A PROOF OF PROPOSITIONS

We first restate propositions and then provide the proof.

Proposition 3.1. For any undirected weighted graph $\mathcal{G}$, $d_{\mathcal{T}}(\cdot,\cdot)$ is a well-defined metric over $\mathcal{Y}$, if all edge weights of $\mathcal{G}$ are positive and $\mathcal{G}$ is connected.

Proof. Given $v_i, v_j \in V$, if $d_{\mathcal{T}}(v_i, v_j)$ is a well-defined metric function, it means $d_{\mathcal{T}}$ must satisfy the following three properties:

1. if $v_i \neq v_j$, $d_{\mathcal{T}}(v_i, v_j) > 0$. Otherwise, $d_{\mathcal{T}}(v_i, v_j) = 0$.
2. $d_{\mathcal{T}}(v_i, v_j) = d_{\mathcal{T}}(v_j, v_i)$.
3. $\forall v_k \in V, d_{\mathcal{T}}(v_i, v_j) \leq d_{\mathcal{T}}(v_i, v_k) + d_{\mathcal{T}}(v_k, v_j)$.

By definition $d_{\mathcal{T}}(v_i, v_j)$ is the distance of the shortest path from v_i to v_j. Since $\mathcal{G}$ has positive weights, the first statement holds.

There is always a path p from v_i to v_j because $\mathcal{G}$ is connected. Since $\mathcal{G}$ is undirected, there is also a path p' from v_j to v_i by backtracking vertices on p. The sum of edge weights on p is the same as that on p'. p' must be the shortest path from v_j to v_i: if there exists a shorter path q from v_j to v_i, backtracking q will give a shorter path q' from v_i to v_j. Thus, $d_{\mathcal{T}}(v_i, v_j) = d_{\mathcal{T}}(v_j, v_i)$.

If the shortest path p from v_i to v_j contains v_k, $d_{\mathcal{T}}(v_i, v_j) = d_{\mathcal{T}}(v_i, v_k) + d_{\mathcal{T}}(v_k, v_j)$. If p does not contain v_k, if $d_{\mathcal{T}}(v_i, v_j) > d_{\mathcal{T}}(v_i, v_k) + d_{\mathcal{T}}(v_k, v_j)$, $v_i \dots v_k \dots v_j$ is a valid path that is shorter than p. Contradiction. ■

Proposition 3.2. Let $\mathcal{T}$ be an unweighted tree with a fixed root node. Then for any pair of nodes $v_i, v_j \in V$, $d_{\mathcal{T}}(v_i, v_j) = 2t(v_i, v_j) - |dt(v_i) - dt(v_j)|$.

Proof. Since $\mathcal{T}$ is an unweighted tree, there is a unique path connecting v_i and v_j. Let the path p be $v_i, v_1, \dots, v_{L-1}, v_j$ of length L. Let $v_l := \text{LCA}\ (v_i, v_j)$. By definition of LCA , there exists a path from v_l to v_i and a path from v_l to v_j as well. Because the path p between v_i and v_j is unique, v_l must be on the path p as well.

Now, since v_l is the least common ancestor, the depth of v_i, v_j can be written as:

$$dt(v_i) = dt(v_l) + l, \quad dt(v_j) = dt(v_l) + L - l.$$

Hence, we have

$$\begin{aligned} 2t(v_i, v_j) - |dt(v_i) - dt(v_j)| &= 2\max\{dt(v_i), dt(v_j)\} - 2dt(\text{LCA}\ (v_i, v_j)) - |dt(v_i) - dt(v_j)| \\ &= 2\max\{dt(v_l) + l, dt(v_l) + L - l\} - 2dt(v_l) - |L - 2l| \\ &= 2\max\{l, L - l\} - |L - 2l| \\ &= \max\{l, L - l\} + \min\{l, L - l\} \\ &= L = d_{\mathcal{T}}(v_i, v_j), \end{aligned}$$

where for the second to last equality we use the fact that $\max\{a, b\} - |a - b| = \min\{a, b\}$ for any $a, b \in \mathbb{R}$. ■

B DISCUSSION ON CONVERGENCE OF THE OBJECTIVE FUNCTION

Convergence of SGD Training with CPCC One question to ask about the CPCC regularizer is that, when coupled with the usual cross-entropy loss, can we guarantee that SGD training over the objective function in Eq. 2 will converge? Note that in general, due to the non-convexity of the loss function with deep neural networks, one cannot hope to obtain convergence to the global optimal (if exists). Absent strict assumptions on the saddle points of Eq. 2, e.g., the strict saddle point property (Lee et al., 2016) that can be hard to verify in practice, in general, it is still unclear that gradient descent will always converge to a local minimum (Murty & Kabadi, 1985). So instead, a more realistic convergence criterion that we can hope for is to show that when optimizing the

objective function in Eq. 2, SGD will converge to a stationary point. To this end, under certain regularity conditions on the data, we will show that SGD will converge to a stationary point of Eq. 2 with probability 1, by invoking a key theorem in recent advances of nonconvex optimization with SGD (Patel & Zhang, 2021, Theorem 2).

Proposition B.1 (informal)**.** For a fixed dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, assume the objective function Eq. 2 is bounded from below for every θ, and that the data is bounded, i.e., $\exists R > 0$ such that $\forall i \in [n], \|x_i\|_2 \leq R$, then under the Robbins-Monro condition of learning rate scheduling, SGD will converge to a stationary point of Eq. 2 with probability 1.

The proof of the above proposition mainly follows by verifying that the Assumptions 1-4[1] (Patel & Zhang, 2021) indeed hold in our case under the regularity condition of the data $\mathcal{D}$, so we omit it here. Essentially, in order for the regularity condition to hold, since the cross-entropy function is always bounded from below, we only need to ensure that the CPCC regularizer is bounded from below along the trajectory of θ. Note that since CPCC is a correlation score, it is bounded between $[-1, 1]$ whenever it is well-defined, i.e., the denominator is non-zero, which is the case as long as the distance $\rho_{\mathcal{Z}}(\cdot, \cdot)$ is not a constant over $\mathcal{D}$. The latter is true whenever $\mathcal{D}$ contains data points from at least two different classes and the feature map given by θ is not degenerate, i.e., at least two class mean vectors are different – a mild condition to hold in practice.

C Formulation of Objectives

- **Flat**: Train using ℓ_{CE} on fine classes only, without leveraging any hierarchical information:

$$\mathcal{L}_{\text{Flat}}(x, y_{\text{fine}}) = \sum_{(x,y)\in\mathcal{D}} \ell_{\text{CE}}(y_{\text{fine}}, h(x)). \tag{3}$$

- **Multi-task Learning**: We jointly train a two-headed network to treat fine and coarse as two separate tasks. The network then becomes $h : \mathcal{X} \to \Delta_{k_1} \times \Delta_{k_2}$ with a shared feature encoder $f_\theta : \mathcal{X} \to \mathcal{Z}$ and two classifiers $g_{\text{coarse}} : \mathcal{Z} \to \Delta_{k_1}, g_{\text{fine}} : \mathcal{Z} \to \Delta_{k_2}$. k_1 is the number of coarse classes and k_2 the number of fine classes. The loss function is the sum of the cross-entropies on the fine and coarse classification tasks, and we simply set the weight of the two parts to 1:

$$\mathcal{L}_{\text{MTL}}(x, y_{\text{coarse}}, y_{\text{fine}}) = \sum_{(x,y)\in\mathcal{D}} \ell_{\text{CE}}(y_{\text{coarse}}, g_{\text{coarse}}(f(x)) + \ell_{\text{CE}}(y_{\text{fine}}, g_{\text{fine}}(f(x))). \tag{4}$$

- **Curriculum Learning**: In the spirit of curriculum learning (Bengio et al., 2009), we first train on the coarse classes using Eq. 3 and use y_{coarse} instead of y_{fine}. In the second step, we remove the linear classifier and fine tune a new one on the fine level labels with Eq. 3 as the loss function.
- **Sum Loss**: We define a hierarchical Sum Loss as

$$\mathcal{L}_{\text{SumLoss}}(x, y_{\text{coarse}}, y_{\text{fine}}) = \sum_{(x,y)\in\mathcal{D}} \ell_{\text{CE}}(y_{\text{coarse}}, \mathbf{W}h(x)) + \ell_{\text{CE}}(y_{\text{fine}}, h(x)). \tag{5}$$

  $\mathbf{W}$ is a k_1 by k_2 matrix representing the relationships in the label tree: if a fine class i belongs to a coarse class j, then $\mathbf{W}_{ji}$ is 1, otherwise the entry is set to 0. In other words, the probability of belonging to a given coarse class is simply the sum of the probabilities of its descendants (*i.e.*, fine classes) in the tree. This way, we use the information from both levels with one head only.
- **Hierarchical Cross Entropy** (Bertinetto et al., 2020) Let $h(\cdot)$ be the height of any label y_{level} in the hierarchy, $\lambda(y_{\text{level}}) = \exp(-\alpha h(y_{\text{level}}))$. In a two-level hierarchy, the HXE loss function is

$$\mathcal{L}_{\text{HXE}}(x, y_{\text{fine}}, y_{\text{coarse}}) = -\sum_{(x,y)\in\mathcal{D}} \lambda(y_{\text{fine}}) \log\left(\frac{h(x)_{y_{\text{fine}}}}{\sum_{j\in y_{\text{coarse}}} h(x)_j}\right) + \lambda(y_{\text{coarse}}) \log\left(\sum_{j\in y_{\text{coarse}}} h(x)_j\right) \tag{6}$$

  We set α as 0.4. When $\alpha = 0$, HXE is reduced to Eq. 3. As α grows up, we emphasize more on the latter term to optimize coarse level prediction.

[1]Note that the Assumption 2 of Theorem 2 in (Patel & Zhang, 2021) asks for unbiasedness of the stochastic gradient, which is not the case for the CPCC regularizer. However, as mentioned in the discussion of Theorem 2 (Patel & Zhang, 2021) as well as formally shown in Assumption 4.3(b) of (Bottou et al., 2018), this assumption could be relaxed. Fortunately, this relaxation is rather easy to account for in our case.

