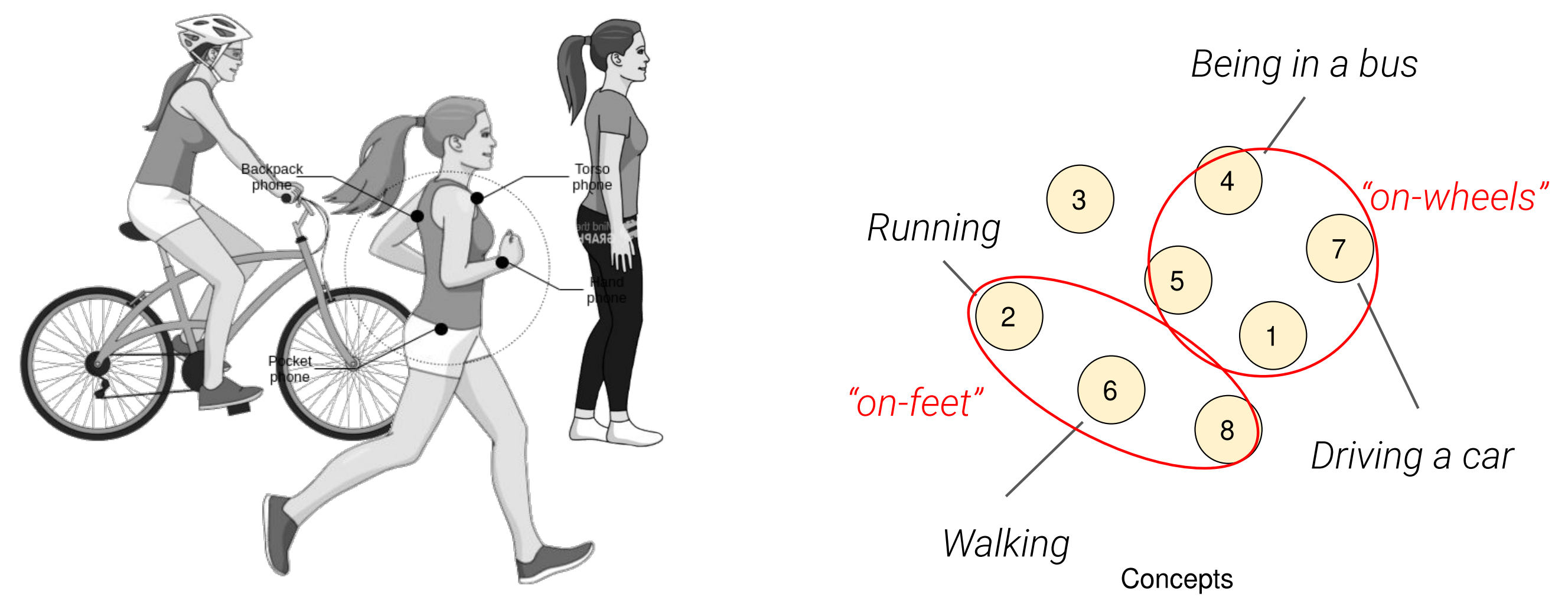
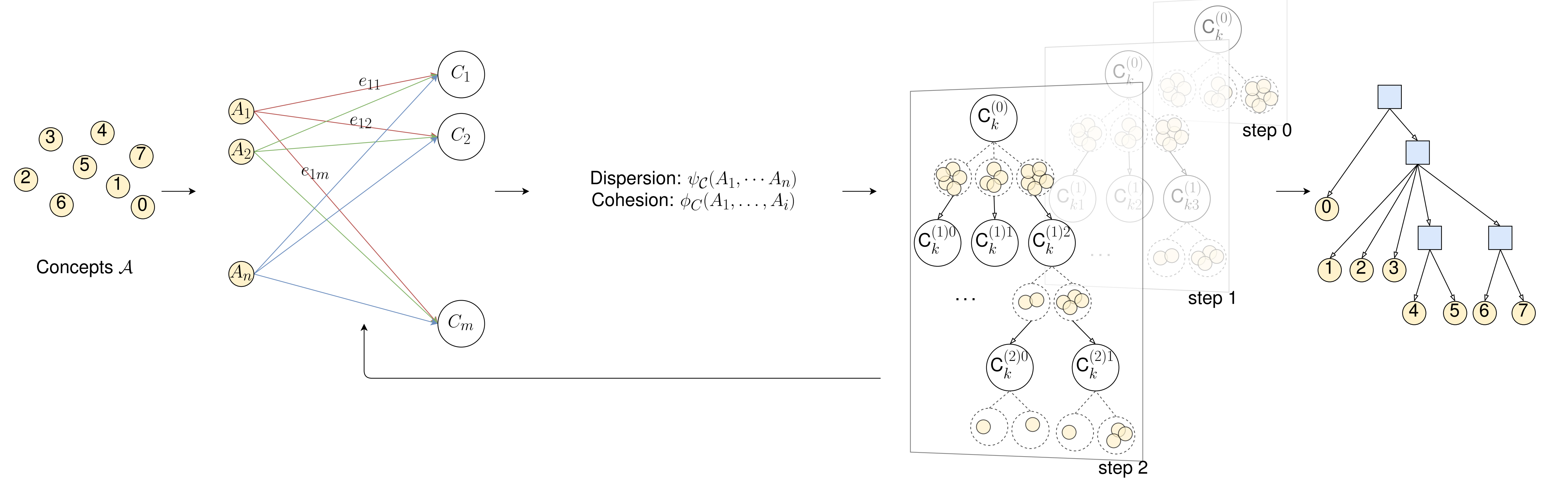
Clustering Approach to Solve Hierarchical Classification Problem Complexity

