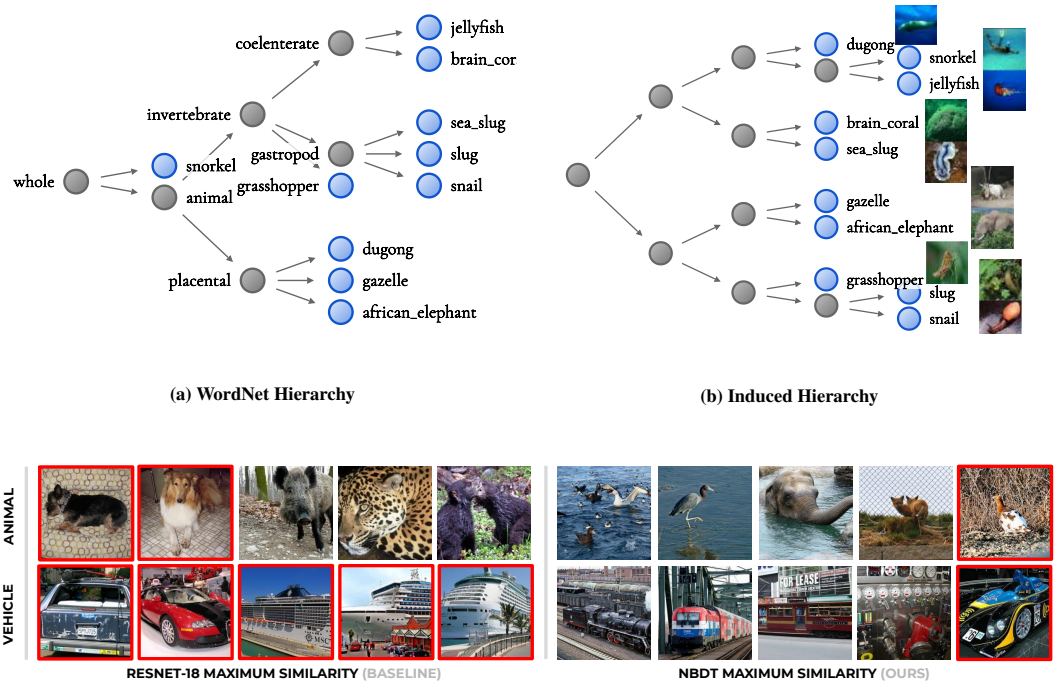

(a) WordNet Hierarchy

(b) Induced Hierarchy

Figure 8: Maximum Similarity Examples. We run two CIFAR10-trained models, one trained with tree supervision loss (NBDT) and one without tree supervision loss (ResNet18). We compute the induced hierarchy of both models and find samples most similar to the *Animal*, and *Motor Vehicle* concepts. Each row represents an inner node, and the red borders indicate images that contain CIFAR10 classes. (1) Note that NBDT's concept of an animal includes classes and contexts it was not trained on; aquatic animals (top-right) and trains (bottom-right) are not a part of CIFAR10. In contrast, ResNet18 largely finds examples closely related to existing CIFAR10 classes (dog, car, boat). This is qualitative evidence that NBDTs better generalize.

A Acknowledgments

In addition to NSF CISE Expeditions Award CCF-1730628, UC Berkeley research is supported by gifts from Alibaba, Amazon Web Services, Ant Financial, CapitalOne, Ericsson, Facebook, Futurewei, Google, Intel, Microsoft, Nvidia, Scotiabank, Splunk and VMware. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE 1752814.

B Explainability

In this section, we expand on details for interpretability as presented in the original paper, with an emphasis on qualitative use of the hierarchy.

B.1 Maximum Similarity Examples to Visualize Generalization

We (1) visually confirm the hypothesized meaning of each node by identifying the most "representative" samples, and (2) check that these "representative" samples represent that category (e.g., *Animal*) and not just the training classes under that category. We define "representative" samples, or maximum similarity examples, to be samples with embeddings most similar to an inner node's representative. We visualize these examples for a model before and after the tree supervision loss (NBDT and ResNet18, respectively). The models are trained on CIFAR10, but samples are drawn from ImageNet. We observe that maximum similarity examples for NBDT contain more unseen classes than ResNet18 (Figure 8). This suggests that our NBDT is better able to capture high-level concepts such as *Animal*, which is quantitatively confirmed by the superclass evaluation in Table 6.

