

NBDT: Neural-Backed Decision Tree

Alvin Wan¹, Lisa Dunlap^{1*}, Daniel Ho^{1*}, Jihan Yin¹, Scott Lee¹,
 Henry Jin¹, Suzanne Petryk¹, Sarah Adel Bargal², Joseph E. Gonzalez¹
 UC Berkeley¹, Boston University²
 {alvinwan, ldunlap, danielho, jihan_yin, scott.lee.3898,
 henlinjin, spetryk, jegonzal}@berkeley.edu
 sbargal@bu.edu

Abstract

Machine learning applications such as finance and medicine demand accurate and justifiable predictions, barring most deep learning methods from use. In response, previous work combines decision trees with deep learning, yielding models that (1) sacrifice interpretability to maintain accuracy or (2) underperform modern neural networks to maintain interpretability. We forgo this dilemma by proposing Neural-Backed Decision Trees (NBDTs), modified hierarchical classifiers that use trees constructed in *weight space*. Our NBDTs achieve (1) interpretability and (2) neural network accuracy. We preserve interpretable properties – e.g. leaf purity and a non-ensembled model – and demonstrate interpretability of model predictions both qualitatively and quantitatively. Furthermore, NBDTs match state-of-the-art neural networks on CIFAR10, CIFAR100, TinyImageNet, and ImageNet to within 1-2%. This yields state-of-the-art *interpretable* models on ImageNet, with NBDTs besting all decision-tree-based methods by ~14% to attain 75.30% top-1 accuracy. Code and pretrained NBDTs are at github.com/alvinwan/neural-backed-decision-trees

1 Introduction

Many computer vision applications (e.g. medical imaging and autonomous driving) require insight into the model’s decision process, complicating applications of deep learning. Recent efforts in explainable computer vision attempt to address this need and can be grouped into one of two categories: (1) saliency maps and (2) sequential decision processes.

Saliency maps retroactively explain model predictions by identifying which pixels most affected the prediction. However, by focusing on the input, saliency maps ignore the model’s decision making process. For example, saliency offers no insight for a misclassification when the model is “looking” at the right object for the wrong reasons. Alternatively, we can gain insight into the model’s decision process by breaking up predictions into a sequence of smaller semantically meaningful decisions. This mirrors classic, interpretable models like decision trees. However, existing efforts to fuse deep learning and decision trees suffer from (1) significant accuracy loss, relative to contemporary models (e.g., residual networks), (2) reduced interpretability due to accuracy optimizations (e.g., impure leaves and ensembles), and (3) tree structures that offer limited insight into the model’s credibility.

In this work, we propose **Neural-Backed Decision Trees (NBDTs)** to make state-of-the-art image classification models interpretable. An NBDT is a hierarchical classifier that, unlike its predecessors, (1) uses a hierarchy derived from model parameters, to avoid overfitting, (2) can be created from any existing classification neural network without architectural modifications, and (3) retains interpretability by using a single model, sequential discrete decisions, and pure leaves. NBDTs are built in 2 steps: construct a tree structure using the weights of a trained network, dubbed an *induced hierarchy*.

*Equal contribution

→

Then, re-train or fine-tune that classification network with an extra hierarchy-based loss term, called a *tree supervision loss*. For the forward pass, we run the fully-connected layer as *embedded decision rules* – variants of oblique decision rules for arbitrary branching factors.

We show that NBDTs are competitive with state-of-the-art neural networks on ImageNet [13] and are substantially (up to $\sim 14\%$) more accurate than state-of-the-art hybrid decision tree and deep learning based approaches on standard benchmarks. Our contributions are as follows.

1. We propose a *tree supervision loss*, which yields NBDTs that match WideResNet accuracy on CIFAR10, CIFAR100, and TinyImageNet; and EfficientNet on ImageNet to within 2%.
2. We propose a variant of oblique decision rules, called *embedded decision rules*. We also design *induced hierarchies* in weight-space that outperform both data-based hierarchies (e.g. built with information gain) and existing hierarchies (e.g. WordNet), in accuracy.
3. We present qualitative and quantitative evidence of interpretability for NBDT model decisions—for both general model behavior and specific sample predictions.

2 Related Works

Saliency maps. Numerous efforts [42, 50, 40, 51, 39, 36, 34, 43] have explored the design of salience maps identifying pixels that most influenced the model’s prediction. White-box techniques [42, 50, 40, 39, 43] use the network’s parameters to determine salient image regions, and black-box techniques [36, 34] determine pixel importance by measuring the prediction’s response to perturbed inputs. However, saliency does not explain the model’s decision process (e.g. Was the model confused early on, distinguishing between *Animal* and *Vehicle*? Or is it only confused between dog breeds?). This is in contrast to decision-tree-based methods, described below.

Transfer to Explainable Models: Prior to the recent success of deep learning, decision trees were state-of-the-art on a wide variety of learning tasks and the gold standard for interpretability. Despite this recency, study at the intersection of neural network and decision tree dates back three decades, where neural networks were seeded with decision tree weights [4, 5, 22, 21], and decision trees were created from neural network queries [24, 7, 11, 9, 10], like distillation [19]. The modern analog of both sets of work [20, 41, 16] evaluate on feature-sparse, sample-sparse regimes such as the UCI datasets [15] or MNIST [27], and perform poorly when applied to standard image classification tasks.

Hybrid Models: Recent work produces hybrid decision tree and neural network models to scale up to datasets like CIFAR10 [25], CIFAR100 [25], TinyImageNet [26], and ImageNet [13]. One category of models organizes the neural network into a hierarchy, dynamically selecting branches to run inference [47, 28, 46, 35, 31]. However, these models forsake interpretability by creating impure leaves. Other approaches fuse deep learning into each decision tree node: an entire neural network [33], several layers [31, 38], a linear layer [1], or some other parameterization of neural network output [23]. These models likewise limit interpretability by employing an ensemble, or supporting no more than depth-2 trees [1, 17].

