Supplementary Materials - Leveraging Class Hierarchies with Metric-Guided Prototype Learning

6 Notebook and illustration

In Figure 5, we represent the embeddings and prototypes generated by variations of our networks as well as their respective performance. We note that the *fixed* prototypes approach performs significantly worse than our metric-guided method. We observe that the resulting prototypes are more compact when they are learned independently, which can lead to an increase in misclassification. We also remark that when the hierarchy contains no useful information, such as the arbitrary order of digits, the metric-based approach has a worse performance than the free (unguided) method. This is particularly drastic for the fixed prototype approach.

An illustrated notebook to reproduce this figure can be accessed at the following URL:

https://colab.research.google.com/drive/1VoQfBx5q5lWFev0cwxLZ0qQOZU7Rlmb_#offline=true&sandboxMode=true

To run this notebook locally, you can also download it from our repository: https://github.com/VSainteuf/metric-guided-prototypes-pytorch.

7 Additional methodological details

7.1 Scale-Independent Distortion

Computing the scale-free distortion defined in Equation 5 amounts to finding a minimizer of the following function $f : \mathbb{R} \mapsto \mathbb{R}$:

$$f(s) = \sum_{i \in I} |s\alpha_i - 1|, \tag{8}$$

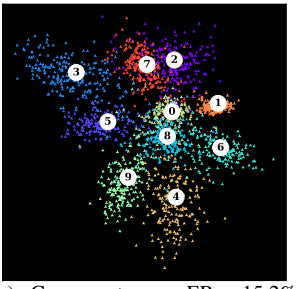
with $\alpha_{k,l} = d(\pi_k, \pi_l)/D[k,l]$, and I an ordering of $\{k,l\}_{k,l \in \mathcal{K}^2}$ such that the sequence $[\alpha_i]_{i \in I}$ is non-decreasing.

Proposition 1. *A global minimizer of f defined in (8) is given by $s^\star = 1/\alpha_i$ with i defined as:*

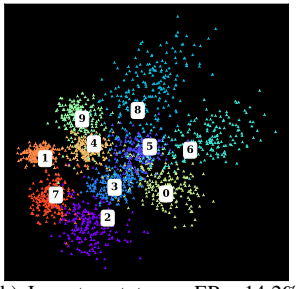
$$i = \min \left\{ j \in I \mid \sum_{k \leq j} \alpha_k \geq \sum_{k > j} \alpha_k \, . \right\} \tag{9}$$

Proof. First, such i exists as it is the smallest member of a discrete, non-empty set (containing at least $j = |I|$). We now verify that $s^\star = 1/\alpha_i$ is a critical point of f. By definition of i we have that $\sum_{k \leq i} \alpha_k \geq \sum_{k > i} \alpha_k$ and $\sum_{k < i} \alpha_k < \sum_{k \geq i} \alpha_k$. By combining these two inequality, we have that

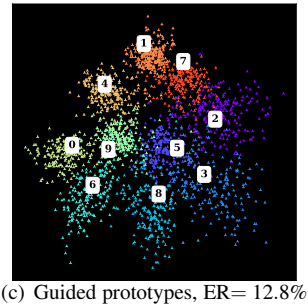
$$-\sum_{k<i} \alpha_k + \sum_{k>i} \alpha_k \in [-\alpha_i, \alpha_i] \, . \tag{10}$$



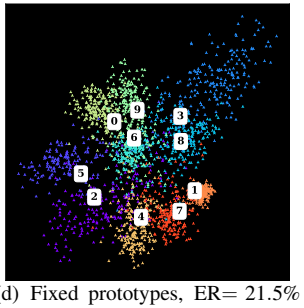
(a) Cross entropy, ER= 15.2%
$\text{disto}_{\text{vis}} = 0.47$, $\text{disto}_{\text{abs}} = 0.61$
$\text{AHC}_{\text{vis}} = 0.81$, $\text{AHC}_{\text{abs}} = 0.65$



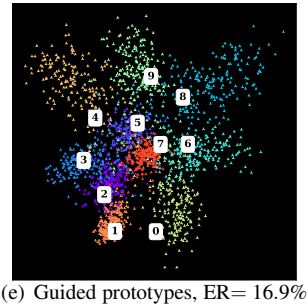
(b) Learnt prototypes, ER= 14.2%
$\text{disto}_{\text{vis}} = 0.42$, $\text{disto}_{\text{abs}} = 0.58$
$\text{AHC}_{\text{vis}} = 0.75$, $\text{AHC}_{\text{abs}} = 0.50$