- **Soft Label** (Bertinetto et al., 2020) replaces the one-hot encoding of y_{fine} in Eq. 3's ℓ_{CE} by $y_{\text{fine}}^{\text{soft}}$. To leverage the hierarchy, denote $d(i, j)$ as the height of LCA(i, j) over the height of the tree. Let $i \in [k]$, the one-hot vector of y_{fine} becomes a categorically distributed vector where each component is:

$$y_{\text{soft},i}^{\text{fine}} = \frac{\exp(-\beta d(i, y_{\text{fine}}))}{\sum_{j\in[k]} \exp(-\beta d(j, y_{\text{fine}}))}, \forall i \in [k] \tag{7}$$

  We only use the two level coarse-fine hierarchy to calculate $d(i, j)$ and set $\beta = 10$. When $\beta \to \infty$, $y_{\text{fine}}^{\text{soft}}$ is the same as one-hot y_{fine}.

- **Quadruplet Loss** (Zhang et al., 2016) is a multi-task objective $\mathcal{L} = \lambda_s \mathcal{L}_{\text{Flat}} + (1 - \lambda_s)\mathcal{L}_{\text{Quad}}$ where $\mathcal{L}_{\text{Quad}}$ is:

$$\frac{1}{2N} \sum_{i=1}^{N} \max\{0, \mathscr{D}(r_i, p_i^+) - \mathscr{D}(r_i, p_i^-) + m_1 - m_2\} + \max\{0, \mathscr{D}(r_i, p_i^-) - \mathscr{D}(r_i, n_i) + m_2\} \tag{8}$$

  In Eq. 8, N is the number of valid quadruplets in the mini-batch, $\mathscr{D}(\cdot, \cdot)$ is the squared Euclidean distance of l_2 normalized features. For each incoming mini-batch, we set all images as anchor r_i, p_i^+ has the same fine label with r_i, p_i^- has the same coarse bot not fine label with r_i, and n_i and r_i have different coarse labels. We set $\lambda_s = 0.8, m_1 = 0.25, m_2 = 0.15$ in our experiments. Margin m_1 is set to be bigger than m_2 to ensure same fine labels are grouped together, then fine labels within one coarse class are close, and finally classes not in the same coarse class are far away, i.e., $\mathcal{D}(r_i, p_i^+) + m_1 < \mathcal{D}(r_i, p_i^-) + m_2 < \mathcal{D}(r_i, n_i)$. Once we found $\mathcal{D}(r_i, p_i^+)$, we pick the hardest $\mathcal{D}(r_i, p_i^-)$ and then $\mathcal{D}(r_i, n_i)$ to accelerate convergence.

D Additional Figures

We include t-SNE visualization of representations and distance matrices for the remaining objectives (Multi-task, Curriculum, Sum Loss) of Table 1 in this section (Figures 4, 5 and 6 for the t-SNE, and Figures 7, 8 and 9 for the matrices). We also include the distance matrix for using CPCC on three layers in Figure 3, showing that CPCC can generalize to deeper hierarchies without sacrificing too much classification accuracy.

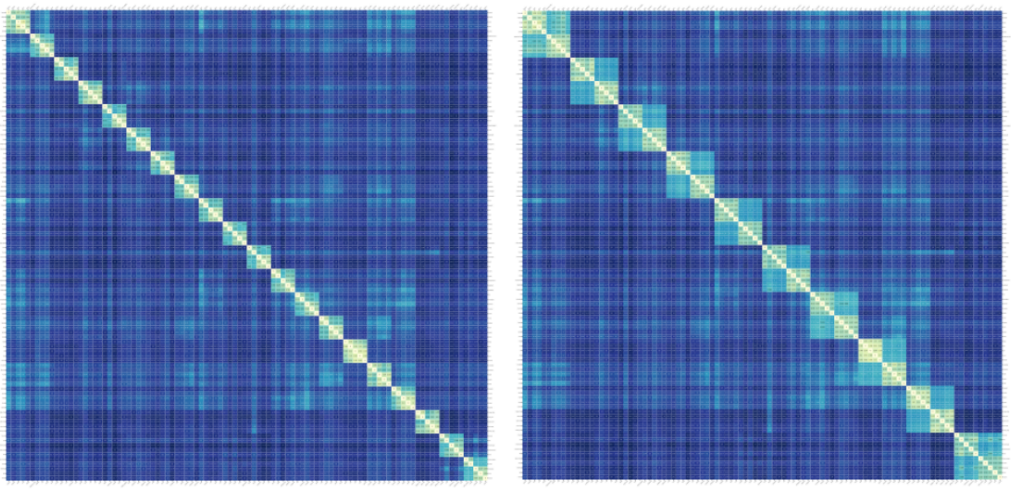


Figure 3: CPCC for three layers for **Flat**. *Left* matrix uses CPCC on fine/mid/coarse (FineAcc^0 = 77.24, CoarseAcc^0 = 86.71). *Right* matrix uses CPCC on fine/coarse/coarser (FineAcc^0 = 76.95, CoarseAcc^0 = 86.58). Both of them correctly encoded information of three levels, by either partitioning the 5 by 5 coarse diagonal blocks into a 2×2 on the left top and 3×3 on the right bottom (*Left*), or grouping neighboring 5 by 5 coarse blocks (*Right*). Lighter color means smaller distance.

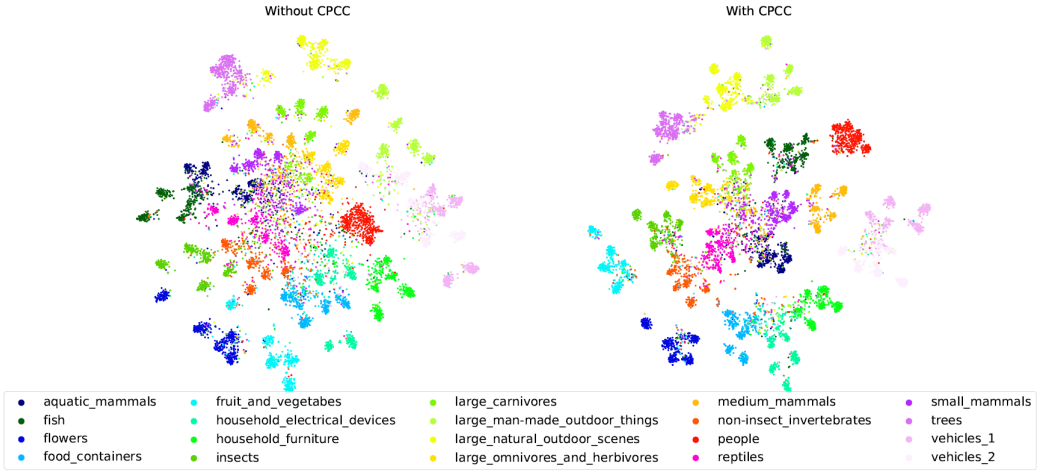


Figure 4: t-SNE visualization of the representations learnt by **Multi-task** with and without CPCC regularization.

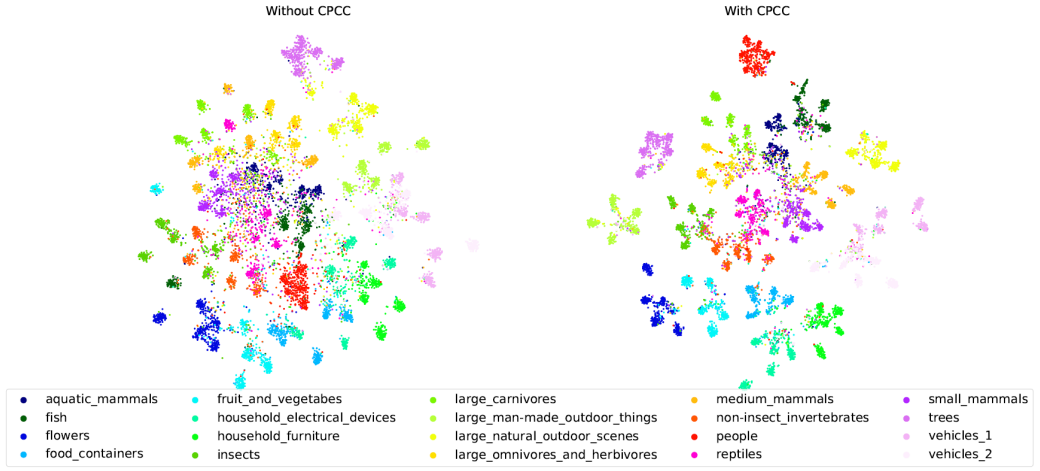


Figure 5: t-SNE visualization of the representations learnt by **Curriculum** with and without CPCC regularization.

E Additional Experiments

In this section, we include remaining baseline experiments of Table 1 to Table 4 and Table 2 to Table 5. Furthermore, we discuss extra settings where CPCC can be applied.

CPCC for OOD Detection Following Hoffmann et al. (2022), we consider CIFAR10 as OOD compared to CIFAR100 (they have disjoint fine classes), and fit a 100-class conditional multivariate Gaussian on the trained CIFAR100 embeddings. The OOD score is the probability that the model gives a higher likelihood to the true class of a random CIFAR100 image than to the maximum class likelihood of a random CIFAR10 one. Results are in Fig. 10: using CPCC significantly boosts the detection score for all objectives.

CPCC for sub-population shift Santurkar et al. (2020) proposed BREEDS benchmark for sub-population shift. Compared to Sec. 4.3, the main difference between two tasks is that BREEDS was only trained on coarse level, while Sec. 4.3 is trained on fine level for all objectives without CPCC. In dataset with sub-population shift, if we train on the train split of source dataset, there will be a big drop of performance of test performance on target set. In the original benchmark setting, models are always trained on *coarse labels*, and fine tuning is unnecessary. In Table 3, we show the performance gain of using CPCC on the Flat method. Since BREEDS contains more than two levels of hierarchy, we present the result of using CPCC on coarse-fine classes, coarse-coarser

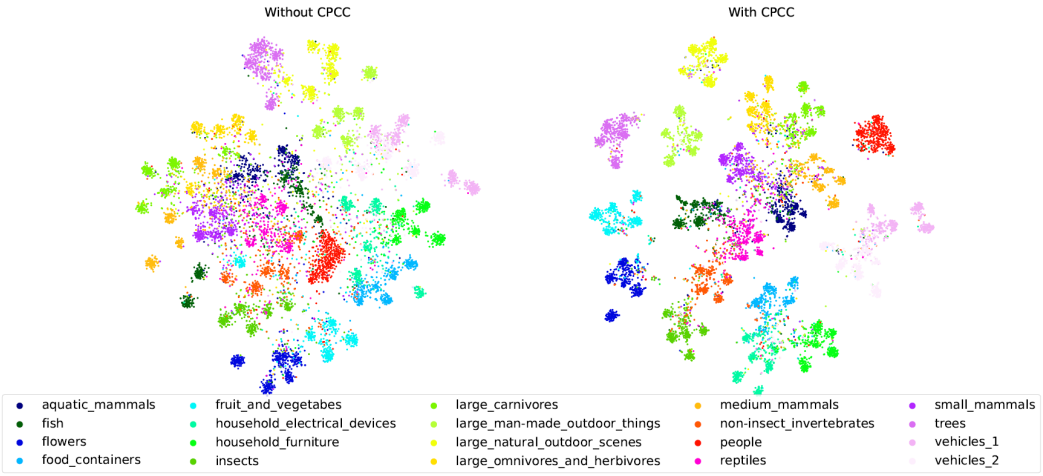


Figure 6: t-SNE visualization of the representations learnt by **Sum Loss** with and without CPCC regularization.