Hierarchical Classification: One set of approaches directly uses a pre-existing hierarchy over classes, such as WordNet [35, 8, 12]. However *conceptual similarity is not indicative of visual similarity*. Other models build a hierarchy using the training set directly, via a classic data-dependent metric like Gini impurity [2] or information gain [37, 6]. These models are instead *prone to overfitting*, per [45]. Finally, several works introduce hierarchical surrogate losses [48, 14], such as hierarchical softmax [30], but as the authors note, these methods quickly suffer from major accuracy loss with more classes or higher-resolution images (e.g. beyond CIFAR10). Furthermore, we demonstrate hierarchical classifiers attain higher accuracy *without* a hierarchical softmax.

3 Method

Neural-backed decision trees are trained in two phases: First, construct an *induced hierarchy* from the weights of a neural network’s last fully-connected layer (Sec. 3.1). Second, re-train or fine-tune the model with a hierarchy-based tree supervision loss (Sec. 3.2). Inference also occurs in two phases: First, featurize the sample using the backbone, or all layers before the final fully-connected layer. Second, run the decision rules embedded in the fully-connected layer (Sec. 3.3).

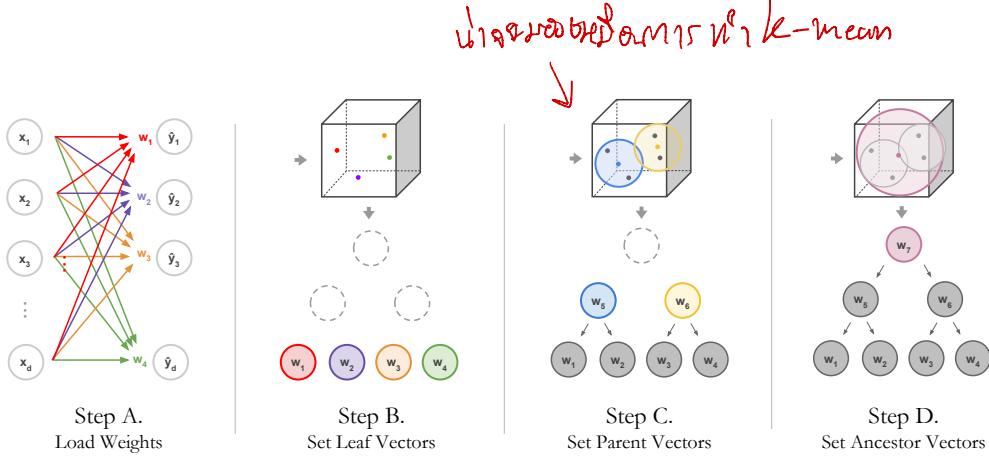


Figure 1: Building Induced Hierarchies. **Step A.** Load the weights of pre-trained neural network’s final fully-connected layer, with weight matrix $W \in \mathbb{R}^{d \times k}$. **Step B.** Use each row w_i of W as representative vectors for each leaf node. For example, the red w_1 from A is assigned to the red leaf in B. **Step C.** Use the average of each pair of leaves for the parents’ representative vectors. For example, w_1 and w_2 (red and purple) in B are averaged to make w_5 (blue) in C. **Step D.** For each ancestor, take the subtree it is the root for. Average representative vectors for all leaves in the subtree. That average is the ancestor’s representative vector. In this figure, the ancestor is the root, so its representative vector is the average of all leaves w_1, w_2, w_3, w_4 .

For simplicity, consider an oblique decision tree [32] with non-axis-aligned hyperplanes for each binary decision: Each node in the decision tree is associated with a *representative vector* r_i . During inference, each sample iteratively traverses the child with the *more similar* r_i , until reaching a leaf. A more generic hierarchical classifier [35] simply extends this to branching factors higher than 2, but decision rules are the same: take inner products between the sample x and each r_i , then pick the child with the highest inner product. Below, we present a variant of hierarchical classifiers.

3.1 Building Induced Hierarchies

We build a hierarchy in *weight-space* to attain a higher accuracy interpretable model. This is in contrast to existing decision-tree-based methods which use either (a) existing hierarchies e.g. WordNet or (b) hierarchies built in feature-space with data-dependent heuristics like information gain.

In particular, we take *row vectors* w_i , each representing a class, from the fully-connected layer weights W . Then, we run hierarchical agglomerative clustering on the class representatives w_i . Each leaf is represented by a vector w_i (Fig. 1 Step B) and each intermediate node’s r_i is the average of its leave’s representatives (Fig. 1 Step C). We refer to this hierarchy as the *induced hierarchy* (Fig. 1).

We additionally compare to alternative hierarchies: a classic information gain hierarchy over neural features and the WordNet [29] hierarchy. WordNet relations also provide interpretable labels for other candidate decision trees, e.g. classifying a *Cat* also as a *Mammal* and a *Living Thing*. To leverage this “free” source of labels, we additionally generate semantic hypotheses for each intermediate node in an induced hierarchy, by finding the earliest ancestor of each subtree’s leaves.

3.2 Training with Tree Supervision Loss

All of the proposed decision trees above suffer from one major issue: Even though the original neural network is encouraged to separate representative vectors for each class, it is not trained to separate *representative vectors for each internal node*. To amend this issue, we add a loss term that encourages the neural network to separate representatives for internal nodes.

