Interpretable Few-Shot Image Classification with Neural-Backed Decision Trees

Scott Lee
Joseph Gonzalez, Ed.
Matthew Wright, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-71
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-71.html

May 27, 2020

Acknowledgement

Thank you to my advisor, Joseph E. Gonzalez, and my reader, Dr. Matthew A. Wright, for their guidance in research and writing. I would also like to thank my co-authors Alvin Wan, Lisa Dunlap, Daniel Ho, Jihan Yin, Henry Jin, and Suzie Petryk.

Interpretable Few-Shot Image Classification with Neural-Backed Decision Trees

by

Scott Lee

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Joseph E. Gonzalez, Chair
Dr. Matthew A. Wright

Spring 2020

Interpretable Few-Shot Image Classification with Neural-Backed Decision Trees

Abstract

Interpretable Few-Shot Image Classification with Neural-Backed Decision Trees

by

Scott Lee

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Joseph E. Gonzalez, Chair

Recent advances in deep learning techniques, especially convolutional neural networks (CNNs), have caused an explosion in their popularity for image-based tasks. Although CNNs achieve state-of-the-art accuracy and can be applied to a variety of tasks, they are difficult to interpret, as humans often cannot explain the reasoning behind the decisions made by these "black box" models. We propose a novel method, Neural-Backed Decision Trees (NBDTs), that combines the explainable nature of decision trees with the high predictive power of neural networks, and demonstrate competitive accuracy on standard image classification tasks (CIFAR10, CIFAR100, TinyImageNet). We quantitatively and qualitatively analyze the quality of explanations for the decisions made by our method, finding that related classes are clustered with higher similarity compared lexical methods. Additionally, we extend NBDTs to few-shot image classification on the Animals with Attributes 2 dataset, maintaining high accuracy on seen classes and achieving competitive accuracy for few-shot classes.

Dedicated to everyone who has supported me and brightened my life, especially my parents and friends who saw me through my toughest times.

Contents

List of Figures

List of Tables

Acknowledgments

Thank you to my advisor, Joseph E. Gonzalez, and my reader, Dr. Matthew A. Wright, for their guidance in research and writing. I would also like to thank my co-authors Alvin Wan, Lisa Dunlap, Daniel Ho, Jihan Yin, Henry Jin, and Suzie Petryk.

Chapter 1

Introduction

In image-based tasks such as image classification and semantic segmentation, deep learning techniques, especially convolutional neural networks (CNNs), have become increasingly popular due to their high accuracy [14, 6, 12, 23, 7, 8]. Although CNNs often yield state-of-the-art accuracy results for such tasks, they provide little insight into the decision-making mechanism of the neural network itself. Indeed, this major limitation of CNNs makes it difficult to diagnose their misclassifications and other errors. Despite this, CNNs are still widely favored due to their high performance, especially in cases where forgoing explainability yields significantly higher accuracy.

With increasing applications of computer vision and artificial intelligence in industry, however, explainable AI has become a major priority in high-stakes applications (e.g. medical services, autonomous vehicles) and right-to-explanation interpretability tasks (e.g. explaining insurance decisions, GDPR compliance). In such sensitive scenarios where it is paramount to understand how a model arrives at its prediction, decision trees are one of the most popular methods due to their transparency. By reducing the task at hand into a series of smaller sub-decisions, each represented by a branch split in the tree, one can better infer which decisions are made for a given input by tracing the path taken in the tree; furthermore, even for incorrect predictions, one can analyze the outcome by locating the branches with incorrect sub-decisions. Even when decision trees do not output the correct prediction for a particular input, they provide useful information for both *human intervention* (when a human manually change the structure of a tree) and *human decision-making* (when a human analyzes and, if necessary, corrects the decision tree's generated prediction and path taken in order to arrive at a final decision).

As mentioned previously, however, decision trees have their shortcomings; CNNs far outperform decision tree-based methods, even when combining many such trees in ensemble-based methods. Moreover, previous attempts of integrating deep learning methods with decision trees often require significant reworking of the underlying neural networks. Thus, we propose a method that combines neural networks and decision trees while addressing both of these issues called Neural-Backed Decision Trees (NBDT). Given a neural network (which can be pretrained), we produce a decision tree architecture that not only yields competitive

accuracies on standard image classification tasks, but also elucidates the underlying model's decision-making process. We demonstrate strong performance across several datasets as well as qualitative methods for evaluating the explainability of the NBDT model, and describe a framework for applying NBDTs towards few-shot image classification.

The remainder of this report is structured as follows: In Chapter 2, we discuss related works. In Chapter 3, we present our method, NBDTs, and demonstrate competitive results from experiments in Chapter 4. In Chapter 5, we analyze the decision-making power of our methods through quantitative (hypothesis testing) and qualitative (tree visualization) methods. In Chapter 6, we conclude the paper and briefly discuss future directions. [1]

[1] This report uses work done as a co-author of [28], in addition to work that will be submitted to future conferences such as NeurIPS 2020.

Chapter 2

Related Work

2.1 Explainability for CNNs

Although convolutional neural nets remain largely black-box entities, an increasing number of works are demystifying the inner mechanics of CNNs through visual explanations. Early work by Zeiler and Fergus involves visualizing layers of a small convolutional neural network to produce filters, which capture visual information such as edges, shapes, and colors [30]; their work led to the use of saliency maps and their variants as standards for demonstrating explainability in image tasks. Because this approach uses the parameters of the underlying CNN model, it is a *white-box method*; on the other hand, black-box methods do not require knowledge of the underlying network's architecture or parameters. We summarize works addressing both types of methods that aim to improve explainability in CNNs.

White-Box Methods

Many modern approaches rely on generating a variant of a saliency map, a technique that highlights visually unique pixels for a given input image.

Saliency maps. Given an image classification CNN and an input image, suppose our task is to determine which pixels of the input image most strongly support the CNN's predicted class (regardless of whether the prediction is correct or not). For example, given a picture of a scuba diver underwater, we want to confirm that our model uses the pixels related to the image class (scuba tank, snorkel, etc) rather than other contextual pixels that may or may not accompany the class (such as the dominating presence of water in the image background). For a class C and input image I, suppose the CNN calculates the class score function

$$S_C(I) = f(w) + b_C \tag{2.1}$$

where $f(w)$ is non-linear for a CNN and b_C is the additional bias term for class C. Since the most discerning pixels have the highest influence on the predicted class, they would also

have the highest influence on its class score $S_C(I)$. Simonyan et al. [22] formalize this idea by computing the first-order Taylor approximation of $S_C(I)$ with respect to the perturbed image I_0:

$$S_C(I) \approx w^T I + b \tag{2.2}$$

with the derivative approximation

$$w = \left.\frac{\partial S_C}{\partial I}\right|_{I_0} \tag{2.3}$$

The magnitude of this derivative w loosely indicates the "most important" pixels in classifying the image as class C, determining which pixels will have the largest effect with the smallest perturbations. To obtain the class saliency map $M \in \mathbb{R}^{m \times n}$ for a multi-channel $m \times n$ image, one can take the maximum of the derivative w across all channels for each pixel at (i, j):

$$M_{ij} = \max_c |w_{(i,j,c)}| \tag{2.4}$$

Overlaying such a map on the original image presents a visualization that highlights the most discriminative pixels of the image with respect to the image class.

Integrated gradients. Extending the idea of using gradients to determine pixel saliency, Sundararajan et al. [26] demonstrate a novel approach to generating higher quality saliency maps. To obtain "tighter" saliency maps, their method is optimized to discard less important salient pixels and only highlight the most important ones. First, given an input image, they generate copies of the original image differing in intensity (starting from a pitch black image, increasing pixel intensity until arriving at the original image). Then, they pass the copies to the CNN, yielding class prediction scores for each copy; we discard images where increasing the intensity does not significantly increase the class prediction score (i.e. whose gradient is close to zero on a plot of prediction score vs. intensity). To obtain the improved map, we take the gradient of the original image with respect to the remaining altered copies, followed by integrating over these gradients.

