NBDT: Neural-Backed Decision Trees

Alvin Wan[1], Lisa Dunlap[1⋆], Daniel Ho[1⋆], Jihan Yin[1], Scott Lee[1], Henry Jin[1], Suzanne Petryk[1], Sarah Adel Bargal[2], Joseph E. Gonzalez[1]

UC Berkeley[1], Boston University[2]
{alvinwan, ldunlap, danielho, jihan_yin, scott.lee.3898, henlinjin, spetryk, jegonzal}@berkeley.edu
sbargal@bu.edu

Abstract. Deep learning is being adopted in settings where accurate and justifiable predictions are required, ranging from finance to medical imaging. While there has been recent work providing post-hoc explanations for model predictions, there has been relatively little work exploring more directly interpretable models that can match state-of-the-art accuracy. Historically, decision trees have been the gold standard in balancing interpretability and accuracy. However, recent attempts to combine decision trees with deep learning have resulted in models that (1) achieve accuracies far lower than that of modern neural networks (e.g. ResNet) even on small datasets (e.g. MNIST), and (2) require significantly different architectures, forcing practitioners pick between accuracy and interpretability. We forgo this dilemma by creating Neural-Backed Decision Trees (NBDTs) that (1) achieve neural network accuracy and (2) require no architectural changes to a neural network. NBDTs achieve accuracy within 1% of the base neural network on CIFAR10, CIFAR100, TinyImageNet, using recently state-of-the-art WideResNet; and within 2% of EfficientNet on ImageNet. This yields state-of-the-art explainable models on ImageNet, with NBDTs improving the baseline by ∼14% to 75.30% top-1 accuracy. Furthermore, we show interpretability of our model's decisions both qualitatively and quantitatively via a semi-automatic process. Code and pretrained NBDTs can be found at github.com/alvinwan/neural-backed-decision-trees.

Keywords: explainability, deep neural networks, decision trees, classification, visual data, maintaining accuracy

1 Introduction

In many applications of computer vision (e.g., medical imaging and autonomous driving) insight into the prediction process or justification for a decision is critical. Whereas deep learning techniques achieve state-of-the-art accuracy in these settings, they provide little insight into the resulting predictions.

⋆ Equal contribution

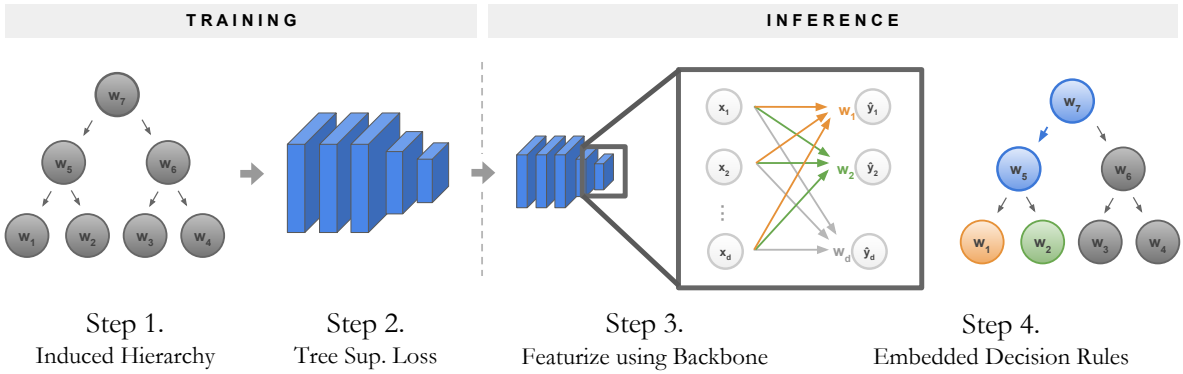


Fig. 1: Neural-Backed Decision Tree. In **step 1**, we use a pre-trained network's fully-connected layer weights to build a hierarchy (Sec 3.2). In **step 2**, we fine-tune the network with a custom loss (Sec. 3.3). In **step 3**, we featurize the sample using the neural network backbone. In **step 4**, we use the fully-connected layer's weights to run decision rules (Sec. 3.1). Illustrated above, the orange arrows in Step 3 are associated with tree's orange node in Step 4. Likewise, the green arrows map to the green node. The tree takes an inner product between the incoming sample and each of the orange w_1 and green w_2 vectors; the leaf with the higher inner product is predicted.

In an attempt to address the loss of interpretability in deep learning, a growing body of work explores explainable predictions. Explainable computer vision often features saliency maps, which depict the parts of an image that significantly impact the final classification. However, visual explanations focus on input to the model rather than the model itself. Therefore these techniques provide little insight into the general behavior of the model and how it might perform on data outside the domain of the available training examples.

In this work we propose neural-backed decision trees (NBDTs) to make state-of-the-art computer vision models interpretable. These NBDTs require no special architectures: Any neural network for image classification can be transformed into an NBDT just by fine-tuning with a custom loss. Furthermore, NBDTs perform inference by breaking image classification into a sequence of intermediate decisions. This sequence of decisions can then be mapped to more interpretable concepts and reveal perceptually informative hierarchical structure in the underlying classes. Critically, in contrast to prior work on decision trees in computer vision, NBDTs are competitive with state-of-the-art results on CIFAR10 [18], CIFAR100 [18], TinyImageNet[19], and ImageNet [8] and are substantially (up to 18%) more accurate than comparable decision tree based approaches while also being more interpretable.

We introduce a two stage training procedure for NBDTs. First, we compute a hierarchy, dubbed the *induced hierarchy* (Fig. 1, Step 1). This hierarchy is derived from the weights of a neural network already trained on the target dataset. Second, we fine-tune the network with a custom loss, designed specifically for this tree, called the *tree supervision loss* (Fig. 1, Step 2). This loss forces the model to maximize decision tree accuracy given a fixed tree hierarchy. We then run inference in two steps: (1) We construct features for each training image

using the network backbone (Fig. 1, Step 3). Then for each node, we compute a vector in the network's weight space that best represents the leaves in its subtree, given the decision tree hierarchy – we refer to this vector as the *representative vector*. (2) Starting at the root node, each sample is sent to the child with the most similar representative vector to the sample. We continue picking and traversing the tree until we reach a leaf. The class associated with this leaf is our prediction (Fig. 1, Step 4). This contrasts related work that introduces obstacles for interpretability such as impure leaves [16] or an ensemble of models [1,16].