In the below Sec 3.3, we describe how to obtain a distribution over classes, $\mathcal{D}_{\text{nbdt}} = \{p(c)\}_{c=1}^C$ (Eq. 2). We then define the *soft tree supervision loss* to be a cross entropy loss over this distribution, $\mathcal{L}_{\text{soft}}$. During training, there are total of 2 different cross entropy loss terms – the original cross entropy loss and the soft tree supervision loss term, with a weighting hyperparameter ω :

$$\mathcal{L} = \underbrace{\text{CROSSENTROPY}(\mathcal{D}_{\text{pred}}, \mathcal{D}_{\text{label}})}_{\mathcal{L}_{\text{original}}} + \omega \underbrace{\text{CROSSENTROPY}(\mathcal{D}_{\text{nbdt}}, \mathcal{D}_{\text{label}})}_{\mathcal{L}_{\text{soft}}} \quad (1)$$

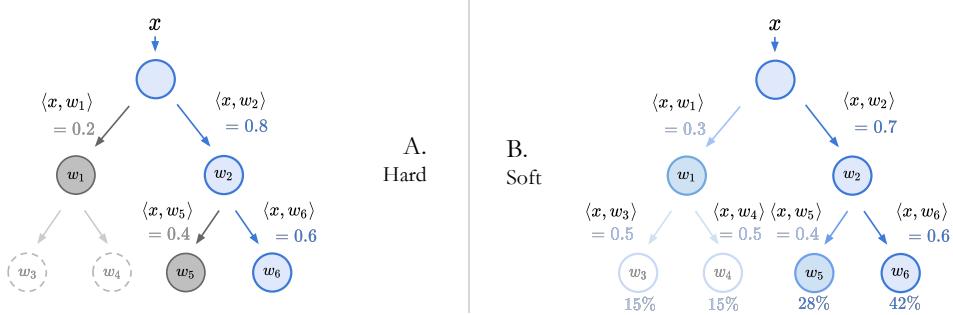


Figure 2: Hard and Soft Decision Trees. **Tree A** is the “hard” decision tree. Each node picks the child node with the largest inner product, and visits that node next. Continue until a leaf is reached. **Tree B** is the “soft” extension, where each node simply returns probabilities, as normalized inner products, of each child. For each leaf, compute the probability of its path to the root. Pick leaf with the highest probability.

The hyperparameter ω is set to 1 for CIFAR10, CIFAR100, and set to 10 for TinyImageNet, ImageNet. We additionally report comparisons against hierarchical softmax and a trained neural network that omits the surrogate tree supervision loss (*i.e.* $\omega = 0$). We re-train where possible, resorting to fine-tuning with $\mathcal{L}_{\text{soft}}$ when the base neural network accuracy is not reproducible.

3.3 Inference with Embedded Decision Rules

To run inference, our NBDT first featurizes each sample using the neural network backbone; the backbone consists of all neural network layers before the final fully-connected layer. Second, we run a decision tree built in *weight-space*. This decision tree can be run with either *hard* or *soft* inference.

Hard Decision Tree Inference (NBDT-H) Starting at the root node, each sample is sent to the child with the most similar representative vector. We continue picking and traversing the tree until we reach a leaf. The class associated with this leaf is our prediction (Fig. 2 A. Hard). More precisely, consider a tree with nodes indexed by i . Each node i produces a probability of child node $j \in C(i)$; this probability is denoted $p(i, j)$. Each node thus picks the next node using $\text{argmax}_{j \in C(i)} p(i, j)$

Soft Decision Tree Inference (NBDT-S) Compute a softmax at each node, over all children, to obtain probabilities of each child. For each leaf, take the path probability of reaching that leaf from its parent. Compute an argmax over this leaf distribution, to select one leaf (Fig. 2 C. Soft).

More precisely, consider a leaf ℓ and its path from the root P_ℓ . The probability of each node $i \in P_\ell$ traversing the next node in the path $C_\ell(i)$ is denoted $p(i, C_\ell(i))$. As a result, the probability of leaf ℓ and its corresponding class c is

$$p(c) = p(\ell) = \prod_{i \in P_\ell} p(i, C_\ell(i)) \quad (2)$$

In soft inference, the final class prediction \hat{c} is defined over these class probabilities,

$$\hat{c} = \text{argmax}_c p(c) = \text{argmax}_\ell \prod_{i \in P_\ell} p(i, C_\ell(i)) \quad (3)$$

This allows us to run any classification neural network as a sequence of inner-product decision rules, which we will refer to as *embedded decision rules* to include soft and hard inference modes for trees with ≥ 2 branching factor.

4 Experiments

Our experiments obtain state-of-the-art results for interpretable models on a number of image classification benchmark datasets. We report results on a variety of different scenarios across models (ResNet[18], recently state-of-the-art WideResNet[49], EfficientNet[44]); datasets (CIFAR10[25], CIFAR100[25], TinyImageNet[26], ImageNet[13]); and inference modes (Soft vs. hard inference).

Table 1: Results The “Exp?” column denotes whether the method retains interpretable properties: pure leaves, sequential decisions, and non-ensemble. On all CIFAR10, CIFAR100, and TinyImageNet datasets, NBDT outperforms competing decision-tree-based methods, even uninterpretable variants such as a decision forest, by up to 18%. On CIFAR10, CIFAR100, and TinyImageNet, NBDTs largely stay within 1% of neural network performance. We italicize the neural network’s accuracy and bold the best-performing decision-tree-based accuracy. Our baselines are either taken directly from the original papers or improved using a modern backbone: Deep Neural Decision Forest (DNDF updated with ResNet18) [23], Explainable Observer-Classifier (XOC) [2], Deep Convolutional Decision Jungle (DCDJ) [3], Network of Experts (NofE) [1], Deep Decision Network (DDN) [33], Adaptive Neural Trees (ANT) [45], and hierarchical classifier CNN-RNN [17].