Aomar OSMANI[1] and Massinissa HAMIDI[1] and Pegah ALIZADEH[2]
[1] LIPN-UMR CNRS 7030, Univ. Sorbonne Paris Nord
[2] De Vinci Research Center, Pôle Universitaire Léonard de Vinci
hamidi@lipn.univ-paris13.fr

In multi-class classification tasks, like human activity recognition, the natural connections among some classes, and not others, deserve to be taken into account. Structuring these classes into groups and learning them using more adapted inductive biases, can considerably improve the quality of the learning process.



1. Concepts Structuring Based on Recursive Clustering

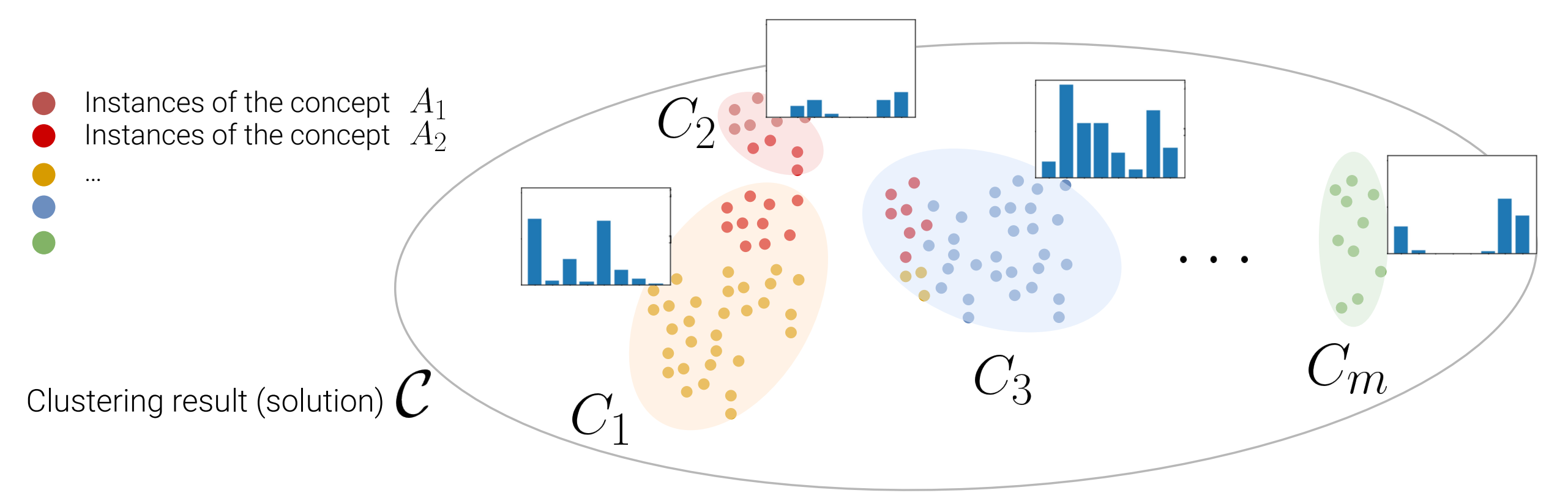

Concepts $\mathcal{A}$
e_{11}
e_{12}
e_{1m}
C_1
C_2
C_m
A_1
A_2
A_n
Dispersion: $\psi_{\mathcal{C}}(A_1, \cdots A_n)$
Cohesion: $\phi_{\mathcal{C}}(A_1, \ldots, A_i)$
step 0
step 1
step 2

Our solution involves several repetitions of 3 main steps:

- It starts by performing clustering on the inputs, in an unsupervised manner and with a varying number of clusters;
- The resulting clustering solutions is used to compute two measures : dispersion and cohesion, which will decide the best clustering to keep;
- The process is repeated recursively on each group of concepts within the selected clustering solution

The process ends as soon as we get individual concepts on the leaves of the decomposition hierarchy.