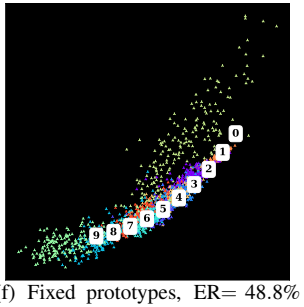
(c) Guided prototypes, ER= 12.8%
$\text{disto}_{\text{vis}} = 0.22$ $\text{AHC}_{\text{vis}} = 0.56$



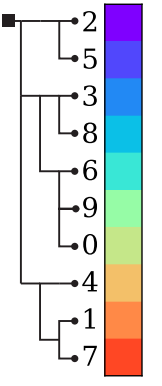
(d) Fixed prototypes, ER= 21.5%
$\text{disto}_{\text{vis}} = 0.17$ $\text{AHC}_{\text{vis}} = 0.82$



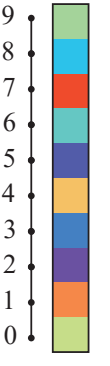
(e) Guided prototypes, ER= 16.9%
$\text{disto}_{\text{abs}} = 0.24$ $\text{AHC}_{\text{abs}} = 0.54$



(f) Fixed prototypes, ER= 48.8%
$\text{disto}_{\text{abs}} = 0.00$ $\text{AHC}_{\text{abs}} = 0.80$



(g)



(h)

Figure 5: Mean class representation ○ , prototypes □ , and 2-dimensional embeddings ⅄ learnt on perturbed MNIST by a 3-layer convolutional net with six different classification modules: (a) cross-entropy, (b) learnt prototypes, (c) learnt prototypes guided by a visual taxonomy, (d) fixed prototypes (see Section 8.2) from a visual taxonomy , (e) learnt prototypes guided by the numbers' values, and (f) fixed prototypes from the numbers' values. The visual hierarchy is represented in (g) and the numerical order in (h). AHC_{vis} corresponds to the cost defined by our proposed visual hierarchy, while AHC_{abs} is defined after the chain-like structure obtained when organizing the digits along their numerical values. While embedding the metric with prototypes prior to learning the representations leads to lower (scale-free) distortion, this translates into worst performance in terms of AHC and ER. Joint learning achieves better performance on both evaluation metrics. We also remark that when the hierarchy is arbitrary (e-f), metric guiding is detrimental to precision.

The subgradient of f at $s^\star$ is the following:

$$\partial_s f(s^\star) = \sum_{k<i} \partial_s |s^\star \alpha_k - 1| + \sum_{k>i} \partial_s |s^\star \alpha_k - 1| + \partial_s |s^\star \alpha_i - 1| \tag{11}$$

$$= -\sum_{k<i} \alpha_k + \sum_{k>i} \alpha_k + [-\alpha_i, \alpha_i]. \tag{12}$$

By using the inequality defined in Equation 10, we have that $0 \in \partial_s f(s^\star)$ and hence $s^\star$ is a critical point of f. Since f is convex, such $s^\star$ is also a global minimizer of f, *i.e.* an optimal scaling. ■

This proposition gives us a fast algorithm to obtain an optimal scaling and hence a scale-free distortion: compute the cumulative sum of the $\alpha_{k,l}$ sorted in ascending order until the equality in (9) is first verified at index i. The resulting optimal scaling is then given by $1/\alpha_i$.

7.2 Smooth Distortion

The minimization problem with respect to s defined in Equation 6 can be solved in closed form:

$$s^\star = \sum \frac{d(\pi_k, \pi_l)}{D[k,l]} \Big/ \sum \frac{d(\pi_k, \pi_l)^2}{D[k,l]^2} \,. \tag{13}$$

7.3 Evolution of Optimal Scaling

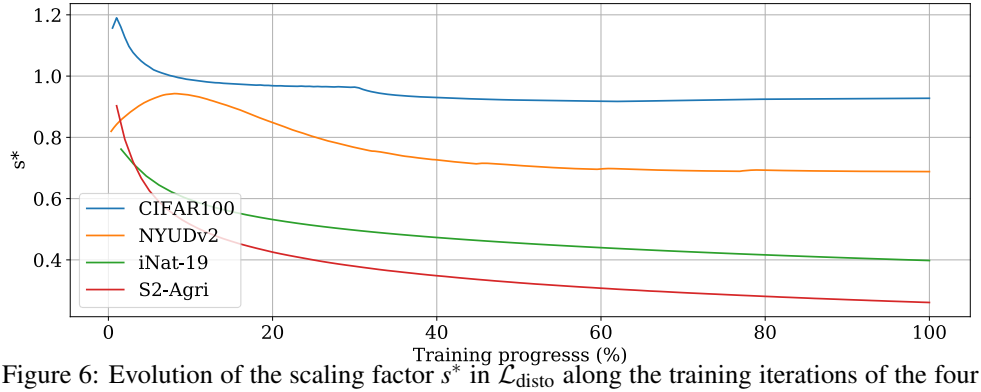


Figure 6: Evolution of the scaling factor s^* in $\mathcal{L}_{\text{disto}}$ along the training iterations of the four networks. We observe that s^* consistently decreases to values smaller than 1, which allow the prototypes to spread apart while respecting the fix distances defined by D.