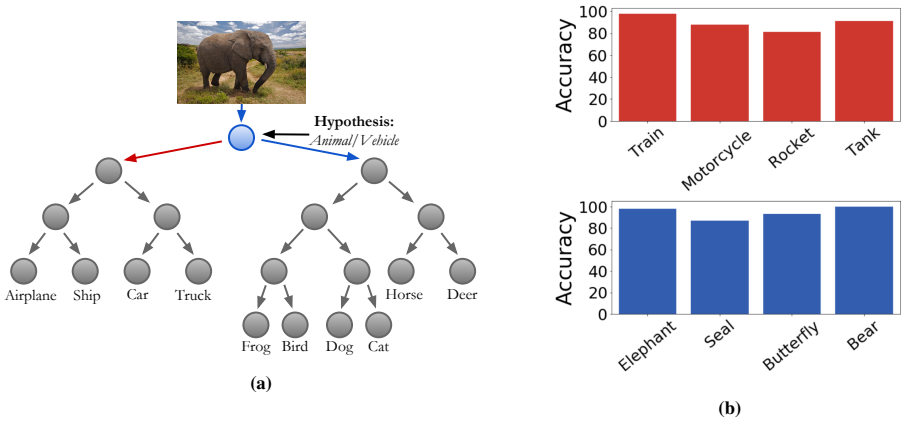

Figure 9: A Node's meaning. (Left) Visualization of node hypothesis test performed on a CIFAR10-trained WideResNet28x10 model, by sampling from CIFAR100 validation set for OOD classes. **(Right)** Classification accuracy is high (80-95%) given unseen CIFAR100 samples of *Vehicles* (top) and *Animals* (bottom), for the WordNet-hypothesized *Animal/Vehicle* node.

B.2 Explainability of Nodes' Visual Meanings

This section describes the method used in Table 6 in more detail. Since the induced hierarchy is constructed using model weights, the intermediate nodes are not forced to split on foreground objects. While hierarchies like WordNet provide hypotheses for a node's meaning, the tree may split on unexpected contextual and visual attributes such as *underwater* and *on land*, depicted in Figure 7b. To diagnose a node's visual meaning, we perform the following 4-step test:

1. Posit a hypothesis for the node's meaning (e.g. *Animal vs. Vehicle*). This hypothesis can be computed automatically from a given taxonomy or deduced from manual inspection of each child's leaves (Figure 9).
2. Collect a dataset with new, unseen classes that test the hypothesised meaning from step 1 (e.g. *Elephant* is an unseen *Animal*). Samples in this dataset are referred to as out-of-distribution (OOD) samples, as they are drawn from a separate labeled dataset.
3. Pass samples from this dataset through the node. For each sample, check whether the selected child node agrees with the hypothesis.
4. The accuracy of the hypothesis is the percentage of samples passed to the correct child. If the accuracy is low, repeat with a different hypothesis.

Figure 9a depicts the CIFAR10 tree induced by a WideResNet28x10 model trained on CIFAR10. The WordNet hypothesis is that the root note splits on *Animal vs. Vehicle*. We use the CIFAR100 validation set as out-of-distribution images for *Animal* and *Vehicle* classes that are unseen at training time. We then compute the hypothesis' accuracy. Figure 9b shows our hypothesis accurately predicts which child each unseen-class's samples traverse.

B.3 How Model Accuracy Affects Interpretability

Induced hierarchies are determined by the proximity of class weights, but classes that are close in weight space may not have similar visual meaning: Figure 10 depicts the trees induced by WideResNet28x10 and ResNet10, respectively. While the WideResNet induced hierarchy (Figure 10a) groups visually-similar classes, the ResNet (Figure 10b) induced hierarchy does not, grouping classes such as *Frog*, *Cat*, and *Airplane*. This disparity in visual meaning is explained by WideResNet's 4% higher accuracy: we believe that higher-accuracy models exhibit more visually-sound weight spaces. Thus, unlike previous work, NBDTs feature better interpretability with higher accuracy, instead of sacrificing one for the other. Furthermore, the disparity in hierarchies indicates that a model with low accuracy will not provide interpretable insight into high-accuracy decisions.