2. Clustering and Bipartite Graph Representation

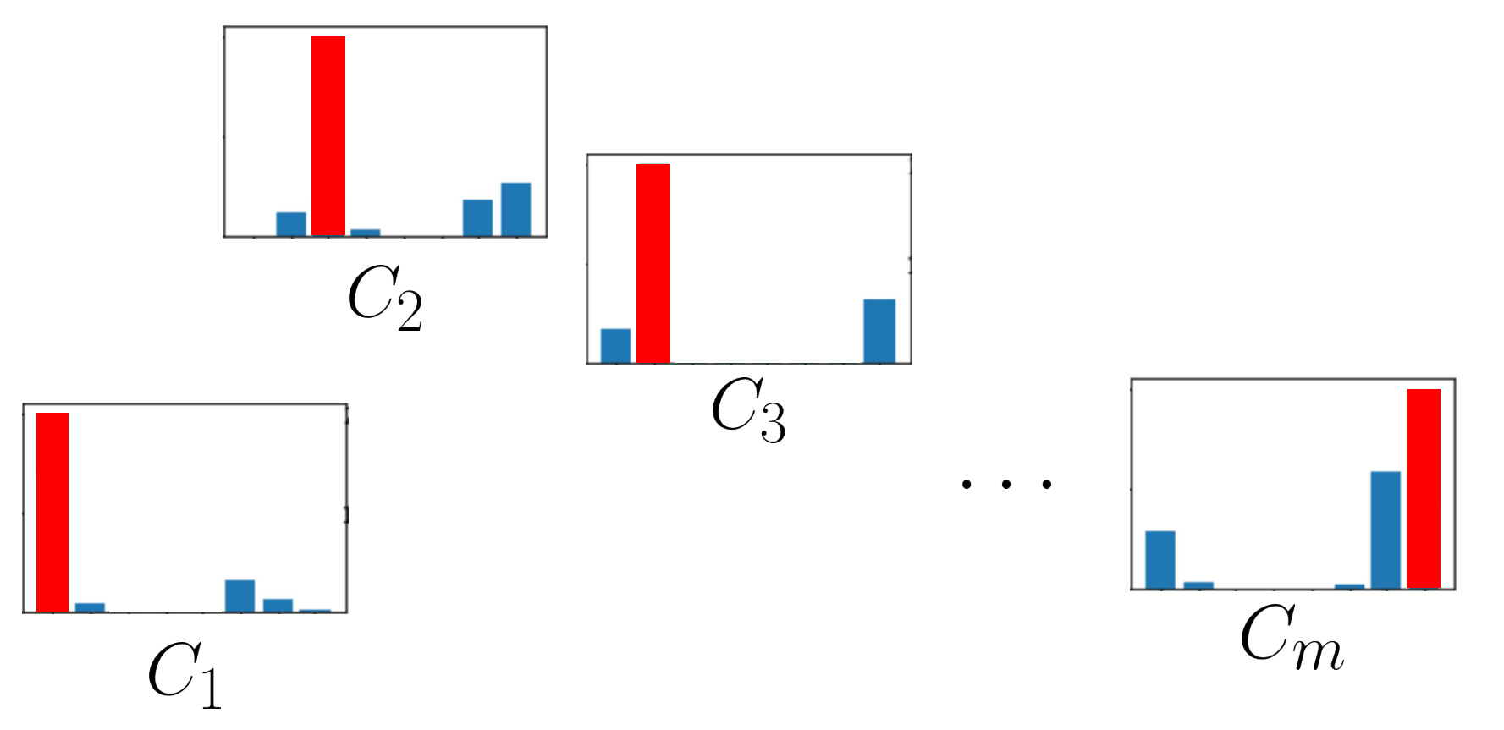

Instances of the concept A_1
Instances of the concept A_2
...
Clustering result (solution) $\mathcal{C}$
C_1
C_2
C_3
C_m