In Figure 6, we represent the evolution of the scaling factor s^* in $\mathcal{L}_{\text{disto}}$ during training of our guided prototype method on the four datasets. Across all four models, s^* presents a decreasing trend overall, which signifies that the average distance between prototypes increases. This is consistent with our analysis of prototypical networks: as the feature learning network and the prototypes are jointly learned, the samples' representations get closer to their true class' prototype. In doing so, they repel the other prototypes, which translate into an *inflation* of the global scale of the problem. Our optimal scaling allows the prototypes'

scale to expand accordingly. Without adaptive scaling, the data loss (3) and regularizer (6) would conflict.

In all our experiments, this scale remained bounded and did not diverge. This can be explained by the fact that for each misclassification $k \to l$ of a sample x_n, the representation $f(x_n)$ is by definition closer to the erroneous prototype π_l than of the true prototype π_k. The first term of $\mathcal{L}_{\text{data}}$ pushes the true prototype π_k towards $f(x_n)$, and by transitivity—towards the erroneous prototype π_l. This phenomenon prevents prototypes from being pushed away from one another indefinitely. However, if the prediction is too precise, *i.e.* most samples are correctly classified, the prototypes may diverge. This setting, which we haven't yet encountered, may necessitate a regularization such as weight decay on the prototypes parameters.

Lastly, we remark that the asymptotic optimal scalings are different from one dataset to another. This can be explained foremost by differences in the depth and density of the class hierarchy of each dataset, as presented in Table 1. As explained above, the inherent difficulty of the classification tasks also have an influence on the problem's scale. However, our parameter-free method is able to automatically find an optimal scaling.

7.4 Inference

As with other prototypical networks, we associate to a sample n the class k whose prototype π_k is the closest to the representation $f(x_n)$ with respect to d, corresponding to the class of highest probability. This process can be made efficient for a large number of classes K and a high embedding dimension m with a KD-tree data structure, which offers a query complexity of $O(\log(K))$ instead of $O(K \cdot m)$ for an exhaustive comparison. Hence, our method does not induce longer inference time than the cross-entropy for example, as the embedding function typically takes up the most time.

7.5 Rank-based Guiding

Following the ideas of Mettes et al. [32], we also experiment with a RankNet-inspired loss [8] which encourages the distances between prototypes to follow *the same order* as the costs between their respective classes, without imposing a specific scaling:

$$\mathcal{L}_{\text{rank}}(\pi) = -\frac{1}{|\mathcal{T}|} \sum_{k,l,m \in \mathcal{T}} \bar{\mathbf{R}}_{k,l,m} \cdot \log(R_{k,l,m}) + (1 - \bar{\mathbf{R}}_{k,l,m}) \cdot \log(1 - R_{k,l,m}) \,, \tag{14}$$

with $\mathcal{T} = \{(k,l,m) \in \mathcal{K}^3 \mid k \neq l, l \neq m, k \neq m\}$ the set of ordered triplet of $\mathcal{K}$, $\bar{\mathbf{R}}_{k,l,m}$ the hard ranking of the costs between $D_{k,l}$ and $D_{k,m}$, equal to 1 if $D_{k,l} > D_{k,m}$ and 0 otherwise, and $R_{k,l,m} = \text{sigmoid}(d(\pi_k, \pi_l) - d(\pi_k, \pi_m))$ the soft ranking between $d(\pi_k, \pi_l)$ and $d(\pi_k, \pi_m)$. For efficiency reasons, we sample at each iteration only a S-sized subset of $\mathcal{T}$. We use $S = 10$ in our experiments.

8 Additional experimental details

We give additional details on our experiments and some supplementary results in the following subsections.

8.1 Competing methods

Hierarchical Cross-Entropy (HXE) Bertinetto et al. [3] model the class structure with a hierarchical loss composed of the sum of the cross-entropies at each level of the class hierarchy. As suggested, a parameter α taken as 0.1 defines exponentially decaying weights for higher levels.

