NBDT: Neural-Backed Decision Trees

Daniel Ho

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-65
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-65.html

May 26, 2020

NBDT: Neural-Backed Decision Trees

by Daniel Ho

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Joseph E. Gonzalez

Professor Joseph Gonzalez
Research Advisor

May 24, 2020

(Date)

* * * * * * *

Jianfei Chen

Doctor Jianfei Chen
Second Reader

Mar 27, 2020

(Date)

Abstract

Deep learning methods are increasingly being used to run inference on a variety of classification and regression problems, ranging from credit scoring to advertising. However, due to their increasing complexity, these models lack the basic properties for establishing fairness and trust with users – transparency and interpretability. While a majority of recent interpretability work involves post-hoc techniques, there has been little work in ante-hoc or intrinsically interpretable models. Historically, decision trees are the most popular method among intrinsically interpretable models as they naturally break down inference into a sequence of decisions. Recent attempts to combine the interpretability of decision trees with the representation power of neural networks have resulted in models that (1) perform poorly in comparison to state-of-the-art models even on small datasets (e.g. MNIST) and (2) require significant architectural changes. We address these issues by proposing Neural-Backed Decision Trees (NBDTs), which take the features produced by the convolutional layers of an existing neural network and convert the final fully-connected layer into a decision tree. NBDTs achieve competitive accuracy on CIFAR10, CIFAR100, TinyImageNet, and ImageNet, while setting state-of-the-art accuracy for interpretable models on Cityscapes, Pascal-Context, and Look Into Person (LIP). Furthermore, we demonstrate the interpretability of NBDTs by presenting qualitative and quantitative evidence of semantic interpretations for our model decisions.

Acknowledgments

I would like to thank Professor Joseph Gonzalez for being my faculty adviser and giving me my first research opportunity, which ultimately led to all my experiences learning from talented research students in RISE Lab. I would like to thank Doctor Jianfei Chen for providing feedback on my thesis and improving the overall clarity. I would also like to thank all my collaborators on the NBDT work, especially Alvin Wan, who has been a mentor to me and guided me through several research projects.

Finally, I would like to thank my family and friends for supporting me throughout my 5 years at Berkeley. I would not be where I am today without all of your support.

Contents

Chapter 1

Introduction

As deep learning has become the de facto standard in computer vision tasks, an increasing number of fields are applying deep learning methods in order to make use of their predictive power. However, neural networks typically provide no insight or justification on their prediction process primarily due to the nature of their deep hierarchical representations. Thus these deep representations are a double-edged sword; although they are responsible for the representation power of neural networks, they make it almost impossible to trace a model's decision making process and isolate the role of any particular hidden unit. Some fields in particular, such as medicine or finance, require a high level of transparency and accountability in order to reliably justify decisions made by a model. As a result, there is typically a trade-off between representation power and interpretability during model selection.

In addition, experiments have shown that neural networks are susceptible to several weaknesses which may cripple their performance. Adversarial attacks [55, 21, 38, 7, 17], for example, demonstrate how small changes in an input, unnoticeable to the human eye, can drastically change the output of a neural network. This ties into another weakness of neural networks, i.e. if there is bias in a training set, an algorithm will acquire that bias, limiting its representation power. Thus if there was a way to enhance the interpretability of deep learning models, it would be much simpler to correct sources of bias in datasets and ensure algorithmic fairness.

In light of this, the field of Explainable Artificial Intelligence (XAI) grew out of the motivation to bring interpretability to deep learning models. To better understand how deep learning models make decisions, XAI has primarily involved generating saliency heatmaps, which highlight the regions of the image that are most significant in making a correct decision. White-box saliency methods [60, 53, 51, 32, 50, 54, 48] take advantage of the internal workings of a model e.g. feature maps, gradients, etc. In contrast, black-box saliency methods [46, 43] rely on extracting information from a model by making queries. In either case, these saliency methods focus on individual inputs and provide little insight into the general behavior of the model, e.g. over the whole dataset or on out-of-distribution samples. An XAI method that better expresses the model's prior over the whole dataset would thus improve the interpretability of deep learning models.

In this thesis, I will introduce Neural-Backed Decision Trees (NBDTs) which not only utilize the interpretability of decision trees but also maintain the predictive power of state-of-the-art computer vision models. NBDTs are essentially decision trees constructed using the weights of an existing neural network. They are defined by two components: (1) the *induced hierarchy* which defines the structure of the NBDT and (2) the *embedded decision rules* which define how NBDT performs inference. At a high level, my collaborators and I propose the following approach: (1) Compute an *induced hierarchy* derived from the weights of an existing, trained network. (2) Fine-tune the network using a custom *tree supervision loss* to maximize the decision tree accuracy under the induced hierarchy. (3) Define a set of *embedded decision rules* to run any neural network as a decision tree. Once the NBDT is trained, the intermediate nodes can be further analyzed by constructing hypotheses for semantic interpretations. The method will be discussed in more detail in Chapter 3.

NBDTs make no modifications to existing neural network architectures, so any network for image classification or semantic segmentation can easily be used to construct an NBDT. Thus, the decision-making process of any existing neural network can be analyzed using NBDTs to gain insight on the hierarchical structure in the underlying classes. Furthermore, NBDTs achieve competitive classification accuracy on CIFAR10 [30], CIFAR100 [30], Tiny-ImageNet [31], and ImageNet [15], while setting state-of-the-art accuracy for interpretable models on Cityscapes [13], Pascal-Context [40], and Look Into Person (LIP) [20].

The remainder of the thesis is organized as follows: Chapter 2 describes related works, Chapter 3 outlines the proposed method, Chapter 4 discusses experiments to evaluate the method, and Chapter 5 discusses the interpretability analysis of NBDTs.

Chapter 2

Related Works

Prior to the past decade of breakthroughs in deep learning, decision trees played an important role in predictive modeling due to not only their interpretability but also their ability to model arbitrarily complex relationships. Decision trees, though prone to overfitting, are white-box models which break down classification and regression problems into a series of decisions through intermediate splitting nodes, meaning they provide users with a simple mechanism to understand the model's decision-making process. On the other hand, deep neural networks have far surpassed other learning techniques in terms of predictive accuracy and have revolutionized learning in fields like computer vision and natural language processing; however, they often lack interpretability as they typically contain millions of parameters and involve billions of operations during inference, making it difficult to trace the decision-making process within a neural network. In the following sections, I will discuss the ideas that pioneered research in decision trees and random forests, work exploring the intersection of decision trees and neural networks, and existing efforts in explainable artificial intelligence, in particular for computer vision.

2.1 Decision Trees

Historically, the first regression tree-based method was proposed in 1963 by Morgan and Sonquist [39]. Their method, called Automatic Interaction Detector (AID), involves fitting a piecewise constant model by recursively splitting up data at each node. The metric for choosing splits at intermediate splitting nodes is defined as impurity, i.e. $\phi(t) = \sum_{i \in t}(y_i - \bar{y})^2$ for each node t, so splits are chosen such that the sum of the impurities in the two children nodes was minimized. Theta Automatic Interaction Detector (THAID) [36] extended the ideas of AID to classification by choosing splits that maximized the sum of the number of observations in each modal category, i.e. the category with the most observations.

While AID and THAID are restricted to binary splits at each node, Chi-square Automatic Interaction Detector (CHAID) [26], another follow-up work of AID, considers n-ary splits at each node. Ordered variables are split into 10 equally-sized intervals, while categorical variables are split into one child node for each category. Because the criteria for splitting is based on the p-values of the chi-square distribution, the stopping rule for tree generation using CHAID automatically accounts for statistical significance. Initially within the statistics community, there was little interest in the above decision tree methods as some criticized AID for severely overfitting to training data and being unable to distinguish the relative importance of highly correlated variables. In the computer science and engineering communities, however, there was growing interest in the idea of recursive partitioning alongside the development of efficient algorithms for searching splits.

The method that finally sparked interest in decision trees and formed the foundation for modern decision tree methods was Classification and Regression Trees (CART) [10], proposed by Breiman et al. CART uses the same greedy search approach as AID, but instead of using stopping rules, the tree is grown to a large size before being pruned to minimize cross validation error. This alleviates the underfitting and overfitting of earlier methods though at an increased computational cost. In addition, Breiman et al. introduced the idea of using surrogate splits to handle variables with missing data. Surrogate splits are splits on alternate variables that serve as a substitute for the preferred split when the latter contained missing data. With these improvements over methods like AID, CART successfully drew attention towards decision tree-based methods.