Method	Backbone	Exp?	CIFAR10	CIFAR100	TinyImageNet
NN	WideResnet28x10	✗	97.62%	82.09%	<i>67.65%</i>
ANT-A*	n/a	✓	93.28%	n/a	n/a
DDN	NiN	✗	90.32%	68.35%	n/a
DCDJ	NiN	✗	n/a	69.0%	n/a
NofE	ResNet56-4x	✗	n/a	76.24%	n/a
CNN-RNN	WideResnet28x10	✓	n/a	76.23%	n/a
NBDT-H (Ours)	WideResnet28x10	✓	97.55%	82.21%	<i>64.39%</i>
NBDT-S (Ours)	WideResnet28x10	✓	97.57%	82.87%	66.66%
NN	ResNet18	✗	94.97%	75.92%	<i>64.13%</i>
DNDf	ResNet18	✗	94.32%	67.18%	<i>44.56%</i>
XOC	ResNet18	✓	93.12%	n/a	n/a
DT	ResNet18	✓	93.97%	64.45%	<i>52.09%</i>
NBDT-H (Ours)	ResNet18	✓	94.50%	74.29%	61.60%
NBDT-S (Ours)	ResNet18	✓	94.76%	74.92%	62.74%

Figure 3: ImageNet Results The “Explainable?” row denotes whether the method retains interpretable properties. Below, “EfficientNet” refers to EfficientNet-EdgeTPU-Small. NBDT outperforms all competing decision-tree-based methods by at least 13%, staying within 2% of EfficientNet accuracy.

	NN	NofE	XOC	NBDT-H (Ours)	NBDT-S (Ours)
Explainable?	✗	✗	✓	✓	✓
Backbone	EfficientNet	AlexNet	ResNet152	EfficientNet	EfficientNet
ImageNet	77.23%	61.29%	60.77%	74.79%	75.30%

4.1 Results

Our decision trees achieve 97.57% on CIFAR10, 82.87% on CIFAR100, and 66.66% on TinyImageNet (Table 1), preserving accuracy of recently state-of-the-art neural networks. On CIFAR10, our soft decision tree matches WideResnet28x10, with a 0.05% margin. On CIFAR100, our soft decision tree achieves accuracy 0.57% higher than WideResnet28x10’s, outperforming the highest competing decision-tree-based method (NofE) by 6.63%. On TinyImageNet, our soft NBDT achieves accuracy within 1% of WideResNet’s. Furthermore, the ResNet18 variant outperforms DNDf by 18.2%. All percentage improvements are absolute, not relative.

On ImageNet (Table 3), NBDTs obtain 75.30% top-1 accuracy, outperforming the strongest competitor NofE by 14%. Note that we take the best competing results for any decision-tree-based method, but the strongest competitors hinder interpretability by using ensembles of models like a decision forest (DNDf, DCDJ) or feature shallow trees with only depth 2 (NofE). We re-train for all experimental results except WideResNet’s, due to unreplicable baseline accuracy.

4.2 Ablations

Hierarchy Ablation: Table 3 shows that induced hierarchies best both existing and data-dependent hierarchies. Each alternative hierarchy has its own limitation: *Data-dependent* hierarchies overfit, and the existing *WordNet hierarchy* focuses on conceptual rather than visual similarity: For example, by virtue of being an animal, *Bird* is closer to *Cat* than to *Plane*, according to WordNet. However,

Table 2: Tree Supervision Loss Training the NBDT with the soft tree supervision loss (“Soft”) is superior to (a) training with a hierarchical softmax tailored to an induced hierarchy (“Hard Loss”) and to (b) omitting the soft tree supervision loss. (“Pre-Loss”). Δ is the accuracy difference between our soft loss and hierarchical softmax.

Dataset	Backbone	NN	Inference	Pre-Loss	Soft Loss	Hard Loss	Δ
CIFAR10	ResNet18	94.97%	Hard	94.32%	94.50%	93.94%	+0.56%
CIFAR10	ResNet18	94.97%	Soft	94.38%	94.76%	93.97%	+0.79%
CIFAR100	ResNet18	75.92%	Hard	57.63%	74.29%	73.23%	+0.94%
CIFAR100	ResNet18	75.92%	Soft	61.93%	74.92%	74.09%	+0.83%
TinyImageNet	ResNet18	64.13%	Hard	39.57%	61.60%	58.89%	+2.71%
TinyImageNet	ResNet18	64.13%	Soft	45.51%	62.74%	61.12%	+1.62%

Table 3: Comparisons of Hierarchies: We demonstrate that our weight-space hierarchy beats taxonomy and data-dependent hierarchies. In particular, the induced hierarchy achieves better performance than (a) the WordNet hierarchy, and (b) a classic decision tree’s information gain hierarchy, built over neural features (“Info Gain”).

Dataset	Backbone	Original	Induced	Info Gain	WordNet
CIFAR10	ResNet18	94.97%	94.76%	93.97%	94.37%
CIFAR100	ResNet18	75.92%	74.92%	64.45%	74.08%
TinyImageNet	ResNet18	64.13%	62.74%	52.09%	60.26%

the opposite is true for visual similarity: by virtue of being in the sky, *Bird* is more visually similar to *Plane* than to *Cat*.

Loss Ablation: Previous work posits a hierarchical softmax is necessary – with one cross entropy term per node. To answer this, we compare soft tree supervision loss with a modified hierarchical softmax over an induced hierarchy, which we call *hard tree supervision loss* (Appendix B). However, relative to NBDTs trained with soft tree supervision loss, NBDTs trained with hard tree supervision loss systematically see lower accuracy, with up to a 2% drop in accuracy on TinyImageNet (Table 2). This shows hierarchical softmax is not necessary for a hierarchical classifier.

5 Explainability

The explainability of a decision tree is well-established, as the final prediction can be broken into a sequence of decisions that can be evaluated independently. When input features are easily understood (*e.g.* tabular data in medicine or finance), determining what a decision rule splits on is straightforward. However, when the input is more complex like an image, this becomes more challenging. Previous work preserving interpretable properties like sequential decisions appeal to explainability only briefly [45] [23] or not at all [1]. By contrast, in this section, we perform qualitative and quantitative analysis of each intermediate node’s hypothesized meaning.