Soft Labels (Soft-labels) Bertinetto et al. [3] propose as second baseline in which the the one-hot target vectors are replaced by soft target vectors in the cross-entropy loss. These target vectors are defined as the softmin of the costs between all labels and the true label, with a temperature $1/\beta$ chosen as 0.1, as recommended in Bertinetto et al. [3].

Earth Mover Distance regularization (XE+EMD): Hou et al. [21] propose to account for the relationships between classes with a regularization based on the squared earth mover distance. We use D as the ground distance matrix between the probabilistic prediction p and the true class y. This regularizer is added along the cross-entropy with a weight of 0.5 and an offset μ of 3.

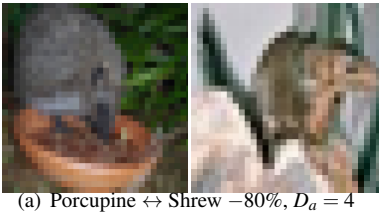
Hierarchical Inference (YOLO): Redmon and Farhadi [37] propose to model the hierarchical structure between classes into a tree-shaped graphical model. First, the conditional probability that a sample belongs to a class given its parent class is obtained with a softmax restricted to the class' co-hyponyms (*i.e.* siblings). Then, the posterior probability of a leaf class is given by the product of the conditional probability of its ancestors. The loss is defined as the cross-entropy of the resulting probability of the leaf-classes.

Hyperspherical Prototypes (Hyperspherical-proto): The method proposed by Mettes et al. [32] is closer to ours, as it relies on embedding class prototypes. They advocate to first position prototypes on the hypersphere using a rank-based loss (see Section 4.6) combined with a prototype-separating term. They then use the squared cosine distance between the image embeddings and prototypes to train the embedding network. Note that in our re-implementation, we used the finite metric defined by D instead of Word2Vec [33] embeddings to position prototypes. Lastly, we do not evaluate on S2-Agri as the integration of the focal loss is non-trivial.

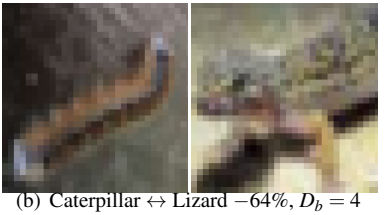
Deep Mean Classifiers (Deep-NCM): Guerriero et al. [16] present another prototype-based approach. Here, the prototypes are the cumulative mean of the embeddings of the classes' samples, updated at each iteration. The embedding network is supervised with $\mathcal{L}_{\text{data}}$ with d defined as the squared Euclidean norm.

8.2 Numerical results

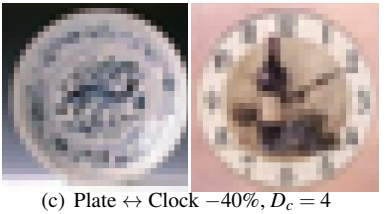
The numerical values of the results shown in Figure 2 are given in Table 2.



(a) Porcupine ↔ Shrew -80%, $D_a = 4$

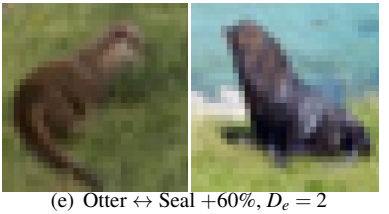


(b) Caterpillar ↔ Lizard -64%, $D_b = 4$

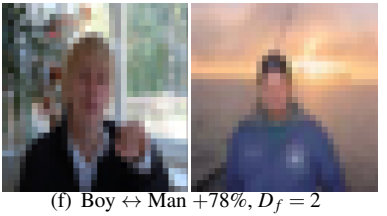


(c) Plate ↔ Clock -40%, $D_c = 4$

(d) Streetcar ↔ Bus $+58\%$, $D_d = 4$



(e) Otter ↔ Seal $+60\%$, $D_e = 2$



(f) Boy ↔ Man $+78\%$, $D_f = 2$

Figure 7: Best (a-c) and worse (d-f) improvements in terms of class confusion provided by Guided-proto compared to the cross-entropy baseline for CIFAR100, given in %, along with their error cost. The metric guided regularization particularly helps decreasing the confusions between classes that are visually similar (*e.g.* Plate and Clock) but are not direct siblings in the class hierarchy ($D = 4$). Conversely, the regularization hinders performance for visually similar siblings classes (*e.g.* Otter and Seal, $D = 2$).

8.3 Ablation Studies

Choice of distance : In Table 3, we report the performance of the Guided-proto model on the four datasets when replacing the Euclidean norm with the squared Euclidean norm. Across our experiments, the squared-norm based model yields a worse performance. This is a notable result as it is the distance commonly used in most prototypical networks [16, 45].