Objective	*LIVING17*	*ENTITY13*	*ENTITY30*	*NONLIVING26*
Flat (coarse)	56.01 (0.71)	60.49 (0.35)	47.72 (0.27)	40.25 (0.66)
Flat + CPCC (coarse, fine)	**57.74 (0.34)**	**61.32 (0.18)**	**48.99 (0.29)**	**42.31 (0.87)**
Flat + CPCC (coarser, coarse)	57.20 (0.90)	60.84 (0.25)	48.45 (0.67)	41.94 (0.90)

Table 3: BREEDS's performance for sub-population shift. Each entry is the target validation accuracy of coarse level classes. Using CPCC consistently outperforms naive Flat.

classes as examples. The result shows CPCC alleviates the sub-population shift problem by using multiple labels in hierarchy, as shown in Mukherjee et al. (2022) who used other methods.

F Implementation Details

Our code is released at `https://github.com/hanzhaoml/HierarchyCPCC`. In this section, we describe the training details for our experiments.

In our experiments, when using CPCC on label hierarchies, although at the training stage we have access to the full label hierarchy, we only calculate pairwise distances on the finest label. For example, in Fig. 1b, assuming L3 doesn't exist, if we have $d_{\mathcal{T}}(\triangle_a, \triangle_b) = 2$, and $d_{\mathcal{T}}(\triangle_a, \square_c) = 4$, given corresponding Euclidean distances, the coarse-fine relationship is already embedded without knowing $d_{\mathcal{T}}(\triangle_{L1}, \square_{L1}) = 2$.

All experiments were trained on NVIDIA RTX A6000. Experiments in Sections 4.3 and 4.4 have the same training setting, except for some optimizer setting differences. For a fair comparison, we searched hyperparameters for Sec. 4.4 to achieve the best performance for all Flat methods, and apply the same setting to other objectives with or without CPCC. We want to highlight again that we use these hyperparameters to show CPCC's advantage over other methods in controlled settings, but not to reproduce SOTA results.

Compared with other approaches, curriculum learning in Sec. 4.3 contains two stages. Empirically, we find that a few first stage training iterations on coarse labels give a better initialization of the second stage on fine labels. The epochs of the first stage do not have a huge impact on the final performance, but less epochs on the second stage can harm the result. Therefore, we train more epochs on curriculum learning settings so that the fine accuracy for all without CPCC objectives are similar. We spend approximately the same number of epochs on fine labels for all experiments, and add some extra iterations for curriculum learning to train on coarse labels. Training epochs are the same for generalization experiments, given the fact that all representations have similar fine and coarse accuracy.

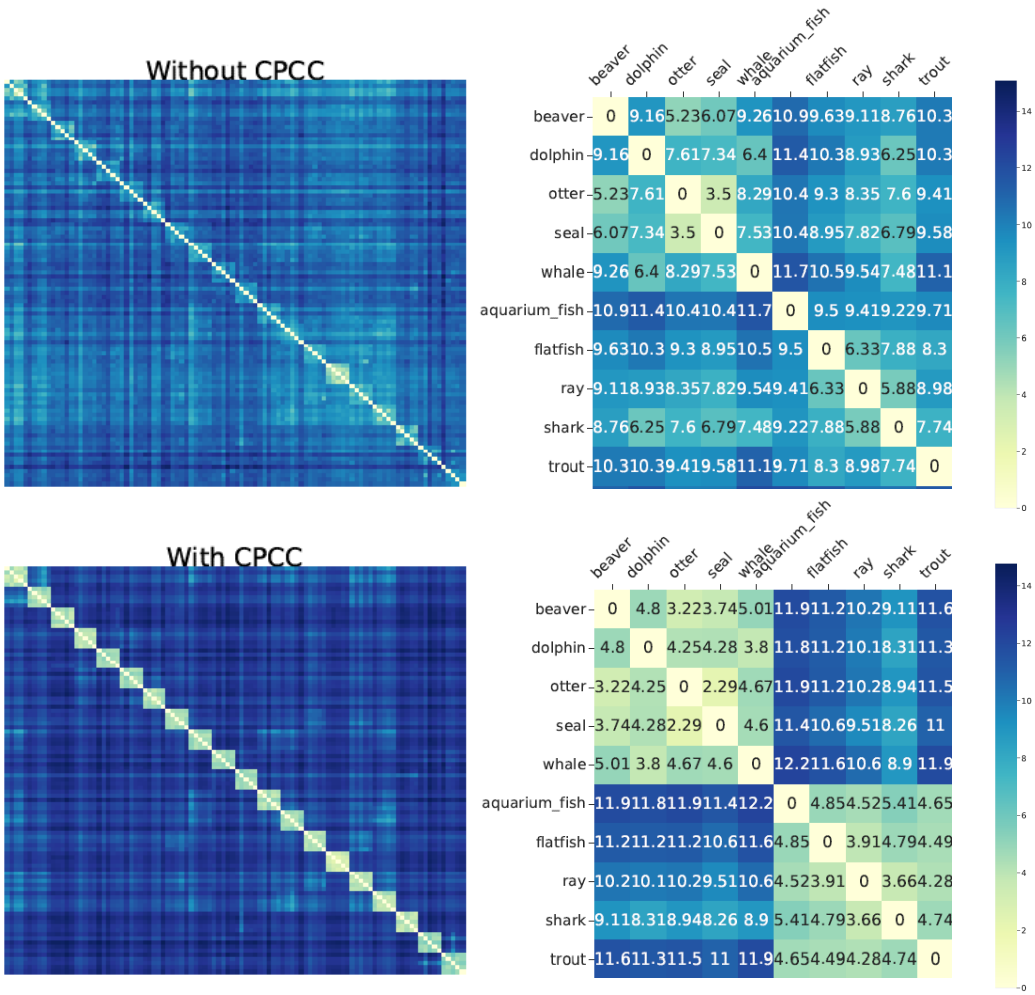


Figure 7: Distance between fine CIFAR100 classes with and without CPCC for **Multi-task**. Two matrices on the right are zoomed in to show the top left corner of the corresponding distance matrix on the left.

MNIST The feature encoder is composed of these layers sequentially: a 5×5 convolutional layer, 2×2 max pooling layer, a 5×5 convolutional layer, 2×2 max pooling layer, fully connected layer with 120 units output, and a fully connected layer with 50 units output. We set batch size as 256, epochs as 70, learning rate start from 0.05, divided by 5 every 25 steps, with momentum of 0.9, weight decay 5×10^{-4} of SGD optimizer. No data augmentation is applied. For curriculum learning, we train 18 epochs on coarse labels and 72 epochs on fine labels. We train one-shot transfer on mid level for 70 epochs, learning rate from 0.005 divided by 10 every 25 steps; fine level 70 epochs learning rate from 0.1 divided by 10 every 25 steps.

CIFAR100 We train ResNet18 from scratch. Following the original ResNet (He et al., 2015) paper, the first convolution layer has a 3×3 kernel size, and the max pooling layer is not included. We set batch size as 128, epochs as 200, learning rate starts from 0.1, divided by 5 every 60 steps, with momentum of 0.9, weight decay 5×10^{-4} of SGD optimizer. All images are normalized by CIFAR100's mean and standard deviation. The following data augmentations are applied on train set sequentially: images are zero padded by 4 pixels and cropped into original size; images are flipped horizontally by the chance of 50%; images are randomly rotated by -15 to 15 degrees. For curriculum learning, we train 50 epochs on coarse labels and 200 epochs on fine labels. We

