weights of the classifiers that have only children leaves. In the second case, we use a loss that encourages the models to leverage orthogonal representations (between children and parent nodes) [27].

4 Experiments and Results

Empirical evaluation of our approach are performed on three steps: we evaluate classification performances in the hierarchical setting (Sec. 4.1); then, we evaluate the transfer-based affinity analysis step and the properties related to the separability of the considered concepts (Sec. 4.2); finally, we evaluate the derived hierarchies in terms of stability, performance, and agreement with their coun-

terparts defined by domain experts (Sec. 4.3) [2]. Training details can be found in Appendix B and evaluation metrics are detailed in Appendix C.

SHL dataset [7]. It is a highly versatile and precisely annotated dataset dedicated to mobility-related human activity recognition. In contrast to related representative datasets like [2], the SHL dataset (26.43 GB) provides , simultaneously, multimodal and multilocation locomotion data recorded in real-life settings. Among the 16 modalities of the original dataset, we select the body-motion modalities including: *accelerometer*, *gyroscope*, *magnetometer*, *linear acceleration*, *orientation*, *gravity*, and *ambient pressure*. This makes the data set suitable for a wide range of applications and in particular transportation recognition concerned with this paper. From the 8 primary categories of transportation, we are selected: *1:Still*, *2:Walk*, *3:Run*, *4:Bike*, *5:Car*, *6:Bus*, *7:Train*, and *8:Subway (Tube)*.

4.1 Evaluation of the Hierarchical Classification Performances

In these experiments, we evaluate the flat classification setting using neural networks which constitute our baseline for the rest of the empirical evaluations. To compare our baseline with the hierarchical models, we make sure to get the same complexity, i.e. comparable number of parameters as the largest hierarchies including the weights of the encoders and those of the ERMs. We also use Bayesian optimization based on Gaussian processes as surrogate models to select the optimal hyperparameters of the baseline model [18,9]. More details about the baseline and its hyperparameters are available in the code repository [9].

Per-node performances. Fig. 3 shows the resulting per-node performances, i.e. how accurately the models associated with the non-leaf nodes can predict the correct subcategory averaged over the entire derived hierarchies. The nodes are ranked according to the obtained per-node performance (top 10 nodes are shown) and accompanied by their appearance frequency. It is worth noticing that the concept 1:*still* learned alone against the rest of the concepts (first bar) achieves the highest gains in terms of recognition performances while the appearance frequency of this learning configuration is high (more than 60 times). We see also that the concepts 4:*bike*, 5:*car*, and 6:*bus* grouped together (5th bar) occur very often in the derived hierarchies (80 times) which is accompanied by fairly significant performance gains ($5.09 \pm 0.3\%$). At the same time, as expected, we see that the appearance frequency gets into a plateau starting from the 6th bar (which lasts after the 10th bar). This suggests that the most influential nodes are often exhibited by our approach.

Per-concept performances. We further ensure that the performance improvements we get at the node levels are reflected at the concept level. Experimental results show the recognition performances of each concept, averaged over the

[2] Software package and code to reproduce empirical results are publicly available at https://github.com/sensor-rich/hierarchicalSHL

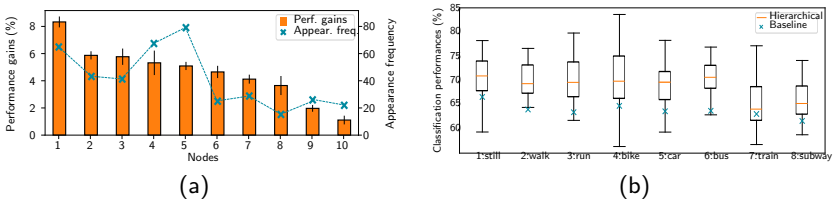


Fig. 3: (a) Per-node performance gains, averaged over the entire derived architectures (similar nodes are grouped and their performances are averaged). The appearance frequency of the nodes is also illustrated. Each bar represents the gained accuracy of each node in our hierarchical approach. For example, the 8th bar corresponds to the concepts 2:*walk*-3:*run*-4:*bike* grouped together. (b) Recognition performances of each individual concept, averaged over the entire derived hierarchies. For reference, the recognition performances of the baseline model are also shown.

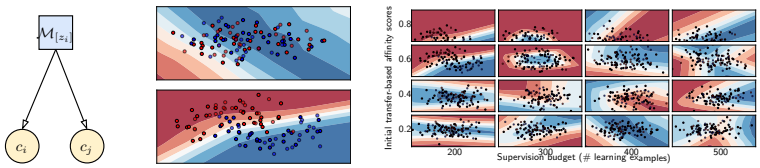
whole hierarchies derived using our proposed approach. We indeed observe that there are significant improvements for each individual concept over the baseline (flat classification setting). We observe that again 1:*still* has the highest classification rate ($72.32 \pm 3.45\%$) and an improvement of 5 points over the baseline. Concept 6:*bus* also exhibits a roughly similar trend. On the other hand, concept 7:*train* has the least gains ($64.43 \pm 4.45\%$) with no significant improvement over the baseline. Concept 8:*subway* exhibits the same behavior suggesting that there are undesirable effects that stem from the definition of these two concepts.