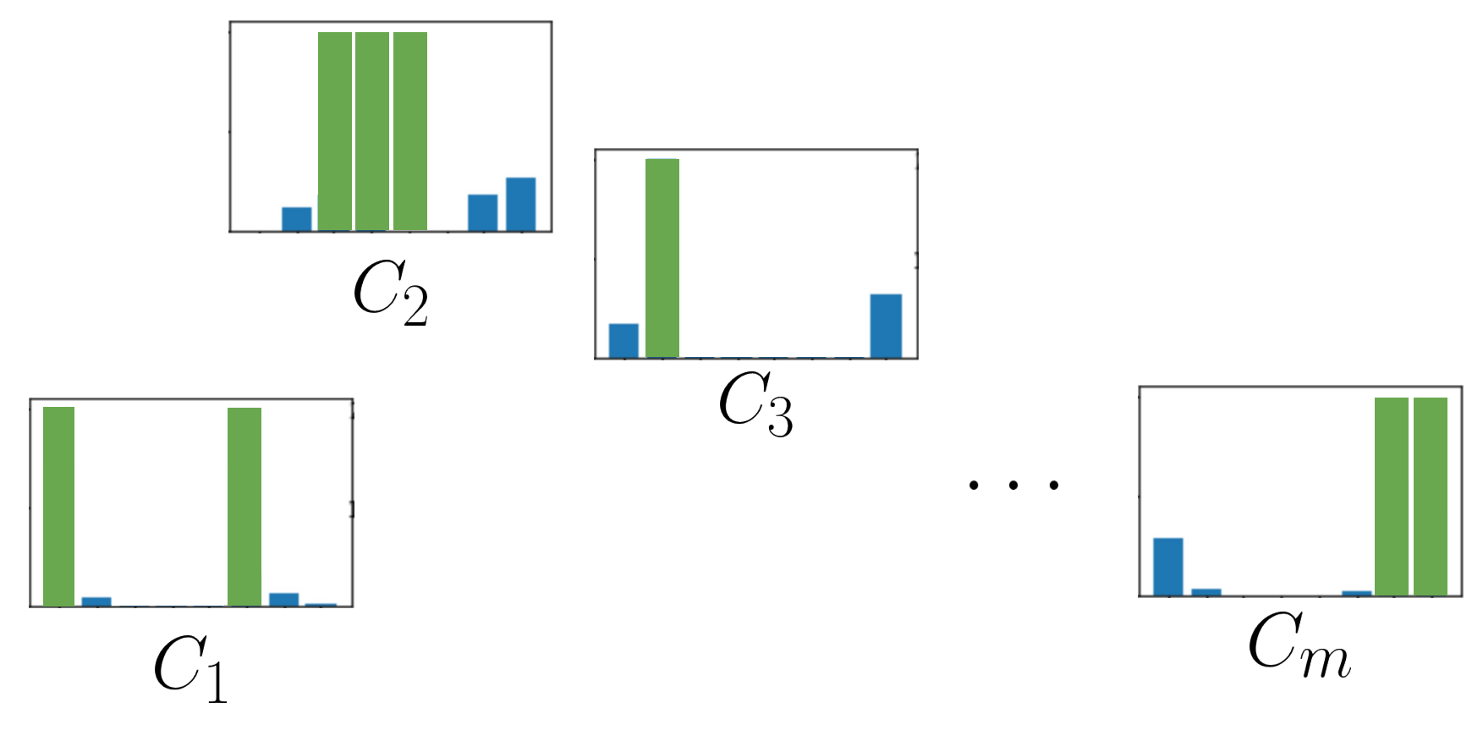
- Let's consider $\mathcal{C} = \{C_1, \ldots, C_m\}$, a clustering result (or solution) obtained in an unsupervised setting (using only the input features X_i of the instances). Instances of the same concept may be grouped in distinct clusters of the clustering solution.
- This clustering result can be represented as $G = (\mathcal{A}, \mathcal{C}, E)$ a bipartite graph whose partition has the parts $\mathcal{A}$ (the classification domain) and $\mathcal{C}$ (the clustering domain), with E denoting the edges of the graph. Each edge $e_{ij} \in E$ represents the percentage of the instances from the input space $\mathcal{X}$ in class A_i, properly covered by the cluster C_j.
- Based on the graph representation, we compute two measures, *dispersion* and *cohesion*, in order to assess the dependencies between clusters and classes.

3. Dispersion and Cohesion Measures

(Dispersion) dispersion measures how instances of a given class are distributed in the different clusters of the clustering solution.

- The lower the dispersion scores are, the better the clustering solution;
- We do not want the instances of a given concept to be spread all over the clusters;





(Cohesion) cohesion measures the co-appearance of classes together in each cluster.

- The higher the co-appearance of two given classes in a given cluster, the higher the cohesion;
- We want to have classes which are close together to be grouped in the clusters.