We summarize our contributions as follows:

1. We propose a method for running any classification neural network as a decision tree by defining a set of *embedded decision rules* that can be constructed from the fully-connected layer. We also design *induced hierarchies* that are easier for neural networks to learn.
2. We propose *tree supervision loss*, which boosts neural network accuracy by 0.5% and produces high-accuracy NBDTs. We demonstrate that our NBDTs achieve accuracies comparable to neural networks on small, medium, and large-scale image classification datasets.
3. We present qualitative and quantitative evidence of semantic interpretations for our model decisions.

2 Related Works

Until the more recent success of deep learning, decision trees defined the state-of-the-art in both accuracy and model interpretability on a wide variety of learning tasks. As deep learning based techniques began to dominate other learning methods, there was a significant amount of work exploring the intersection of deep learning and decision trees. We summarize the co-evolution of deep learning and decision trees.

The study of decision tree and neural network combinations dates back three decades, where neural networks were seeded with weights provided by decision trees [4,5,15]. Work converting from decision trees to neural networks also dates back three decades [14,17,6,7]. Like distillation [12], these works focused on treating the neural network as an oracle and querying it to create splits.

Decision trees to neural networks. Recent work also seeds neural networks with weights provided by decision trees [13], with renewed interest in gradient-based approaches [29]. These methods show empirical evidence on UCI datasets[9], in very feature-sparse and sample-sparse regime.

Neural networks to decision trees. Recent work [10] uses distillation, training a decision tree to mimic a neural network's input-output function. All of these works evaluate on simple datasets such as UCI[9] or MNIST[20], while our method is evaluated on more complex datasets such as CIFAR10[18], CIFAR100[18], and TinyImageNet[19].

Combining neural networks with decision trees. More recent are works combining neural networks with decision trees, scaling up inference to datasets

with many high-dimensional samples. The Deep Neural Decision Forest [16] saw performance that matched neural networks on ImageNet. However, this occurred before the inception of residual networks and sacrificed interpretability of the model, by using impure leaves and requiring a forest. Murthy et al. [23] propose creating a new neural network for each node in the decision tree and showed interpretable outputs. Ahmed et al. [1] (NofE) modify this by sharing the backbone between all nodes but support only depth-2 trees; NofE sees ImageNet performance competitive with pre-ResNet architectures. Our method further builds on this by sharing not just the backbone but also the fully-connected layer; we furthermore show competitive performance with state-of-the-art neural networks, including residual networks, while preserving interpretability.

Instead of combining neural networks and decision trees explicitly, several works borrow ideas from decision trees for neural networks and vice versa. In particular, several redesigned neural network architectures utilize decision tree branching structures [35,21,34]. Whereas accuracy improves, this approach sacrifices decision tree interpretability. Others use decision trees to analyze neural network weights [39,24]. This has the opposite downfall, of either sacrificing accuracy or not supporting a mechanism for prediction. As we hypothesize and show, a high-accuracy decision tree is necessary to explain and interpret high-accuracy models. Furthermore, our competitive performance shows that accuracy and explainability do not need to be traded off.

Visual Explanations. An orthogonal but predominant explainability direction involves generating saliency maps, which highlight the spatial evidence used for decisions made by neural networks [30,37,28,38,27,26,25,31]. White-box techniques such as Guided Backpropagation [30], Deconvolution [37,28], Grad-CAM [27] and Integrated Gradients [31] use the gradients of the network to determine the most salient regions in an image and Black-box techniques such as LIME [26] and RISE [25] determine pixel importance by perturbing the input and measuring the change in prediction. Saliency maps only explain a single image and are unhelpful when a network is looking at the right thing for the wrong reasons (e.g. a bird misclassified as an airplane). On the other hand, our method expresses the model's prior over the entire dataset and explicitly breaks down each classification into a sequence of intermediate decisions.

3 Method

In this section we describe the proposed steps, illustrated in Fig. 1, for converting any classification neural network into a decision tree: (1) Build an induced hierarchy (Sec. 3.2), (2) fine-tune the model with a tree supervision loss (Sec. 3.3). For inference, (3) featurize samples with the neural network backbone, and (4) run decision rules embedded in the fully-connected layer (Sec. 3.1).

As implied by steps 3 and 4, our neural-backed decision tree (NBDT), has the exact same architecture as a standard neural network. As explained below in more detail, a *subset* of a fully-connected layer represents a node in the decision tree. This means that our method (1) is broadly applicable, as all classification

neural network architectures are supported as-is, i.e. no architectural modifications are needed to use NBDT and (2) benefits from deep learning know-how, as our NBDT accuracy improves as the neural network accuracy improves.

3.1 Inference with Embedded Decision Rules

First, our NBDT approach featurizes each sample using the neural network backbone; the backbone consists of all neural network layers before the final fully-connected layer. Second, at each node, we take the inner product between the featurized sample $x \in \mathbb{R}^d$ and each child node's representative vector r_i. Note that all representative vectors r_i are computed from the neural network's fully-connected layer weights. Thus, these decision rules are "embedded" in the neural network. Third, we use these inner-products to make either *hard* or *soft* decisions, described below.

To motivate why we use inner-products, we will first construct a degenerate decision tree that is equivalent to a fully-connected layer.

1. **Fully-connected layer.** The fully-connected layer's weight matrix is $W \in \mathbb{R}^{k \times d}$. Running inference with a featurized sample is a matrix-vector product:

$$\underbrace{\begin{bmatrix} \text{—} & w_1 & \text{—} \\ \text{—} & w_2 & \text{—} \\ & \vdots & \\ \text{—} & w_d & \text{—} \end{bmatrix}}_{W} \begin{bmatrix} | \\ x \\ | \end{bmatrix} = \underbrace{\begin{bmatrix} \langle x, w_1 \rangle \\ \langle x, w_2 \rangle \\ \vdots \\ \langle x, w_d \rangle \end{bmatrix}}_{\hat{y}} \implies \text{argmax}(\hat{y}) \tag{1}$$

   The matrix-vector product yields inner-products between x and each w_i, which is written as $\langle x, w_i \rangle = \hat{y}_i$. The index of the largest inner product $\hat{y}_i$ is our class prediction.
2. **Decision Tree.** Consider a minimal tree, with a root node and k child nodes. Each child node is a leaf, and each child node has a representative vector, namely a row vector $r_i = w_i$ from W. Running inference with a featurized sample x means taking inner products between x and each child node's representative vector r_i, which is written as $\langle x, r_i \rangle = \langle x, w_i \rangle = \hat{y}_i$. Like the fully-connected layer, the index of the largest product $\hat{y}_i$ is our class prediction. This is illustrated in Fig. 2 (B. Naive).

Even though the two computations are represented differently, both predict the class by taking the index of the largest inner product $\text{argmax}\langle x, w_i \rangle$. We refer to the decision tree inference as running *embedded decision rules*.