5.1 Explainability of Nodes’ Visual Meanings

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 Fig. 4. 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 (Fig. 5).
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.

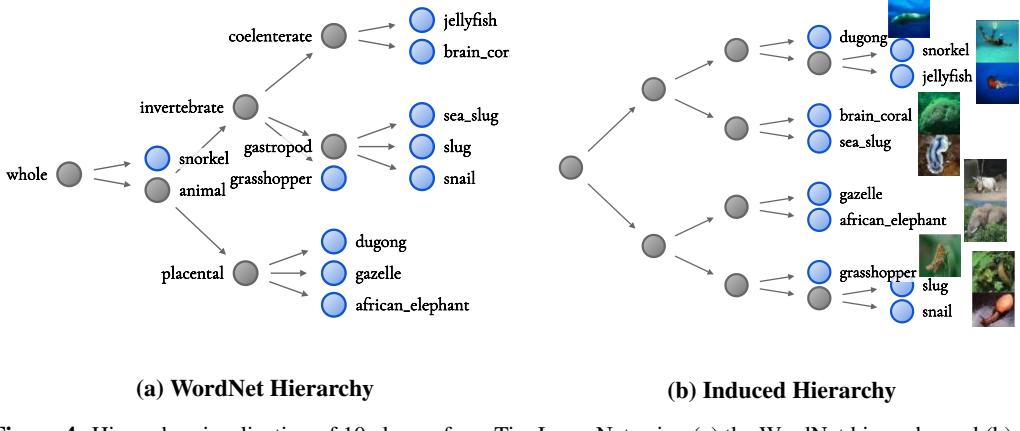


Figure 4: Hierarchy visualization of 10 classes from TinyImageNet using (a) the WordNet hierarchy and (b) the induced tree from a trained ResNet10 model. Note that (b) exhibits more visually-justifiable splits; the root node splits between aquatic animals (top) and land animals (bottom).

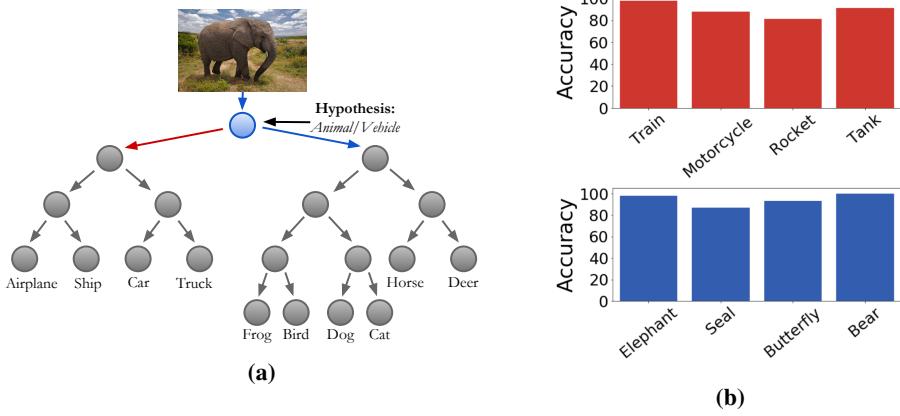


Figure 5: A Node's meaning. (Left) Visualization of node hypothesis test performed on a CIFAR10-trained WideResNet28x10 model, by sampling from CIFAR100 validation set for OOD classes. (Right) Classification accuracy is high (80–95%) given unseen CIFAR100 samples of *Vehicles* (top) and *Animals* (bottom), for the WordNet-hypothesized *Animal/Vehicle* node.

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.

Fig. 5a 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. Fig. 5b shows our hypothesis accurately predicts which child each unseen-class’s samples traverse.

5.2 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: Fig. 6 depicts the trees induced by WideResNet28x10 and ResNet10, respectively. While the WideResNet induced hierarchy (Fig. 6a) groups visually-similar classes, the ResNet (Fig. 6b) 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.

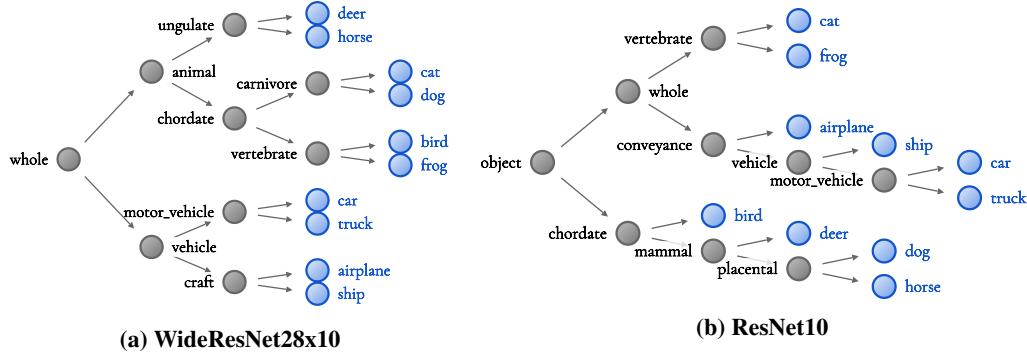


Figure 6: CIFAR10 induced hierarchies, with automatically-generated WordNet hypotheses for each node. The higher-accuracy (a) WideResNet (97.62% acc) has a more sensible hierarchy than (b) ResNet’s (93.64% acc): The former groups all *Animals* together, separate from all *Vehicles*. By contrast, the latter groups *Airplane*, *Cat*, and *Frog*.