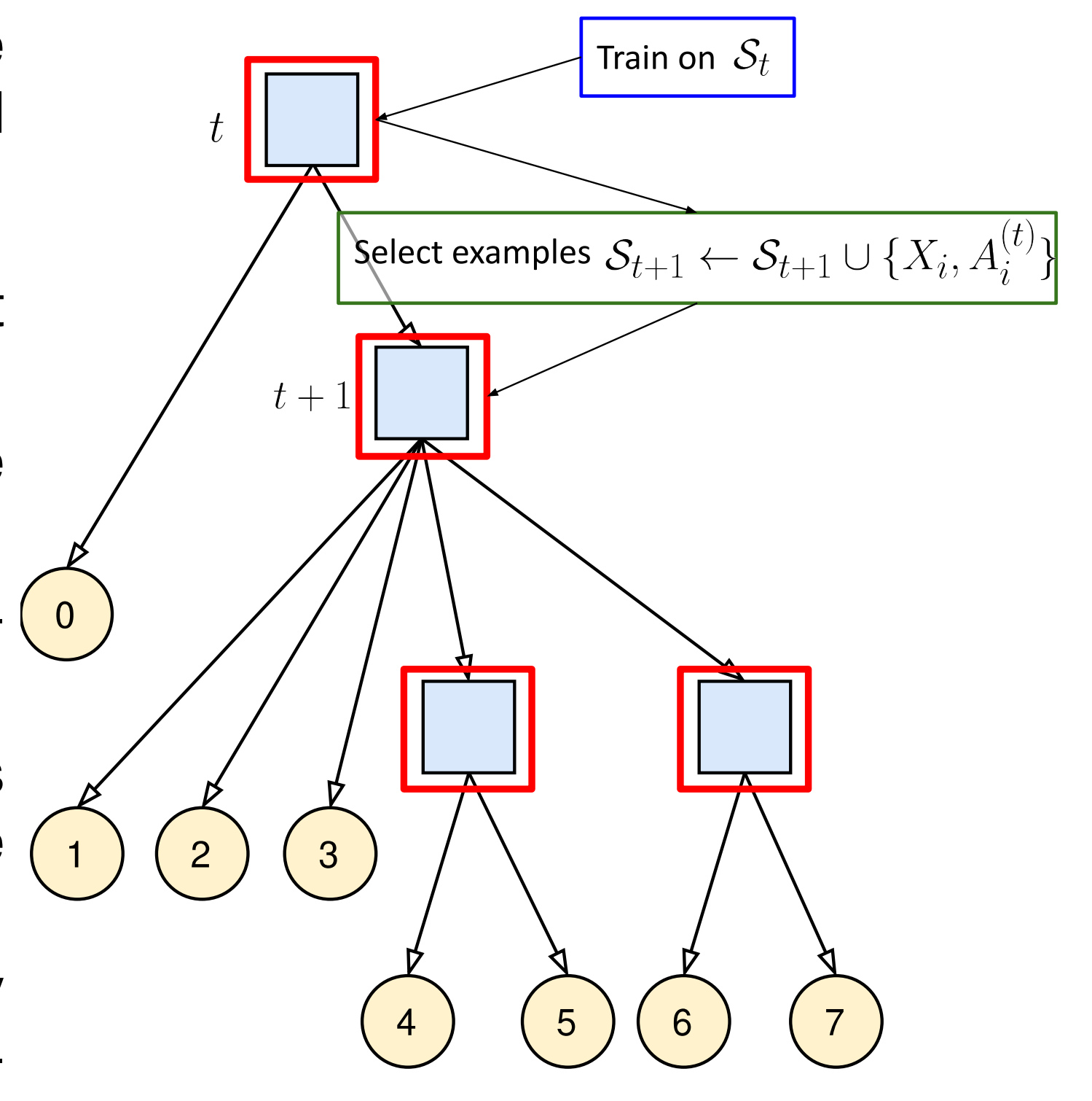
4. Hierarchy Derivation and Training

A non-leaf node t of the derived hierarchy is assigned with a learner trained on $\mathcal{S}_t$ to discriminate between the concepts or groups of concepts found within its descendants.

Leveraging the hierarchical structure for efficient training:

- Evaluating the learners' weights exactly can be prohibitive due to the expensive inner optimization.
- We investigate for this, two strategies that are designed to improve the learning process, namely,

(1) *boosting* strategy: the hard-to-classify examples are weighted so that the learners located in the descendant nodes focus on them;
(2) *student-teacher* strategy: the easy-to-classify examples are selected for training the subsequent learners.