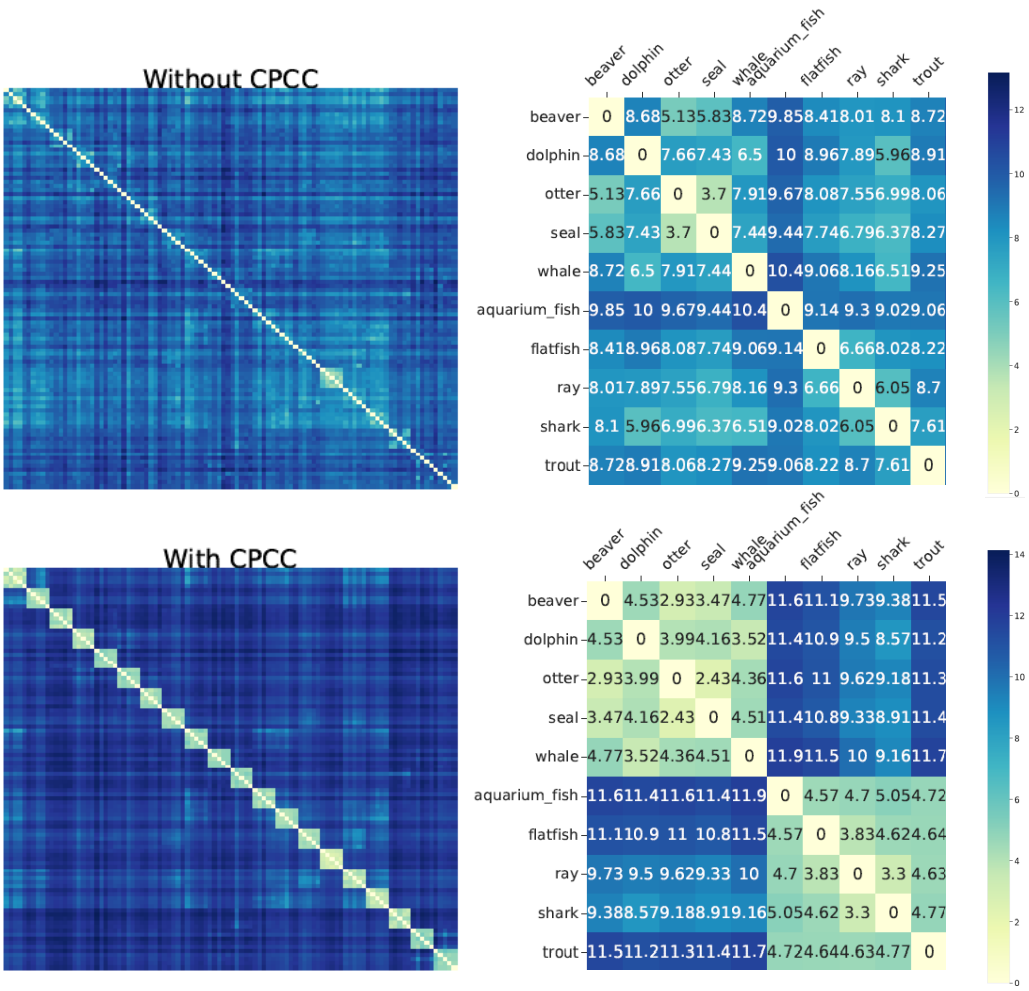


Figure 8: Distance between fine CIFAR100 classes with and without CPCC for **Curriculum**.

train one-shot transfer on mid and fine level for 100 epochs, learning rate from 0.05 divided by 10 every 60 steps.

BREEDS We train ResNet18 from scratch. The first convolution layer has a 7×7 kernel size. We follow BREEDS training setting: we set batch size as 128, initial learning rate as 0.1, weight decay 10^{-4} of SGD optimizer. On *ENTITY13* and *ENTITY30*, we train 300 epochs, learning rate divided by 10 every 100 steps; on *LIVING17* and *NONLIVING26*, we train 450 epochs, learning rate divided by 10 every 150 steps. For curriculum learning, on *ENTITY13* and *ENTITY30*, we train 75 epochs on coarse labels and 300 epochs on fine labels; on *LIVING17* and *NONLIVING26*, we train 108 epochs on coarse labels and 432 epochs on fine labels. Train set is augmented by a RandomResizedCrop into 224×224, flipped horizontally by the chance of 50 $\%$, and randomly change the brightness, contrast, saturation by 0.9 to 1.1 (ColorJitter). Test set is first resized into 256×256 and CenterCropped into 224×224.

On *LIVING17*, we train one-shot transfer on mid level for 150 epochs, learning rate 0.01; fine level 150 epochs learning rate from 0.1 divided by 10 at epoch 100. On *ENTITY13*, we train one-shot transfer on mid level for 100 epochs, learning rate starting from 0.1 divided by 10 every 50 steps; fine level 200 epochs learning rate from 0.05 divided by 10 at epoch 150. On *ENTITY30*, we train one-shot transfer on mid level for 100 epochs, learning rate 0.1 divided by 10 every 50 steps; fine level 150 epochs learning rate from 0.1 divided by 10 at epoch 100. On *NONLIVING26*, we train

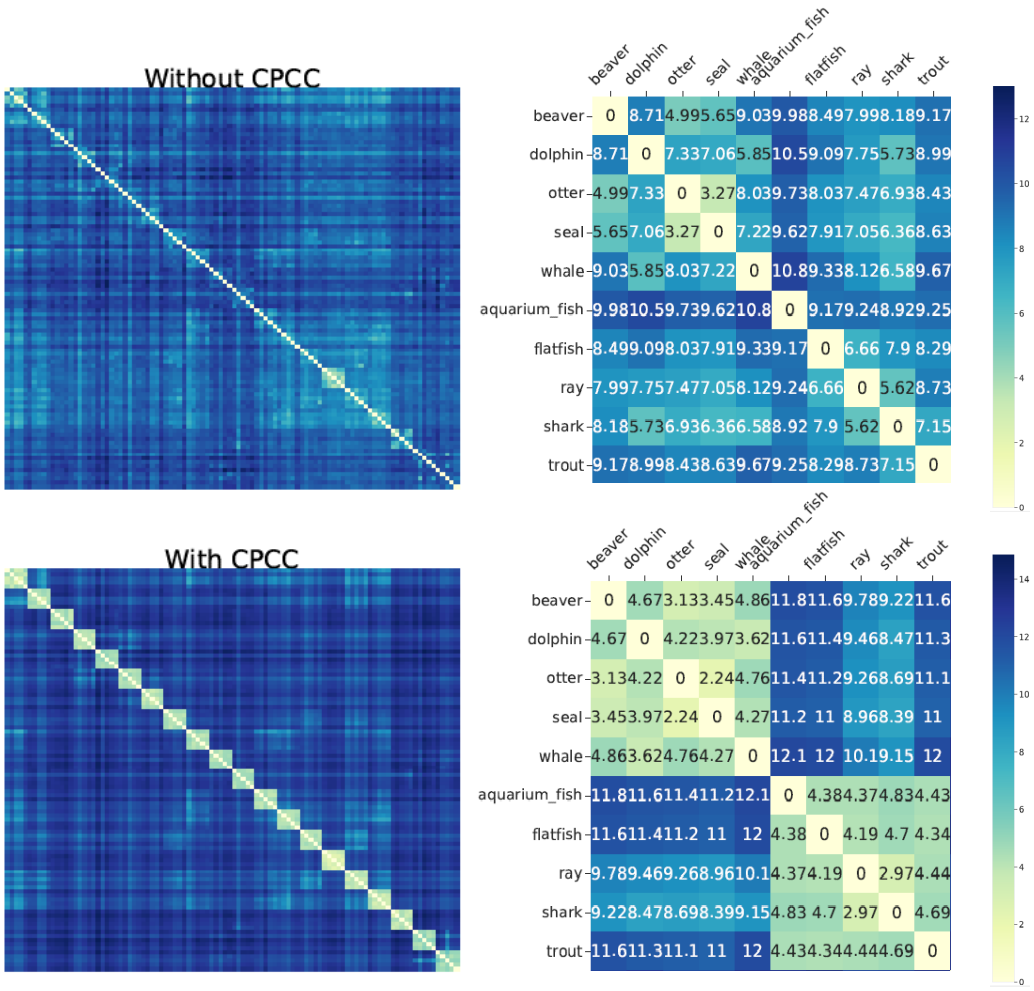


Figure 9: Distance between fine CIFAR100 classes with and without CPCC for **Sum Loss**.

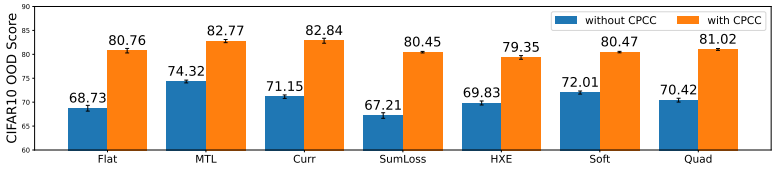


Figure 10: OOD detection score for all experiments. CPCC greatly boosts performance.

one-shot transfer on mid level for 100 epochs, learning rate 0.1 divided by 10 every 50 steps; fine level 150 epochs learning rate from 0.1 divided by 10 at epoch 100.

Table 4: Remaining results for Table 1. Since the MNIST's hierarchy is artificial, CPCC sometimes leads to slightly worse performance (though well within the standard deviation). On BREEDS and CIFAR100, CPCC is able to use the extra visually meaningful hierarchical information to provide gain across all metrics and objectives. Also, we observe that curriculum learning works well on most BREEDS setting and gives the highest CPCC and silhouette values.