We next extend the nave decision tree beyond the degenerate case. Our decision rule requires that each child node has a representative vector r_i. As a result, if we add a non-leaf child to the root, this non-leaf child would need a representative vector. We naively consider the non-leaf's representative vector to be the average of all the subtrees' leaves' representative vectors. With a more complex tree structure containing intermediate nodes, there are now two ways to run inference:

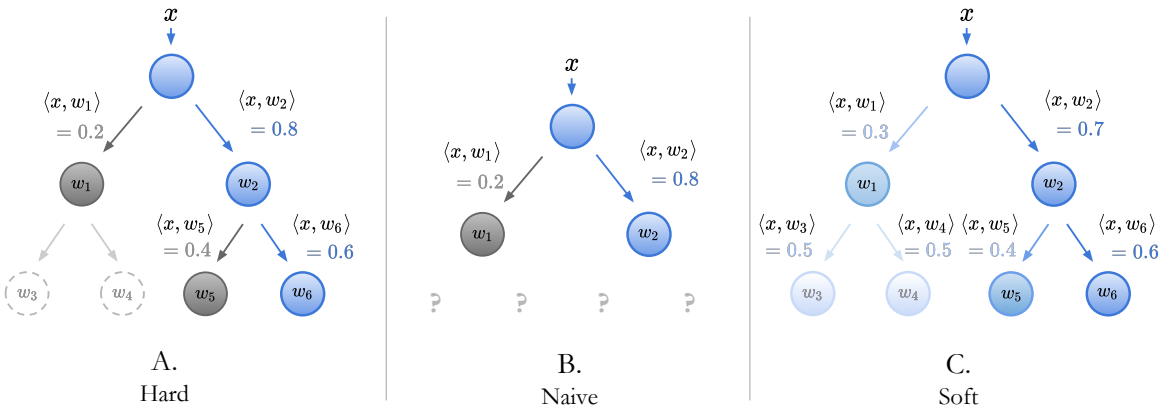


Fig. 2: Hard and Soft Decision Trees. Tree B is the nave tree with one root and k leaves. This is identical to a fully-connected layer: This tree takes inner products between the input x and each leaf's representative vector w_i, then picks the leaf corresponding to the largest inner product. **Tree A** is the "hard" extension of the nave tree. Each node picks the child node with the largest inner product, and visits that node next. Continue until a leaf is reached. **Tree C** 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.

1. **Hard Decision Tree.** Compute an argmax at each node, over all children. For each node, take the child node corresponding to the largest inner product, and traverse that child node. This process selects one leaf (Fig. 2, A. Hard).
2. **Soft Decision Tree.** Compute a softmax at each node, over all children, to obtain probabilities of each child per node. For each leaf, take the probability of traversing that leaf from its parent. Then take the probability of traversing the leaf's parent from its grandparent. Continue taking products until you reach the root. This product is the probability of that leaf and its path to the root. Tree traversal will yield one probability for each leaf. Compute an argmax over this leaf distribution, to select one leaf (Fig. 2, C. Soft).

This allows us to run any classification neural network as a sequence of embedded decision rules. However, simply running a standard-issue pretrained neural network in this way will result in poor accuracy. In the next section, we discuss how to maximize accuracy by fine-tuning the neural network to perform well after determining the hierarchy.

3.2 Building Induced Hierarchies

With the above inner-product decision rule, there are intuitively easier decision tree hierarchies for the network to learn. These easier hierarchies may more accurately reflect how the network attained high accuracy. To this end, we ran hierarchical agglomerative clustering on the class representatives w_i extracted from the fully-connected layer weights W. As described in the previous Sec. 3.1, each leaf is one w_i (Fig. 3, Step B) and each intermediate node's representative

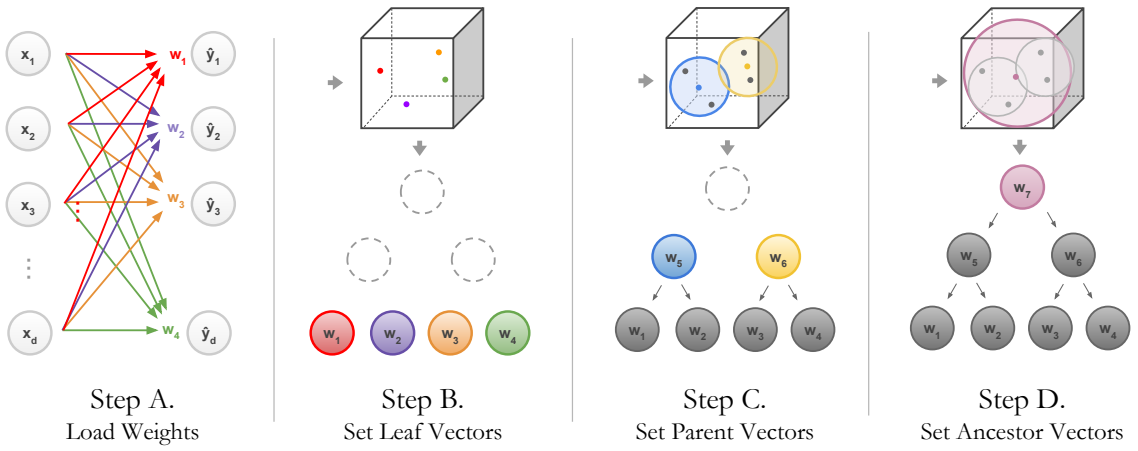


Fig. 3: Building Induced Hierarchies. Step A. Load the weights of pre-trained neural network's final fully-connected layer, with weight matrix $W \in \mathbb{R}^{d \times k}$. **Step B.** Use each column w_i of W as representative vectors for each leaf node. For example, the red w_1 from A is assigned 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.

vector is the average of all the representatives of its subtrees' leaves (Fig. 3, Step C). We refer to this hierarchy as the *induced hierarchy* (Fig. 3).

We additionally conduct experiments with an alternative WordNet-based hierarchy. WordNet [22] provides an exising hierarchy of nouns, which we leverage to relate the classes in each dataset, linguistically. We find a minimal subset of the WordNet hierarchy that includes all classes as leaves, pruning redundant leaves and single-child intermediate nodes. As a result, WordNet relations provide "free" and interpretable labels for this candidate decision tree, *e.g.* classifying a *Cat* also as a *Mammal* and a *Living Thing*. To leverage this "free" source of labels, we automatically generate hypotheses for each intermediate node in an induced hierarchy, by finding the earliest ancestor of each subtrees' leaves.