Another important work that further popularized the adoption of decision trees is Iterative Dichotomiser 3 (ID3) [44], proposed by Quinlan. The main contribution of ID3 was the use of entropy as the splitting criteria, i.e. at each splitting node, the algorithm selected the attribute that minimizes entropy or maximizes information gain. Similar to previous methods, ID3 suffers from limitations such as not guaranteeing the optimal solution since it uses a greedy approach and is restricted to datasets with categorical variables. Quinlan's follow-up work, C4.5 [45] makes several improvements over ID3 and has become one of the most widely used decision tree methods. Unlike ID3, C4.5 is capable of handling both continuous and discrete attributes and handles training data with missing attribute values by ignoring missing values during entropy calculation. In addition, trees are pruned after creation, though they are still often substantially larger than trees produced by other methods [35]. In any case, empirical evidence shows that C4.5 has strong prediction accuracy and speed among decision tree methods [34].

2.2 Random Forests

Although decision trees offer interpretability and are simple to use, they suffer from overfitting, can be non-robust to small changes in training data, and typically rely on heuristics like greedy search strategies. Consequently, decision trees are unable to generalize well and suffer from limited accuracy. In response to this limitation, Ho [58] was the first to propose a tree-based ensemble learning method for increasing generalization accuracy of decision tree-based classifiers without trading away accuracy on training data by training a random forest of classifiers. Variation among the trees is introduced by projecting the training data into a randomly chosen subspace before fitting each tree or each node, i.e. suppose the input feature is m-dimensional, then there are 2^m possible feature subspaces on which to project the training data. Ho notes that the projection allows trees in different subspaces to generalize their classification in complementary ways, resulting in improved overall classification accuracy. The experiments on MNIST demonstrate that increasing the number of decision trees yields almost monotonic increase in accuracy to overcome the overfitting issue of individual decision trees. Around the same time, Amit and Geman [3] independently proposed searching over a random subset of decisions when searching for the best split, in the context of growing a single tree. They define a set of binary features called queries which are designed to capture prior information about shape invariance and regularity within an image. Multiple trees are grown and each node searches over a random subset of these geometric features to find the best split.

Influenced by the two stochastic methods described above, Breiman was the first to formalize the random forests method and provide both theoretical and empirical results [9]. Breiman's method utilizes bagging and random node optimization to construct random forests, which are defined as a collection of tree structured classifiers whose outputs are aggregated for classification. In terms of bagging, the method forms bootstrap training sets from the original training data and constructs a tree for each bootstrap training set. One benefit of bagging is that it introduces the notion of out-of-bag error. Given a specific sample from the training set, an out-of-bag classifier refers to a classifier whose bootstrap training set does not contain that sample. For each sample in the training set, votes are aggregated for the sample using the out-of-bag classifiers. Then the out-of-bag error is the error rate of out-of-bag classifiers on the training set. This removes the need for cross validation or a separate test set for estimating generalization error and provides a means for measuring variable significance. In terms of random node optimization, Breiman explores two methods to introduce variance in feature selection. One way is to randomly select a set of variables to split on at each node. The other way is to generate multiple linear combinations of the input variables where coefficients are drawn from a uniform distribution. Then the algorithm searches over the linear combinations for the best split.

Breiman's work solidified the value of random forests as an ensemble learning method and directly dealt with the low bias, high variance problem of decision trees. Empirical results in the early random forests papers demonstrate that forests not only greatly improve the performance of decision trees but also performed competitively with methods like boosting and adaptive bagging. However, the drawback of random forests is the fact that they drastically reduce the interpretability of decision trees by aggregating votes from multiple decision trees. Nonetheless, despite their limitations, both decision trees and random forests are still widely used in modern applications from simpler tasks like classification to specialized tasks like human pose estimation, as we will see in the following section.

2.3 Intersection of Decision Trees and Neural Networks

As mentioned previously, deep learning based techniques have consistently pushed the state-of-the-art accuracy in a variety of tasks over the past decade while sacrificing the interpretability of results. Because of the inherent interpretability of decision trees, researchers have explored ways to combine the two methods in hopes of maintaining both accuracy and interpretability.

Decision Trees to Neural Networks

Work in converting from decision trees to neural networks dates back three decades as methods explored the use of decision trees to speed up neural network training. [6, 5, 25] propose initializing the weights of a neural network using a decision tree. [6, 5] notes that while decision tree methods like ID3 and CART run quickly during training, neural network based algorithms like perceptron and error-backpropagation often attain higher predictive accuracy. Therefore, they suggest constructing a decision tree, converting it into a neural network, and fine-tuning it in order to decrease training time. Recent work [24] also proposes initializing the weights of neural networks using decision trees to speed up training. The proposed mapping from decision tree to neural network involves corresponding paths in the tree to connections between neurons in the neural network. Thus, this method not only determines the initialization weights of a neural network but also the structure of the architecture. Unlike the previous works which use decision trees to speed up neural network training, TreeGrad [52] converts decision trees to neural networks in order to train boosted decision trees in an online manner, e.g. to update decision split values and the choice of split candidates. Results on the UCI dataset [16] demonstrate that TreeGrad performs competitively and even outperforms existing greedy tree ensemble algorithms.

Neural Networks to Decision Trees

There has also been work done in the other direction, i.e. converting from neural networks to decision trees, in hopes of developing an understanding of what a neural network is learning. [29] proposes a method for extracting decision trees from a neural network in order to better understand the internal workings of the neural network. This is done by generating a set of prototypes, i.e. a set of inputs which yield a desired classification when passed through a neural network. Once the set of the prototypes is generated, a decision tree is induced using a subset of the prototypes. The prototypes not only reveal information about the neural network but also cause the induced decision tree to be smaller as it is constructed with a smaller training set. [8, 14] propose treating a trained neural network as an oracle and making queries to the network in order to extract a set of rules. Using those rules, the method induces a decision tree that essentially mimics the behavior of the neural network. Though the induced trees do not perform as well as the neural network, they still reveal some insight on how neural networks make decisions. A more recent work [18] similarly samples from the input space and queries the neural network before fitting a decision tree to the output in order to mirror the behavior of the network. A soft decision tree is constructed and trained by stochastic gradient descent using the predictions of the neural network, allowing it to generalize better than a decision tree trained on the training data directly. By using neural networks to improve the performance of more explicable models like decision trees, this method eases the tension between representation power and interpretability.

Decision Trees for Image Classification

Recent work in image classification explores methods in which decision trees and neural networks work jointly to either improve predictive accuracy or model interpretability. Deep Neural Decision Forests (DNDF) [27] unifies classification trees and convolutional networks and trains them in an end-to-end manner by introducing stochastic, differentiable trees which can learn through backpropagation. Input samples are first featurized using a neural network before being routed through decision nodes to the prediction nodes, which hold a distribution over the possible output classes. DNDF achieved state-of-the-art accuracy on ImageNet; however, this was prior to the inception of residual networks which greatly improved accuracy across computer vision tasks. In addition, the main drawback of DNDF is the use of random forests and leaves with distributions over all classes which ultimately diminished the interpretability of the method.

Despite the limited interpretability of DNDF, Li et al. [33] visualize saliency maps along the decision-making process of DNDF to better understand what the network is attending to in the input image. The saliency maps are computed by taking the derivative of the routing probability with respect to the input, where the routing probability of each splitting node represents whether the tree should traverse along the left sub-node. While saliency maps highlight spatial evidence, they fail to provide semantic meaning for the decision made at each intermediate node.

Murthy et al. [41] propose a stage-wise training strategy in which Deep Decision Networks (DDN) are trained to split data into disjoint clusters of classes that are subsequently handled by successive expert networks. That is, at each stage, a network is trained on a given set of data and a confusion matrix is used to identify clusters with samples that are difficult to classify. Then an expert network is fine-tuned to handle those confusion clusters. This results in a tree-like structure in which each node corresponds to a network; consequently, this method reveals insight on the underlying structure of the data, e.g. images and classes that are difficult to classify. Network of Experts (NofE) [1] presents a similar strategy involving expert networks and identifying confusing classes. NofE contains a generalist backbone which is shared across the expert networks and outputs a distribution over the specialties, or disjoint subsets of original classes. The expert network is responsible for discriminating the classes within its specialty. Like the previous work mentioned, this method reveals insight on the data and model in that it partitions the classes into specialties such that confusing classes or classes of the same domain can be clustered together. Then, the expert network can learn highly specialized features to distinguish between the classes.