Dataset	Objective	CPCC	silhouette	FineAcc	MidAcc	CoarseAcc	CoarserAcc
MNIST	MTL	42.65 (2.96)	20.87 (1.02)	99.21 (0.05)	**99.41 (0.02)**	**99.62 (0.02)**	N/A
	MTLCPCC	**99.96 (0.01)**	60.60 (0.50)	**99.26 (0.05)**	99.34 (0.07)	99.55 (0.04)	
	Curr	31.22 (4.99)	19.09 (1.17)	99.20 (0.12)	99.37 (0.08)	99.59 (0.06)	
	CurrCPCC	**99.96 (0.01)**	**60.90 (0.31)**	99.18 (0.10)	99.36 (0.05)	**99.62 (0.03)**	
	SumLoss	23.76 (4.10)	17.05 (0.62)	99.11 (0.08)	99.36 (0.06)	99.60 (0.06)	
	SumLossCPCC	**99.96 (0.01)**	60.07 (0.75)	99.10 (0.12)	99.30 (0.09)	99.56 (0.06)	
LIVING17^s	MTL	45.83 (0.78)	11.66 (0.09)	83.87 (0.28)	89.60 (0.23)	90.29 (0.27)	93.15 (0.37)
	MTLCPCC	90.71 (4.28)	45.05 (0.54)	84.77 (0.59)	90.84 (0.39)	91.53 (0.30)	94.07 (0.33)
	Curr	40.19 (0.71)	10.95 (0.12)	83.87 (0.40)	89.53 (0.54)	90.10 (0.49)	93.01 (0.52)
	CurrCPCC	**93.41 (0.34)**	**51.36 (0.46)**	**85.21 (0.42)**	90.91 (0.53)	91.59 (0.65)	94.07 (0.59)
	SumLoss	39.02 (1.10)	8.95 (0.26)	83.49 (0.63)	89.24 (0.46)	89.92 (0.31)	92.74 (0.19)
	SumLossCPCC	93.03 (0.46)	44.36 (0.28)	85.05 (0.23)	**91.07 (0.21)**	**91.79 (0.25)**	**94.56 (0.19)**
ENTITY13^s	MTL	43.55 (0.71)	5.19 (0.11)	82.47 (0.28)	84.80 (0.04)	91.13 (0.10)	93.97 (0.15)
	MTLCPCC	90.73 (0.34)	36.62 (0.27)	82.83 (0.23)	85.06 (0.21)	91.51 (0.24)	94.21 (0.17)
	Curr	37.03 (0.51)	3.98 (0.05)	83.41 (0.29)	85.39 (0.31)	91.15 (0.15)	93.98 (0.13)
	CurrCPCC	**92.38 (0.25)**	**41.06 (0.15)**	**83.86 (0.25)**	**86.12 (0.18)**	**91.90 (0.20)**	**94.55 (0.19)**
	SumLoss	37.99 (0.26)	3.51 (0.02)	82.50 (0.23)	84.86 (0.16)	90.83 (0.25)	93.68 (0.18)
	SumLossCPCC	89.66 (0.30)	35.08 (0.23)	82.97 (0.43)	85.26 (0.32)	91.56 (0.10)	94.24 (0.08)
ENTITY30^s	MTL	30.55 (0.26)	5.25 (0.12)	81.31 (0.32)	82.81 (0.19)	86.95 (0.15)	90.12 (0.15)
	MTLCPCC	72.00 (0.52)	25.92 (0.19)	81.90 (0.17)	83.58 (0.20)	87.66 (0.15)	90.72 (0.15)
	Curr	27.09 (0.20)	4.55 (0.05)	82.19 (0.45)	83.59 (0.42)	87.52 (0.25)	90.54 (0.31)
	CurrCPCC	**77.80 (0.50)**	**34.08 (0.19)**	**82.86 (0.20)**	**84.31 (0.20)**	**88.55 (0.09)**	**91.23 (0.14)**
	SumLoss	26.05 (0.29)	3.18 (0.09)	81.11 (0.17)	82.59 (0.21)	86.62 (0.14)	89.71 (0.10)
	SumLossCPCC	66.10 (0.63)	21.38 (0.20)	81.76 (0.26)	83.28 (0.26)	87.63 (0.25)	90.68 (0.18)
NONLIVING26^s	MTL	30.38 (0.73)	7.36 (0.10)	82.06 (0.24)	84.91 (0.15)	86.42 (0.22)	88.46 (0.31)
	MTLCPCC	79.49 (0.66)	30.55 (0.22)	82.84 (0.61)	85.91 (0.42)	87.60 (0.39)	89.45 (0.45)
	Curr	28.81 (0.41)	7.68 (0.08)	81.82 (0.65)	84.72 (0.69)	86.23 (0.73)	88.35 (0.54)
	CurrCPCC	**81.70 (0.71)**	**36.92 (0.25)**	**83.54 (0.28)**	**86.45 (0.30)**	**88.06 (0.29)**	**89.83 (0.32)**
	SumLoss	28.57 (0.69)	5.65 (0.15)	81.58 (0.97)	84.56 (0.71)	86.28 (0.63)	88.48 (0.56)
	SumLossCPCC	79.04 (0.95)	29.14 (0.36)	82.60 (0.51)	85.82 (0.54)	87.42 (0.41)	89.27 (0.51)

G Dataset Hierarchies

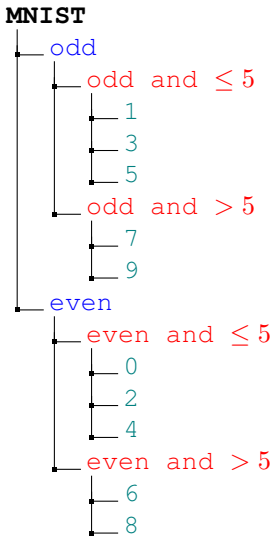
We visualize 4 levels of hierarchy (coarser, coarse, mid, fine) of MNIST, CIFAR, and BREEDS^s in Fig 11, and provide more details for Section 4.1 and 4.4 about how we constructed the hierarchy based on original labels. We can notice in BREEDS hierarchies, there are some repetitive labels[2]. One case is that the parent node is a dummy node which shares the same name as its child. For example, dummy 52 and n02773037 (WordNet ID) are both called "bags". Santurkar et al. (2020) inserted dummy nodes into the ImageNet hierarchy to adjust the granularity of each level. Since we create mid labels by backtracking the fine classes by 1 level, the other case of repetitive labels happens when a coarse label coincides with the mid label (*i.e.*, if the coarse label is exactly one level higher than the fine label). Unlike on MNIST and CIFAR, we do not want to create some artificial new classes on BREEDS. Therefore, we can also treat repetitive nodes as dummy nodes as in the first case, which only affects mid class results.

[2]All label names of BREEDS can be found in `https://github.com/MadryLab/BREEDS-Benchmarks/blob/master/imagenet_class_hierarchy/modified/node_names.txt`

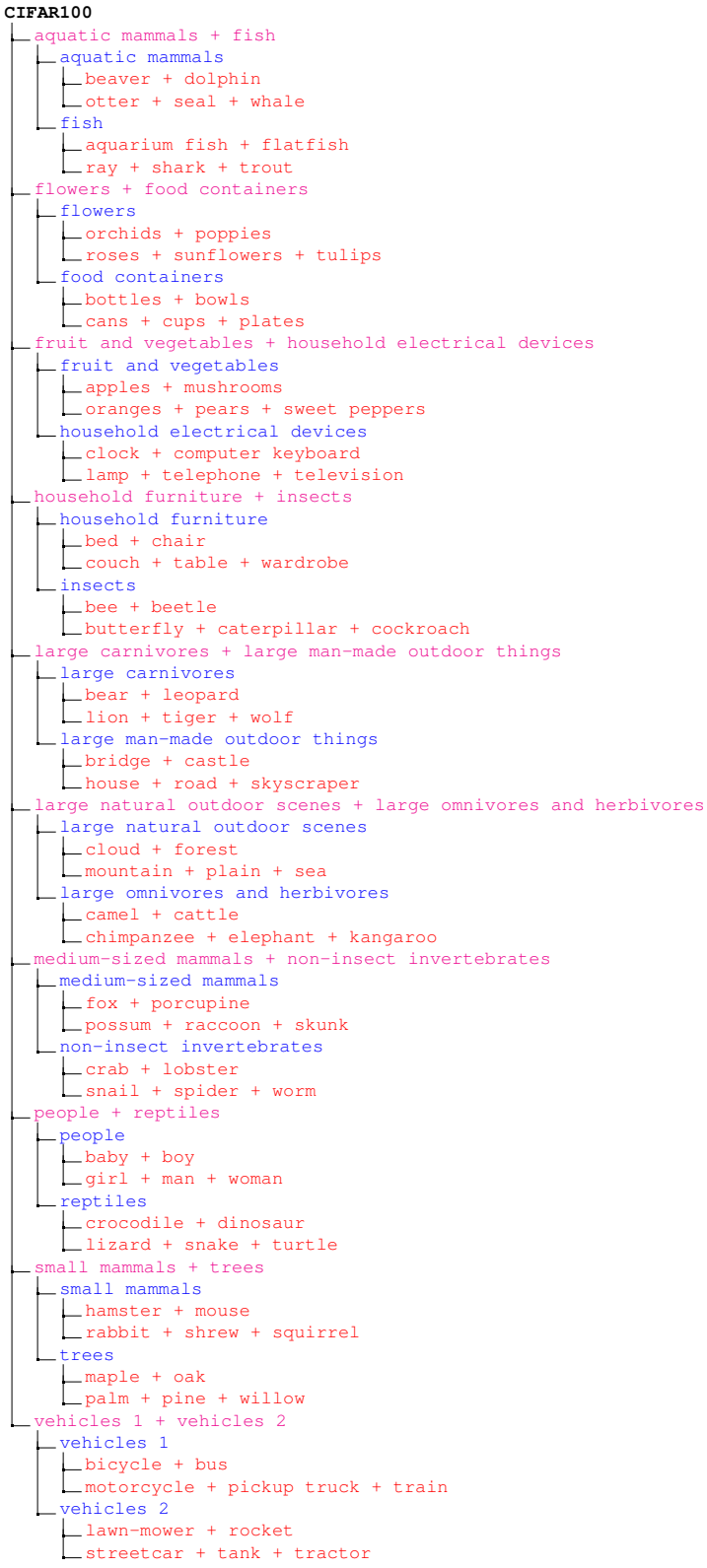
Table 5: Remaining results for Table 2. As in Table 4, MNIST's results occasionally (SumLoss-CPCC MidAcc, Curriculum CoarseAcc) show CPCC is worse than original objective due to artificial hierarchy, but on BREEDS where we have a more reasonable visual hierarchy, the conclusion is consistent. CPCC leads to a better coarse/coarser in-hierarchy performance, mostly better mid level performance, but not on fine level.

Dataset	Objective	FineAcc^1	MidAcc^1	CoarseAcc^0	CoarserAcc^0
MNIST^t	MTL	63.44 (11.39)	57.40 (4.35)	53.05 (1.69)	N/A
	MTLCPCC	**63.68 (4.89)**	58.56 (0.39)	56.94 (3.00)	
	Curr	56.12 (8.64)	50.69 (8.27)	**60.36 (3.38)**	
	CurrCPCC	57.87 (5.86)	58.09 (9.80)	59.91 (2.36)	
	SumLoss	62.10 (8.93)	**60.52 (6.53)**	52.44 (2.43)	
	SumLossCPCC	62.11 (14.76)	56.27 (6.53)	55.24 (2.18)	
LIVING17^t	MTL	30.92 (2.24)	35.00 (2.79)	55.39 (0.46)	70.75 (0.42)
	MTLCPCC	31.84 (3.04)	39.45 (3.56)	**61.76 (0.14)**	72.24 (0.58)
	Curr	29.70 (2.29)	32.71 (2.66)	54.33 (0.27)	70.20 (0.15)
	CurrCPCC	**32.36 (1.71)**	**40.15 (3.15)**	57.24 (0.77)	**72.84 (0.68)**
	SumLoss	31.26 (2.21)	32.09 (2.17)	54.41 (0.36)	70.46 (0.27)
	SumLossCPCC	32.07 (2.69)	38.91 (2.15)	57.18 (0.69)	72.79 (0.38)
ENTITY13^t	MTL	16.17 (1.87)	**22.18 (0.96)**	59.91 (0.53)	70.68 (0.29)
	MTLCPCC	13.04 (1.67)	18.74 (1.02)	61.76 (0.14)	72.47 (0.30)
	Curr	**16.29 (1.50)**	21.84 (1.15)	59.63 (0.49)	70.83 (0.34)
	CurrCPCC	12.71 (1.43)	18.96 (0.76)	**62.04 (0.31)**	**72.54 (0.28)**
	SumLoss	16.01 (1.64)	20.99 (0.34)	59.74 (0.56)	70.74 (0.42)
	SumLossCPCC	13.74 (1.57)	18.17 (0.43)	61.43 (0.59)	72.18 (0.53)
ENTITY30^t	MTL	21.84 (1.09)	22.98 (1.49)	47.17 (0.40)	60.73 (0.49)
	MTLCPCC	20.54 (0.66)	25.15 (1.33)	48.70 (0.43)	62.65 (0.28)
	Curr	**22.41 (1.01)**	22.84 (1.97)	47.37 (0.39)	60.83 (0.53)
	CurrCPCC	20.40 (0.95)	25.59 (1.79)	**49.70 (0.21)**	**63.40 (0.23)**
	SumLoss	21.51 (1.16)	22.35 (1.62)	46.51 (0.60)	60.44 (0.45)
	SumLossCPCC	21.48 (0.64)	**26.02** (0.93)	48.97 (0.42)	62.41 (0.39)
NONLIVING26^t	MTL	24.00 (1.64)	26.07 (0.59)	39.95 (0.64)	53.74 (0.53)
	MTLCPCC	23.83 (1.86)	27.35 (1.55)	41.24 (0.25)	54.95 (0.33)
	Curr	23.55 (1.44)	26.50 (1.37)	39.97 (0.46)	54.49 (0.60)
	CurrCPCC	23.93 (0.99)	**28.51 (2.51)**	**42.84 (0.60)**	**56.88(0.95)**
	SumLoss	**24.18 (1.40)**	25.75 (1.62)	40.02 (0.61)	53.78 (0.66)
	SumLossCPCC	24.05 (1.01)	27.37 (2.65)	41.92 (0.55)	55.64 (0.55)