Class activation maps. Many standard image recognition tasks involve training on datasets with labels at the image level (as opposed to finer labels such as bounding boxes). Research has shown that despite this limitation, CNNs trained on such data are able to perform weak object localization. For instance, Zhou et al. [34] extend this idea with class activation mapping (CAM), a heatmap of pixel relevance for a given input image. By applying global average pooling just before the final output layer of the network, one obtains the spatial averages of the feature map for each neuron of that layer; overlaying a weighted average (weighted by the weights of the fully connected layer) of these spatial averages produces the desired class activation map. We can then predict the object's bounding box by select the region surrounding the highest pixel importance in the image. Examining the CAMs generated from performing top-k classification is especially revealing in cases where the top-1 prediction is incorrect while the correct class is within the top-k, since it enables users to determine which parts of the image can be ambiguous. CAM is an example of

a *spatial grounding method*, which are used to determine the location of objects (usually specified by written description) in an image.

Selvaraju et al. [21] generalize class activation mappings with a procedure called *Guided Grad-CAM*, which takes advantage of additional information, in the form of gradients, given to the final convolutional layer. These additional changes yield improved object localization and class discriminative heatmaps.

Black-Box Methods

So far, we have discussed white-box methods that require access to the underlying network's architecture or parameters. Next, we examine several techniques that act directly on an arbitrary model without needing such information, known as black-box methods.

LIME. Various other works have extended the perturbation-based method used for saliency maps [22], as discussed previously. In their work, Ribeiro et al. describe a method for providing *local model interpretability* called LIME (Local Interpretable Model-Agnostic Explanations) [20]. It does so by applying perturbations on input data, observing the changes in output, and outputting a list of features and the magnitude of their impact on the model's prediction. Creating local linear approximations of the underlying model makes is easier to explain its decision, since interpreting a linear model is much simpler than understanding the architecture of a complicated neural network. Because such a linear model has the potential to underfit the complexities of a CNN, LIME may not provide compelling explanations in such cases, especially when involving large, complex datasets. Moreover, the perturbations applied to the data during LIME are often specific to the task at hand, so it is difficult to generalize them to many tasks.

RISE. Early work for improving weakly-supervised object localization by Singh et al. involves enforcing regularization on a CNN by randomly masking input images as data augmentation before training [24]. They demonstrate that this improves performance because the method discourages the model from localizing only the most discriminative features, which are masked out randomly during the data augmentation. A method proposed by Petsiuk et al. [19] called RISE uses a similar approach to produce higher quality saliency maps as compared to other techniques, outshining white-box methods in some cases.

RISE (Randomized Input Sampling for Explanation of Black-box Models) takes any trained model and probes it with randomly masked versions of the same input image, saving the results from each masked input. The final saliency map is a linear combination of these outputs, weighted by the output probabilities predicted by the model on the corresponding masked input image. They further describe a *deletion metric*, which measures the decrease in probability of a class prediction after deleting the input image's important pixels as indicated by the resulting saliency map. RISE improves on other methods such as Grad-CAM and LIME, both quantitatively in terms of the deletion metric and qualitatively in terms of higher quality, tightly accurate saliency maps.

Guided Zoom. An approach proposed by Bargal et al. called Guided Zoom [5] examines the spatial grounding of a model's decision in order to make more explainable and

justified predictions. For a given deep model as input, the *Evidence CNN* recovers the most discriminating patches of correctly classified images with a combination of spatial grounding methods (including Grad-CAM, RISE, and contrastive excitation backpropagation [31]) and highlights the most salient parts of the input images. By utilizing the evidence from top-k prediction results, as opposed to using only a top-1 prediction, Guided Zoom improves the classification accuracy for a given CNN model.

2.2 Decision Trees and Neural Networks

In this section, we survey a sample of works combining the usage of decision trees and neural networks in order to improve accuracy and interpretability.

We begin with a psychology perspective of pattern recognition in images. *Category learning* is the ability to formulate mental groups in order to classify a set of objects (in this case, extracting visual features from images in order to predict classes). Peterson et al. [18] illustrate the following example of category learning: when a child is shown a picture of a Dalmatian called Sebastian, it needs to determine if the label "dog" refers to Sebastian, all Dalmatians, all four-legged animals, or all animals—essentially figuring out where the label belongs in a hierarchy of classes. They focus on training an image classifier with a CNN that exploits multiple levels of labels per image, ranging from highest to lowest (e.g. animal, four-legged animal, dog, Dalmatian).

This idea of generating a hierarchy of classes naturally extends to the use of decision trees in deep learning models. Murthy et al. [16] propose a deep decision network that is trained level by level in a tree-like network structure. In addition, they present a novel approach for clustering groups of relevant classes to generate the hierarchy. By using a new loss function that encourages approximate block diagonalization of the confusion matrix during each step of optimization, they encourage the generated clusters to have high confusion within each cluster, but low confusion across different clusters. The motivation behind constructing clusters in this manner is that by first sorting classes into broader categories (e.g. animal, vehicle) that share common visual characteristics, it allows the model to better identify the discriminating features within each category (as opposed to extracting these features across all categories). Because classes within the same cluster most likely already share many common visual features, this framework allows the model to hone in on more subtle differences between the similar classes.

Ahmed et al. [1] take advantage of this key observation in their "Network of Experts" (`NOfE`) framework, which divides image classification into two smaller tasks. First, a *generalist* network is trained to categorize input images into one of several coarse categories of classes. Then, for each category, a *category expert* network learns specialized features for images in a category. They propose that the generalist network provides an initial knowledge base for all experts (in the form of shared features), while the category expert networks will learn the most distinguishing features within a group of very similar classes. The generalist network forms the initial "trunk" of the decision tree structure, with branches coming out of

the trunk for each coarse category and leaves of the coarse category branches corresponding to a class prediction.

Instead of using a single network to generate the decision tree structure, Alaniz and Akata propose a two-network framework called Explainable Observer-Classifier (XOC) [3]. In XOC, the first network, a decision tree classifier, formulates binary questions related to the input image's visual features (e.g. "Does the object in the image have whiskers?"), while the second network, the binary-attribute observer, provides an answer to the decision tree classifier's question based on the input image. The two networks feed off of each other to improve predictions, and the resulting tree provides yes/no-type insight on decision making. They perform zero-shot learning with the generated attribute tree, beating a tree generated with naive clustering of Word2Vec features.

The decision tree-based works mentioned in this section so far are white-box methods, as they define entirely new network architectures or frameworks that are incompatible with existing pretrained models. Work by Zhang et al. [32] instead attempts to explain the reasoning behind the decisions made by a model. Given a pretrained network, they learn an explanatory decision tree that works in two stages. First, they extract *disentangling* feature filters learned in early layers of the network and map middle-layer features to a semantic part-of-image concept (e.g. one filter may look for the presence of an animal head shape). These middle-layer features are then "bridged" with final predictions by identifying which set of object parts (i.e. the part-of-image concepts represented by middle-layer features) are activated and contributing to the final prediction result. The CNN is able to choose different object parts for a given image; for instance, the early-layer filters identified for a flying bird may differ from those for a stationary bird, and any differences in the decision-making process of the pretrained network can be traced by the path on the resulting decision tree.

2.3 Few-Shot Learning

Next, we examine works related to *few-shot learning*, the ability to classify images of a class given only a few samples at training time. These tasks are also commonly extended to the more difficult one-shot and zero-shot variants. For instance, suppose we use an animal dataset and we designate `cat` as a few-shot class. The model would be trained on a set of training images, which only includes a few samples of cats. Then at inference, the model is shown a mixture of images, at which point it should classify between all classes (including `cat`). Because the model is given less information to work with during training, it is common to incorporate knowledge from outside data sources (such as WordNet or Word2Vec) to assist with few-shot classification, as many of the works discussed below do.

Akata et al. [2] frame few-shot image classification as a label-embedding problem, where each image class is represented as a vector of attributes, and the model matches a given image to the best-fit class based on its proximity in the space of attribute vectors. An advantage to their method is that they can take advantage of outside data sources without additional

engineering efforts; in fact, any combination of labeled examples is compatible within their framework, even if the images within a class have different degrees of attribute labels. We also note that early work by Socher et al. [25] demonstrated two different strategies for classification of zero-shot classes, one that prioritizes high accuracy on unseen classes and another that maintains accuracy of seen classes while conservatively selecting unseen classes as the prediction.