While DDN and NofE reveal insight on how classes are clustered and which images or classes tend to be difficult for classification, they still treat the neural network as a black-box in that the internal decision-making process of the network is still unclear. In light of this, Zhang et al. [61] propose learning a decision tree, which clarifies the reason for each prediction made by a convolutional neural network at the semantic level. A filter loss is defined to push high-level filters towards representations of object parts, resulting in disentangled filters. Then, a decision tree is constructed to highlight which object parts highlight which filters most strongly for a given prediction. Thus this method not only provides insight on which object parts are contributing to a prediction but also unmasking what high-level filters are learning at a semantic level, aside from the typical edges, colors, gradient orientations, etc, which can be somewhat uninformative in the big picture.

Decision Trees for Segmentation

Prior to deep learning becoming the de facto standard in vision-related tasks, decision trees and random forests were being applied to tasks like human pose estimation and semantic segmentation. Brostow et al. [11] propose using random forests and 3D point clouds derived from ego-motion for accurate segmentation of video frames. Unlike previous methods, this method exploits structure and motion cues in video sequences and works well on sparse, noisy point clouds. A random forest classifier scans across the image to classify each pixel based on features inferred from the point cloud, such as distance from and height above the camera. Shotton et al. [49] similarly uses a sliding window decision forest classifier but in the context of human pose estimation. The decision forest predicts a body part label at each pixel as an intermediate step towards predicting joint positions. Despite using handcrafted features, this method outperformed state-of-the-art methods even without kinematic and temporal constraints and ran several orders of magnitude faster than existing methods.

Similar to DDN and NofE, where splitting nodes correspond to neural networks, recent work proposes using neural networks as split nodes to learn more complex split features and improve performance of decision trees and forests on image segmentation. Neural Decision Forests (NDF) [12] proposes using randomized multi-layer perceptrons (rMLP) as split nodes to learn non-linear, data-specific representations and find optimal splits. rMLPs differ from conventional MLPs, which suffer from overfitting, in that the topology or number of layers in an rMLP is determined by the distribution of labels arriving at a node and the input data to the first layer of the rMLP is randomly selected. Evaluated on ETrims8 [28] and CamVid [11] segmentation datasets, NDF outperforms conventional decision tree classifiers which typically involve hand-crafted data representations. Similarly, Hehn et al. [23] proposes using convolutional neural networks as a split feature extractor since their method learns decision trees end-to-end using a gradient-based optimization method. Using CNNs as splits, a single decision tree learned using their method outperforms standard random forest ensembles whose splits are restricted to axis-aligned or oblique splits.

Other work explores combining decision trees and deep networks in order to improve performance on semantic segmentation in the presence of limited training data. In the context of robotic learning where time and data are limited, Zuo et al. [63] leverage the strength of CNNs and random forests to reduce training time and provide higher performance when training data is limited. The method replaces the softmax layer of a CNN with a decision forest which serves to provide the loss function for learning the network weights through backpropagation. Thus the network acts as a feature extractor while the decision forest acts as the final classifier. Evaluated on KITTI [19] and NYUv2-40 [42] segmentation datasets, this method outperforms pure deep learning methods while using a fraction of typical training time by exploiting the discriminating power of decision forests. Richmond et al. [47] instead propose a mapping from stacked decision forests to CNNs as a form of weight initialization to enable training even with limited amounts of training data. This follows from the assertion that stacked decision forests are a special case of CNNs with sparse convolution kernels. Segmentation results on body part labeling and microscopy images indicate that this method outperforms existing strategies in the face of limited training samples. They further demonstrate a mapping from the newly trained CNN to another stacked decision forest which offers a more computationally efficient inference alternative that performs better than the original stacked decision forest.

While these aforementioned works have tackled increasing accuracy or decreasing runtime, they are inadequate from an interpretability perspective as many of these methods rely on random decision forests and do not propose hypotheses for node meanings. In the following section, we discuss widely used explainable AI methods that attempt to address the black-box nature of neural networks.

2.4 Explainable Artificial Intelligence

Explainable AI (XAI) is a relatively recent field that grew out of the motivation to address the lack of interpretability in state-of-the-art artificial intelligence models. The impact of research on XAI is not limited to providing insight on the behavior of complex models and algorithms; XAI improves the transparency of models by providing explanations to some degree, ensures algorithmic fairness, and allows users to identify potential sources of bias or issues with training data. While there are many ways to categorize XAI methods, two predominant categories are white-box or gradient-based methods and black-box or perturbation-based methods. White-box methods have access to the internals of a system, e.g. the model, feature maps, gradients, etc. On the other hand, black-box methods do not require access to the internals of a system and draw explanations from making queries on the system.

White-Box/Gradient-Based

Early white-box methods explored constructing visualizations of images that maximally activated neurons by projecting activations back to the pixel space or using gradients. Deconvnet [60] presents a way to map activations back to the input pixel space in order to visualize the input pattern that causes a given activation in the feature maps. A deconvnet essentially runs a convnet in reverse by applying unpooling, rectification, and transposed convolution layers repeatedly until the pixel space is reached. Visualizations constructed using deconvnet reveal the hierarchical nature of features in a network, with early layers capturing features like edges and color and later layers capture more complex, class-specific features. The drawback of this method is that its performance relies on the existence of max-pooling layers in the network to obtain the positions of maxima within pooling regions. As a result, deconvnet and its reconstructions are conditioned on the input image and do not directly visualize learned features.

Springenberg et al. [53] addresses this issue by introducing guided backpropagation, in which they mask out gradient values for which the corresponding gradient or input feature value is negative. This prevents the backward flow of gradients and adds an additional guidance signal from higher layers during backpropagation. Using guided backpropagation rather than deconvnet produces sharper, more recognizable image structure for higher layers of a network. Similarly, Simonyan et al. [51] introduces two gradient-based visualizations based on computing gradient of a class score with respect to an image. The first visualization performs optimization on the image while fixing the weights to generate an image that is most representative of a given class according to the network's scoring function. The second visualization computes the gradient of a class score with respect to each pixel in an input image, highlighting pixels according to their influence on the class score.

The following works attempt to address the shortcomings of the early gradient-based methods whose saliency maps can be noisy and contain indistinguishable attention regions. Li et al. [32] proposes Salience Relevance maps, a two-step method, which begins by computing a pixel relevance map using Layer-wise Relevance Propagation (LRP) and then constructs a context-aware saliency map from the LRP-generated map. The second step filters out irrelevant regions of the relevance map to better reveal the true attention areas. DeepLIFT [50], a method similar to LRP, addresses the issue of gradient-based methods in which activation functions like ReLU and sigmoid cause zero gradients even when activations may carry information. Instead of propagating gradients to compute pixel importance, DeepLIFT propagates activation differences where each neuron is assigned a reference activation. Integrated Gradients [54] is an alternative method which first constructs a series of images by scaling down the pixel intensities from the input image to zero or a pure black image. Then, the average of the gradients of the constructed images is computed and multiplied by the input image to construct the final saliency map. The motivation behind this is that at a certain scaling factor, there will be a large jump in prediction score, meaning mainly the pixels of most significance remain visible. Thus, the gradients of that image will dominate the final saliency map.

GradCAM [48] also utilizes gradients of a target class to highlight spatial evidence for classification; however, rather than computing gradients with respect to the input image, gradients are taken with respect to the final convolution layer. The heatmap is computed as follows:

$$\alpha_k^c = \frac{1}{Z}\sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k}$$

$$L_{GradCAM}^c = ReLU(\sum_k \alpha_k^c A^k)$$

where α_k^c represents the weights of each layer in the final convolution layer and are computed as the global average pool of gradients. Compared to CAM [62], GradCAM achieves better localization and clearer class discriminative saliency maps.

Black-Box/Perturbation-Based

In contrast to white-box models, black-box models rely on probing behavior of a model to gain understanding. LIME [46] identifies the regions of an input most influential for a decision by constructing a local linear model that serves as a proxy to the full model in the neighborhood of a given input. The regions of similar pixels are referred to as super-pixels and are masked to observe how the network output changes in response. The disadvantage of this method is that its reliance on locating super-pixels impacts the correctness and consistency of the method, e.g. LIME may face difficulty locating the super-pixel region for an object containing many textures and colors. In addition, though the linear approximation constructed by LIME may perform well locally, it may not faithfully mimic the behavior of a more complex model. In contrast, RISE [43] is another black-box method that generates saliency maps by feeding masked versions of an input image and observing the subsequent changes to the output. The saliency map is a linear combination of the randomly generated masks where the weights correspond to the score of the target class corresponding to the respective masked input. Thus, RISE, like GradCAM, highlights the pixels most significant in classifying an input image correctly, but requires no knowledge of the internals of the model.