3.3 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. This is illustrated in Fig. 4. To amend this issue, we add loss terms that encourage the neural network to separate representatives for internal nodes, during training. We now explain the additional loss terms for the hard and soft decision rules in turn (Fig. 5).

For hard decision rules, we use the *hard tree supervision loss*. 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

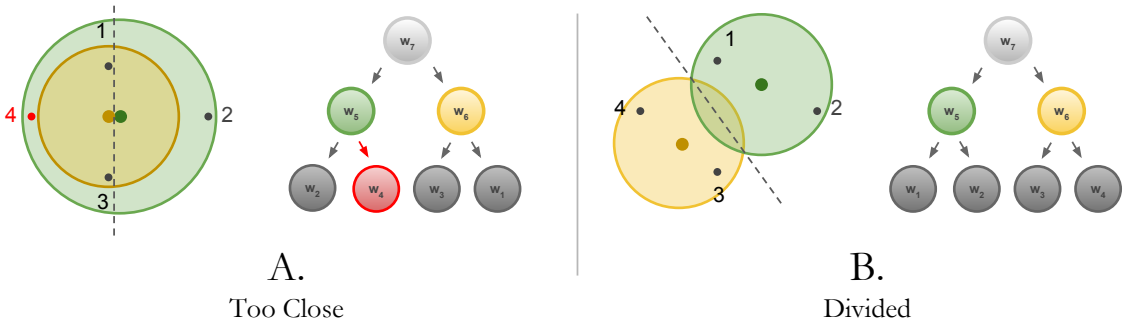


Fig. 4: Pathological Trees. In the plots, one cluster of points is marked with a green circle – the other with yellow. The center of each circle is given by the average of its two gray points. The corresponding decision tree is drawn to the right of each plot. **A**: Once given a point, the decision tree's root will compute the child node with the closest representative vector (green or yellow point). Note that all samples for class 4 (red) will be closer to the wrong parent (yellow) than the right one (green). This is because A attempts to cluster 2 with 4 and 1 with 3. Thus, it will be difficult for the neural network to obtain high accuracy, as it needs to move all points drastically to separate the yellow and green points. **B**: With the same points, this tree clusters 1 with 2 and 3 with 4, leading to more separable clusters. Note that the decision boundary (dotted line) in B features a much larger margin, with respect to the green and yellow points. Thus, this tree is far easier for the neural network to classify points correctly with.

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. 2). 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. 4.2. 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:

$$\mathcal{L}_{\text{hard}} = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\textsc{CrossEntropy}(\mathcal{D}(i)_{\text{pred}}, \mathcal{D}(i)_{\text{label}})}_{\text{over the } c \text{ children for each node}}. \tag{2}$$

For soft decision rules, we use the *soft tree supervision loss*. In Sec 3.1, we described how the soft decision tree provides a single distribution over leaves, $\mathcal{D}_{\text{pred}}$. We add a cross entropy loss over this distribution. In total, there are 2 different cross entropy loss terms – the original cross entropy loss and the soft tree supervision loss term. This is $\mathcal{L}_{\text{original}} + \mathcal{L}_{\text{soft}}$, where:

$$\mathcal{L}_{\text{soft}} = \textsc{CrossEntropy}(\mathcal{D}_{\text{pred}}, \mathcal{D}_{\text{label}}). \tag{3}$$

Supplementary materials include details of the mathematical formulation.

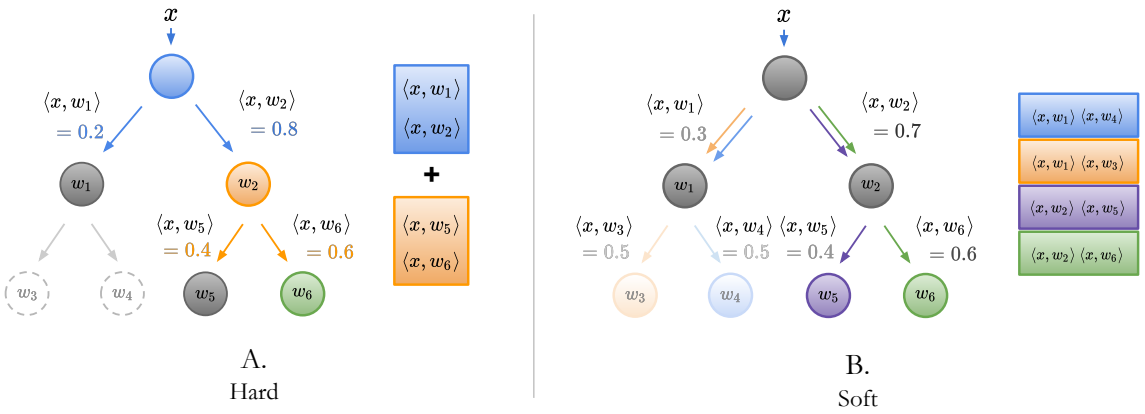


Fig. 5: Tree Supervision Loss has two variants: **Hard Tree Supervision Loss (A)** defines a cross entropy term per node. This is illustrated with the blue box for the blue node and the orange box for the orange node. The cross entropy is taken over the child node probabilities. The green node is the label leaf. 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 sitting directly adjacent to one another.

4 Experiments

Our experiments obtain state-of-the-art results for decision trees on a number of image classification benchmark datasets. We report results on a variety of different scenarios across models, datasets and inference modes:

1. Datasets: CIFAR10[18], CIFAR100[18], TinyImageNet[19], ImageNet[8]
2. Models: ResNet[11], recently state-of-the-art WideResNet[36], EfficientNet[32]
3. Inference modes: Soft *vs.* hard inference.

We also perform ablation studies on tree supervision loss weight and different hierarchies, noting that a *tree supervision loss* with half weight consistently boosts the base neural network's accuracy by 0.5% across datasets.

4.1 Results

Our decision trees achieve 97.57% on CIFAR10, 82.87% on CIFAR100, and 66.66% on TinyImageNet, 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 decision tree achieves accuracy within 1% of WideResNet's. Furthermore, the ResNet18 variant outperforms DNDF by 18.2%.

Table 1: Results On all CIFAR10, CIFAR100, TinyImageNet, and ImageNet 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) [16], Explainable Observer-Classifier (XOC) [2], Deep Convolutional Decision Jungle (DCDJ) [3], Network of Experts (NofE) [1], Deep Decision Network (DDN) [23], and Adaptive Neural Trees (ANT) [33].