Zhang and Saligrama [33] take a similar approach with class attributes, but their method views each sample image as a combination of classes seen at training time (with the hypothesis that images from the same class will have similar combinations). Their model learns a function that maps any sample data into a vector in this semantic space, then selects the most likely class based on other nearby examples. Instead of viewing each sample image as a combination of classes, Park et al. [17] propose learning a set of meta-learners that are trained on a distribution of related tasks, then adapting the ensemble of meta-learners for few-shot classification. The weight prediction network that controls the mixture of meta-learners is tuned using seen and unseen examples.

A major drawback in using an attribute space-based method is that the model's performance degrades due to domain shift, especially for one and zero-shot learning, where the model has even less data than one would in a few-shot setting. Kodirov et al. [9] propose learning a semantic autoencoder/decoder pair that projects an image into semantic space similar to previously mentioned methods, with the constraint that the decoder can reconstruct the original visual features using the encoded projection. Using linear encoder and decoders, their method is efficient and able to better generalize to few-shot classes.

Chapter 3

Neural-Backed Decision Trees

In this chapter, we detail our method of generating a decision tree structure given any (pretrained) neural network; we refer to such a decision tree as a *neural-backed decision tree* (NBDT). As discussed in Chapter 1, decision trees provide a structure that is easily explainable as a sequence of decisions that can be traced down the path taken during inference. Therefore, it is in our best interest to generate a tree structure that (a) best separates classes into distinct super-categories, represented by the parent-children structure of tree nodes, (b) has decisions that are easy to interpret with little ambiguity, accomplished by constructing node splits such that the children have little overlap of super-categories.

At a high level, the steps to create an NBDT are: (1) select a neural network architecture to use as the backbone and train the network; (2) construct an induced hierarchy using the pretrained network weights; (3) fine-tune the model with a novel loss function. To conduct inference, one would featurize the input image using the neural network backbone, then feed the input through the model, running the decision rules (from the induced hierarchy) embedded in the final layer of the model.

3.1 Motivation

During inference with a neural network, an input image is featurized with the weights of the final (fully-connected) layer, resulting in a class scores vector $\hat{y}$. This process is a linear transformation, weighted by the weight matrix $W \in \mathbb{R}^{k \times d}$ of the final layer. We can represent this calculation as a set of inner products:

$$\underbrace{\begin{bmatrix} — & w_1 & — \\ — & w_2 & — \\ & \vdots & \\ — & w_d & — \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} \langle x, w_1 \rangle \\ \langle x, w_2 \rangle \\ \vdots \\ \langle x, w_d \rangle \end{bmatrix}}_{\hat{y}} \tag{3.1}$$

The model's predicted class is the index of the largest class score, $\arg\max(\hat{y})$.

We first construct a naive decision tree structure. Consider a tree with a single root node having d children (all of which are leaves). Each leaf node corresponds to a row in the weight matrix, so we assign each node a vector $r_i = w_i$ that is equal to its corresponding row in W. Using this tree, we can run the same inference process as we would with the original neural network by simply finding d inner products of the form $\hat{y}_i = \langle x, r_i \rangle$, then taking the argmax of the resulting scores.

In both inference pipelines above, the resulting calculation involves finding an argmax of the inner product scores:

$$\text{prediction} = \arg\max_i \langle x, w_i \rangle = \arg\max_i \langle x, r_i \rangle \tag{3.2}$$

Hence, the two methods utilize the same calculations during inference, which we denote as applying the *embedded decision rules*.

3.2 Taller Trees

Because the naive tree merely computes the same calculations as the original backbone network, it is clear that the naive tree brings no advantage during inference. Therefore, we insert additional intermediate nodes into the tree. Similar to the idea of separating the network into a generalist and a category expert in the "Network of Experts" framework [1], adding intermediate nodes will allow the network to first separate super-categories in early levels of the tree, then apply finer-grained inference within a super-category in lower levels. An intermediate node i will also need a node vector r_i, but since it does not correspond to a particular row in the weight matrix W (only leaf nodes do), we assign it the mean of its childrens' node vectors

$$r_i = \frac{1}{|\mathcal{C}_i|} \sum_{c \in \mathcal{C}_i} r_c \tag{3.3}$$

where $\mathcal{C}$ is the set of node i's children. Because such a tree now requires a sequence of decisions, we must alter our method for inference. We present two related methods, also summarized in Figure 3.1:

Hard Decision Tree. Starting at the root node, calculate inner products with each of its children's vectors. Move to the child node with the largest score and repeat the process, traversing the tree until arriving at a leaf node. The predicted class is the class associated with this leaf. This method only ever considers one path down the tree (following the path with the highest probability at the current level).

Soft Decision Tree. Starting at the root node, calculate inner products with each of its children's vectors. Repeat the process for every node and compute a softmax, yielding the probabilities of traversing from a node to its children for every node in the tree. Then for every leaf node, find its total probability by multiplying the node traversal probabilities on the path from the root to that leaf. Finally, we compute an argmax over the distribution of

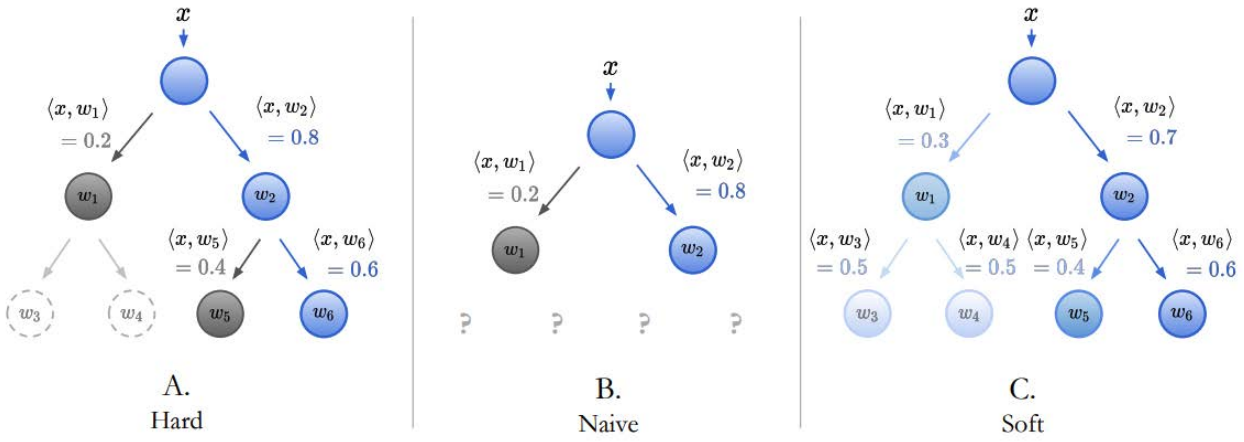

Figure 3.1: **Hard, Naive, and Soft Decision Trees.** The naive decision tree structure described in the Motivation section consists of a root node and k leaves (Tree B). The hard and soft decision trees are constructed by adding intermediate nodes to the naive tree. In the hard tree (Tree A), we traverse the tree by picking the node with the largest inner product until we reach a leaf. In the soft tree (Tree C), we calculate the probability of each path from the root to a leaf, and choose the path (and therefore the leaf) associated with the highest probability.

leaf probabilities in order to find the predicted class associated with the maximum probability leaf. This method considers all paths in the tree, yielding probabilities for each of them.

By using one of these two inference methods, we can convert the calculation of any neural network into a sequence of embedded decision rules represented by the tree structure. Because both the Hard and Soft decision rules rely heavily on the hierarchy defined by the tree (i.e. the positions of leaf nodes and the super-category groupings represented by intermediate nodes), we now turn our attention to generating a high quality hierarchy.

3.3 Induced Hierarchies

Given a pretrained backbone network, we can generate an *induced hierarchy*, the tree structure used during inference. We propose two methods for constructing the induced hierarchy from the final layer weights of the backbone network.