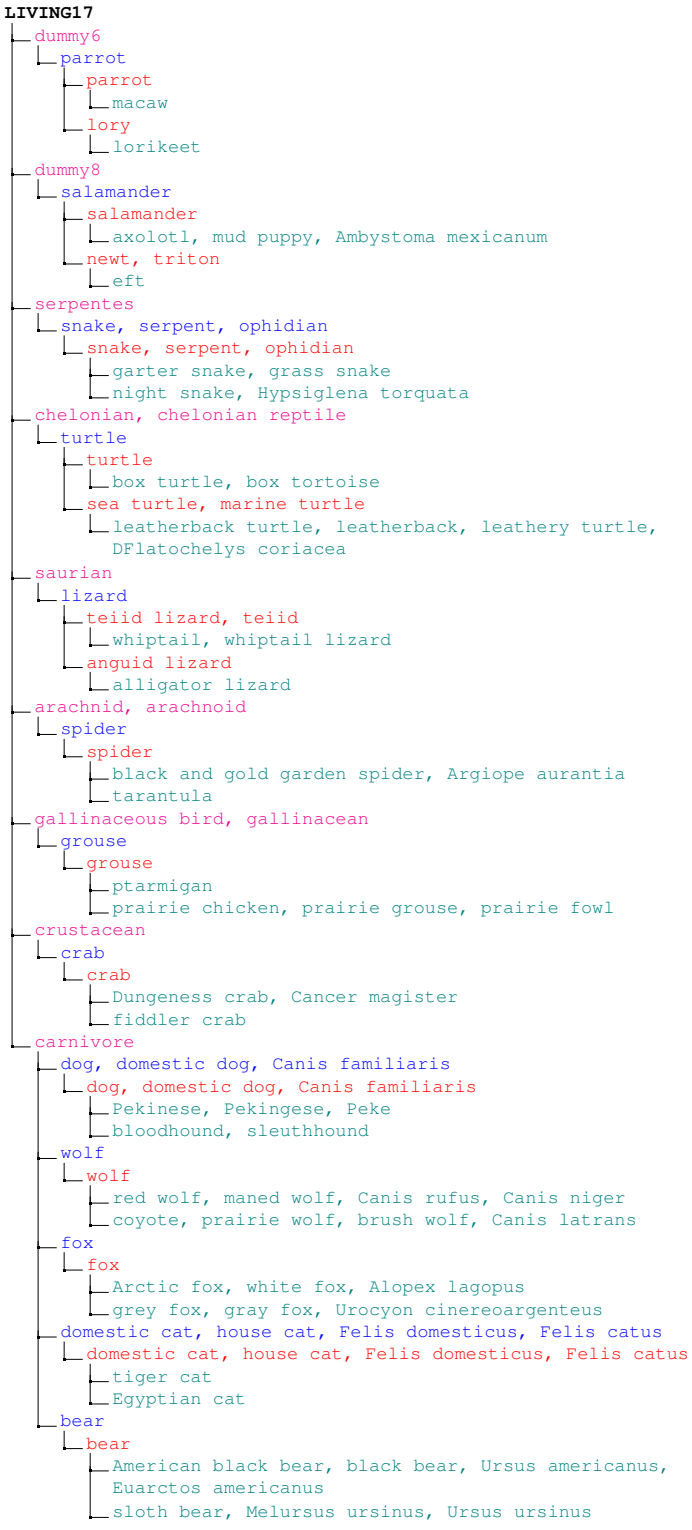
Figure 11: Hierarchies for all datasets.



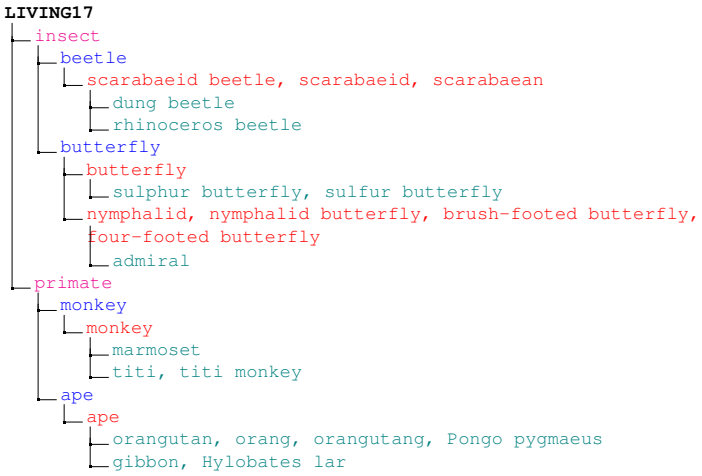
(a) MNIST hierarchy.



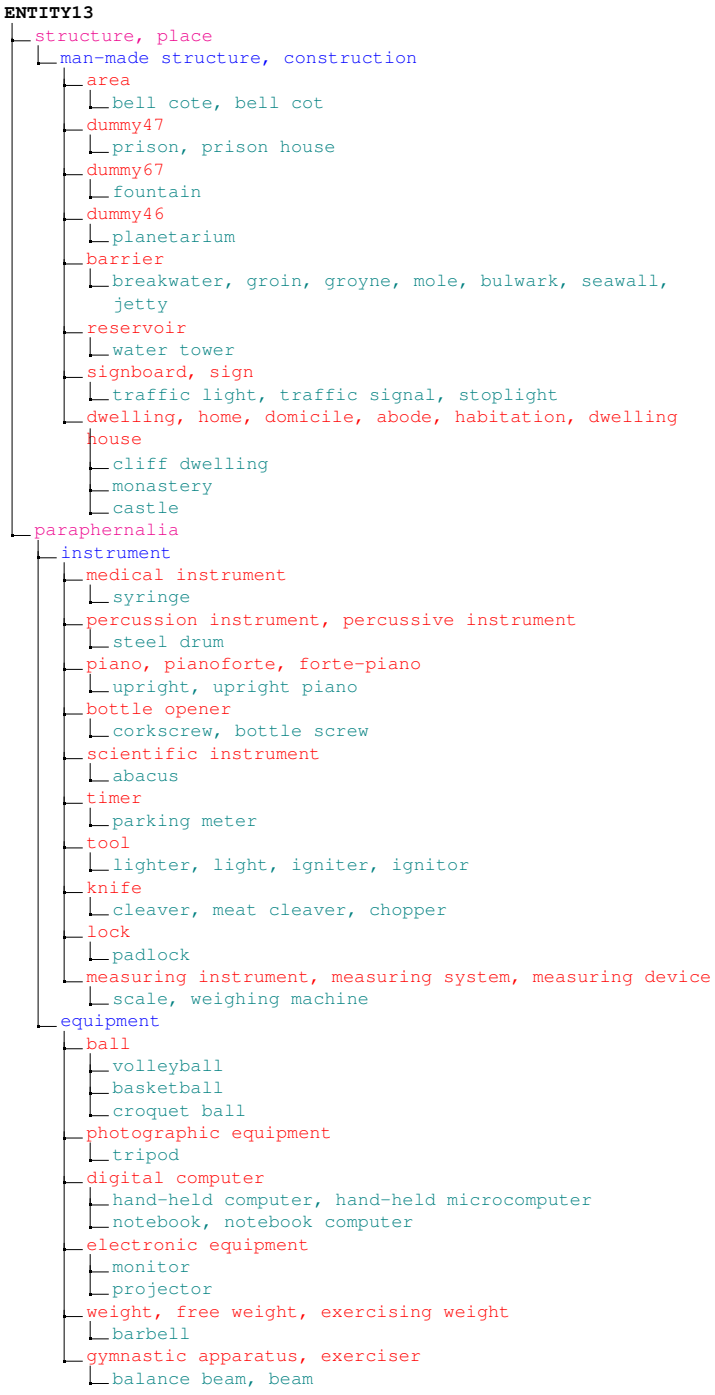
(b) CIFAR hierarchy. The leaf nodes (100 fine classes) are evident from the mid level node names (they are simply separated by a +). We create the mid-level split based on alphabetical order. Label names are from `https://www.cs.toronto.edu/~kriz/cifar.html`.