Rank-based Regularization: Mettes et al. [32] use a rank-based loss [8] to encourage prototype mappings whose pairwise distance follows the same order as an external qualification of errors D. We argue that our formulation of $\mathcal{L}_{\text{disto}}$ provides a stronger supervision than only considering the order of distances, and allows the prototypes to find a more profitable arrangement in the embedding space. In Table 3, we observe that replacing our distortion-based loss by a rank-based one results in a slight decrease of overall performance.

Robustness: As shown in Table 4, our presented method has low sensitivity with respect to regularization strength: models trained with λ ranging from 0.5 to 3 yield sensibly equivalent performances. Choosing $\lambda = 1$ seems to be the best configuration in terms of AHC.

Table 2: Error Rate (ER) in % and Average Hierarchical Cost (AHC) on three datasets for our proposed methods (top) and the competing approaches (bottom). The values are computed with the median over 5 runs for CIFAR100, the average over 5 cross-validation folds for S2-Agri, and a single run for NYUDv2 and iNat-19. (HSP: Hyperspherical Prototypes, GP: Guided Prototypes).

	CIFAR100		NYUDv2		S2-Agri		iNat-19	
	ER	AHC	ER	AHC	ER	AHC	ER	AHC
Cross-Entropy	24.2	1.160	32.7	1.486	19.4	0.699	40.9	1.993
HXE	24.1	1.168	32.4	1.456	19.5	0.731	41.8	2.013
Soft-label	23.5	**1.046**	32.4	**1.424**	19.2	0.703	52.8	2.029
XE+EMD	24.5	1.196	33.3	1.498	19.0	0.687	40.1	1.893
YOLO	26.2	1.214	**32.0**	**1.425**	19.1	**0.685**	42.0	1.942
HSP	29.4	1.472	49.7	2.329	-	-	42.4	2.027
Deep-NCM	25.6	1.249	33.5	1.498	19.4	0.702	40.8	1.929
Free-proto	23.8	1.091	32.5	1.462	19.1	0.691	**38.8**	**1.728**
Fixed-proto	24.7	1.083	33.1	1.462	19.4	0.710	43.9	2.148
GP-rank	**23.3**	**1.056**	32.7	1.445	19.1	0.691	39.3	**1.718**
GP-disto	23.6	**1.052**	32.5	1.440	**18.9**	**0.685**	**38.9**	**1.721**

Hidden prototypes: In cases where the cost matrix D is derived from a tree-shaped class hierarchy, it is possible to also learn prototypes for the internal nodes of this tree, corresponding to super-classes of leaf-level labels. These prototypes do not appear in $\mathcal{L}_{\text{data}}$, but can be used in the prototype penalization to instill more structure into the embedding space. In Table 4, line *leaf-proto*, we note a small but consistent improvement in terms of AHC resulting in associating prototypes for classes corresponding to the internal-nodes of the tree hierarchy as well.

8.4 Illustration of Results

In Figure 7 and Figure 3, we illustrate that our model particularly improves the classification rates of classes with high visual similarity and comparatively large error costs.

9 Additional Implementation Details

CIFAR100 ResNet-18 is trained on CIFAR100 using SGD with initial learning rate $l_r = 10^{-1}$, momentum set to 0.9 and weight decay $w_d = 5 \cdot 10^{-4}$. The network is trained for 200 epochs in batches of size 128, and the learning rate is divided by 5 at epochs 60, 120, and 160. The model is evaluated using its weights of the last epoch of training, and the results reported in the paper are median values over 5 runs.

NYUDv2 We train FuseNet on NYUDv2 using SGD with momentum set to 0.9. The learning rate is set initially to 10^{-3} and multiplied at each epoch by a factor that exponentially

Table 3: Influence of the choice of scaling in $\mathcal{L}_{\text{disto}}$, metric guiding regularizer, guiding scheme, and distance function d on the performance of Guided-proto on the four datasets. For d, we compare the performance of the Euclidean norm, the pseudo-Huberized Euclidean norm, and the square Euclidean norm.

	CIFAR100		NYUDv2		S2-Agri		iNat-19	
	ER	AHC	ER	AHC	ER	AHC	ER	AHC
Guided-proto	23.6	**1.052**	32.5	1.440	**18.9**	**0.685**	**38.9**	1.721
Fixed-scale	+0.1	+0.003	0.0	0.000	+0.2	**+0.001**	+0.9	0.000
Fixed-proto	+1.1	+0.031	+0.6	+0.013	+0.5	+0.025	+5.0	+0.427
Rank-based guiding	**-0.3**	+0.004	+0.2	+0.005	+0.2	+0.006	+0.4	**-0.003**
Squared Norm	+1.0	+0.118	0.0	+0.005	+0.6	+0.022	+2.2	+0.233

Table 4: Robustness assessment of guided prototypes on CIFAR100 (left) and S2-Agri (right). The top line is our chosen hyper-parameter configuration.