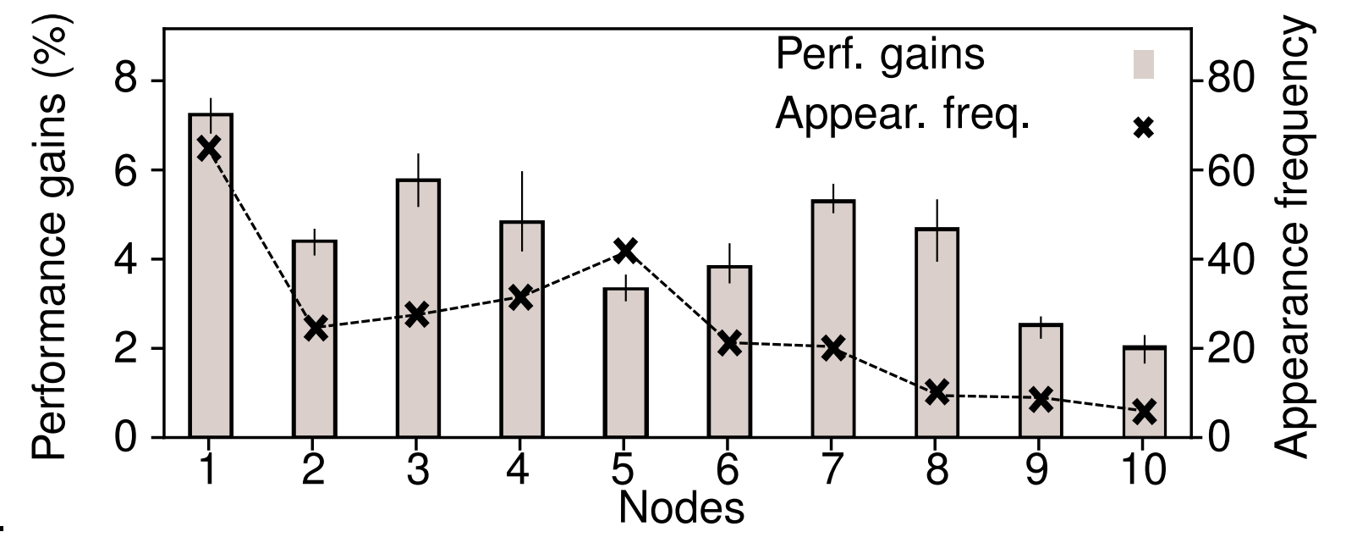

Train on $\mathcal{S}_t$
t
Select examples $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_{t+1} \cup \{X_i, A_i^{(t)}\}$
$t+1$
0
1
2
3
4
5
6
7

5. Performance of the Derived Hierarchies

- **Datasets.** in addition to USC-HAD and HTC-TMD, we use the SHL dataset which consists of motion sensor data. 8 different modes of transportation collected in real-life setting (*0:Still*, *1:Walk*, *2:Run*, *3:Bike*, *4:Car*, *5:Bus*, *6:Train*, and *7:Subway*).

(1) Recognition performances measured via the f1-score obtained with the baseline models on the considered representative datasets;
(2) Resulting per-node performances, illustrating how accurately the learners assigned to the non-leaf nodes can predict the correct groups of concepts associated to them.

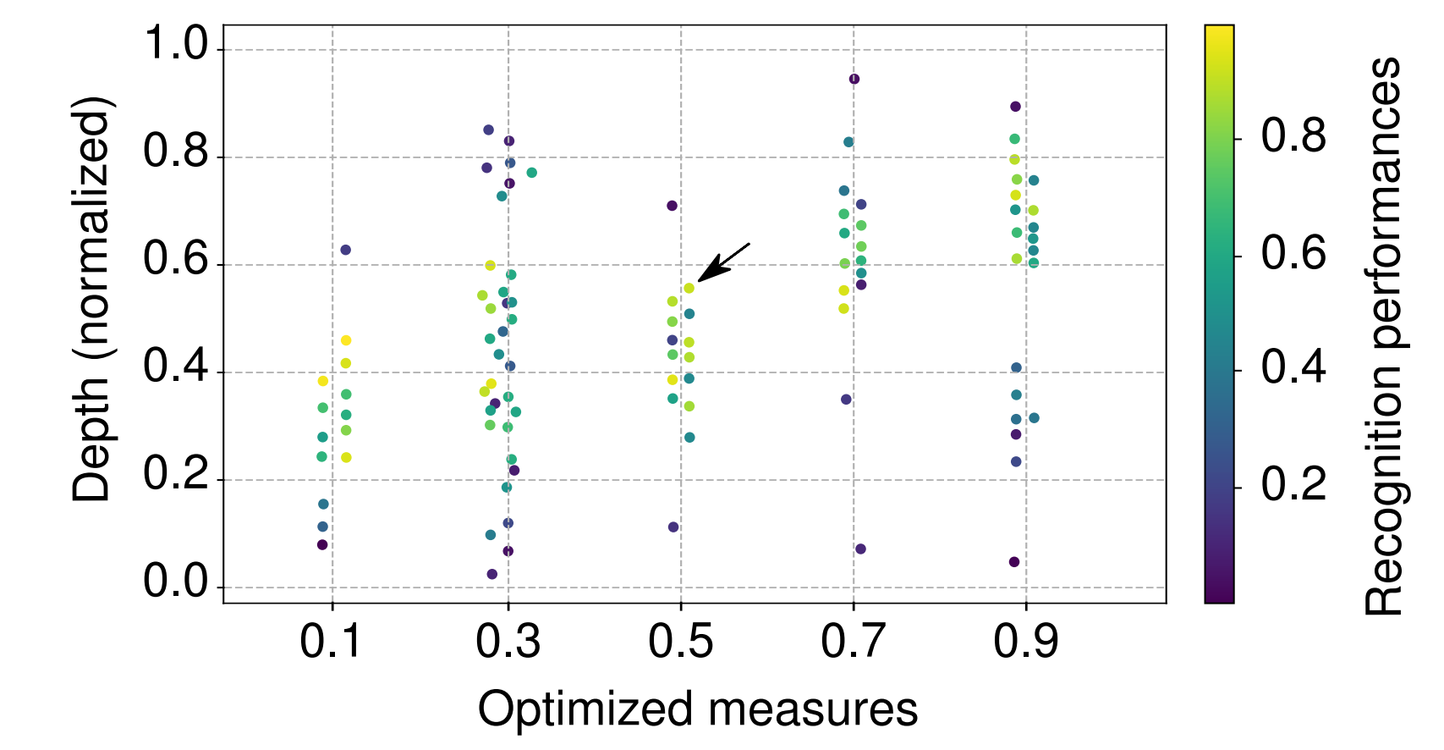
Model	USC-HAD	HTC-TMD	SHL
DeepConvLSTM	65.8±.0028	68.2±.0016	65.3±.012
DeepSense	67.0±.017	68.5±.0032	66.5±.005
AttnSense	68.5±.04	70.1±.005	68.4±.002
Feature fusion	67.2±.001	69.2±.0074	66.8±.0042
Corr. align.	69.5±.004	70.5±.0026	69.1±.06
Proposed	71.8±.001	74.5±.0017	73.7±.006