Method	Backbone	CIFAR10	CIFAR100	TinyImageNet	ImageNet
NN	WideResnet28x10	*97.62%*	*82.09%*	*67.65%*	–
ANT-A*	–	93.28%	–	–	–
XOC	–	93.12%	–	–	60.77%
NofE	ResNet56	–	76.24%	–	–
DDN	–	90.32%	68.35%	–	–
DCDJ	NiN	–	69.0%	–	–
NBDT-H (Ours)	WideResnet28x10	97.55%	82.21%	64.39%	–
NBDT-S (Ours)	WideResnet28x10	**97.57%**	**82.87%**	**66.66%**	–
NN	EfficientNet-ES	–	–	–	*77.23%*
NofE	AlexNet	–	–	–	61.29%
NBDT-H (Ours)	EfficientNet-ES	–	–	–	74.79%
NBDT-S (Ours)	EfficientNet-ES	–	–	–	**75.30%**
NN	ResNet18	*94.97%*	*75.92%*	*64.13%*	–
DNDF	ResNet18	94.32%	67.18%	44.56%	–
NBDT-H (Ours)	ResNet18	94.50%	74.29%	61.60%	–
NBDT-S (Ours)	ResNet18	**94.76%**	**74.92%**	**62.74%**	–

On ImageNet, 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).

4.2 Ablation Studies

Tree Supervision Loss. The tree supervision loss, as described in Sec. 3.3, 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 2).

WordNet Hierarchy. Table 3 shows that WordNet is comparable to induced hierarchies on CIFAR but underperforms by 4.17% on TinyImageNet. This is because WordNet similarity is not indicative of visual similarity: For example, by virtue of being an animal, *Bird* is closer to *Cat* than to *Plane*, according to WordNet. However, the opposite is true for visual similarity: by virtue of being in the sky, *Bird* is more visually similar to *Plane* than to *Cat*.

Table 2: 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%

Table 3: WordNet Hierarchy. We compare the WordNet hierarchy with the induced hierarchy. All results use a ResNet10 backbone with tree supervision loss weight of 10. Both inference and tree supervision losses are hard.

Dataset	Backbone	Original	WordNet	Induced
CIFAR10	ResNet10	93.61%	93.65%	93.32%
CIFAR100	ResNet10	73.36%	71.79%	71.70%
TinyImageNet	ResNet10	61.01%	52.33%	56.50%

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 4).

5 Explainability

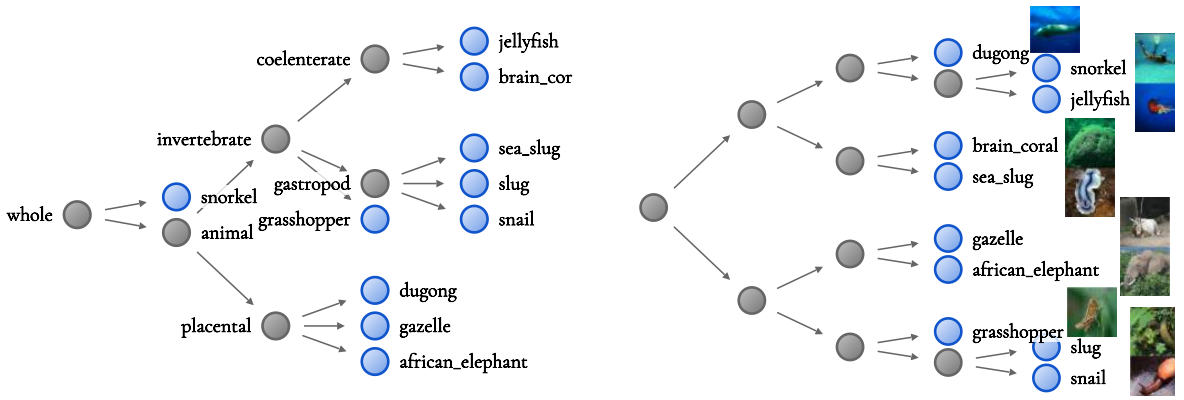
The explainability of a decision tree is well-established, as the final prediction can be broken into a sequence of decisions that can be evaluated independently for correctness. In the case where the input features are easily understood (e.g. medical/financial data), analyzing the rules which dictate the splits in the tree is relatively straightforward, but when the input is more complex like an image, this becomes more challenging. In this section, we perform qualitative and quantitative analysis of decisions made by intermediate tree nodes.

5.1 Explainability of Nodes' Semantic Meanings

Since the induced hierarchy is constructed using model weights, it is not forced to split on particular attributes. While hierarchies like WordNet provide hypotheses for a node's meaning, Fig.6 shows that WordNet doesn't suffice, as the tree may split on contextual attributes such as *underwater* and *on land*. To diagnose node meanings, we perform the following 4-step test:

Table 4: 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%



(a) WordNet Tree **(b) Induced Tree**

Fig. 6: Tree Visualization of 10 classes from TinyImageNet using (a) the WordNet hierarchy and (b) the induced tree from a trained ResNet10 model.

1. Make a hypothesis for the node's meaning (*e.g. Animal vs. Vehicle*). This hypothesis can be computed automatically from a given taxonomy like WordNet or deduced from manual inspection of leaves for each child (Fig. 7).
2. Collect a dataset with new, unseen classes that test the hypothesised meaning of the node in step 1 (*e.g. Elephant* is an unseen *Animal*). Samples in this dataset are referred to as out-of-distribution samples, as they are drawn from a separate labeled dataset.
3. Pass samples from this dataset through the node in question. 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.

This process automatically validates WordNet hypotheses, but manual intervention is needed for hypotheses beyond WordNet. Fig. 7a depicts the CIFAR10 tree induced by a WideResNet28x10 model trained on CIFAR10. Our hypothesis is that the root note splits on *Animal vs. Vehicle*. We collect out-of-distribution images for *Animal* and *Vehicle* classes that are unseen at training time, from CIFAR100. We then compute the hypothesis' accuracy. Fig. 7b shows our hypothesis accurately predicts which child each unseen-class's samples traverse.