4.2 Evaluation of the Affinity Analysis Stage

These experiments evaluate the proposed transfer-based affinity measure. We assess, the separability of the concepts depending on their similarity score (for both the transfer-affinity and supervision budget) and the learned representation.

Appropriateness of the Transfer-based Affinity Measure. We reviewed above the nice properties of the transfer-based measure especially the universality and stability of the resulting affinity structure. The question that arises is related to the separability of the concepts that are grouped together. Are the obtained representations, are optimal for the final ERMs used for the classification? This is what we investigate here. Fig. 4b shows the decision boundaries generated by the considered ERMs which are provided with the learned representations of two concepts. The first case (top right), exhibits a low-affinity score, and the second case (bottom right) shows a high-affinity score. In the first case, the boundaries are unable to separate the two concepts while it gets a fairly distinct frontier.

Impact on the ERMs' Decision Boundaries. We train different models with various learned representations in order to investigate the effect of the initial affinities (obtained solely with a set of 100 learning examples) and the supervision budget (additional learning examples used to fine-tune the obtained representation) on



(a) (b) (c)

Fig. 4: (a) Non-leaf node grouping concepts c_i and c_j. (b) Decision boundaries generated by the ERM of the non-leaf node using an encoder (representation) fine-tuned to account for (top) the case where c_i and c_j are dissimilar (low-affinity score) and (bottom) the case where c_i and c_j are similar (high-affinity score). (c) Decision boundaries obtained by SVM-based classifiers trained on the representations $\mathcal{Z}_t$ as a function of the distance between the concepts (y-axis) and the supervision budget (x-axis).

the classification performances of the ERMs associated with the non-leaf nodes of our hierarchies. Fig. 4c shows the decision boundaries generated by various models as a function of the distance between the concepts (y-axis) and the supervision budget (x-axis). Increasing the supervision budget to some larger extents (more than ~ 300 examples) results in a substantial decrease in classification performances of the ERMs. This suggests that, although our initial affinity scores are decisive (e.g. 0.8), the supervision budget is tightly linked to generalization. This shows that a trade-off (controlled by the supervision budget) between separability and initial affinities arises when we seek to group concepts together. In other words, the important question is whether to increase the supervision budget indefinitely (in the limits of available learning examples) in order to find the most appropriate concepts to fuse with, while expecting good separability.

4.3 Universality and Stability

We demonstrated in the previous section the appropriateness of the transfer-based affinity measure to provide distance between concepts as well as the existence of a trade-off between concepts separability and their initial affinities. Here we evaluate the **universality** of the derived hierarchies as well as their **stability** during adaptation with respect to our hyperparameters (affinity threshold and supervision budget). We compare the derived hierarchies with their domain experts-defined counterparts, as well as those obtained via a random sampling process. Fig. 5 shows some of the hierarchies defined by the domain experts (first row) and sampled using the random sampling process. For example, the hierarchy depicted in Fig. 5d corresponds to a split between static (1:*still*, 5:*car*, 6:*bus*, 7:*train*, 8:*subway*) and dynamic (2:*walk*, 3:*run*, 4:*bike*) activities. The difference between the hierarchies depicted in Fig. 5a and 5b is related to 4:*bike* activity which is linked first to 2:*walk* and 3:*run* then to 5:*car* and 6:*bus*. A possible interpretation is that in the first case, biking is considered as "on feet" activity while in the second case as "on wheels" activity. What we observed is that the derived hierarchies tend to converge towards the expert-defined ones.

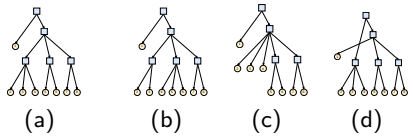


Fig. 5: Examples of hierarchies: (a) defined via domain expertise, (b-c) derived using our approach, and (d) randomly sampled. Concepts 1—8 from left to right.

Method	Agree.	perf. avg.± std.
Expertise	-	72.32±0.17
Random	0.32	48.17±5.76
Proposed	0.77	75.92±1.13

Table 1: Summary of the recognition performances obtained with our proposed approach compared to randomly sampled and expert-defined hierarchies.