	CIFAR100		S2-Agri	
	ER	AHC	ER	AHC
Guided-proto $\lambda = 1$, hidden proto,	23.6	**1.052**	**18.9**	**0.685**
$\lambda = 0.5$	**-0.2**	+0.015	+0.5	+0.019
$\lambda = 2$	+0.3	+0.013	+0.2	+0.010
$\lambda = 3$	+0.1	+0.004	+0.1	+0.010
leaf proto only	+0.2	+0.015	+0.3	+0.011

decreases from 1 to 0.9. The network is trained for 300 epochs in batches of 4 with weight decay set to $5 \cdot 10^{-3}$. We report the performance of the best-of-five last testing epochs.

S2-Agri We train PSE+TAE on S2-Agri using Adam with $l_r = 10^{-3}$, $\beta = (0.9; 0.999)$ and no weight decay. The dataset is randomly separated in five splits. For each of the five folds, 3 splits are used as training data on which the network is trained in batches of 128 samples for 100 epochs. The best epoch is selected based on its performance on the validation set, and we use the last split to measure the final performance of the model. We report the average performance over the five folds.

iNaturalist-19 Given the complexity of the dataset, we follow [3] and use a ResNet-18 pre-trained on ImageNet. The network is trained for 65 epochs in batches of 64 epochs using Adam with $l_r = 10^{-4}$, $\beta = (0.9; 0.999)$ and no weight decay. The best epoch is selected based on the performance on the validation set, and we report the performance on the held-out test set.

10 Hierarchies used in Experiments

We present here the hierarchy used in the numerical experiments to derive the cost matrix. We define the cost between two classes as the length of the shortest path in the proposed tree-shape hierarchy. The hierarchy of CIFAR100 is presented in Figure 8, NYUDv2 in Figure 9,

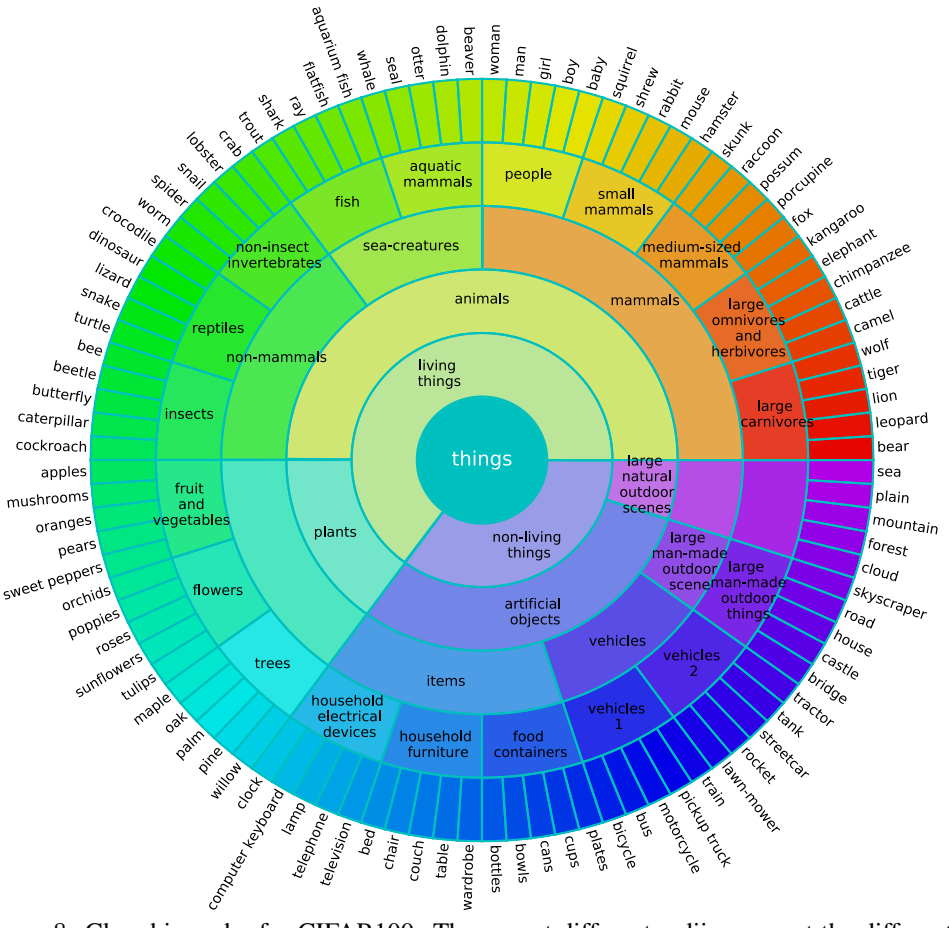


Figure 8: Class hierarchy for CIFAR100. The arcs at different radii represent the different classes of each level of the hierarchy. Unlabelled arcs share the same name as their parent class.