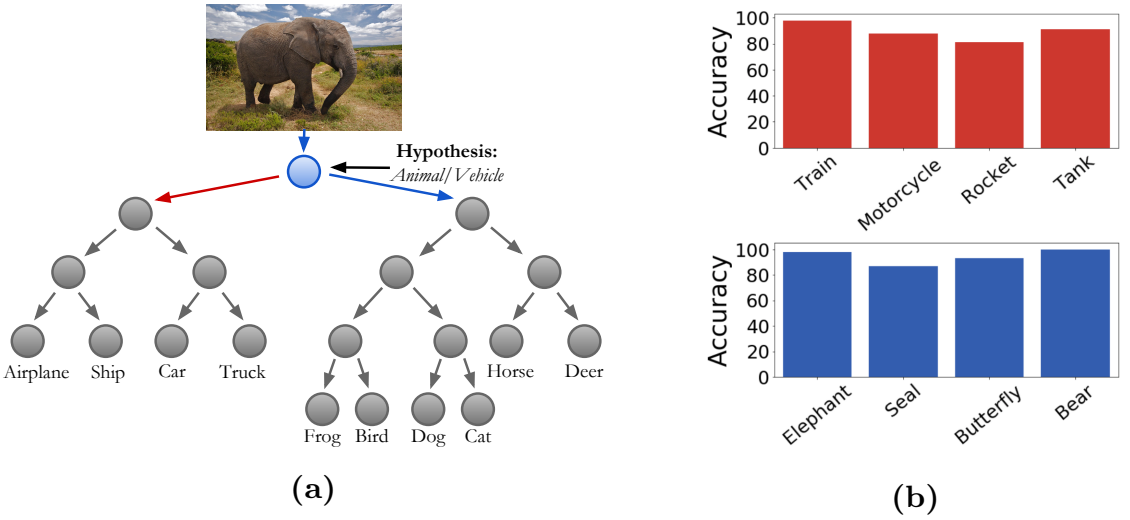


Fig. 7: A Node's semantic meaning. (a) CIFAR10 Tree Visualization of a WideResNet28x10 model. (b) Classifications of the hypothesized *Animal/Vehicle* node on samples of unseen classes of *Vehicles* (top) and *Animals* (bottom).

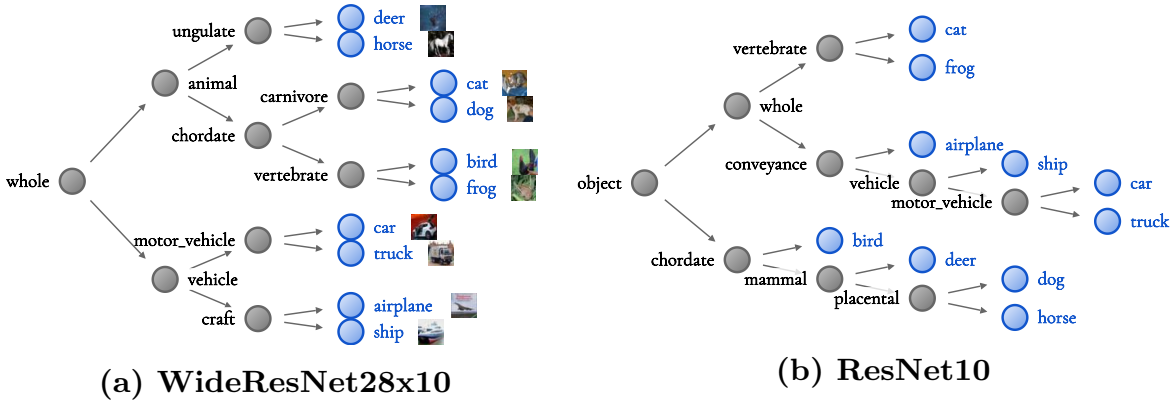


Fig. 8: CIFAR10 induced hierarchies for (a) WideResNet (97.62% acc) and (b) ResNet (93.64% acc), with automatically-generated WordNet hypotheses for node meanings.

5.2 Sidestepping the Accuracy-Interpretability Tradeoff

Induced hierarchies cluster vectors in weight space, but classes that are close in weight space may not have similar semantic meaning: Fig. 8 depicts the trees induced by WideResNet20x10 and ResNet10, respectively. While the WideResNet induced hierarchy (Fig. 8a) groups semantically-similar classes, the ResNet (Fig. 8b) induced hierarchy does not, grouping classes such as *Frog*, *Cat*, and *Airplane*. This disparity in semantic meaning is explained by WideResNet's 4% higher accuracy: we believe that higher-accuracy models exhibit more semantically-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 low-accuracy, interpretable models do not provide insight into high-accuracy decisions; an interpretable, state-of-the-art model is needed to interpret state-of-the-art neural networks.

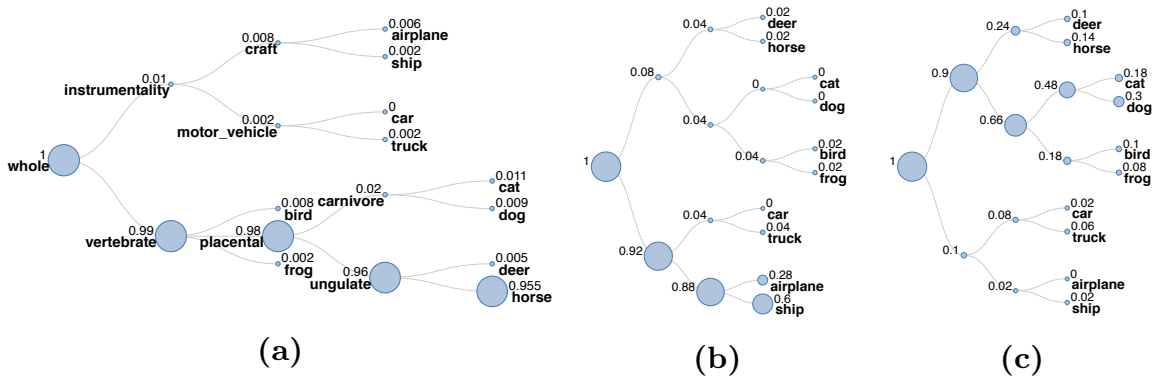


Fig. 9: Visualization of path traversal frequency for three different classes. (a) **In-Distribution Class: *Horse*** uses samples of a class found in training. Hypothesized meanings of intermediate nodes are from WordNet. (b) **Context Class: *Seashore*** uses samples unseen at training time, indicating reliance on context. (c) **Confusing Class: *Teddy*** uses samples that identify edge cases in node meanings.

5.3 Visualization of Tree Traversal

To interpret not only tree hierarchies but also tree traversals, we visualize the percentage of samples that pass every node (Fig. 9). This highlights both the correct path (the most frequently traversed) and allows us to interpret common incorrect paths (Fig. 9a). To be specific, we can interpret attributes shared between the leaves of traversed nodes. These attributes may be *Backgrounds* or *Scenes* but could also be *Color* or *Shape*. Fig. 9b depicts the paths of samples that describe context. In this case, very few of the animals are recognized in a *Seashore* environment while *Ship* is almost always seen in that environment. Fig. 9c depicts the paths of samples belonging to an out-of-distribution class that does not fit the attributes of the hypothesized node but maintains path consistency. In this case, *Teddy* tends toward the animal classes, specifically *Dog*, because it shares similar shape and visual features.