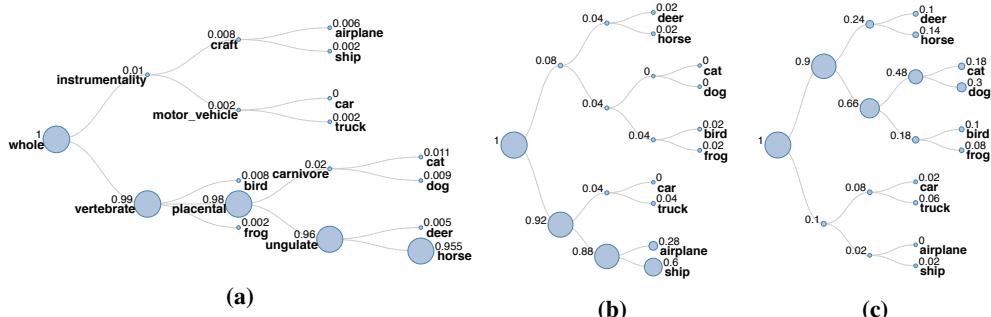


Figure 7: Visualization of path traversal frequency on an induced hierarchy for CIFAR10. (a) **In-Distribution:** *Horse* is a training class and thus sees highly focused path traversals. (b) **Unseen Class:** *Seashore* is largely classified as *Ship* despite not containing any objects, exhibiting model reliance on context (water). (c) **Unseen Class:** *Teddy Bear* is classified as *Dog*, for sharing visual attributes like color and texture.

5.3 Visualization of Tree Traversal

Frequency of path traversals additionally provide insight into general model behavior. Fig. 7 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 (Fig. 7a). As this class is present during training, tree traversal highlights the correct path with extremely high frequency. **Unseen class, seen context:** In Fig. 7b, 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 Fig. 7c 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%).

6 Conclusion

In this work, we propose Neural-Backed Decision Trees, removing the dichotomy between accuracy and interpretability that prior hierarchical classifiers and decision-tree-based methods suffer from. To assess interpretability, we automatically generate hypotheses for each node’s meaning using WordNet, then introduce a 4-step, human-in-the-loop algorithm that validates these hypotheses both qualitatively and quantitatively. We achieve interpretability by maintaining key properties—pure leaves, sequential decisions, and a single, non-ensemble model—while matching state-of-the-art neural networks to within 1-2% on CIFAR10, CIFAR100, TinyImageNet, and ImageNet.

Acknowledgments and Disclosure of Funding

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.

Broader Impact

As machine learning, and in particular deep learning, sees growing adoption in “sensitive” applications such as finance and medicine, justifiable predictions become increasingly important. There are two main uses for justification: (1) convince the human expert, such as a health care professional; (2) uncover undesirable model biases in automated resume review or fraud detection. By combining neural networks and decision trees with several new insights, we produce state-of-the-art interpretable neural networks, that can satisfy these two requirements. Critically, this means models with ethical violations and negative societal impact can be more easily diagnosed: For example, a hierarchy may misplace a person class in a wrong portion of the tree. A proxy dataset used for auditing may reveal problematic path traversals. By handling the accuracy-interpretability dichotomy for image classification, our work makes model problems and biases more explicit.

There are corresponding negative implications of justifiable predictions: (1) Faulty model predictions with justification could negatively influence decisions for human expert, e.g. health care professional. This influence extends even beyond a single human expert decision; invalid justification may influence an expert to generalize incorrectly to inapplicable scenarios. (2) Furthermore, making a bias explicit can also be used for societal harm: Rather than avoid models with problematic biases, practitioners could favor such models.

References

- [1] K. Ahmed, M. Baig, and L. Torresani. Network of experts for large-scale image categorization. volume 9911, April 2016.
- [2] S. Alaniz and Z. Akata. XOC: explainable observer-classifier for explainable binary decisions. *CoRR*, abs/1902.01780, 2019.
- [3] S. Baek, K. I. Kim, and T. Kim. Deep convolutional decision jungle for image classification. *CoRR*, abs/1706.02003, 2017.
- [4] A. Banerjee. Initializing neural networks using decision trees. 1990.
- [5] A. Banerjee. Initializing neural networks using decision trees. In *Proceedings of the International Workshop on Computational Learning and Natural Learning Systems*, pages 3–15. MIT Press, 1994.
- [6] U. C. Biçici, C. Keskin, and L. Akarun. Conditional information gain networks. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1390–1395. IEEE, 2018.
- [7] O. Boz. Converting a trained neural network to a decision tree dectext - decision tree extractor. In *ICMLA*, 2000.
- [8] C.-A. Brust and J. Denzler. Integrating domain knowledge: using hierarchies to improve deep classifiers. In *Asian Conference on Pattern Recognition*, pages 3–16. Springer, 2019.
- [9] M. Craven and J. W. Shavlik. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pages 24–30, 1996.
- [10] M. W. Craven and J. W. Shavlik. Using sampling and queries to extract rules from trained neural networks. In *Machine learning proceedings 1994*, pages 37–45. Elsevier, 1994.
- [11] D. Dancey, D. McLean, and Z. Bandar. Decision tree extraction from trained neural networks. January 2004.
- [12] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs.
- [13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- [14] J. Deng, J. Krause, A. C. Berg, and L. Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3450–3457. IEEE, 2012.
- [15] D. Dua and C. Graff. UCI machine learning repository, 2017.