As we can see, all the interpretability methods described above generate attribution or saliency maps to highlight important pixels in an image. As a result, they are limited to interpreting 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 a plane). In contrast, neural-backed decision trees express the model's prior over the entire dataset and utilizes the interpretability of decision trees, breaking down classification into a sequence of intermediate decisions.

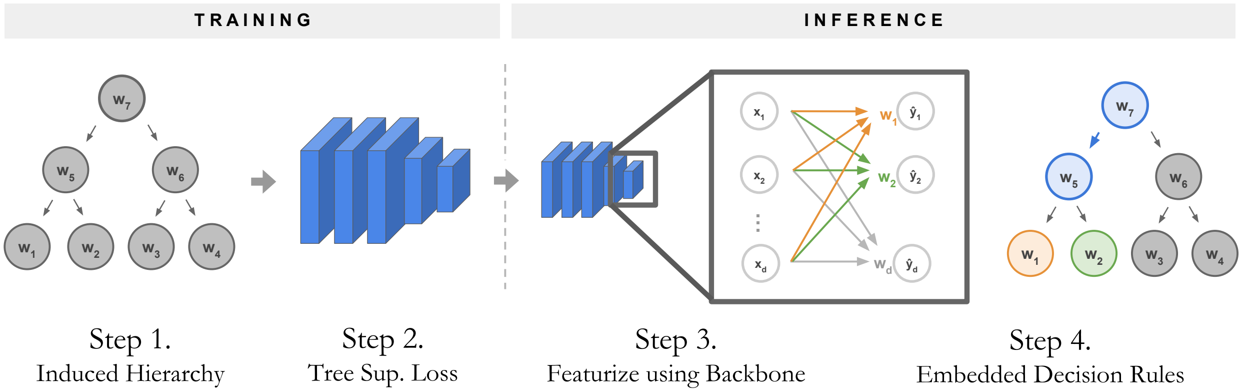
Chapter 3

Methodology

Suppose we have an existing neural network for image classification with standard layers e.g. convolution, batch normalization, and activation layers. This neural network can range from something as simple as ResNet10 to the state-of-the-art EfficientNet. Consider all layers of the network before the final fully-connected layer. We refer to these layers as the backbone of the neural network and use them to featurize input images/samples in some embedded space. Then, the final fully-connected layer is responsible for making a prediction based on the features of the embedded sample. Now, to explain what an NBDT is, we must define it in terms of two components: the structure of the decision tree and how it performs inference. We refer to these as the *induced hierarchy* and *embedded decision rules* respectively. As we are trying to better understand the distribution of embedded features and how the fully-connected layer behaves, we can construct the *induced hierarchy* and *embedded decision rules* using the weights of the final fully-connected layer.

As illustrated in Figure 3.1, the NBDT pipeline consists of four steps divided into a training phase and an inference phase. In **step 1**, we build an induced hierarchy using the weights of a pre-trained network's fully-connected layer (Sec 3.2). In **step 2**, we fine-tune the network with a custom tree supervision loss (Sec. 3.3). In **step 3**, we featurize the sample using the neural network backbone. In **step 4**, we use the embedded decision rules to run inference (Sec. 3.1).

Notice that the architecture of neural network remains unchanged during construction of an NBDT. We simply use the weights of the fully-connected layer to construct the NBDT and update the weights during fine-tuning. Thus, our method supports any existing classification or segmentation neural network architecture, and the accuracy of NBDT scales with the accuracy of the underlying neural network. To reiterate, the interpretability of NBDTs focuses on the final fully-connected layer and embedded feature space rather than the convolutional layers of the neural network. The structure of the NBDT illustrates properties of the distribution of features in the embedded feature space and provides one interpretation of how the neural network features are distributed rather than fully explaining the decision making process of the overall neural network.



Step 1.
Induced Hierarchy

Step 2.
Tree Sup. Loss

Step 3.
Featurize using Backbone

Step 4.
Embedded Decision Rules

Figure 3.1: **Neural-Backed Decision Tree.** The NBDT process consists of a training phase and inference phase. The induced hierarchy is constructed during training, while the embedded decision rules are used to run inference using the NBDT.

Notation

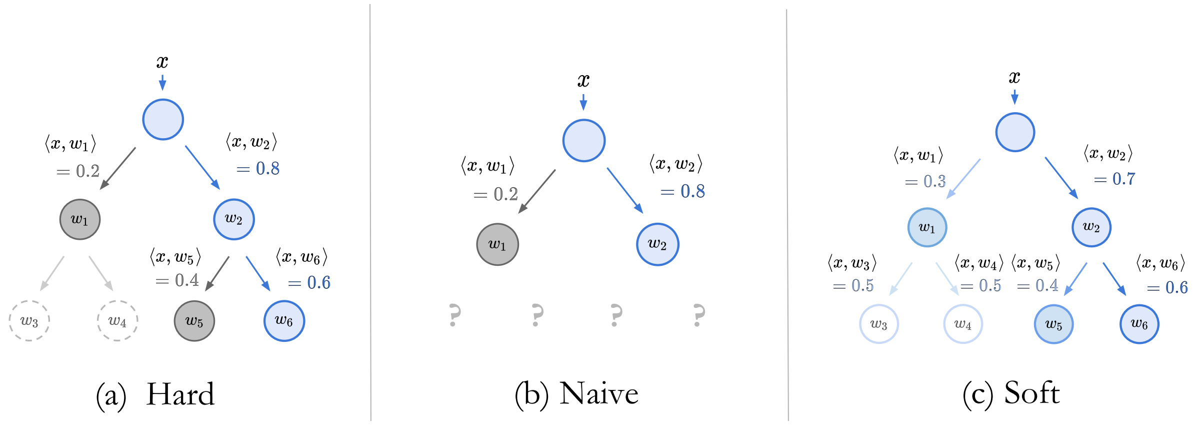
Throughout this chapter, we will use the following notation to explain the methodology. Let x be the featurized sample, i.e. the output of passing an input image through the neural network backbone, while $\hat{y}$ is the predicted class. Let W be the weight matrix of the fully-connected layer, where each row w_i of W corresponds to an output class. In addition, each node in the NBDT will correspond to a representative vector r_i contained in the embedded feature space. The representative vectors are used to run inference using the NBDT.

3.1 Inference with Embedded Decision Rules

Suppose we have an NBDT with some arbitrary structure/hierarchy. We must define how the NBDT performs inference to complete construction of the NBDT. The first step of the inference phase in the NBDT pipeline is to featurize each sample using the neural network backbone, where the backbone consists of all layers in the network before the final fully-connected layer. To traverse the decision tree, at each non-leaf node, we take the inner product between the featurized sample $x \in R^d$ 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. We use these inner-products to make either *hard* or *soft* decisions, described below.

We first motivate the use of inner products by establishing the equivalence between a fully-connected layer and a degenerate decision tree.

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


x
$\langle x, w_1 \rangle$ $= 0.2$
$\langle x, w_2 \rangle$ $= 0.8$
w_1
w_2
$\langle x, w_5 \rangle$ $= 0.4$
$\langle x, w_6 \rangle$ $= 0.6$
w_3
w_4
w_5
w_6
x
$\langle x, w_1 \rangle$ $= 0.2$
$\langle x, w_2 \rangle$ $= 0.8$
w_1
w_2
? ? ? ?
x
$\langle x, w_1 \rangle$ $= 0.3$
$\langle x, w_2 \rangle$ $= 0.7$
w_1
w_2
$\langle x, w_3 \rangle$ $= 0.5$
$\langle x, w_4 \rangle$ $= 0.5$
$\langle x, w_5 \rangle$ $= 0.4$
$\langle x, w_6 \rangle$ $= 0.6$
w_3
w_4
w_5
w_6

(a) Hard

(b) Naive

(c) Soft

Figure 3.2: **Hard and Soft Decision Trees.** Illustrated above are the various ways to run inference on an NBDT.

$$\underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}}_{W} \begin{bmatrix} | \\ x \\ | \end{bmatrix} = \underbrace{\begin{bmatrix} \langle x, w_1 \rangle \\ \langle x, w_2 \rangle \\ \vdots \\ \langle x, w_d \rangle \end{bmatrix}}_{\hat{y}} \implies \text{argmax}(\hat{y}) \tag{3.1}$$

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. 3.2 ((b) Naive).