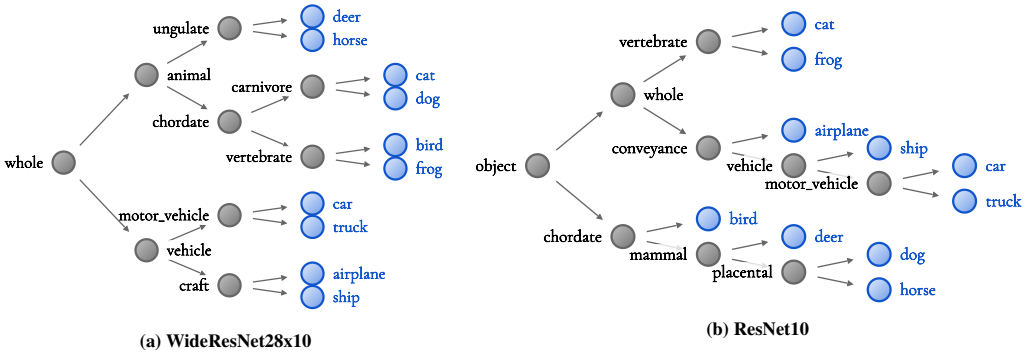

Figure 10: CIFAR10 induced hierarchies, with automatically-generated WordNet hypotheses for each node. The higher-accuracy (a) WideResNet (97.62% acc) has a more sensible hierarchy than (b) ResNet's (93.64% acc): The former groups all *Animal*s together, separate from all *Vehicles*. By contrast, the latter groups *Airplane*, *Cat*, and *Frog*.

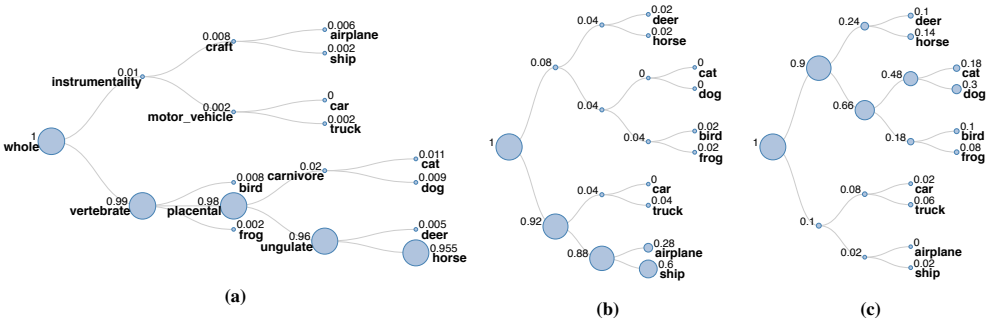

Figure 11: Visualization of path traversal frequency on an induced hierarchy for CIFAR10. (a) **In-Distribution: *Horse*** is a training class and thus sees highly focused path traversals. (b) **Unseen Class: *Seashore*** is largely classified as *Ship* despite not containing any objects, exhibiting model reliance on context (water). (c) **Unseen Class: *Teddy Bear*** is classified as *Dog*, for sharing visual attributes like color and texture.

B.4 Visualization of Tree Traversal

Frequency of path traversals additionally provide insight into general model behavior. Figure 11 shows frequency of path traversals for all samples in three classes: a seen class, an unseen class but with seen context, and an unseen class with unseen context.

Seen class, seen context: We visualize tree traversals for all samples in CIFAR10's *Horse* class (Figure 11a). As this class is present during training, tree traversal highlights the correct path with extremely high frequency. **Unseen class, seen context**: In Figure 11b, we visualize tree traversals for TinyImagenet's *Seashore* class. The model classifies 88% of *Seashore* samples as "vehicle with blue context," exhibiting reliance on context for decision-making. **Unseen class, unseen context**: In Figure 11c, we visualize traversals for TinyImagenet's *Teddy Bear*. The model classifies 90% as *Animal*, belying the model's generalization to stuffed animals. However, the model disperses samples among animals more evenly, with the most furry animal *Dog* receiving the most *Teddy Bear* samples (30%).

C Hierarchical Softmax and Conditional Execution

In the context of neural netework and decision tree hybrids, many works (Shazeer et al., 2017; Keskin & Izadi, 2018; Yang et al., 2019; Tanno et al., 2019) leverage conditional execution to improve computational efficiency in a hierarchical classifier. One motivation is to handle large-scale classification problems.