6 Conclusions

In this work, we propose Neural-Backed Decision Trees, removing the dichotomy between accuracy and interpretability; narrowing the accuracy gap between neural networks and decision trees to 1% on CIFAR10, CIFAR100, TinyImageNet and to 2% on ImageNet; advancing state-of-the-art for interpretable methods by ∼14% on ImageNet to 75.30% top-1 accuracy. Any classification neural network can be converted into an NBDT: We design decision tree structures called *induced hierarchies* that neural networks then fine-tune on, using a *tree supervision loss*. The network is then run as a decision tree using *embedded decision rules*. As a fortuitous side effect, the tree supervision loss also boosts the original neural network accuracy by 0.5%. Furthermore, to assess semantic meaning, 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.

References

1. Ahmed, K., Baig, M., Torresani, L.: Network of experts for large-scale image categorization. vol. 9911 (April 2016)
2. Alaniz, S., Akata, Z.: XOC: explainable observer-classifier for explainable binary decisions. CoRR **abs/1902.01780** (2019)
3. Baek, S., Kim, K.I., Kim, T.: Deep convolutional decision jungle for image classification. CoRR **abs/1706.02003** (2017)
4. Banerjee, A.: Initializing neural networks using decision trees (1990)
5. Banerjee, A.: Initializing neural networks using decision trees. In: Proceedings of the International Workshop on Computational Learning and Natural Learning Systems. pp. 3–15. MIT Press (1994)
6. Boz, O.: Converting a trained neural network to a decision tree dectext - decision tree extractor. In: ICMLA (2000)
7. Dancey, D., McLean, D., Bandar, Z.: Decision tree extraction from trained neural networks (January 2004)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
9. Dua, D., Graff, C.: UCI machine learning repository (2017), `http://archive.ics.uci.edu/ml`
10. Frosst, N., Hinton, G.E.: Distilling a neural network into a soft decision tree. CoRR **abs/1711.09784** (2017)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
12. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
13. Humbird, K., Peterson, L., McClarren, R.: Deep neural network initialization with decision trees. IEEE Transactions on Neural Networks and Learning Systems **PP**, 1–10 (October 2018)
14. Ivanova, I., Kubat, M.: Decision-tree based neural network (extended abstract). In: Machine Learning: ECML-95. pp. 295–298. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
15. Ivanova, I., Kubat, M.: Initialization of neural networks by means of decision trees. Knowledge-Based Systems **8**(6), 333 – 344 (1995), knowledge-based neural networks
16. Kontschieder, P., Fiterau, M., Criminisi, A., Rota Bulo, S.: Deep neural decision forests. In: The IEEE International Conference on Computer Vision (ICCV) (December 2015)
17. Krishnan, R., Sivakumar, G., Bhattacharya, P.: Extracting decision trees from trained neural networks. Pattern Recognition **32**(12), 1999 – 2009 (1999)
18. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep. (2009)
19. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge (2015)
20. LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database. ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist **2** (2010)
21. McGill, M., Perona, P.: Deciding how to decide: Dynamic routing in artificial neural networks. In: ICML (2017)
22. Miller, G.A.: WordNet: A lexical database for english. Commun. ACM **38**(11), 3941 (Nov 1995). https://doi.org/10.1145/219717.219748, `https://doi.org/10.1145/219717.219748`

23. Murthy, V.N., Singh, V., Chen, T., Manmatha, R., Comaniciu, D.: Deep decision network for multi-class image classification. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
24. Peterson, J.C., Soulos, P., Nematzadeh, A., Griffiths, T.L.: Learning hierarchical visual representations in deep neural networks using hierarchical linguistic labels. CoRR **abs/1805.07647** (2018)
25. Petsiuk, V., Das, A., Saenko, K.: Rise: Randomized input sampling for explanation of black-box models. In: Proceedings of the British Machine Vision Conference (BMVC) (2018)
26. Ribeiro, M.T., Singh, S., Guestrin, C.: "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. pp. 1135–1144 (2016)
27. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 618–626 (2017)
28. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
29. Siu, C.: Transferring tree ensembles to neural networks. In: Neural Information Processing. pp. 471–480 (2019)
30. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.A.: Striving for simplicity: The all convolutional net. CoRR **abs/1412.6806** (2014)
31. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. International Conference on Machine Learning (ICML) 2017 (2017)
32. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2019)
33. Tanno, R., Arulkumaran, K., Alexander, D.C., Criminisi, A., Nori, A.: Adaptive neural trees (2019)
34. Teja Mullapudi, R., Mark, W.R., Shazeer, N., Fatahalian, K.: Hydranets: Specialized dynamic architectures for efficient inference. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
35. Veit, A., Belongie, S.: Convolutional networks with adaptive inference graphs. In: The European Conference on Computer Vision (ECCV) (September 2018)
36. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
37. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European Conference on Computer Vision (ECCV). pp. 818–833. Springer (2014)
38. Zhang, J., Lin, Z., Brandt, J., Shen, X., Sclaroff, S.: Top-down neural attention by excitation backprop. In: European Conference on Computer Vision (ECCV). pp. 543–559. Springer (2016)
39. Zhang, Q., Yang, Y., Ma, H., Wu, Y.N.: Interpreting cnns via decision trees. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)

NBDT: Neural-Backed Decision Trees Supplementary Material

1 Tree Supervision Loss Formulation

As described in Sec 3.3, the tree supervision loss has both a soft variant and a hard variant. Usage depends on whether the neural-backed decision tree's inference mode is soft or hard. Consider a tree with N nodes $n(i)$.