Clustering. Run hierarchical agglomerative clustering (HAC) on the leaf node vectors r_i. Since each leaf vector corresponds to a row in the weight matrix W, we can run HAC over w_i's (3.2). As mentioned in the previous section, to get the vectors for an intermediate node, we take the average of its children's vectors.

WordNet. WordNet [15] is a large lexical dataset that provides information about

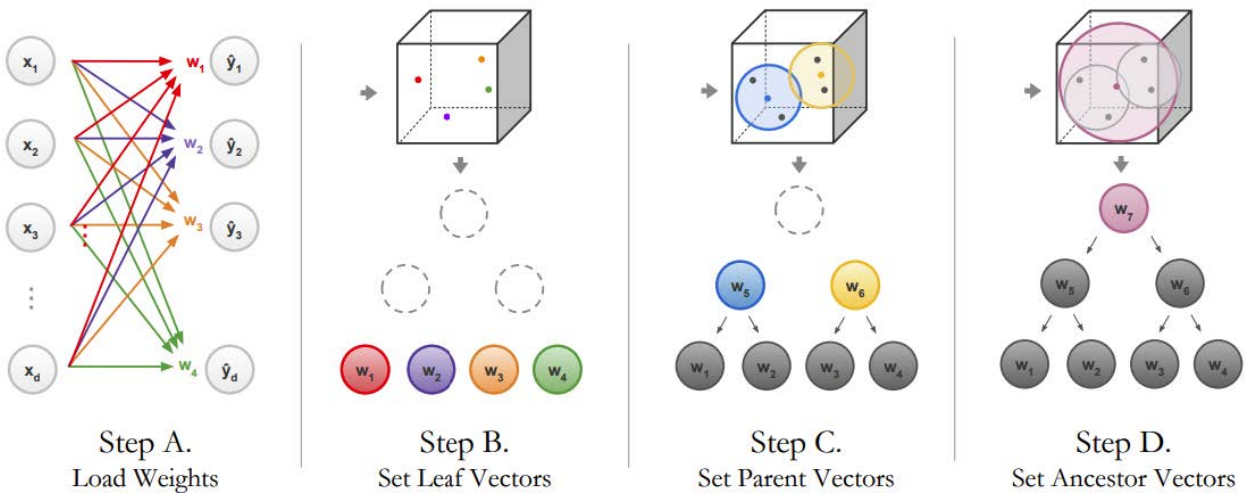

Figure 3.2: **Induced hierarchies via clustering.** To construct an induced hierarchy, there are four steps: (a) obtain a trained neural network and extract the weights of the final layer, denoted as $W \in \mathbb{R}^{d \times k}$; (b) for each class, create a leaf node with vector $r_i = w_i$ as in the naive tree; (c) set each parent node's vector as the average of its children's vectors; (d) for each ancestor node, collect the vectors of all leaf nodes contained in the subtree rooted at the ancestor; set the ancestor node's vector to be the average of these vectors (e.g. the root would have a vector that is the average of every leaf vector).

relationships between words. In particular, we can use WordNet to extract a linguistics-based hierarchy for image class labels. This results in an induced hierarchy that mirrors category learning; for example, WordNet yields additional categories for a `Dalmatian` as `dog`, `animal`, and `organism`, so the tree would place these labels as a chain of children under `organism`.

3.4 Tree Supervision Loss

The final component needed for using NBDTs is a modified loss function that takes into account intermediate nodes of the induced hierarchy. During the training phase of the original backbone network, it optimizes the loss function by separating vectors for each class (i.e. node vector for leaves); however, it neglects the existence of intermediate nodes, and as such, does a poor job in separating the vectors corresponding to these nodes (see 3.3 for a toy example). Therefore, we introduce additional loss terms for the two decision rules discussed previously:

Hard Decision Tree. During backbone training, the objective is to minimize the k-way cross entropy loss across the k classes in the dataset, which correspond to the k leaf nodes.

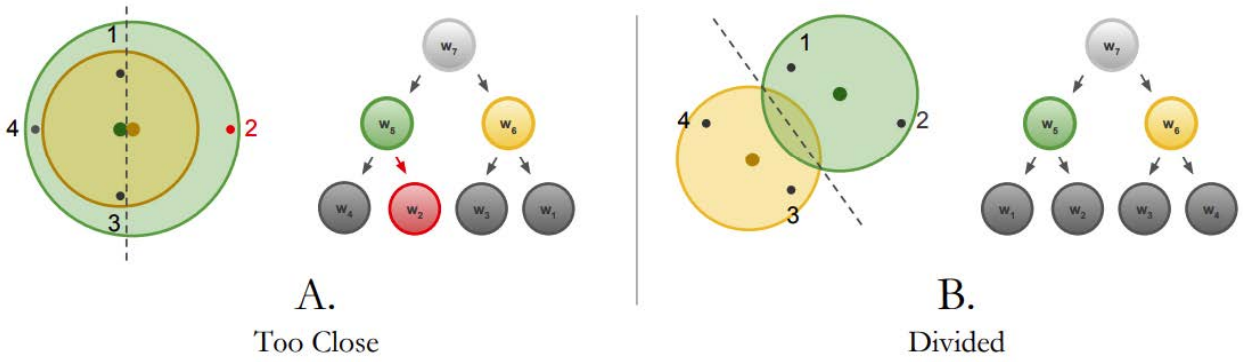

Figure 3.3: **Class separation.** For four leaf nodes shown on the plot of points, we construct two induced hierarchies, with the green and yellow circles representing the "range" of child nodes covered by the corresponding colored nodes. The center of each colored circle represents that node's representative vector, which is the average of its childrens' vectors. In (a), all samples of class 2 (red) are difficult to classify correctly because they are closest to the yellow circle center; hence, all such samples are sent down the wrong branch (i.e. to the yellow branch with classes 1 and 3). In (b), however, the hierarchy groups classes 1-2 and 3-4 together, resulting in more separable clusters. We can concretely visualize this by considering the margin between the two centers; we observe that the margin in (b) is much greater than the margin in (a), implying better separability.

Similarly for an intermediate node with $|\mathcal{C}|$ children, we add a $|\mathcal{C}|$-way cross-entropy loss term to the overall objective.

Soft Decision Tree. Using the soft decision rule yields a probability distribution over leaf nodes (after the softmax). The only additional term we add is a k-way cross entropy loss over this leaf distribution.

In both of the rules above, the additional loss terms are scaled by a hyperparameter λ, the *tree supervision loss weight*, to control the trade-off between optimizing the overall network cross-entropy loss term and the intermediate node loss terms. In other words, the total loss takes the form

$$\mathcal{L}_{\text{cross entropy}} + \lambda \mathcal{L}_{\text{tree supervision loss}} \tag{3.4}$$

In the next chapter, we present results for various values of the tree supervision loss weight.

Chapter 4

Experiments

To demonstrate the effectiveness of NBDTs, we run classification tasks on several standard image datasets (CIFAR10 [11], CIFAR100 [11], TinyImageNet [13]) and observe state-of-the-art results. Furthermore, we show that the novel tree supervision loss term improves classification performance by 0.5% across various datasets and model architectures.

For fine-tuning, our experiments use SGD with a momentum value of 0.9, weight decay of 5^{-4}. We train for 200 epochs with an initial learning rate of 10^{-2}, decaying by 90% at $\frac{3}{7}$ and $\frac{5}{7}$ of training. All training is completed using one Titan Xp, with batch sizes of 256 for TinyImageNet and 512 for CIFAR10 and CIFAR100. We use a tree supervision loss weight of $\lambda = 10$, as we find that this provides the best tradeoff between optimizing the cross entropy loss term and the tree loss term. We also note that these additional loss terms lengthen our training (in terms of wall clock time), by roughly 10% and 25% for soft and hard loss, respectively.