- [16] N. Frosst and G. E. Hinton. Distilling a neural network into a soft decision tree. *CoRR*, abs/1711.09784, 2017.
- [17] Y. Guo, Y. Liu, E. M. Bakker, Y. Guo, and M. S. Lew. Cnn-rnn: a large-scale hierarchical image classification framework. *Multimedia Tools and Applications*, 77(8):10251–10271, 2018.
- [18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [19] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- [20] K. Humbird, L. Peterson, and R. McClaren. Deep neural network initialization with decision trees. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–10, October 2018.
- [21] I. Ivanova and M. Kubat. Decision-tree based neural network (extended abstract). In *Machine Learning: ECML-95*, pages 295–298, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.
- [22] I. Ivanova and M. Kubat. Initialization of neural networks by means of decision trees. *Knowledge-Based Systems*, 8(6):333 – 344, 1995. Knowledge-based neural networks.
- [23] P. Kotschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo. Deep neural decision forests. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [24] R. Krishnan, G. Sivakumar, and P. Bhattacharya. Extracting decision trees from trained neural networks. *Pattern Recognition*, 32(12):1999 – 2009, 1999.
- [25] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.
- [26] Y. Le and X. Yang. Tiny imagenet visual recognition challenge. 2015.
- [27] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.
- [28] M. McGill and P. Perona. Deciding how to decide: Dynamic routing in artificial neural networks. In *ICML*, 2017.
- [29] G. A. Miller. WordNet: A lexical database for english. *Commun. ACM*, 38(11):39–41, Nov. 1995.
- [30] A. A. Mohammed and V. Umaashankar. Effectiveness of hierarchical softmax in large scale classification tasks. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1090–1094. IEEE, 2018.
- [31] C. Murdock, Z. Li, H. Zhou, and T. Duerig. Blockout: Dynamic model selection for hierarchical deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2583–2591, 2016.
- [32] S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of artificial intelligence research*, 2:1–32, 1994.
- [33] V. N. Murthy, V. Singh, T. Chen, R. Manmatha, and D. Comaniciu. Deep decision network for multi-class image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [34] V. Petsiuk, A. Das, and K. Saenko. Rise: Randomized input sampling for explanation of black-box models. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.
- [35] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.
- [36] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.
- [37] S. Rota Bulo and P. Kotschieder. Neural decision forests for semantic image labelling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 81–88, 2014.
- [38] A. Roy and S. Todorovic. Monocular depth estimation using neural regression forest. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5506–5514, 2016.
- [39] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 618–626, 2017.
- [40] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
- [41] C. Siu. Transferring tree ensembles to neural networks. In *Neural Information Processing*, pages 471–480, 2019.
- [42] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.
- [43] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. *International Conference on Machine Learning (ICML) 2017*, 2017.
- [44] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.
- [45] R. Tanno, K. Arulkumaran, D. C. Alexander, A. Criminisi, and A. Nori. Adaptive neural trees, 2019.

- [46] R. Teja Mullapudi, W. R. Mark, N. Shazeer, and K. Fatahalian. Hydranets: Specialized dynamic architectures for efficient inference. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [47] A. Veit and S. Belongie. Convolutional networks with adaptive inference graphs. In *The European Conference on Computer Vision (ECCV)*, September 2018.
- [48] M. Wu, M. Hughes, S. Parbhoo, and F. Doshi-Velez. Beyond sparsity: Tree-based regularization of deep models for interpretability. In *In: Neural Information Processing Systems (NIPS) Conference. Transparent and Interpretable Machine Learning in Safety Critical Environments (TIML) Workshop*, 2017.
- [49] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.
- [50] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014.
- [51] J. Zhang, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff. Top-down neural attention by excitation backprop. In *European Conference on Computer Vision (ECCV)*, pages 543–559. Springer, 2016.

Table 4: Training Protocol We test the soft tree supervision loss with a ResNet10 backbone. The “Scratch” column denotes accuracy for a model trained from scratch, and the “Fine-tuned” column denotes accuracy for a model fine-tuned from a pre-trained backbone. To fine-tune, we change starting learning rate from 0.1 to 0.01.

Dataset	Backbone	Original	Scratch	Fine-tuned
CIFAR10	ResNet10	93.61%	93.76%	93.45%
CIFAR100	ResNet10	73.36%	73.98%	73.31%

Table 5: Tree Supervision Loss. The original neural network’s accuracy increases by 0.5% for CIFAR100 and TinyImageNet across a number of models, after training with soft tree supervision loss.

Dataset	Backbone	NN	NN+TSL	Δ
CIFAR100	WideResnet28x10	82.09%	82.63%	+0.59%
CIFAR100	ResNet18	75.92%	76.20%	+0.28%
CIFAR100	ResNet10	73.36%	73.98%	+0.62%
TinyImageNet	ResNet18	64.13%	64.61%	+0.48%
TinyImageNet	ResNet10	61.01%	61.35%	+0.34%

A Ablation Studies

Training Protocol: We find that both (1) training the neural network from scratch, with tree supervision loss at the onset and (2) fine-tuning a pre-trained network with tree supervision loss achieve similar performance. We use settings identical to those found in Sec. D except with LR 0.1 and a tree supervision loss coefficient of 1.

Tree Supervision Loss Boosts Accuracy: The tree supervision loss, as described in Sec. 3.2, boosts the accuracy of a neural network by 0.5% with tree supervision loss weight of 0.5, when training from scratch on CIFAR100 and TinyImageNet (Table 5).

Tree Supervision Loss Weight: As we vary the coefficient for the tree supervision loss, we note that disproportionately assigning weight to the tree supervision loss (by two orders of magnitude) significantly degrades the performance of both the neural network and the NBDT. However, our method is robust to imbalance between the two loss terms of up to an order of magnitude. We conclude the method is not hyper-sensitive to the loss coefficient (Table 6).

B Hard Tree Supervision Loss

An alternative loss would be hierarchical softmax. 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 $\mathcal{L}_{\text{original}}$ minimize 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. We test various weighting schemes in Sec. A. 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:

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

C 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 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

Table 6: Tree Supervision Loss Weight. Below, w refers to the coefficient for the hard tree supervision loss. All NBDT-H trees use the ResNet18 backbone with hard inference. Note that $w = 0$ is simply the original neural network.