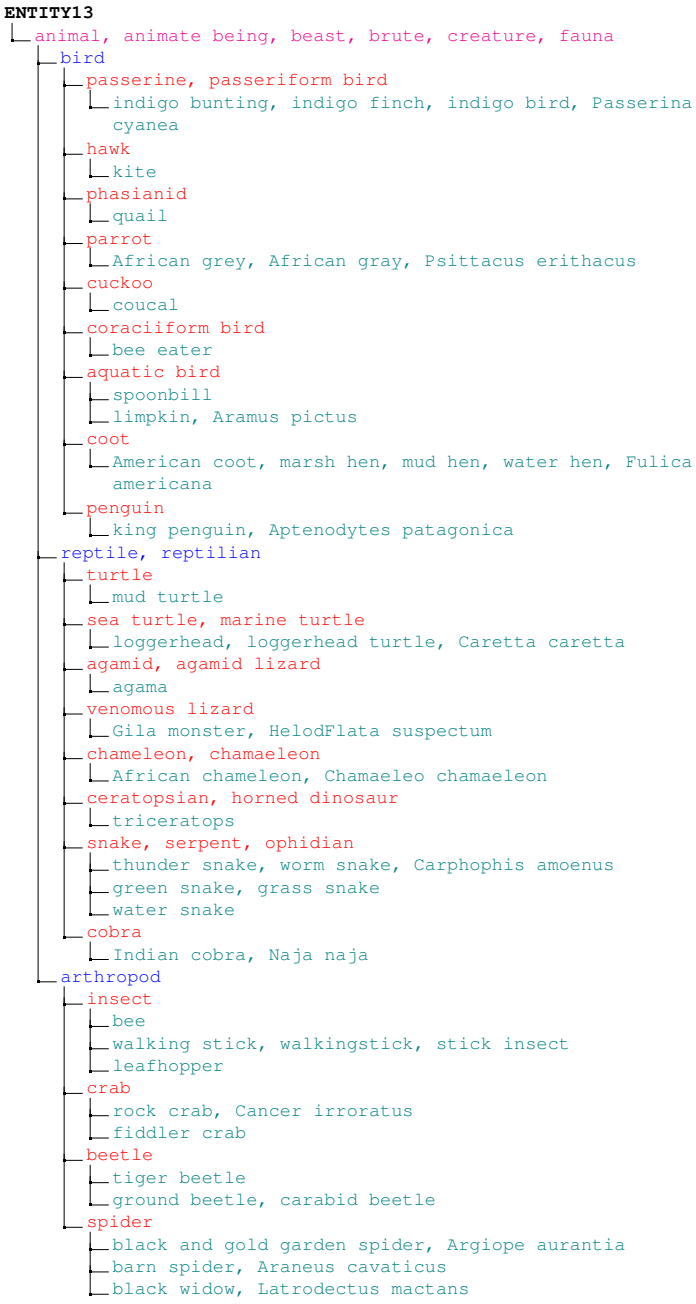
(c) LIVING17 hierarchy (part 1).



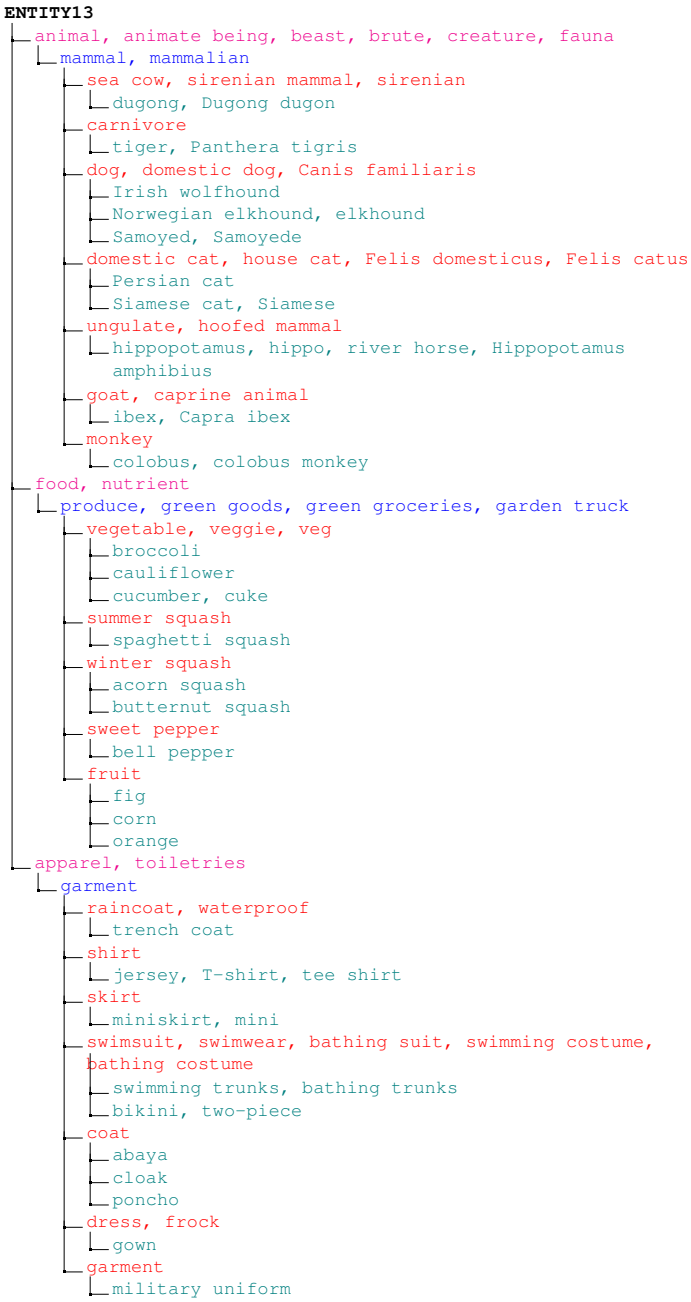
(d) LIVING17 hierarchy (part 2).



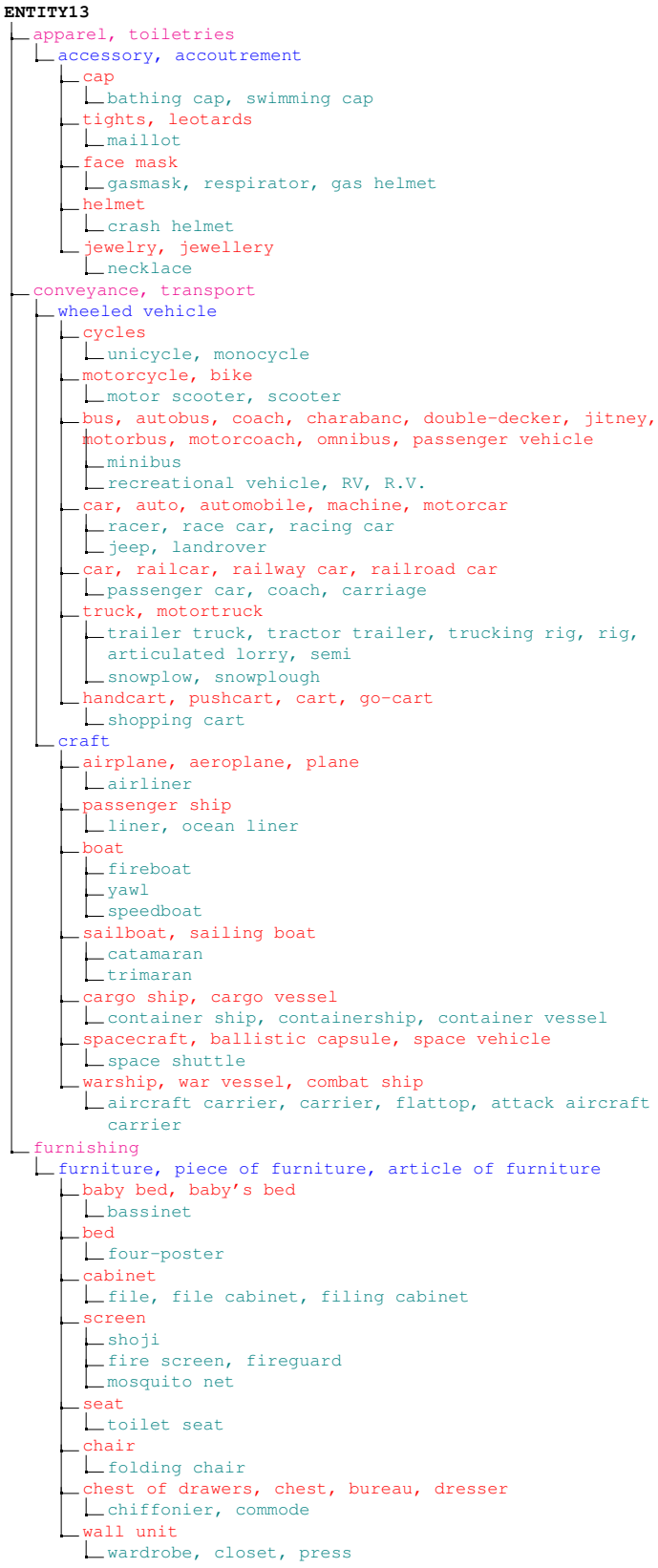
(e) ENTITY13 hierarchy (part 1).



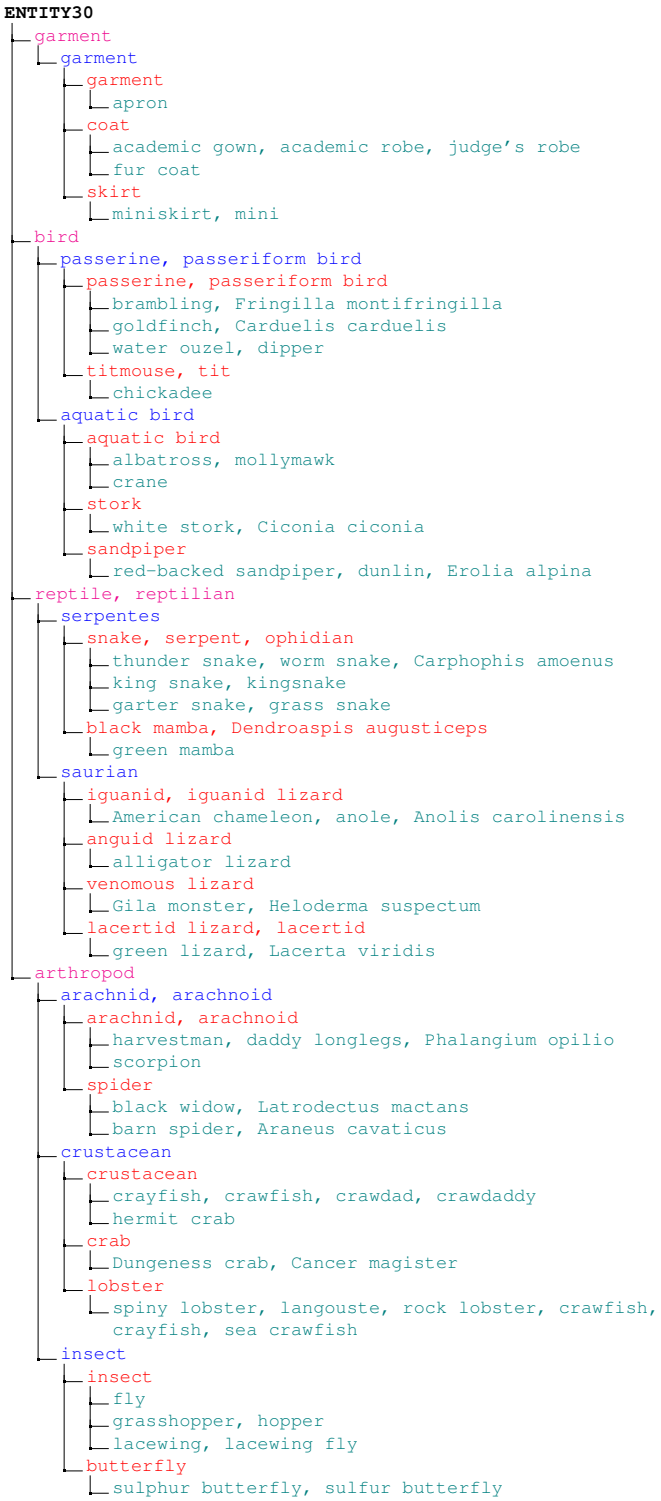
(f) ENTITY13 hierarchy (part 2).