4.1 Results

We compare our results on CIFAR10, CIFAR100, and TinyImageNet to other baseline accuracies from decision tree-based methods, using the results reported in the original paper or derived from an improved ResNet18 backbone. The baselines for comparison shown in Table 4.1 are: Deep Neural Decision Forest (DNDF) [10], Explainable Observer-Classifier (XOC) [3], Deep Convolutional Decision Jungle (DCDJ) [4], Network of Experts (NofE) [1], Deep Decision Network (DDN) [16], and Adaptive Neural Trees (ANT) [27].

Our NBDTs hit accuracies of 97.57% and 82.87% on CIFAR10 and CIFAR100, respectively, which remain competitive with state-of-the-art methods not based on decision trees. Our soft decision tree matches a WideResNet28x10 model on CIFAR10 with a 0.05% margin, and beats it by 0.57% on CIFAR100. NBDTs outshine existing decision tree-based methods on CIFAR100, improving on the next best method (NofE) by 6.63%. On TinyImageNet, a larger dataset with 200 image classes, our method achieves accuracy within a 1.5% margin of ResNet18's accuracy while beating the next best method (DNDF) by 18.2%. The top

Method	Backbone Architecture	CIFAR10	CIFAR100	TinyImageNet
(Backbone network)	WideResNet28x10	97.62%	82.09%	–
ANT-A*	–	93.28%	–	–
XOC	–	93.12%	–	–
NOfE	ResNet56	–	76.24%	–
DDN	–	90.32%	68.35%	–
DCDJ	NiN	–	69.00%	–
NBDT-Hard	WideResNet28x10	97.55%	82.21%	–
NBDT-Soft	WideResNet28x10	**97.57%**	**82.87%**	–
(Backbone network)	ResNet18	94.97%	75.92%	64.13%
DNDF	ResNet18	94.32%	67.18%	44.56%
NBDT-Hard	ResNet18	94.50%	74.29%	61.60%
NBDT-Soft	ResNet18	**94.76%**	**74.92%**	**62.74%**

Table 4.1: **Accuracies.** We compare our method with various decision tree-based methods and find that NBDTs outperform them with a margin between 4% and 18%, including methods that sacrifice interpretability to boost accuracy; moreover, it remains within 1.5% of the original backbone network's performance.

baselines are often obtained with ensemble-based methods that hinder interpretability, while NBDT relies on a single tree structure.

4.2 Ablation Studies

Tree Supervision Loss. To see the effects of the additional tree supervision loss term described in Chapter 3, we train NBDTs on CIFAR100 and TinyImageNet with a tree supervision loss weight of $\lambda = 0.5$. We find that the new loss terms increases accuracy by 0.5% on average (Table 4.2). We further experiment with various values of λ and observe a significant decrease in performance when the tree supervision loss terms are highly weighted (by several orders of magnitude) over the cross entropy loss term. As seen in Table 4.3, however, our method is stable for reasonable values of λ, with a generous range of values that do not adversely affect performance.

WordNet Hierarchy. For CIFAR10 and CIFAR100, the accuracy based on a WordNet based hierarchy and the NBDT induced hierarchy are similar. For TinyImageNet, we observe a 4.17% accuracy gap, which will most likely only widen as the size and complexity of the dataset increase (Table 4.4). One reason for this gap is because linguistic similarity diverges from visual similarity, especially as additional finer-grain classes are incorporated. For instance, the WordNet hierarchy may hypothesize that `bird` and `cat` are both animals, so they are similar; from a visual perspective, however, `bird` and `airplane` are closely related because they are often photographed in the sky. In the next chapter, we discuss methods for visually analysing induced hierarchies.

Dataset	Backbone Architecture	No TSL	With TSL	Δ
CIFAR100	WideResNet28x10	82.09%	82.93%	**+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 4.2: **Tree Supervision Loss.** We compare the accuracy of methods with and without the tree supervision loss (TSL) on various combinations of datasets and backbones. The presence of TSL increases accuracy by 0.5% on average.

Dataset	Method	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 10$	$\lambda = 100$
CIFAR10	ResNet18	94.97%	94.91%	94.44%	93.82%	91.91%
CIFAR100	NBDT-Hard	–	94.50%	94.06%	93.94%	92.28%
CIFAR100	ResNet18	75.92%	76.20%	75.58%	75.63%	73.86%
CIFAR100	NBDT-Hard	–	66.84%	69.49%	73.23%	72.05%
TinyImageNet	ResNet18	64.13%	64.61%	63.90%	63.98%	63.11%
TinyImageNet	NBDT-Hard	–	43.05%	58.25%	56.25%	58.89%

Table 4.3: **Tree Supervision Loss Weight.** We compare the accuracy of methods with various values for the tree supervision loss weight λ. All experiments using NBDT-Hard trees have a ResNet18 backbone and uses hard inference. A TSL weight of $\lambda = 0$ corresponds to using the original loss function.

Dataset	Original	WordNet	Induced
CIFAR10	93.61%	93.65%	93.32%
CIFAR100	73.73%	71.79%	71.70%
TinyImageNet	61.01%	52.33%	56.50%

Table 4.4: **Induced Hierarchy Methods.** We compare the accuracy of using the WordNet and induced hierarchies with the performance of the original network. In all experiments below, we use a ResNet10 backbone and tree supervision loss weight $\lambda = 10$, with hard TSL and inference.

Chapter 5

Interpreting NBDTs

As mentioned previously, decision trees methods are generally accepted as one of the most interpretable, far outshining the interpretability of deep learning methods. One can easily encode a sequence of independent decisions as a path from the root of the tree to a leaf node representing the final prediction, with each intermediate node representing a single decision. Decision trees are easy to use and interpret when the data used to make the decisions are relatively simple; in the case of complex input such as images, however, it is difficult to concretely determine the purpose for each node (i.e. what is the decision being made at a particular node). In the previous section, we showed that NBDTs can provide near-state-of-the-art performance with any pre-existing neural network architecture. Now, we demonstrate their interpretability by presenting several analytical methods to evaluate the quality of the decision trees produced by the NBDT method, examining the decisions made by intermediate nodes of induced hierarchies.

5.1 Interpreting Node Decisions

When constructing the induced hierarchy, the weights corresponding to each node are drawn from the weights of the backbone network (for leaf nodes) or are linear combinations thereof (for intermediate nodes). Hence, it is difficult to extract meaning from the intermediate node weights, as the network weights do not dictate a particular decision rule or split on a specific attribute. However, one can form a *hypothesis* about the node's semantic meaning by examining its descendent leaf prediction classes and forming super-categories, then evaluating new related image samples using the induced hierarchy to see if the resulting predictions are consistent with the proposed hypothesis.

More concretely, suppose we have a hypothesis for a particular node in an induced hierarchy for the CIFAR10 dataset (e.g. the node splits on whether the image contains an `animal` or a `vehicle`). Here, we can analyze the descendent leaf nodes manually to form a hypothesis, or we can use an outside data source such as WordNet to extract linguistic relationships. Next, we create an *out-of-distribution dataset* for our hypothesis—that is, we

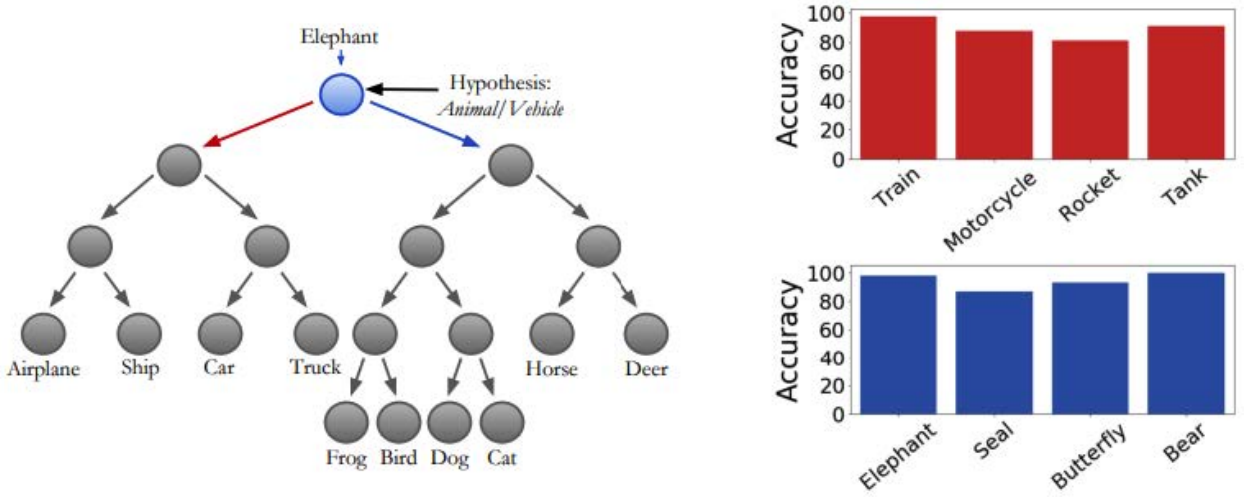

Figure 5.1: **Inferring a node's semantic meaning.** In the root node of the induced hierarchy shown on the left (for a WideResNet28x10 model), we test the hypothesis *Animal vs. Vehicle* on images of unseen (out-of-distribution) categories. The plots on the right show the proportion of images in each category that traverse to the correct child node (i.e. chooses the correct *Animal/Vehicle* path).