S2-Agri in Figure 10, and iNat-19 in Figure 11.

For S2-Agri, we built the hierarchy by combining the two levels available in the dataset S2 of Garnot *et al.* withwith the fine-grained description of the agricutltural parcel classes on the French Payment Agency's website (in French):

https://www1.telepac.agriculture.gouv.fr/telepac/pdf/tas/2017/Dossier-PAC-2017_notice_cultures-precisions.pdf.

Note that for S2-Agri, following [40] we have removed all classes that had less than 100 samples among the almost 200 000 parcels to limit the imbalance of the dataset.

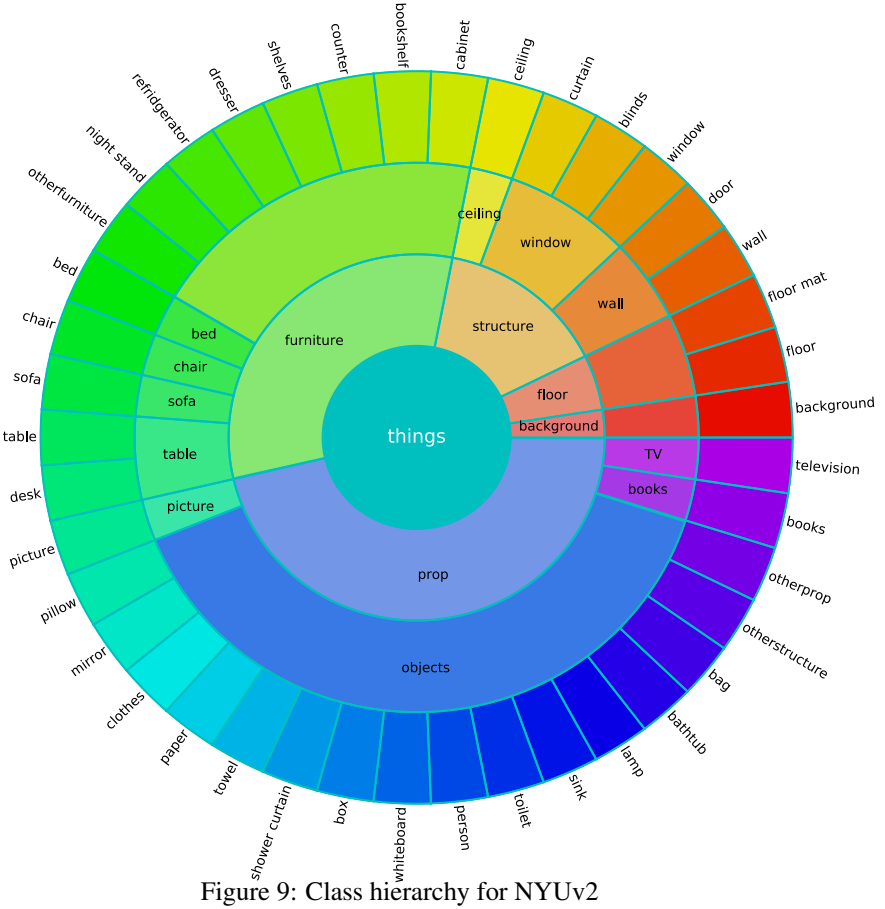


Figure 9: Class hierarchy for NYUv2

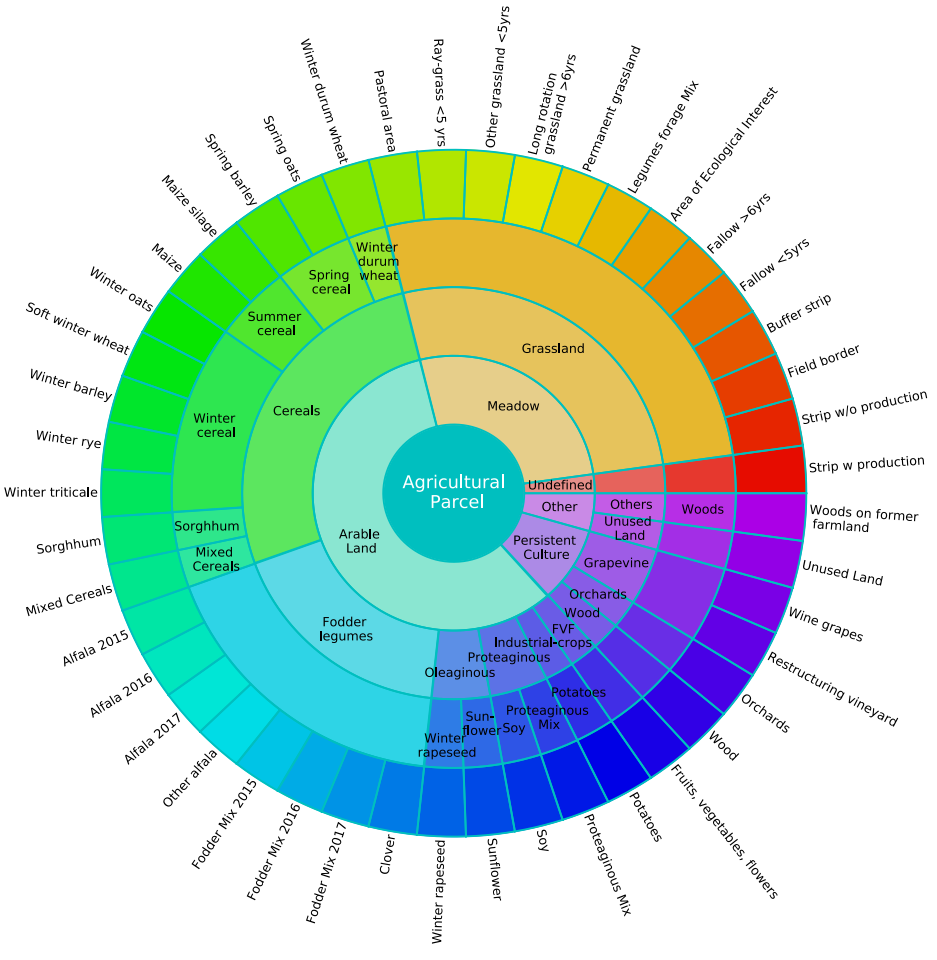


Figure 10: Class hierarchy for S2-Agri

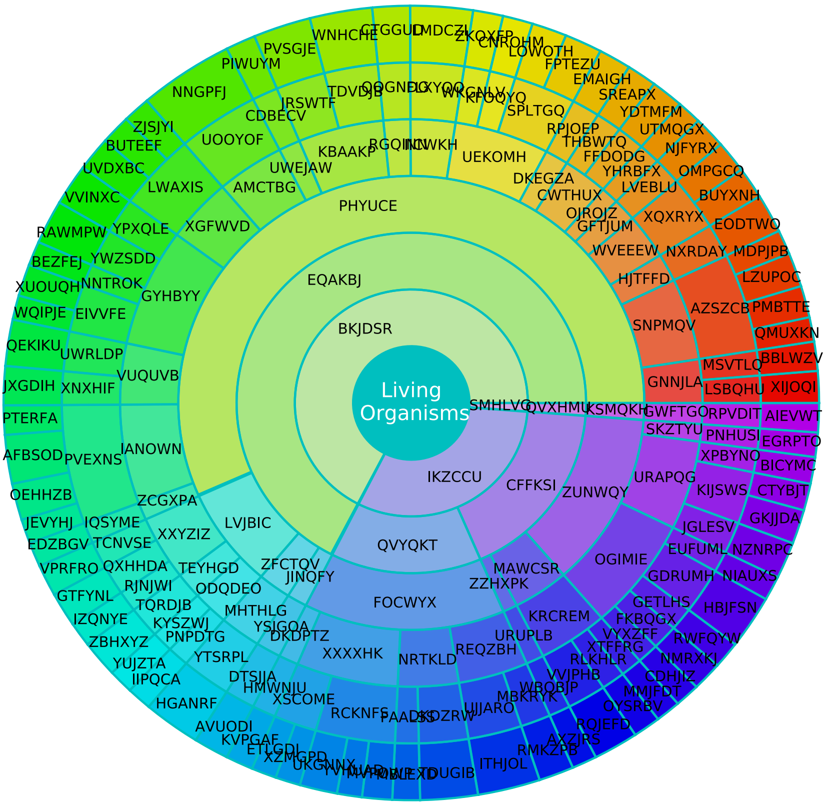


Figure 11: Class hierarchy for iNat-19, only the first 6 levels of the hierarchy are represented. At the time of writing, only the classes' obfuscated names were publicly available