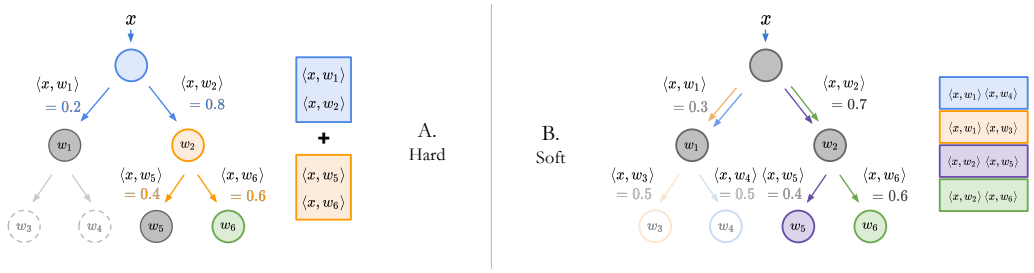

Figure 12: Tree Supervision Loss has two variants: **Hard Tree Supervision Loss (A)** defines a cross entropy term per node. This is illustrated with the blue box for the blue node and the orange box for the orange node. The cross entropy is taken over the child node probabilities. The green node is the leaf representing a class label. The dotted nodes are not included in the path from the label to the root, so do not have a defined loss. **Soft Tree Supervision Loss (B)** defines a cross entropy loss over all leaf probabilities. The probability of the green leaf is the product of the probabilities leading up to the root (in this case, $\langle x, w_2 \rangle \langle x, w_6 \rangle = 0.6 \times 0.7$). The probabilities for the other leaves are similarly defined. Each leaf probability is represented with a colored box. The cross entropy is then computed over this leaf probability distribution, represented by the colored box stacked on one another.

C.1 Hard Tree Supervision Loss

An alternative loss would be hierarchical softmax – in other words, one cross entropy loss per decision rule. We denote this the *hard tree supervision loss*, as we construct a variant of hierarchical softmax that (a) supports arbitrary depth trees and (b) is defined over a single, un-augmented fully-connected layer (e.g. k-dimensional output for a k-leaf tree). The original neural network's loss $\mathcal{L}_{\text{original}}$ minimizes cross entropy across the classes. For a k-class dataset, this is a k-way cross entropy loss. Each internal node's goal is similar: minimize cross-entropy loss across the child nodes. For node i with c children, this is a c-way cross entropy loss between predicted probabilities $\mathcal{D}(i)_{\text{pred}}$ and labels $\mathcal{D}(i)_{\text{label}}$. We refer to this collection of new loss terms as the *hard tree supervision loss* (Eq. 4). The individual cross entropy losses for each node are scaled so that the original cross entropy loss and the tree supervision loss are weighted equally, by default. If we assume N nodes in the tree, excluding leaves, then we would have $N + 1$ different cross entropy loss terms – the original cross entropy loss and N hard tree supervision loss terms. This is $\mathcal{L}_{\text{original}} + \mathcal{L}_{\text{hard}}$, where:

$$\mathcal{L}_{\text{hard}} = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\textsc{CrossEntropy}(\mathcal{D}(i)_{\text{pred}}, \mathcal{D}(i)_{\text{label}})}_{\text{over the } c \text{ children for each node}}. \tag{4}$$

C.2 Hard Inference

Hard inference is more intuitive: Starting at the root node, each sample is sent to the child with the most similar representative. We continue picking and traversing the tree until we reach a leaf. The class associated with this leaf is our prediction (Figure 1, A. Hard). More precisely, consider a tree with nodes indexed by i with set of child nodes $C(i)$. Each node i produces a probability of child node $j \in C(i)$; this probability is denoted $p(j|i)$. Each node thus picks the next node using $\text{argmax}_{j \in C(i)} p(j|i)$.

Whereas this inference mode is more intuitive, it underperforms soft inference (Figure 7). Furthermore, note that hard tree supervision loss (*i.e.* modified hierarchical softmax) appears to more specifically optimize hard inference. Despite that, hard inference performs worse (Figure 8) with hard tree supervision loss than the "soft" tree supervision loss (Sec 3.4) used in the main paper.

D Implementation

Our inference strategy, as outlined above and in Sec. 3.1 of the paper, includes two phases: (1) featurizing the sample using the neural network backbone and (2) running the embedded decision rules. However, in practice, our inference implementation does not need to run inference with the

Table 7: Comparisons of Inference Modes Hard inference performs worse than soft inference. See Table 1 in the main manuscript for a comparison against baselines.