gather images that were not included in the training set, but are related to the hypothesis at hand. For example, an elephant is a type of `animal` and a bus is a type of `vehicle`, but neither are present in the original CIFAR10 dataset. We pass samples from our out-of-distribution dataset through the node we are examining, keeping track of the proportion of each out-of-distribution class that makes the correct decision based on the proposed hypothesis (Fig. 5.1). If the resulting proportions are low, the hypothesis may not be a good fit for the node, and we can repeat the process with a different guess. In the running example, for instance, if only a small proportion of `bus` images goes down the `vehicle` branch, this is an indication that the node is splitting on an attribute other than the one proposed by the hypothesis.

Tree Visualizations. To assist in this hypothesis confirmation process, we can visualize the induced hierarchy with the proportion of out-of-distribution sample inputs that pass through each node (Fig. 5.2). One can easily glean the ground-truth path by tracing the path from the root to the leaf, and in addition, compare it to the most common paths taken in the induced hierarchy to evaluate the node hypothesis. Furthermore, the tree visualization can reveal common contextual elements shared by leaf nodes on a particular tree path. Some common context attributes for images include background scenery, color, and shape (attributes that are visually related to the class object or its surroundings). For example, images of ships will often contain water, leading to large proportions of blue pixels. With this context in mind, if we pass sample images of `seashore`, we see that the majority of the samples go through the ship node, indicating that they share this common context.

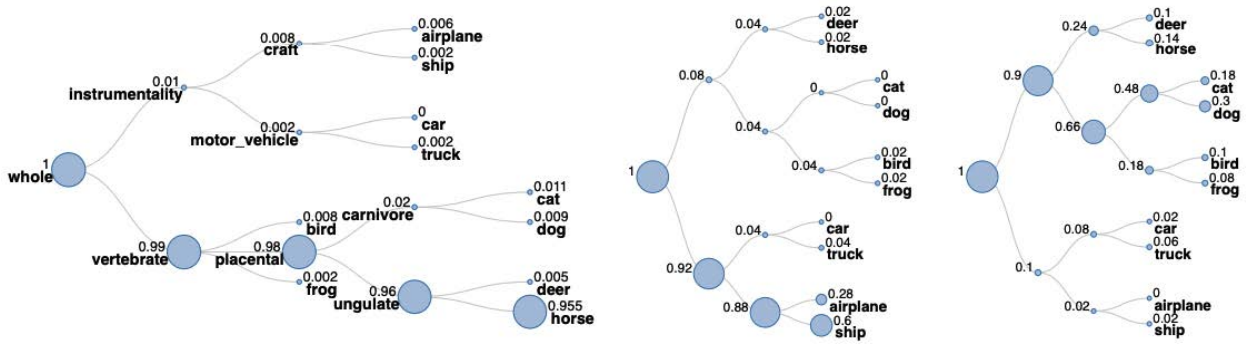

Figure 5.2: **Comparing paths of classes.** We visualize the path taken by samples from three different image classes; the leftmost hierarchy shows possible semantic labels for intermediate nodes. Left: `horse` images that exist in the training data (in-distribution class). Middle: `seashore` images that are not part of the training data and indicate context (context class). Right: `teddy` images that are difficult to group semantically with other classes (confusing class).

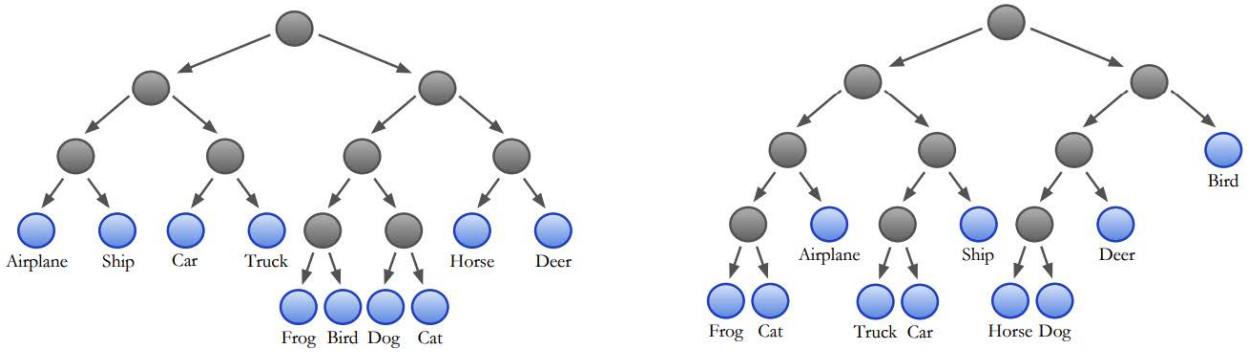

Figure 5.3: **Comparing Hierarchies.** Visualizations for induced hierarchies, using a WideResNet28x10 backbone (left) and a ResNet10 backbone (right). The grouping of classes in the WideResNet hierarchy have higher semantic correlation than that of the ResNet backbone.

5.2 Interpreting Model Quality

The NBDT method allows one to experiment with many neural network backbones of varying architectures, performance, efficiency, etc.; each of these different backbones yield different final layer weights, which in turn yields different induced hierarchies from NBDT. Therefore, it is important to examine the super-category groupings of classes created by our induced hierarchy to ensure that they make semantic sense. For instance, we can generate induced hierarchies for CIFAR10 using two different models, shown in Fig. 5.3. The tree generated with the WideResNet model groups classes with semantic meaning (grouping related animal and vehicle types under the same parent node), whereas the tree generated with the less powerful (in the number of trainable parameters) ResNet model fails to separate some

unrelated classes (in the figure above, `frog`, `cat`, and `airplane` are descendants of the same intermediate node). Although the ResNet model's induced hierarchy has more ambiguity in its semantic meaning, we can also analyze its errors in order to discover new potential correlation between classes. By doing so, we can ensure that the model not only provides high accuracy, but also provides meaningful and intuitive explanations via the induced hierarchy.

5.3 Few-Shot Learning

We have demonstrated both the predictive power and interpretability of NBDTs thus far; in particular, the underlying decision tree structure of NBDTs enables one to clearly trace the model's decision process. It is natural to extend this method to few-shot (as well as one-shot and zero-shot) learning tasks, where the model is only provided with a few (or one or zero) sample images of each class before inference. The process for using NBDTs to classify unseen class images at inference involves several more steps:

Few- and one-shot learning. For each class that was not seen during training, feed all sample images given for the class through the NBDT. Find the most common path taken in the induced hierarchy by the samples, and insert a new leaf node for the unseen class at the most common destination. During inference, use the newly generated induced hierarchy with added nodes for unseen classes and follow the standard inference process with NBDTs.