Although the two computations are represented differently, both make a prediction by taking the index of the largest inner product $\text{argmax}\langle x, w_i \rangle$. Hereafter, we denote decision tree inference as *embedded decision rules.*

Now consider the general case in which the decision tree contains non-leaf intermediate nodes. The embedded decision rules require that each node corresponds to a representative vector r_i. Therefore, we naively consider the non-leaf's representative vector to be the average of all the subtree's leaves' representative vectors. With a more complex tree structure containing intermediate nodes, there are now two ways to run inference:

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. 3.2, (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. 3.2, (c) Soft).

Using embedded decision rules thus breaks down any classification into a sequence of decisions; however, running a neural network as an NBDT as-is yields poor accuracy as the neural network is not optimized to maximize NBDT accuracy. In the following section, we discuss how to build an induced hierarchy which can be used in fine-tuning the neural network to maximize accuracy.

3.2 Building Induced Hierarchies

With the inner-product decision rule defined above, we can now define the structure of the NBDT. Intuitively, some of these hierarchies may be easier for the network to learn. These easier hierarchies may also better reflect how the neural network attains higher accuracy. Thus, we run hierarchical agglomerative clustering on the class representatives $r_i = w_i$ extracted from the fully-connected layer weights W. The motivation for this is as follows: consider the inner product space $V = R^d$ containing the rows w_i of weight matrix $W \in R^{k \times d}$. w_i's corresponding to similar classes like cat and dog or truck and car should lie close to each other in V, so agglomerative clustering should cluster similar classes together. In other words, leaves lying in the same subtree will correspond to classes of similar attributes. Note that because hierarchical agglomerative clustering is a bottom-up approach, it merges pairs of clusters during construction of the hierarchy, resulting in a binary tree structure.

As mentioned in Sec. 3.1, each leaf corresponds to one w_i and each intermediate node's representative vector is the average of the the representative vectors of the leaves in its subtree. We refer to the hierarchy constructed in this manner as the *induced hierarchy* (Fig. 3.3). The steps in building an induced hierarchy are as follows: **Step (a)** Load the weights of a pre-trained neural network's final fully-connected layer, with weight matrix $W \in R^{d \times k}$. **Step (b)** Use each column w_i of W as representative vectors r_i 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 for which it is the root. Set the representative vector of the ancestor to the average

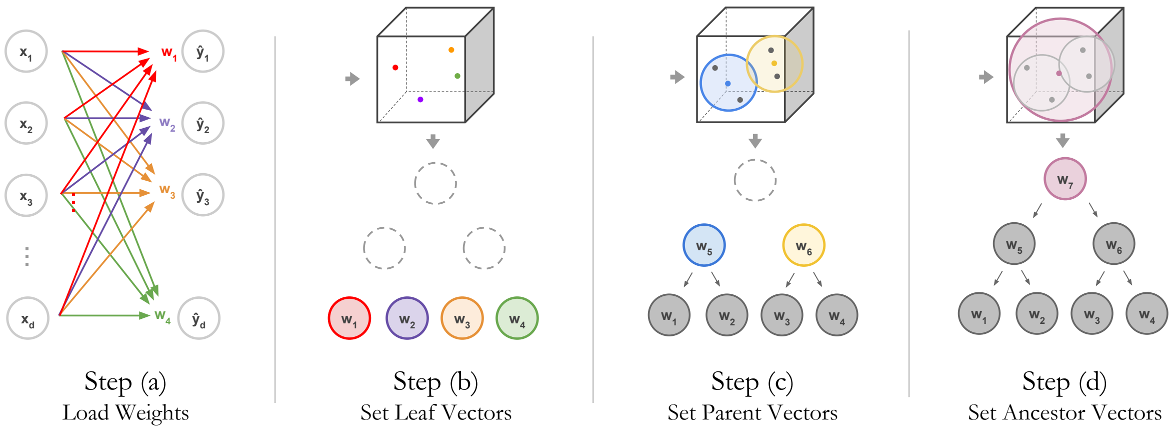


Figure 3.3: **Building Induced Hierarchies.** The induced hierarchy of an NBDT corresponds to the structure of the tree and can be built using the weights of a pre-trained neural network's final fully-connected layer.

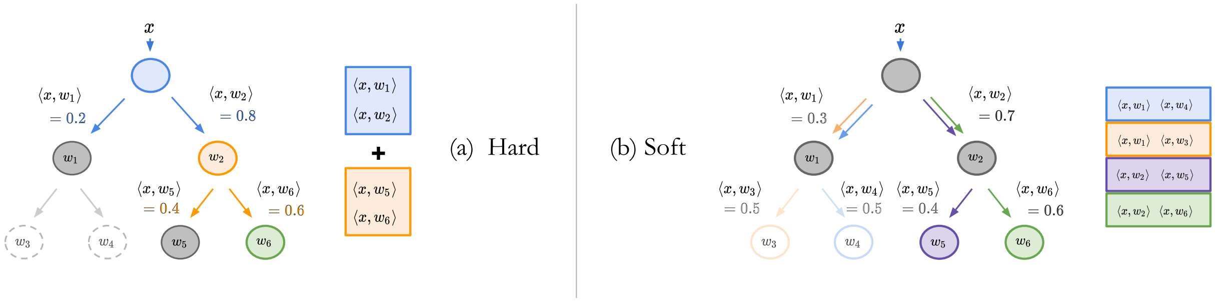
of the representative vectors for all leaves in the subtree. In Fig. 3.3, the ancestor is the root, so its representative vector is the average of all leaves w_1, w_2, w_3, w_4.

We additionally conduct experiments with an alternative WordNet-based hierarchy. WordNet [37] provides an existing 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, 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

In Sec. 3.1, we established the equivalence between a fully-connected layer and a degenerate decision tree with no intermediate nodes. Using the same example, we can deduce that the original neural network has only been trained to separate representative vectors for each class, i.e. training the fully-connected layer to maximize accuracy. Therefore, the network is not trained to separate the representative vectors for internal node. To address this issue, we add loss terms that encourage the neural network to separate representatives for internal nodes, during training.

We propose two variants of tree supervision loss for the hard and soft decision rules in turn (Fig. 3.4). For hard decision rules, we use the *hard tree supervision loss*. The original neural network's loss $\mathcal{L}_{\text{original}}$ minimizes cross entropy across the classes. For a k-class dataset, this is a k-way cross entropy loss. Each internal node's goal is similar: minimize cross-entropy


x
$\langle x, w_1 \rangle = 0.2$
$\langle x, w_2 \rangle = 0.8$
w_1
w_2
$\langle x, w_5 \rangle = 0.4$
$\langle x, w_6 \rangle = 0.6$
w_3
w_4
w_5
w_6
$\langle x, w_1 \rangle$
$\langle x, w_2 \rangle$
+
$\langle x, w_5 \rangle$
$\langle x, w_6 \rangle$
x
$\langle x, w_1 \rangle = 0.3$
$\langle x, w_2 \rangle = 0.7$
w_1
w_2
$\langle x, w_3 \rangle = 0.5$
$\langle x, w_4 \rangle = 0.5$
$\langle x, w_5 \rangle = 0.4$
$\langle x, w_6 \rangle = 0.6$
w_3
w_4
w_5
w_6
$\langle x, w_1 \rangle$ $\langle x, w_4 \rangle$
$\langle x, w_1 \rangle$ $\langle x, w_3 \rangle$
$\langle x, w_2 \rangle$ $\langle x, w_5 \rangle$
$\langle x, w_2 \rangle$ $\langle x, w_6 \rangle$

(a) Hard

(b) Soft

Figure 3.4: **Tree Supervision Loss** has two variants: **Hard Tree Supervision Loss (a)** defines a cross entropy term per node. Because the dotted nodes are not included in the path from the label to the root, they do not have a defined loss. **Soft Tree Supervision Loss (b)** defines a cross entropy loss over all leaf probabilities. 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.