We compare the derived hierarchies in terms of their level of agreement. We use for this assessment, the Cohen's kappa coefficient [4] which measures the agreement between two raters. The first column of Table 1 provides the obtained coefficients. We also compare the average recognition performance of the derived hierarchies (second column of Table 1). In terms of stability, as we vary the design choices (hyperparameters), defined in our approach, we found that the affinity threshold has a substantial impact on our results with many adjustments involved (12 hierarchy adjustments on avg.) whereas the supervision budget has a slight effect, which confirms the observations in Sec. 4.2.

5 Conclusion and Future Work

This paper proposes an approach for organizing the learning process of dependent concepts in the case of human activity recognition. We first determine a suitable structure for the concepts according to a transfer affinity-based measure. We then characterize optimal representations and classifiers which are then refined to account for both local and global errors. We provide theoretical bounds for the problem and empirically show that using our approach we are able to improve the performances and robustness of activity recognition models over a flat classification baseline. In addition to supporting the necessity of organizing concepts learning, our experiments raise interesting questions for future work. Noticeably, Sec. 4.2 asks what is the optimal amount of supervision for deriving the hierarchies. Another future work is to study different approaches for searching and exploring the search space of different hierarchical types (lattices, etc.).

References

1. Cai, L., Hofmann, T.: Hierarchical document categorization with support vector machines. In: CIKM. pp. 78–87 (2004)
2. Carpineti, C., et al.: Custom dual transportation mode detection by smartphone devices exploiting sensor diversity. In: PerCom wksh. pp. 367–372. IEEE (2018)
3. Cesa-Bianchi, N., Gentile, C., Zaniboni, L.: Incremental algorithms for hierarchical classification. JMLR **7**(Jan), 31–54 (2006)

4. Cohen, J.: A coefficient of agreement for nominal scales. Educational and psychological measurement **20**(1), 37–46 (1960)
5. Costa, E., Lorena, A., Carvalho, A., Freitas, A.: A review of performance evaluation measures for hierarchical classifiers. In: Evaluation Methods for machine Learning II: papers from the AAAI-2007 Workshop. pp. 1–6 (2007)
6. Essaidi, M., Osmani, A., Rouveirol, C.: Learning dependent-concepts in ilp: Application to model-driven data warehouses. In: ILP, pp. 151–172 (2015)
7. Gjoreski, H., et al.: The university of sussex-huawei locomotion and transportation dataset for multimodal analytics with mobile devices. IEEE Access (2018)
8. Hamidi, M., Osmani, A.: Data generation process modeling for activity recognition. In: ECML-PKDD. Springer (2020)
9. Hamidi, M., Osmani, A., Alizadeh, P.: A multi-view architecture for the shl challenge. In: UbiComp/ISWC Adjunct. p. 317–322 (2020)
10. Kosmopoulos, A., Partalas, I., Gaussier, E., Paliouras, G., Androutsopoulos, I.: Evaluation measures for hierarchical classification: a unified view and novel approaches. Data Mining and Knowledge Discovery **29**(3), 820–865 (2015)
11. Lance, G.N., Williams, W.T.: A general theory of classificatory sorting strategies: 1. hierarchical systems. The computer journal **9**(4), 373–380 (1967)
12. Nakamura, Y., et al.: Multi-stage activity inference for locomotion and transportation analytics of mobile users. In: UbiComp/ISWC. pp. 1579–1588 (2018)
13. Nguyen-Dinh, L.V., Calatroni, A., Tröster, G.: Robust online gesture recognition with crowdsourced annotations. JMLR **15**(1), 3187–3220 (2014)
14. Peters, M.E., Ruder, S., Smith, N.A.: To tune or not to tune? adapting pretrained representations to diverse tasks. arXiv preprint arXiv:1903.05987 (2019)
15. Samie, F., Bauer, L., Henkel, J.: Hierarchical classification for constrained iot devices: A case study on human activity recognition. IEEE IoT Journal (2020)
16. Scheurer, S., et al.: Using domain knowledge for interpretable and competitive multi-class human activity recognition. Sensors **20**(4), 1208 (2020)
17. Silla, C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery **22**(1-2), 31–72 (2011)
18. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: NIPS. pp. 2951–2959 (2012)
19. Stikic, M., Schiele, B.: Activity recognition from sparsely labeled data using multi-instance learning. In: Int. Symposium on LoCA. pp. 156–173. Springer (2009)
20. Taran, V., Gordienko, Y., Rokovyi, A., Alienin, O., Stirenko, S.: Impact of ground truth annotation quality on performance of semantic image segmentation of traffic conditions. In: ICCSEA. pp. 183–193. Springer (2019)
21. Vapnik, V.: Principles of risk minimization for learning theory. In: NIPS (1992)
22. Vincent, P., et al.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR **11**(12) (2010)
23. Wang, L., et al.: Summary of the sussex-huawei locomotion-transportation recognition challenge. In: UbiComp/ISWC. pp. 1521–1530 (2018)
24. Wehrmann, J., Cerri, R., Barros, R.: Hierarchical multi-label classification networks. In: ICML. pp. 5075–5084 (2018)
25. Yao, H., Wei, Y., Huang, J., Li, Z.: Hierarchically structured meta-learning. In: ICML. pp. 7045–7054 (2019)
26. Zamir, A.R., Sax, A., Shen, W., Guibas, L.J., Malik, J., Savarese, S.: Taskonomy: Disentangling task transfer learning. In: CVPR. pp. 3712–3722 (2018)
27. Zhou, D., Xiao, L., Wu, M.: Hierarchical classification via orthogonal transfer. In: ICML. pp. 801–808 (2011)