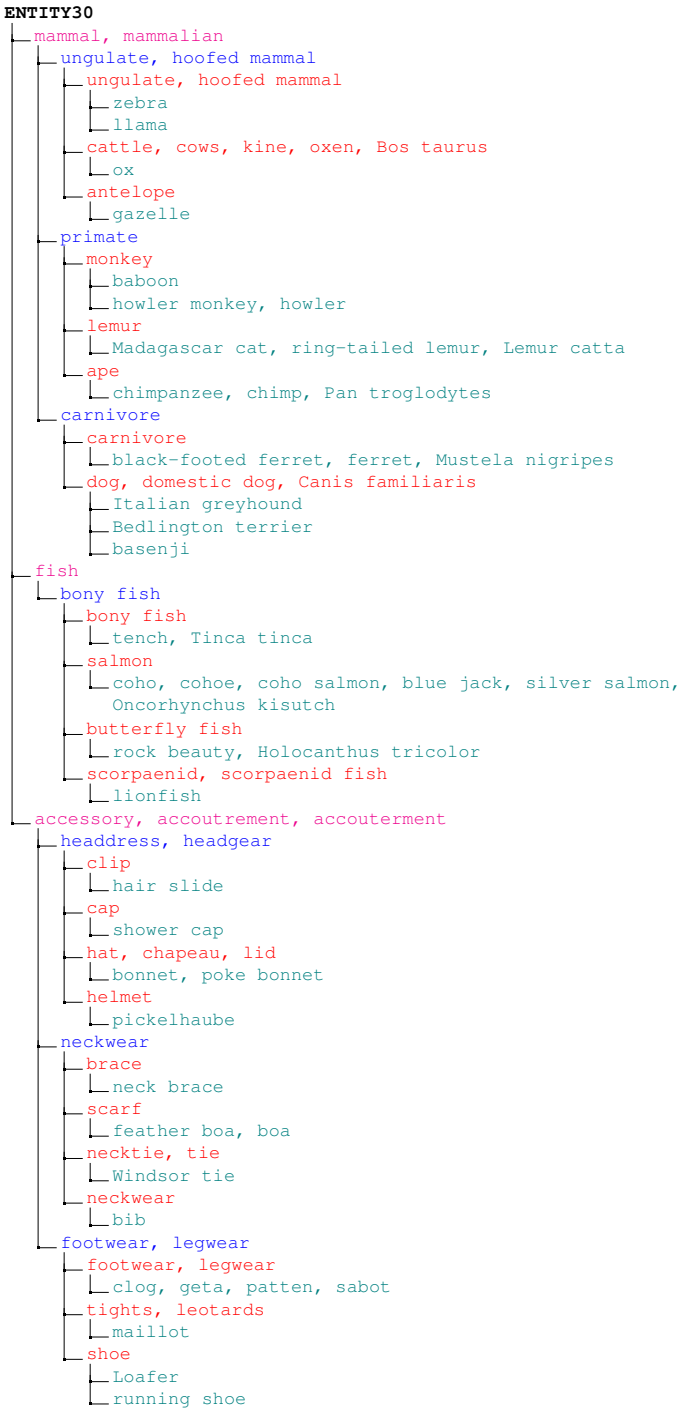
(g) ENTITY13 hierarchy (part 3).



(h) ENTITY13 hierarchy (part 4).

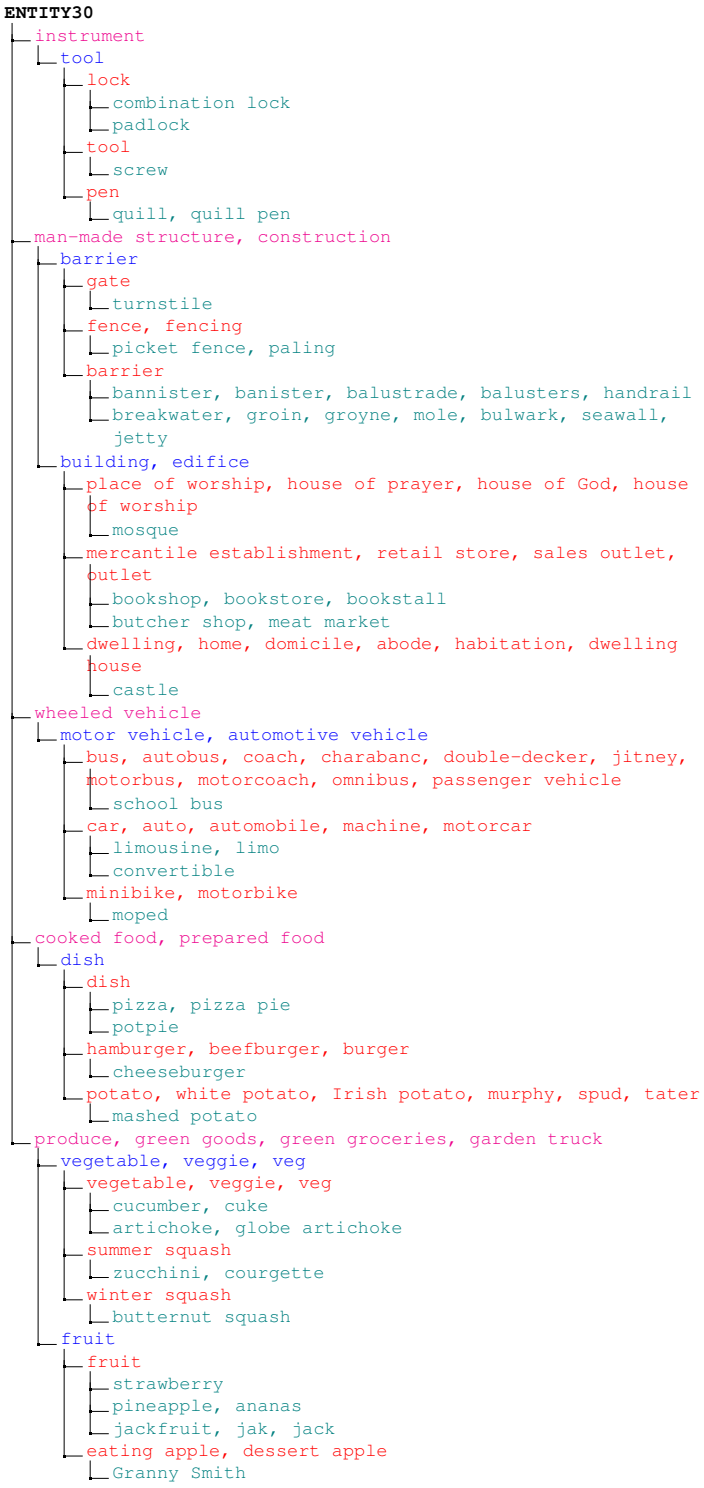


(i) ENTITY30 hierarchy (part 1).



(j) ENTITY30 hierarchy (part 2).

(k) ENTITY30 hierarchy (part 3).



(l) ENTITY30 hierarchy (part 4).

NONLIVING26

- armor
  - body armor, body armour, suit of armor, suit of armour, coat of mail, cataphract
    - body armor, body armour, suit of armor, suit of armour, coat of mail, cataphract
      - breastplate, aegis, egis
      - bulletproof vest
- bag
  - bag
    - bag
      - plastic bag
      - purse
- area
  - roof
    - roof
      - dome
      - vault
- barrier
  - fence, fencing
    - fence, fencing
      - chainlink fence
    - rail fence
      - worm fence, snake fence, snake-rail fence, Virginia fence
- building, edifice
  - dwelling, home, domicile, abode, habitation, dwelling house
    - dwelling, home, domicile, abode, habitation, dwelling house
      - monastery
      - palace
  - mercantile establishment, retail store, sales outlet, outlet
    - mercantile establishment, retail store, sales outlet, outlet
      - butcher shop, meat market
      - barbershop
  - outbuilding
    - shed
      - apiary, bee house
    - outbuilding
      - greenhouse, nursery, glasshouse
- electronic equipment
  - digital computer
    - digital computer
      - desktop computer
      - laptop, laptop computer
- garment
  - coat
    - coat
      - fur coat
      - lab coat, laboratory coat
  - skirt
    - skirt
      - miniskirt, mini
      - hoopskirt, crinoline
- headdress, headgear
  - hat, chapeau, lid
    - hat, chapeau, lid
      - bearskin, busby, shako
      - bonnet, poke bonnet
- kitchen utensil
  - pot
    - pot
      - teapot
      - Dutch oven
- measuring instrument, measuring system, measuring device
  - timepiece, timekeeper, horologe
    - timer
      - stopwatch, stop watch
    - watch, ticker
      - digital watch

(m) NONLIVING26 hierarchy (part 1).

(n) NONLIVING26 hierarchy (part 2).