Soft inference produces this predicted class distribution by obtaining a probability for each class, or leaf, independently. For leaf ℓ, consider the path P_ℓ connecting that leaf to the root. Along this path, we have nodes with indices $i_{\ell,j}$ for the jth node from leaf ℓ to the root. Written succinctly, path $P_\ell = \{n(i_{\ell,j})\}_{j=1}^{k}$ for a k-length path. Each node $n(i_{\ell,j})$ along this path produces the probability of traversing the next node $n(i_{\ell,j+1})$; this probability is denoted $p(i_{\ell,j}, i_{\ell,j+1})$. As a result, the probability of leaf ℓ and its corresponding class c is

$$p(c) = p(\ell) = \Pi_{j=1}^{k-1} p(i_{\ell,j}, i_{\ell,j+1}) \tag{1}$$

The distribution over classes is then written as $\mathcal{D}_{\text{pred}} = \{p(c)\}_{c=1}^{\mathcal{C}}$, for a distribution with $\mathcal{C}$ total classes. In soft inference, the final class prediction $\hat{c}$ is defined over these class probabilities,

$$\hat{c} = \text{argmax}_c(\mathcal{D}_{\text{pred}}) = \text{argmax}_c p(c) = \text{argmax}_c \Pi_{j=1}^{k-1} p(i_{c,j}, i_{c,j+1}) \tag{2}$$

The **soft tree supervision loss** is a single cross entropy loss term defined between the label distribution and the predicted class distribution, produced by soft inference.

$$\textsc{CrossEntropy}(\mathcal{D}_{\text{pred}}, \mathcal{D}_{\text{label}}) \tag{3}$$

On the other hand, **hard inference** produces a prediction at each node, for which node to traverse next. Each node $n(i)$ produces a probability of child node j; this probability is denoted $p(i, j)$. This distribution over s child nodes is summarized as $\mathcal{D}(i)_{\text{pred}} = \{p(i, j)\}_{j=1}^{s}$ Each node thus picks the next node using

$$n(j) = \text{argmax}_j(\mathcal{D}(i)) = \text{argmax}_j p(i, j) \tag{4}$$

The **hard tree supervision loss** defines a cross entropy loss for every node. We know in advance which class is correct and thus know which leaf is correct. Since each leaf only features one path to the root, we know which child node is correct for each node. We will denote this label distribution for node i as $\mathcal{D}(i)_{\text{label}}$. The complete loss matches the following:

$$\frac{1}{N}\sum_{i=1}^{N}\textsc{CrossEntropy}(\mathcal{D}(i)_{\text{pred}}, \mathcal{D}(i)_{\text{label}}). \tag{5}$$

2 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 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.

$$\langle x, \frac{1}{n}\sum_{i=1}^{|C|} w_i\rangle = \frac{1}{n}\sum_{i=1}^{|C|}\langle x, w_i\rangle = \frac{1}{n}\sum_{i=1}^{|C|}\hat{y}_i, i \in C \tag{6}$$

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

3 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. Although these results are not reported, 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)

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 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`.

4 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 semantic meanings can be drawn from intermediate nodes of larger trees such as the one for CIFAR100. Fig. 1 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. 1 that subtrees can be grouped by semantic meaning, which can be a Wordnet attribute like *Vehicle* or *Household Item*, or a more contextual meaning such as shape or background like *Cylindrical Object in Blue Background*.

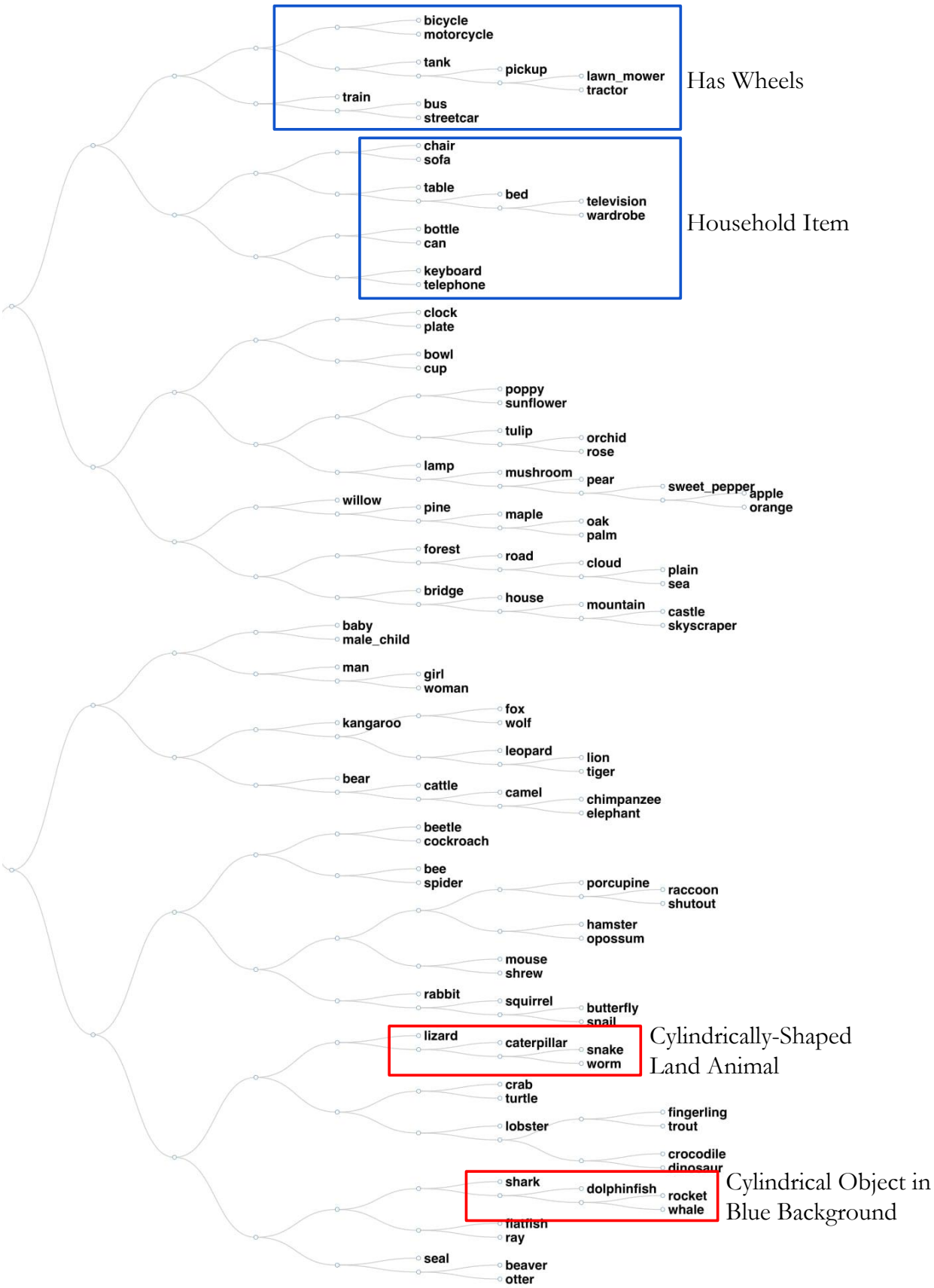


Fig. 1: CIFAR100 tree visualization on WideResNet28x10. The classes in the blue boxes represent subgroups with a semantic meaning derived from WordNet while the classes in the red boxes represent subgroups with a semantic meaning related to the image itself, such as shape or background. The root node is cropped out for readability.