Appendix A

Proof. Theorem 1. It can be explained by observing that, for $K+1$ concepts containing K existed concepts $c_1, \cdots c_K$ and a new added concept γ, we can produce the first level trees combinations as below. Notice that each atomic element o can be one of the $c_1, \cdots c_K$ concepts. In order to compute the total number of trees combinations, we show what is the number of tree combinations by assigning the K concepts to each item:

- $(\gamma(\overbrace{o\cdots o}^{K\text{concepts}}))$: the number of trees combinations by taking the concept labels into the account are: $\binom{K}{0}L(1) \times 2 \times L(K)$; the reason for multiplying the number of trees combinations for K concepts to 2 is because while the left side contains an atomic γ concept, there are two choices for the right side of the tree in the first level: either we compute the total number of trees for K concepts from the first level or we keep the first level as a $\overbrace{o\cdots o}^{K\text{concepts}}$ atomics and keep all K concepts together, then continue the number of K trees combinations from the second level of the tree.
- $((\gamma o)(\overbrace{o\cdots o}^{K-1\text{concepts}}))$: similar to the previous part we have $\binom{K}{1}L(2)\times 2\times L(K-1)$ trees combinations by taking the concepts labels into the account. $\binom{K}{1}$ indicates the number of combinations for choosing a concept from the K concept and put it with the new concept separately. While $L(2)$ is the number of trees combinations for the left side of tree separated with the new concept γ.
- $((\gamma oo)(\overbrace{o\cdots o}^{K-2\text{concepts}}))$, $\cdots$
- $((\gamma \overbrace{o\cdots o}^{K-1\text{concepts}})o)$: $\binom{K}{K-1}L(K)L(1)$ in this special part, we follow the same formula except the single concept in the right side has only one possible combination in the first level equal to $L(1)$.

All in all, the sum of these items calculates the total number of tree hierarchies for $K+1$ concepts.

The first few number of total number of trees combinations for $1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \cdots$ concepts are: 1 , 1, 4, 26, 236, 2752, 39208, 660032, 12818912, 282137824, $\cdots$. In the case of the SHL dataset that we use in the empirical evaluation, we have 8 different concepts and thus, the number of different types of hierarchies for this case is $L(8) = 660,032$.

Appendix B Training Details

We use Tensorflow for building the encoders/decoders. We construct encoders by stacking Conv1d/ReLU/MaxPool blocks. These blocks are followed by a Fully Connected/ReLU layers. Encoders performance estimation is based on the validation loss and is framed as a sequence classification problem. As a preprocessing

step, annotated input streams from the huge SHL dataset are segmented into sequences of 6000 samples which correspond to a duration of 1 min. given a sampling rate of 100 Hz. For weight optimization, we use stochastic gradient descent with Nesterov momentum of 0.9 and a learning-rate of 0.1 for a minimum of 12 epochs (we stop training if there is no improvement). Weight decay is set to 0.0001. Furthermore, to make the neural networks more stable, we use batch normalization on top of each convolutional layer. We use SVMs as our ERMs in the derived hierarchies.

Appendix C Evaluation Metrics

In hierarchical classification settings, the hierarchical structure is important and should be taken into account during model evaluation [17]. Various measures that account for the hierarchical structure of the learning process have been studied in the literature. They can be categorized into: distance-based; depth-dependent; semantics-based; and hierarchy-based measures. Each one is displaying advantages and disadvantages depending on the characteristics of the considered structure [5]. In our experiments, we use the *H-loss*, a hierarchy-based measure defined in [3]. This measure captures the intuition that *"whenever a classification mistake is made on a node of the taxonomy, then no loss should be charged for any additional mistake occurring in the sub-tree of that node."* $\ell_H(\hat{y}, y) = \sum_{i=1}^{N} \{\hat{y}_i \neq y_i \wedge \hat{y}_j = y_j, j \in Anc(i)\}$, where $\hat{y} = (\hat{y}_1, \cdots \hat{y}_N)$ is the predicted labels, $y = (y_1, \cdots y_N)$ is the true labels, and $Anc(i)$ is the set of ancestors for the node i.