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. 3.2). 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. 4.1. Suppose there are N nodes along the path from the root to the label in the tree, excluding leaves, then there are $N+1$ 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:

$$\mathcal{L}_{\text{hard}} = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\textsc{CrossEntropy}(\mathcal{D}(i)_{\text{pred}}, \mathcal{D}(i)_{\text{label}})}_{\text{over the } c \text{ children for each node}} . \tag{3.2}$$

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:

$$\mathcal{L}_{\text{soft}} = \textsc{CrossEntropy}(\mathcal{D}_{\text{pred}}, \mathcal{D}_{\text{label}}). \tag{3.3}$$

3.4 Extension to Semantic Segmentation

Thus far we have discussed how to construct an NBDT using any image classification neural network. However, we can easily extend our method to an NBDTSeg model, i.e. constructing an NBDT using a semantic segmentation network. The fully connected layer of a classification network serves as the foundation for our method. For a segmentation network, we can instead utilize the final 1×1 convolutional layer, which plays the same role as a fully connected layer by essentially performing classification at each input pixel.

Consider an input of dimension (H, W, C_{in}) and C_{out} $1 \times 1 \times C_{in}$ convolution kernels. The output of the 1×1 convolutional layer can be computed by taking the inner product between each convolution kernel and the vector along the channel dimension at each spatial location of the input. Then the output will have dimension (H, W, C_{out}). The final prediction of the segmentation network will be the argmax of the output feature map along the channel dimension. Thus, for an NBDT for classification, each row in the weight matrix of a fully connected layer corresponds to a class; for an NBDT for segmentation, each convolution kernel in a 1×1 convolutional layer corresponds to a class. From this perspective, NBDTs can be easily applied to both classification and segmentation networks.

Chapter 4

Experiments

4.1 Classification

Experiments on multiple image classification benchmark datasets reveal that NBDTs achieve state-of-the-art accuracy for decision trees, in some cases outperforming other methods by a large margin. We report results across a variety of datasets, models, and inference modes:

1. Datasets: CIFAR10[30], CIFAR100[30], TinyImageNet[31], ImageNet[15]
2. Models: ResNet[22], recently state-of-the-art WideResNet[59], EfficientNet[56]
3. Inference modes: soft *vs.* hard inference

In terms of ablation studies, we first note that tree supervision loss with weight 0.5 consistently improves the accuracy of the original neural network by 0.5% across datasets. In addition, we compare induced hierarchy *vs.* WordNet hierarchy and run hyperparameter sweep on tree supervision loss weight.

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% (Table 4.1). Our baselines are either taken directly from the original papers or improved using a modern backbone: Deep Neural Decision Forest (DNDF updated with ResNet18) [27], Explainable Observer-Classifier (XOC) [2], Deep Convolutional Decision Jungle (DCDJ) [4], Network of Experts (NofE) [1], Deep Decision Network (DDN) [41], and Adaptive Neural Trees (ANT) [57].

Using WideResnet28x10 as the backbone, 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

Table 4.1: **Results.** 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.

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%**	–

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% on TinyImageNet.

For our ImageNet experiments, we use EfficientNet-ES, a smaller variant of the EfficientNet models, which have achieved state-of-the-art accuracy 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).

Ablation Studies

Tree Supervision Loss. As described in Sec. 3.3, the overall loss term is $\mathcal{L}_{\text{original}} + \mathcal{L}_{\text{hard}}$, when weighting the original cross entropy loss and hard tree supervision loss equally. In these set of experiments (Table 4.2), we use tree supervision loss weight 0.5, i.e. overall loss is $\mathcal{L}_{\text{original}} + 0.5 * \mathcal{L}_{\text{hard}}$. Training the networks from scratch on CIFAR100 and TinyImageNet using this loss improves accuracy of the original network by 0.5%.

Table 4.2: **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 4.3: **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%

WordNet Hierarchy. Table 4.3 indicates that WordNet and induced hierarchies perform similarly on smaller datasets like CIFAR10 and CIFAR100; however, on TinyImageNet, the induced hierarchy outperforms the WordNet hierarchy by 4.17% because WordNet similarity does not translate directly to visual similarity. Consider the following example: according to WordNet, *Bird* is closer to *Cat* than to *Plane* as they are both animals. However, images of *Bird* are more visually similar to *Plane* than *Cat* as they tend to share contextual information such as the sky.

Tree Supervision Loss Weight. Sweeping over the tree supervision loss weight, we find that increasing the weight up to one order of magnitude encourages the network to improve NBDT accuracy; however, disproportionately assigning weight to the tree supervision loss by two orders of magnitude significantly degrades performance of both the neural network and the NBDT. Thus, our method is robust to imbalance between the two loss terms up to an order of magnitude (Table 4.4).

4.2 Semantic Segmentation

1. Datasets: Cityscapes [13], Pascal-Context [40], LookIntoPerson [20]

2. Models: HRNet-w18-v1, HRNet-w48

Table 4.4: **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%

3. Inference modes: soft *vs.* hard inference

Results

The NBDTSeg model attains state-of-the-art for decision-tree-based models on three segmentation benchmarks, achieving accuracy within 4% of the base neural network on Pascal Context [40], Cityscapes [13], and LookIntoPerson [20] in Table 4.5. This demonstrates the extensibility of neural-backed decision trees to not only high-dimensional inputs like images but also high-dimensional, dense predictions like segmentation.

Table 4.5: **NBDTSeg Results** For each dataset, we use a state-of-the-art segmentation network HRNetv2 and construct a corresponding NBDTSeg model.

Dataset	NBDTSeg-S (Ours)	NBDTSeg-H (Ours)	HRNetV2 Size	NN Acc	Δ
Pascal-Context	49.12%	–	W48	52.54%	3.42%
Cityscapes	67.53%	67.33%	W18-Small	70.30%	2.77%
Cityscapes	79.01%	–	W48	81.12%	2.11%
Look Into Person	51.64%	–	W48	55.37%	3.73%

Chapter 5

Interpretability Analysis

As mentioned previously, decision trees are inherently interpretable as they break down classification and regression problems into a series of intermediate decisions, which can be independently analyzed. When the input features are easily understood (e.g. medicine/finance), analyzing the rules for splitting intermediate splitting nodes is a relatievly straightforward task; however, in the case of NBDTs, the input features are the embedded featurization of images through a complex neural network.

5.1 Interpretability of Nodes' Semantic Meanings

With the WordNet hierarchy, hypotheses for nodes' meanings can be automatically generated using the WordNet taxonomy. For induced hierarchies, however, hypotheses must be manually proposed because the hierarchy is constructed using model weights. In either case, to verify hypotheses, we follow a four-step procedure for deducing nodes' meanings:

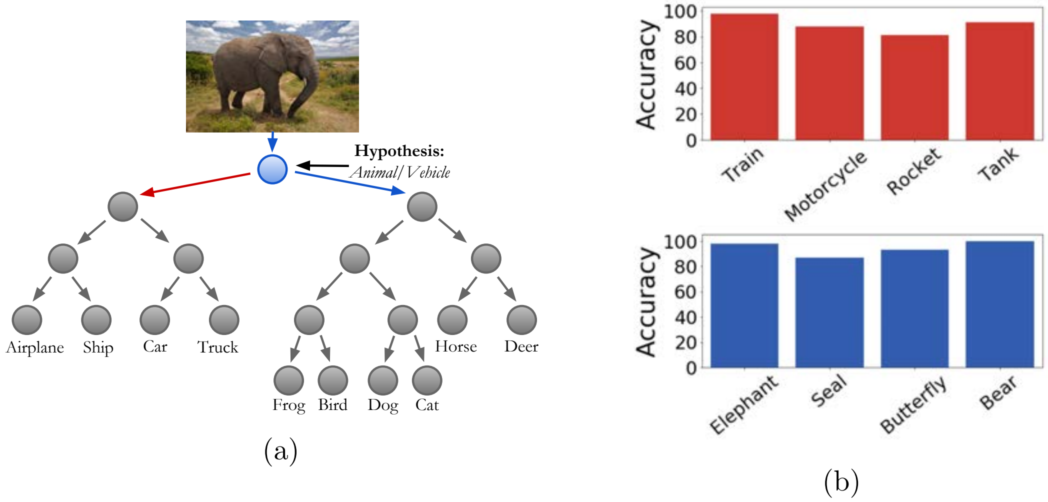
1. Make a hypothesis for the node's meaning (e.g. *Animal vs. Vehicle*). This hypothesis can be computed automatically from a given taxonomy like WordNet or deduced from manual inspection of leaves for each child (Fig. 5.1).

2. Collect a dataset with new, unseen classes that test the hypothesis from step 1 (e.g. *Elephant* is an unseen *Animal*). Samples in this dataset are referred to as out-of-distribution samples.

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.

Fig. 5.1a depicts the CIFAR10 tree induced by a WideResNet28x10 model trained on CIFAR10. Consider the hypothesis in which the root node splits on *Animal vs. Vehicle*.