Method	Backbone	CIFAR10	CIFAR100	TinyImageNet
NN	WideResNet28x10	*97.62%*	*82.09%*	*67.65%*
NBDT-H (Ours)	WideResNet28x10	97.55%	82.21%	64.39%
NBDT-S (Ours)	WideResNet28x10	**97.55%**	**82.97%**	**67.72%**
NN	ResNet18	*94.97%*	*75.92%*	*64.13%*
NBDT-H (Ours)	ResNet18	94.50%	74.29%	61.60%
NBDT-S (Ours)	ResNet18	**94.82%**	**77.09%**	**63.77%**

Table 8: Tree Supervision Loss Training the NBDT with the tree supervision loss ("TSL") is superior to (a) training with a hierarchical softmax ("HS") and to (b) omitting extra loss terms. ("None"). Δ is the accuracy difference between our soft loss and hierarchical softmax.

Dataset	Backbone	NN	Inference	None	TSL	HS	Δ
CIFAR10	ResNet18	94.97%	Hard	94.32%	94.50%	93.94%	+0.56%
CIFAR10	ResNet18	94.97%	Soft	94.38%	94.82%	93.97%	+0.85%
CIFAR100	ResNet18	75.92%	Hard	57.63%	74.29%	73.23%	+0.94%
CIFAR100	ResNet18	75.92%	Soft	61.93%	77.09%	74.09%	+1.83%
TinyImageNet	ResNet18	64.13%	Hard	39.57%	61.60%	58.89%	+2.71%
TinyImageNet	ResNet18	64.13%	Soft	45.51%	63.77%	61.12%	+2.65%

backbone, separately. In fact, our inference implementation only requires the logits $\hat{y}$ outputted by the network. This is motivated by the knowledge that the average of inner products is equivalent to the inner product of averages. Knowing this, we have the following equivalence, given the fully-connected layer weight matrix W, its row vectors w_i, featurized sample x, and the classes C we are currently interested in.

$$\langle x, \frac{1}{n}\sum_{i=1}^{|C|} w_i \rangle = \frac{1}{n}\sum_{i=1}^{|C|} \langle x, w_i \rangle = \frac{1}{n}\sum_{i=1}^{|C|} \hat{y}_i, i \in C \tag{5}$$

Thus, our inference implementation is simply performed using the logits $\hat{y}$ output by the network.

E Experimental Setup

To reiterate, our best-performing models for both hard and soft inference were obtained by training with the soft tree supervision loss. All CIFAR10 and CIFAR100 experiments weight the soft loss terms by 1. All TinyImagenet and Imagenet experiments weight the soft loss terms by 10. We found that hard loss performed best when the hard loss weight was $10\times$ that of the corresponding soft loss weight (e.g. weight 10 for CIFAR10, CIFAR100; and weight 100 for TinyImagenet, Imagenet); these hyper-parameters are use for the tree supervision loss comparisons in Table 3.

Where possible, we retrain the network from scratch with tree supervision loss. For our remaining training hyperparameters, we largely use default settings found in `github.com/kuangliu/pytorch-cifar`: SGD with 0.9 momentum, 5^{-4} weight decay, a starting learning rate of 0.1, decaying by 90% $\frac{3}{7}$ and $\frac{5}{7}$ of the way through training. We make a few modifications: Training lasts for 200 epochs instead of 350, and we use batch sizes of 512 and 128 on one Titan Xp for CIFAR and TinyImagenet respectively.

In cases where we were unable to reproduce the baseline accuracy (WideResNet), we fine-tuned a pretrained checkpoint with the same settings as above, except with starting learning rate of 0.01.

On Imagenet, we retrain the network from scratch with tree supervision loss. For our remaining hyperparameters, we use settings reported to reproduce EfficientNet-EdgeTPU-Small results at `github.com/rwightman/pytorch-image-models`: batch size 128, RMSProp with start-

Figure 13: An example of a survey question presented to mechanical turks.

ing learning rate of 0.064, decaying learning rate by 97% every 2.4 epochs, weight decay of 10^{-5}, drop-connect with probability 0.2 on 8 V100s. Our results were obtained with only one model, as opposed to averaging over 8 models, so our reported baseline is 77.23%, as reported by the EfficientNet authors: https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet/edgetpu#post-training-quantization.