Zero-shot learning. Without passing any sample images for unseen classes through the NBDT, it is more difficult to guess where to insert the new node in the induced hierarchy. Similar to how the induced hierarchy is initially constructed, we can use (a) hierarchical agglomerative clustering or (b) a lexical dataset (e.g. WordNet) to determine the new location for a leaf node. For clustering, we first calculate the image centers for each class during training. For an unseen class image at inference time, we find the seen class with the closest image center, then insert the new node as a sibling or child of the seen class's node. To use a lexical dataset instead, we use Word2Vec to find the semantically closest seen class, then insert the new node as a sibling or child of that seen class's node.

When inserting a new node into the hierarchy, it is especially important that the sample images traverse as many nodes as possible in the tree, even if they do not reach the correct final leaf. In order to quantitatively identify which inserted nodes are properly placed, we define a *depth metric* for each class as follows. For a class C, find its ground truth path p_{gt} on the induced hierarchy (here, we define a path as the set of nodes included in the path). For each image $I_{C,i}$ of class C, feed it through the induced hierarchy and examine the predicted path p_{pred}; keep track of the number of correctly traversed nodes $n_{C,i} = |p_{gt} \setminus p_{pred}|$ (compared to the ground truth path). Then the depth metric for class C is the average proportion of correctly traversed nodes:

$$\text{depth}_C = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{n_{C,i}}{|p_{gt}|} \tag{5.1}$$

where

$$n_{C,i} = |p_{gt} \setminus p_{pred}| \tag{5.2}$$

is calculated for the ground truth and the predicted path generated at inference for each image $I_{C,i}$. For few-shot learning tasks, we use samples from the Animals with Attributes 2 dataset [29]. Generally, we observe that with any of the above methods, the predicted paths for unseen classes are indeed the most probable ones, resulting in depth metrics that roughly match those for seen classes (Fig. 5.4 and Fig. 5.5).

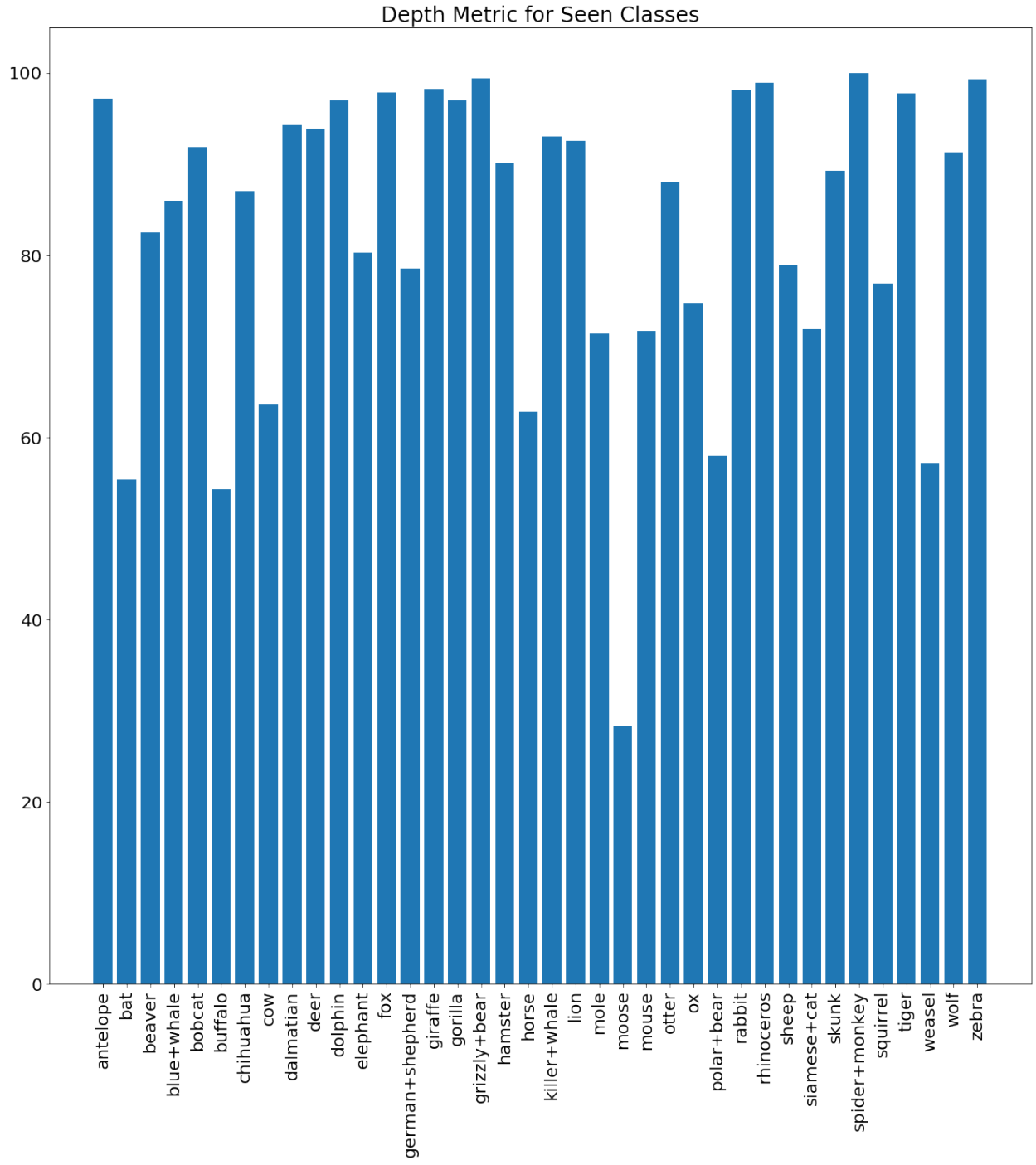

Figure 5.4: **Depth metrics for seen classes.** Depth metric values calculated for Animals With Attributes 2 classes seen at training time.