(a)

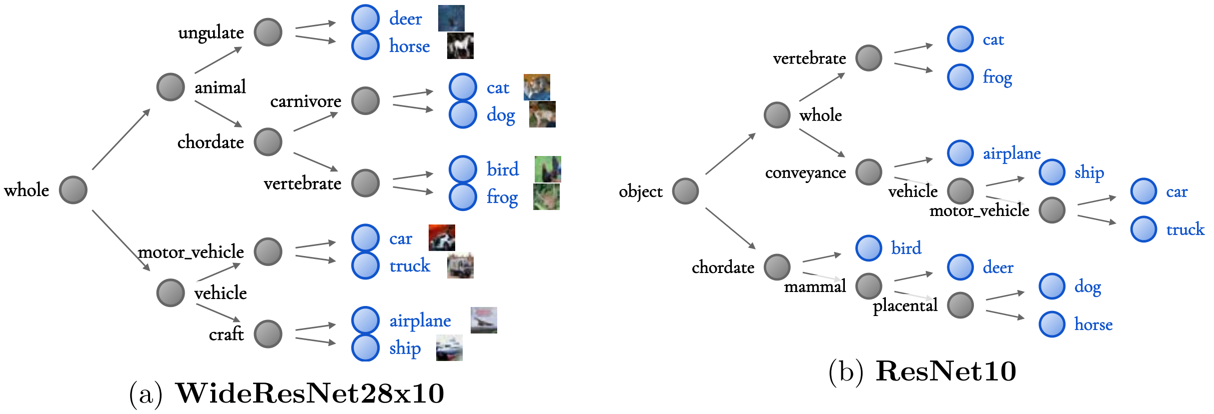
(b)

Figure 5.1: **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).

Using CIFAR100, we can collect out-of-distribution images for *Animal* and *Vehicle* classes that are unseen at training time. Fig. 5.1b validates our initial hypothesis as samples of unseen vehicle classes primarily traverse the left child, while samples of unseen animal classes traverse the right child.

5.2 Sidestepping the Accuracy-Interpretability Tradeoff

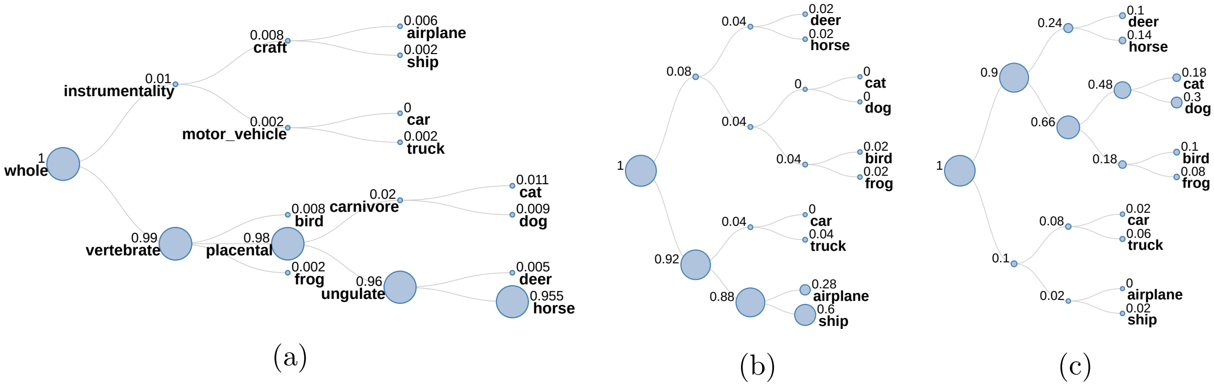
While the induced hierarchies are constructed by clustering representative vectors in the weight space, vectors that are close in weight space do not necessarily indicate that they are semantically similar. Fig. 5.2 depicts the CIFAR10 induced hierarchies for WideResNet28x10 and ResNet10 respectively. The WideResNet hierarchy (Fig. 5.2a) splits between animals and vehicles at the root node, while the ResNet hierarchy (Fig. 5.2b) groups vehicles and vertebrates like *Cat* and *Frog*. Thus the WideResNet hierarchy does a better job in grouping semantically-similar classes together. We attribute this disparity to the higher accuracy of WideResNet models, i.e. higher-accuracy models exhibit more semantically-sound weight spaces. Thus, NBDTs do not require sacrificing accuracy of the model for interpretability. Instead, using higher accuracy models improves the interpretability of NBDTs.



(a) **WideResNet28x10**

(b) **ResNet10**

Figure 5.2: CIFAR10 induced hierarchies for (a) WideResNet (97.62% acc) and (b) ResNet (93.64% acc), with automatically-generated WordNet hypotheses for node meanings.



(a)

(b)

(c)

Figure 5.3: Visualization of path traversal frequency for three different classes. (a) **In-Distribution Class:** ***Horse*** uses samples of a class found in training. (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

In addition to understanding the meaning of individual nodes, we can draw meaning from tree traversals by visualizing the percentage of samples that pass each node (Fig. 5.3). This reveals information about the most frequently traversed path and common incorrect paths (Fig. 5.3a). We can investigate context and attributes that are shared between leaves such as *Backgrounds*, *Scenes*, *Color*, or *Shape*. Fig. 5.3b depicts the paths of *Seashore* samples. Notice that a majority of the samples end up at the *Ship* leaf, while few samples are classified as animals. Fig. 5.3c depicts the paths of out-of-distribution *Teddy* samples. Notice that the samples tend towards the animal classes as they share similar shape and visual features.

Chapter 6

Conclusion and Future Work

In this thesis, I discuss Neural-Backed Decision Trees, a method that forgoes the dichotomy between interpretability and representation power by combining decision trees and deep neural networks to create a high-accuracy, interpretable model. For image classification, NBDTs narrow the accuracy gap between neural networks and decision trees to 1% on CIFAR10, CIFAR100, TinyImageNet and to 2% on ImageNet, while advancing state-of-the-art for interpretable methods by ∼14% on ImageNet to 75.30% top-1 accuracy. For semantic segmentation, NBDTs match neural network accuracy to within 4% on Cityscapes, Pascal-Context, and Look Into Person (LIP). We show that any classification or segmentation network can be used to construct an NBDT by (1) creating a decision tree like structure called an *induced hierarchy*, (2) fine-tuning the network using a *tree supervision loss*, and (3) running inference using *embedded decision rules.* To draw meaning from intermediate nodes in an NBDT, we follow a four step procedure to validate our hypotheses qualitatively and quantitatively.

In terms of future work, one way of further strengthening the interpretability of NBDTs is to visually ground the meaning of intermediate nodes, i.e. currently the meaning derived from nodes in trained NBDTs are not associated with semantic parts of the image. In particular, this would improve interpretability for dense prediction tasks like semantic segmentation as there is currently little work in XAI in terms of producing context explanations for semantic segmentation. Alvin Wan and I are currently exploring this direction by using a modified version of GradCAM to highlight what each node is looking at visually in an image. Associating each decision rule with a visual correspondence in the image can reveal insight into important object parts or context used to perform segmentation, e.g. windows, wheels, and tail lights of vehicles.

In addition, currently the interpretability of NBDTs are limited to the embedded feature space of the neural network backbone as we rely on using the weights of the final fully-connected layer to construct an NBDT. Consequently, the overall network still remains largely a black-box model and little insight is gained with respect to the overall decision-making process. Extending NBDT to handle intermediate convolutional layers would remove this restriction and reveal more about the internal workings of neural networks.

Bibliography

[1] Karim Ahmed, Mohammadharis Baig, and Lorenzo Torresani. “Network of Experts for Large-Scale Image Categorization”. In: vol. 9911. Apr. 2016.

[2] Stephan Alaniz and Zeynep Akata. “XOC: Explainable Observer-Classifier for Explainable Binary Decisions”. In: *CoRR* abs/1902.01780 (2019).

[3] Y. Amit and D. Geman. “Shape Quantization and Recognition with Randomized Trees”. In: *Neural Computation* 9.7 (1997), pp. 1545–1588.

[4] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. “Deep Convolutional Decision Jungle for Image Classification”. In: *CoRR* abs/1706.02003 (2017).

[5] Arunava Banerjee. “Initializing Neural Networks using Decision Trees”. In: *Proceedings of the International Workshop on Computational Learning and Natural Learning Systems*. MIT Press, 1994, pp. 3–15.

[6] Arunava Banerjee. “Initializing neural networks using decision trees”. In: (1990).