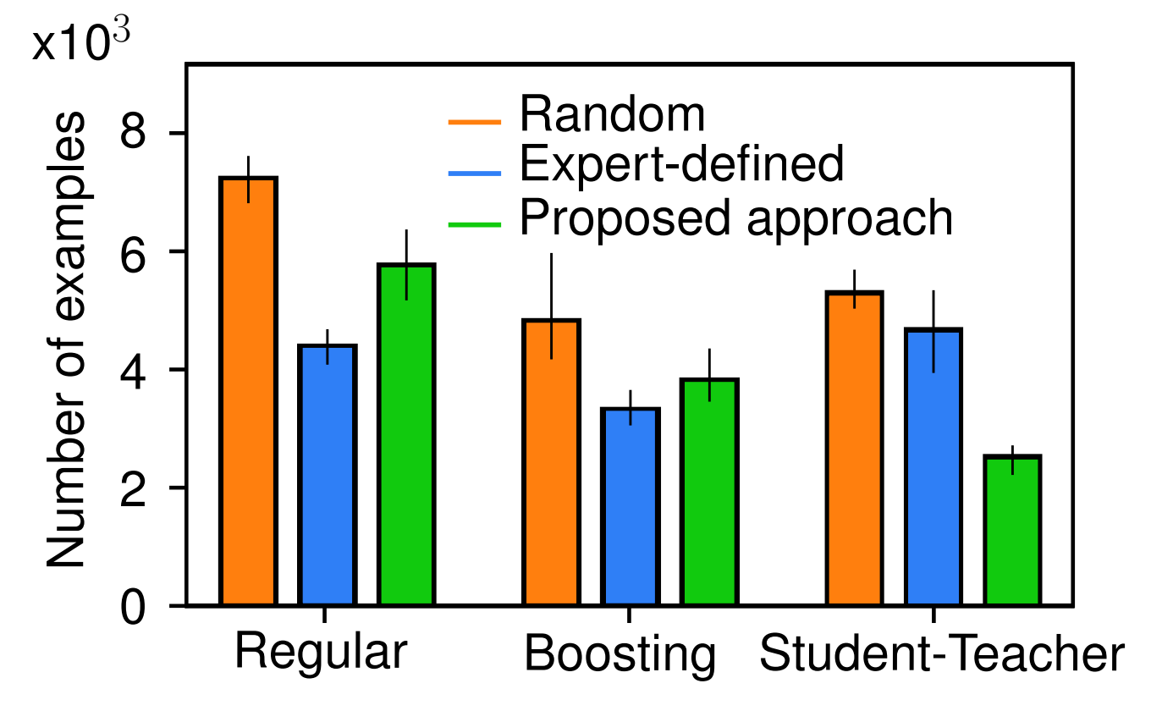


(1) Our approach contributes noticeably to improving the recognition performances on the SHL dataset with $\sim$4%. (2) A similar trend can be observed in the per-node recognition performances with the *"on-feet"* activities such as *Walking* and *Running* grouped together in node #1.

6. Hierarchy Derivation and Amount of Supervision

(1) How do the measures change when we go down the hierarchy? Are the best clustering solutions the ones that very quickly decompose the groups of concepts into atomic ones? Or, on the contrary, those which try to keep the concepts grouped until the leaves?;
(2) How leveraging the hierarchical structuring of the concepts can improve the learning process in terms of data supervision.