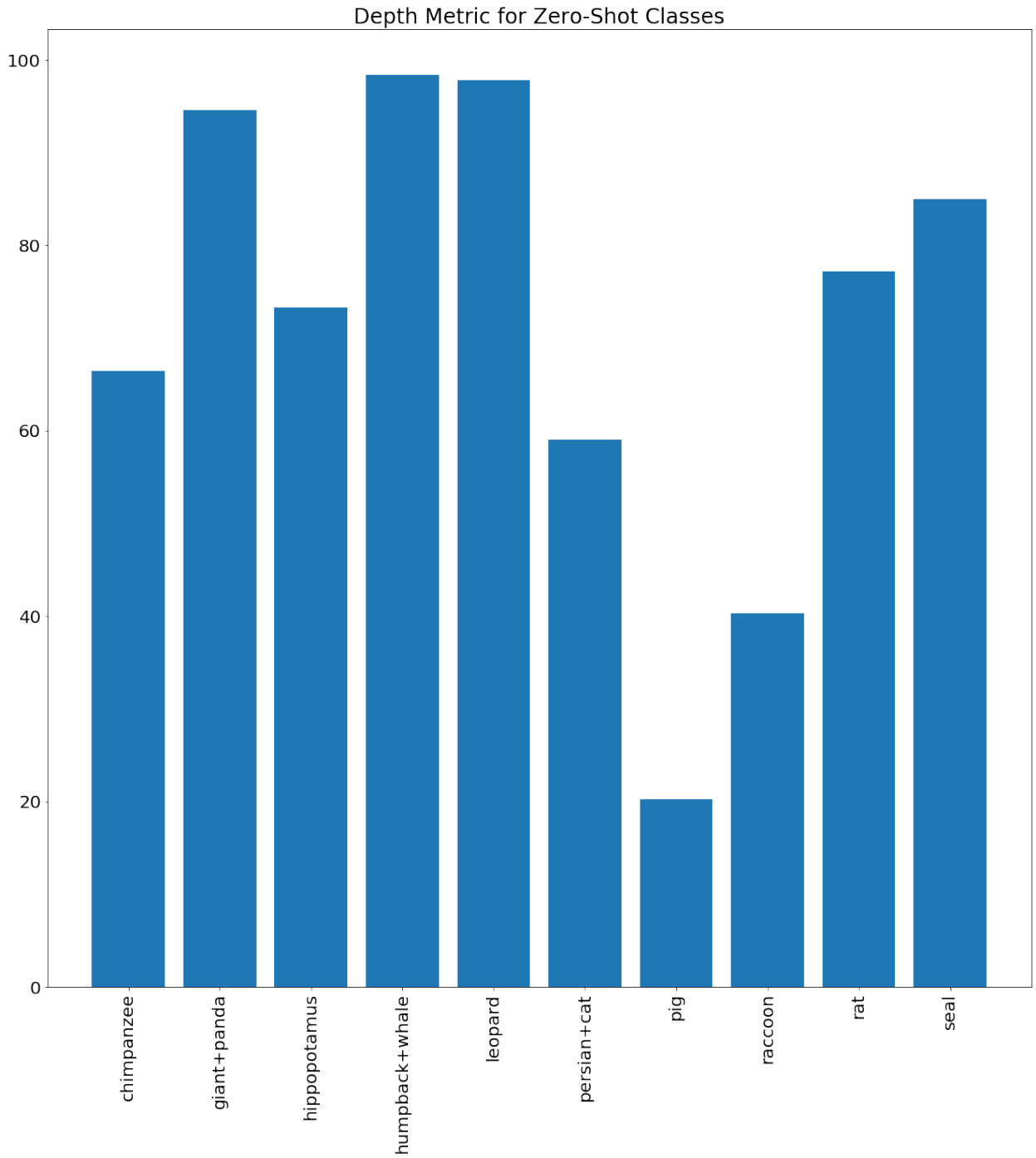

Figure 5.5: **Depth metrics for unseen classes.** Depth metric values calculated for Animals With Attributes 2 classes not seen at training time. The accuracies are similar to those of classes seen during training.

Chapter 6

Conclusion

We propose Neural-Backed Decision Trees, a black-box method that maintains the state-of-the-art accuracy of any deep neural network while taking advantage of the natural explainability of decision trees. NBDTs extract a set of *embedded decision rules* from the final layer of a pretrained backbone network, building an *induced hierarchy* via hierarchical agglomerative clustering of these weights. By fine-tuning the induced hierarchy with the addition of a tree supervision loss, our method (a) achieves competitive accuracy on datasets of varying complexity, and (b) beats existing decision tree-based methods by up to 18%. We demonstrate several methods for analyzing the resulting induced hierarchy, including diagnosing poor class hierarchy relationships and generating a tree visualization.

6.1 Future Work

Modifying tree structure. As of now, the induced hierarchy uses a tree structure, with one leaf corresponding to one class in the dataset. There are numerous ways to increase complexity in the hierarchy structure, potentially boosting model accuracy, without compromising explainability. For instance, we can add multiple paths to a single node, generalizing the hierarchy structure from a tree to a graph. Alternatively, we can insert additional leaf nodes for a single class, so that multiple representations of a single class can be encoded by the tree. For example, in some experiments, our model had difficulty classifying cat images taken in an indoor setting, but had very high accuracy for those taken outside. Including alternative representations for the same class can help the model learn more generalized embedded decision rules that consider a wider range of features.

Few-shot learning extensions. Currently, we are experimenting with various augmentations to boost performance when applying NBDTs to few-shot and zero-shot learning classification tasks. One such addition is applying label smoothing on the training set to discourage the induced hierarchy from overfitting. We observe that classes similar to one other (e.g. Golden Retriever and German Shepherd, two breeds of dogs) are sometimes wildly misclassified due to their close proximity on the induced hierarchy; label smoothing allows

greater flexibility for our model when choosing a path, enabling it to make such confusing decisions correctly. We are also developing new methods for few-shot and zero-shot inference to improve accuracy; for instance, one method makes use of a default "other" class during training to determine potential node insertion positions.

References

[1] Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. *Network of Experts for Large-Scale Image Categorization*. 2016. arXiv: 1604.06119 [cs.CV].

[2] Zeynep Akata et al. "Label-Embedding for Image Classification". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.7 (July 2016), pp. 1425–1438. ISSN: 2160-9292. DOI: 10.1109/tpami.2015.2487986. URL: http://dx.doi.org/10.1109/TPAMI.2015.2487986.

[3] Stephan Alaniz and Zeynep Akata. *Explainable Observer-Classifier for Explainable Binary Decisions*. 2019. arXiv: 1902.01780 [cs.LG].

[4] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. *Deep Convolutional Decision Jungle for Image Classification*. 2017. arXiv: 1706.02003 [cs.CV].

[5] Sarah Adel Bargal et al. *Guided Zoom: Questioning Network Evidence for Fine-grained Classification*. 2018. arXiv: 1812.02626 [cs.CV].

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN: 0262035618.

[7] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[8] Max Jaderberg et al. *Spatial Transformer Networks*. 2015. arXiv: 1506.02025 [cs.CV].

[9] Elyor Kodirov, Tao Xiang, and Shaogang Gong. *Semantic Autoencoder for Zero-Shot Learning*. 2017. arXiv: 1704.08345 [cs.CV].

[10] P. Kontschieder et al. "Deep Neural Decision Forests". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1467–1475.

[11] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: 2009.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[13] Ya Le and Xuan Yang. "Tiny ImageNet Visual Recognition Challenge". In: 2015.

[14] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". English (US). In: *Nature Cell Biology* 521.7553 (May 2015), pp. 436–444. ISSN: 1465-7392. DOI: 10.1038/nature14539.

[15] George A. Miller. "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11 (Nov. 1995), pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748. URL: https://doi.org/10.1145/219717.219748.

[16] Venkatesh N. Murthy et al. "Deep Decision Network for Multi-class Image Classification". In: June 2016, pp. 2240–2248. DOI: 10.1109/CVPR.2016.246.

[17] Minseop Park et al. *MxML: Mixture of Meta-Learners for Few-Shot Classification*. 2019. arXiv: 1904.05658 [cs.LG].

[18] Joshua C. Peterson et al. *Learning Hierarchical Visual Representations in Deep Neural Networks Using Hierarchical Linguistic Labels*. 2018. arXiv: 1805.07647 [cs.CV].

[19] Vitali Petsiuk, Abir Das, and Kate Saenko. *RISE: Randomized Input Sampling for Explanation of Black-box Models*. 2018. arXiv: 1806.07421 [cs.CV].

[20] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016. arXiv: 1602.04938 [cs.LG].

[21] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://dx.doi.org/10.1007/s11263-019-01228-7.

[22] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2013. arXiv: 1312.6034 [cs.CV].

[23] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. arXiv: 1409.1556 [cs.CV].

[24] Krishna Kumar Singh and Yong Jae Lee. *Hide-and-Seek: Forcing a Network to be Meticulous for Weakly-supervised Object and Action Localization*. 2017. arXiv: 1704.04232 [cs.CV].

[25] Richard Socher et al. *Zero-Shot Learning Through Cross-Modal Transfer*. 2013. arXiv: 1301.3666 [cs.CV].

[26] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. *Axiomatic Attribution for Deep Networks*. 2017. arXiv: 1703.01365 [cs.LG].

[27] Ryutaro Tanno et al. *Adaptive Neural Trees*. 2018. arXiv: 1807.06699 [cs.NE].

[28] Alvin Wan et al. *NBDT: Neural-Backed Decision Trees*. 2020. arXiv: 2004.00221 [cs.CV].

[29] Yongqin Xian et al. *Zero-Shot Learning - A Comprehensive Evaluation of the Good, the Bad and the Ugly*. 2017. arXiv: 1707.00600 [cs.CV].

[30] Matthew D Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks*. 2013. arXiv: 1311.2901 [cs.CV].

[31] Jianming Zhang et al. *Top-down Neural Attention by Excitation Backprop*. 2016. arXiv: 1608.00507 [cs.CV].

[32] Quanshi Zhang et al. *Interpreting CNNs via Decision Trees*. 2018. arXiv: 1802.00121 [cs.CV].

[33] Ziming Zhang and Venkatesh Saligrama. *Zero-Shot Learning via Semantic Similarity Embedding*. 2015. arXiv: 1509.04767 [cs.CV].

[34] Bolei Zhou et al. *Learning Deep Features for Discriminative Localization*. 2015. arXiv: 1512.04150 [cs.CV].