[7] Avishek Joey Bose and Parham Aarabi. *Adversarial Attacks on Face Detectors using Neural Net based Constrained Optimization*. 2018. arXiv: `1805.12302 [cs.CV]`.

[8] Olcay Boz. “Converting A Trained Neural Network To a Decision Tree DecText - Decision Tree Extractor”. In: *ICMLA*. 2000.

[9] Leo Breiman. “Random Forests”. In: *Mach. Learn.* 45.1 (Oct. 2001), pp. 5–32. ISSN: 0885-6125. DOI: `10.1023/A:1010933404324`. URL: `https://doi.org/10.1023/A:1010933404324`.

[10] Leo Breiman et al. “Classification and Regression Trees”. In: 1983.

[11] Gabriel Brostow et al. “Segmentation and Recognition Using Structure from Motion Point Clouds”. In: vol. 5302. Oct. 2008, pp. 44–57. DOI: `10.1007/978-3-540-88682-2_5`.

[12] S. Bulò and P. Kontschieder. “Neural Decision Forests for Semantic Image Labelling”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 81–88.

[13] Marius Cordts et al. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. 2016. arXiv: `1604.01685 [cs.CV]`.

[14] Darren Dancey, David McLean, and Zuhair Bandar. "Decision tree extraction from trained neural networks". In: (Jan. 2004).

[15] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[16] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: `http://archive.ics.uci.edu/ml`.

[17] Kevin Eykholt et al. *Robust Physical-World Attacks on Deep Learning Models*. 2017. arXiv: `1707.08945 [cs.CR]`.

[18] Nicholas Frosst and Geoffrey E. Hinton. "Distilling a Neural Network Into a Soft Decision Tree". In: *CoRR* abs/1711.09784 (2017).

[19] Andreas Geiger et al. "Vision meets Robotics: The KITTI Dataset". In: *International Journal of Robotics Research (IJRR)* (2013).

[20] Ke Gong et al. *Look into Person: Self-supervised Structure-sensitive Learning and A New Benchmark for Human Parsing*. 2017. arXiv: `1703.05446 [cs.CV]`.

[21] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2014. arXiv: `1412.6572 [stat.ML]`.

[22] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[23] Thomas Hehn, Julian Kooij, and Fred Hamprecht. "End-to-End Learning of Decision Trees and Forests". In: *International Journal of Computer Vision* (Oct. 2019). DOI: `10.1007/s11263-019-01237-6`.

[24] Kelli Humbird, Luc Peterson, and Ryan McClarren. "Deep Neural Network Initialization With Decision Trees". In: *IEEE Transactions on Neural Networks and Learning Systems* PP (Oct. 2018), pp. 1–10.

[25] Irena Ivanova and Miroslav Kubat. "Initialization of neural networks by means of decision trees". In: *Knowledge-Based Systems* 8.6 (1995). Knowledge-based neural networks, pp. 333–344.

[26] G. Kaas. "An exploratory technique for investigating large quantities of categorical data". In: *Appl Stat 1980,* (Jan. 1980).

[27] Peter Kontschieder et al. "Deep Neural Decision Forests". In: *The IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[28] F. Korč and W. Förstner. *eTRIMS Image Database for Interpreting Images of Man-Made Scenes*. Tech. rep. TR-IGG-P-2009-01. Apr. 2009. URL: `http://www.ipb.uni-bonn.de/projects/etrims_db/`.

[29] R. Krishnan, G. Sivakumar, and P. Bhattacharya. "Extracting decision trees from trained neural networks". In: *Pattern Recognition* 32.12 (1999), pp. 1999–2009.

[30] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

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

[32] Heyi Li et al. "Beyond saliency: Understanding convolutional neural networks from saliency prediction on layer-wise relevance propagation". In: *Image and Vision Computing* 83-84 (Mar. 2019), pp. 70–86. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2019.02.005. URL: http://dx.doi.org/10.1016/j.imavis.2019.02.005.

[33] Shichao Li and Kwang-Ting Cheng. *Visualizing the decision-making process in deep neural decision forest*. 2019. arXiv: 1904.09201 [cs.CV].

[34] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. "A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms". In: *Mach. Learn.* 40.3 (Sept. 2000), pp. 203–228. ISSN: 0885-6125. DOI: 10.1023/A:1007608224229. URL: https://doi.org/10.1023/A:1007608224229.

[35] Wei-Yin Loh. "Improving the precision of classification trees". In: *The Annals of Applied Statistics* 3.4 (Dec. 2009), pp. 1710–1737. ISSN: 1932-6157. DOI: 10.1214/09-aoas260. URL: http://dx.doi.org/10.1214/09-AOAS260.

[36] Robert Messenger and Lewis Mandell. "A Modal Search Technique for Predictive Nominal Scale Multivariate Analysis". In: *Journal of the American Statistical Association* 67.340 (1972), pp. 768–772. DOI: 10.1080/01621459.1972.10481290. eprint: https://doi.org/10.1080/01621459.1972.10481290. URL: https://doi.org/10.1080/01621459.1972.10481290.

[37] 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.

[38] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. *DeepFool: a simple and accurate method to fool deep neural networks*. 2015. arXiv: 1511.04599 [cs.LG].

[39] James Nelson Morgan and John A. Sonquist. "Problems in the Analysis of Survey Data, and a Proposal". In: 1963.

[40] Roozbeh Mottaghi et al. "The Role of Context for Object Detection and Semantic Segmentation in the Wild". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[41] Venkatesh N. Murthy et al. "Deep Decision Network for Multi-Class Image Classification". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[42] Pushmeet Kohli Nathan Silberman Derek Hoiem and Rob Fergus. "Indoor Segmentation and Support Inference from RGBD Images". In: *ECCV*. 2012.

[43] Vitali Petsiuk, Abir Das, and Kate Saenko. "RISE: Randomized Input Sampling for Explanation of Black-box Models". In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2018.

[44] J. R. Quinlan. "Induction of Decision Trees". In: *MACH. LEARN* 1 (1986), pp. 81–106.

[45] J. Ross Quinlan. *C4.5: programs for machine learning.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN: 1-55860-238-0. URL: `http://portal.acm.org/citation.cfm?id=152181`.

[46] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016.* 2016, pp. 1135–1144.

[47] David L. Richmond et al. "Mapping Stacked Decision Forests to Deep and Sparse Convolutional Neural Networks for Semantic Segmentation". In: 2015.

[48] Ramprasaath R Selvaraju et al. "Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2017, pp. 618–626.

[49] J. Shotton et al. "Efficient Human Pose Estimation from Single Depth Images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.12 (2013), pp. 2821–2840.

[50] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. *Learning Important Features Through Propagating Activation Differences.* 2017. arXiv: `1704.02685 [cs.CV]`.

[51] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps". In: *arXiv preprint arXiv:1312.6034* (2013).

[52] Chapman Siu. "Transferring Tree Ensembles to Neural Networks". In: *Neural Information Processing.* 2019, pp. 471–480.

[53] Jost Tobias Springenberg et al. "Striving for Simplicity: The All Convolutional Net". In: *CoRR* abs/1412.6806 (2014).

[54] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks". In: *International Conference on Machine Learning (ICML) 2017* (2017).

[55] Christian Szegedy et al. *Intriguing properties of neural networks.* 2013. arXiv: `1312.6199 [cs.CV]`.

[56] Mingxing Tan and Quoc V Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *arXiv preprint arXiv:1905.11946* (2019).

[57] Ryutaro Tanno et al. *Adaptive Neural Trees.* 2019.

[58] Tin Kam Ho. "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition.* Vol. 1. 1995, 278–282 vol.1.

[59] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks". In: *arXiv preprint arXiv:1605.07146* (2016).

[60] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 818–833.

[61] Quanshi Zhang et al. "Interpreting CNNs via Decision Trees". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[62] Bolei Zhou et al. "Learning Deep Features for Discriminative Localization". In: *Computer Vision and Pattern Recognition*. 2016.

[63] Yan Zuo and Tom Drummond. "Fast Residual Forests: Rapid Ensemble Learning for Semantic Segmentation". In: *Proceedings of the 1st Annual Conference on Robot Learning*. Ed. by Sergey Levine, Vincent Vanhoucke, and Ken Goldberg. Vol. 78. Proceedings of Machine Learning Research. PMLR, 13–15 Nov 2017, pp. 27–36. URL: `http://proceedings.mlr.press/v78/zuo17a.html`.