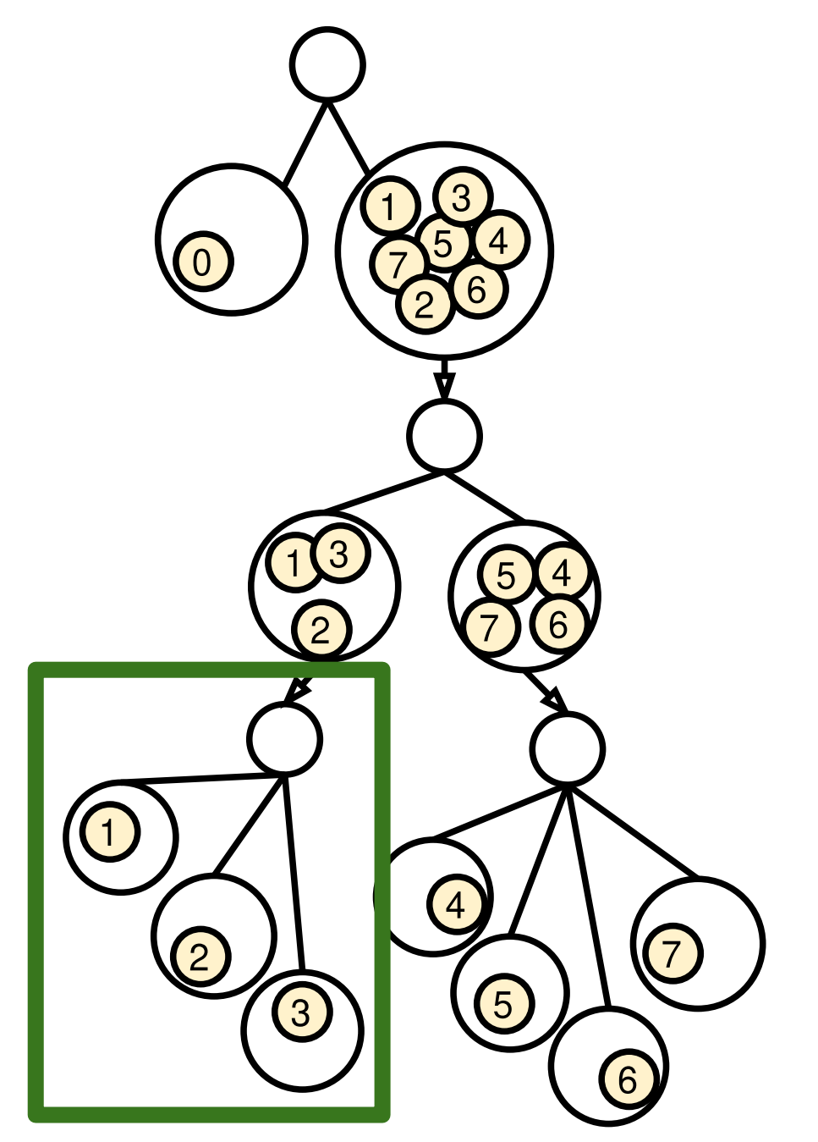


(1) In theory, optimal hierarchies would be those keeping the concepts grouped while going down the hierarchy, which result in deeper hierarchies in a way that the biases of groups are leveraged to a greater extent. For high values of the optimized measures ($\geq$ 0.8), we get a fairly large number of deep hierarchies which are accompanied by fair recognition performances ($\sim$70%). (2) Proposed hierarchies achieve competitive performances while using far less training examples ($\sim$2$\times$10^{-3} examples).

7. Inheritance of Inductive Biases

The bias learned at each non-leaf node is more adapted to each group. But, is there a way to structure the biases such that a given learner can share them with its descendants in the hierarchy? We investigate how this happens via the study of hyperparameters' importance. For example, the hyperparameters' importance are those corresponding to the group of concepts {1, 2, 3}. The more a hyperparameter is important the higher its impact on the learning.

	Groups of concepts			
Hyperparam.	[0][1-7]	[1,2,3][4-7]	[1][2][3]	[4][5][6][7]
First layer				
Kernel size	0.496	0.021	0.026	0.079
# of filters	0.325	0.078	0.014	0.124
Stride	0.852	0.745	0.752	0.664
Second layer				
Kernel size	0.147	0.578	0.454	0.125
# of filters	0.452	0.327	0.273	0.368
Stride	0.662	0.491	0.765	0.054
Third layer				
Kernel size	0.654	0.584	0.027	0.041
# of filters	0.076	0.025	0.581	0.031
Stride	0.324	0.558	0.754	0.017



Some hyperparameters, such as *Stride*, appear to be very important for nearly all learners of the hierarchy and can potentially be optimized jointly.