F CIFAR100 Tree Visualization

We presented the tree visualizations for various models on the CIFAR10 dataset in Sec. 5 of the paper. Here we also show that similar visual meanings can be drawn from intermediate nodes of larger trees such as the one for CIFAR100. Figure 14 displays the tree visualization for a WideResNet28x10 architecture on CIFAR100 (same model listed in Table 1 of Sec. 4.2). It can be seen in Figure 14 that subtrees can be grouped by visual meaning, which can be a Wordnet attribute like *Vehicle* or *Household Item*, or a more contextual meaning such as shape or background like *Cylindrical* or *Blue Background*.

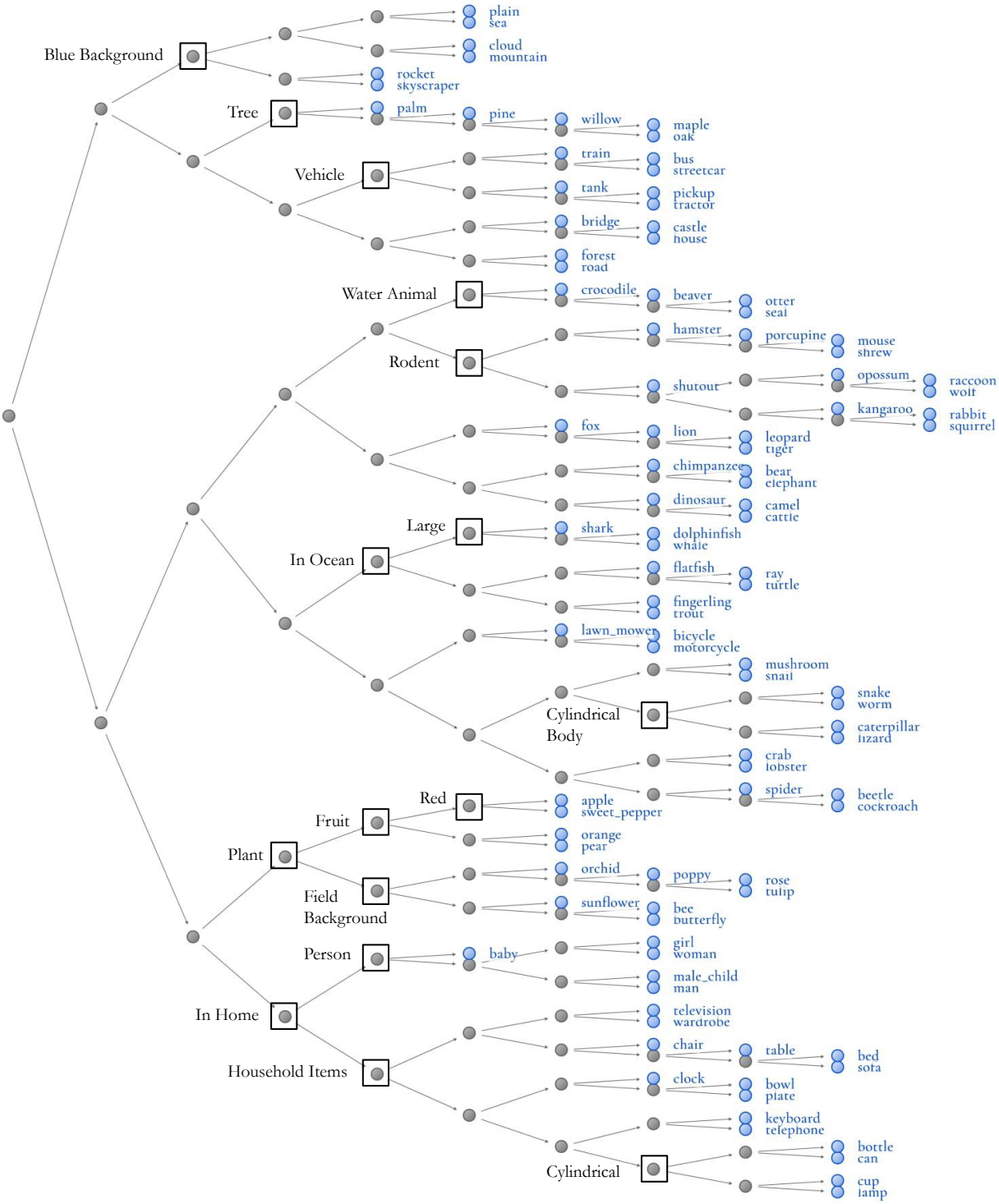

Figure 14: CIFAR100 tree visualization on WideResNet28x10 with samples of intermediate node hypothesis. Some nodes split on Wordnet attributes while other split on visual attributes like color, shape, and background.