Dataset	Method	$w = 0$	$w = 0.5$	$w = 1$	$w = 10$	$w = 100$
CIFAR10	ResNet18	94.97%	94.91%	94.44%	93.82%	91.91%
CIFAR10	NBDT-H	—	94.50%	94.06%	93.94%	92.28 %
CIFAR100	ResNet18	75.92%	76.20%	75.78%	75.63%	73.86%
CIFAR100	NBDT-H	—	66.84%	69.49%	73.23%	72.05%
TinyImageNet	ResNet18	64.13%	64.61%	63.90%	63.98%	63.11%
TinyImageNet	NBDT-H	—	43.05%	58.25%	56.25%	58.89%

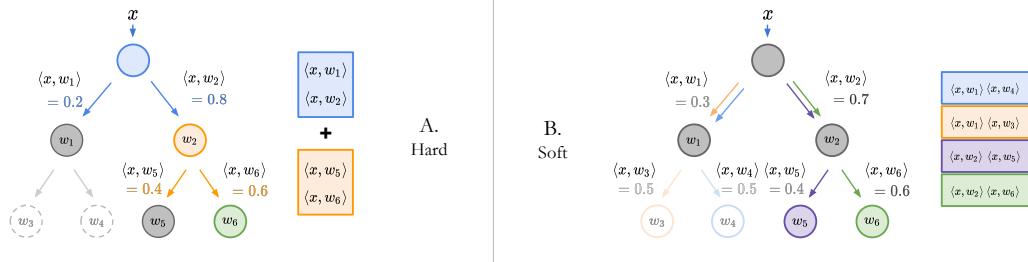


Figure 8: Tree Supervision Loss has two variants: **Hard Tree Supervision Loss (A)** defines a cross entropy term per node. This is illustrated with the blue box for the blue node and the orange box for the orange node. The cross entropy is taken over the child node probabilities. The green node is the 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.

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.

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

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

D 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 2.

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 $\frac{2}{7}$ 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 starting 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>

E CIFAR10 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. Fig. 9 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 Fig. 9 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*.

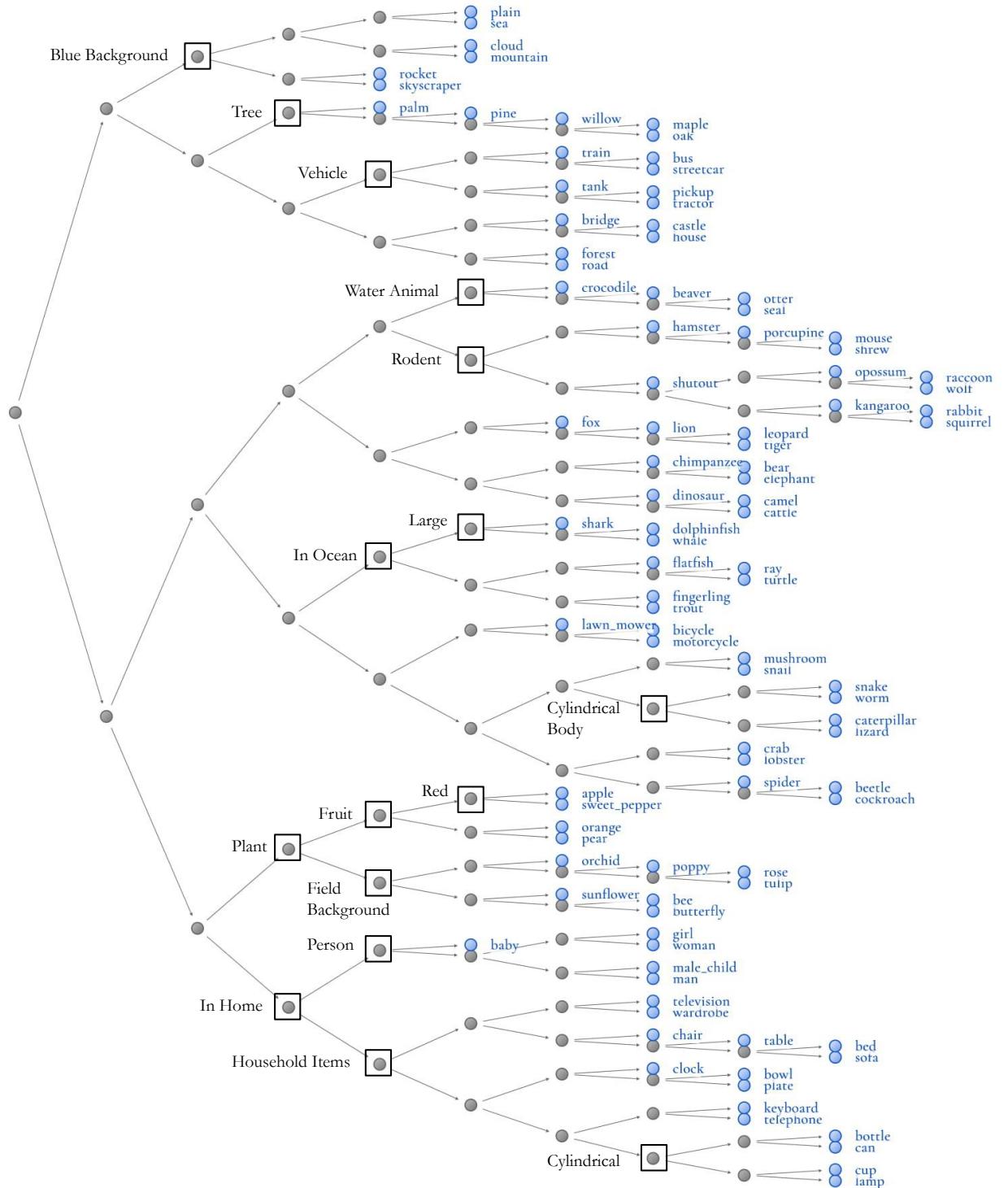


Figure 9: CIFAR100 tree visualization on WideResNet28x10 with samples of intermediate node hypothesis. Some nodes split on Wordnet attributes while other split on visual attributes like color, shape, and background.