NBDT: Neural-Backed Decision Trees

Radek Bartyzal

GLAMI AI

5. 5. 2020

Motivation

Interpretability of models:

- decision trees = good
- neural nets = bad

Saliency maps:

- tells you what the nets is "looking" at
- good for debugging = is the net focusing on the right object?
- does not help if the net looks at the right object but predicts wrong class

Overview

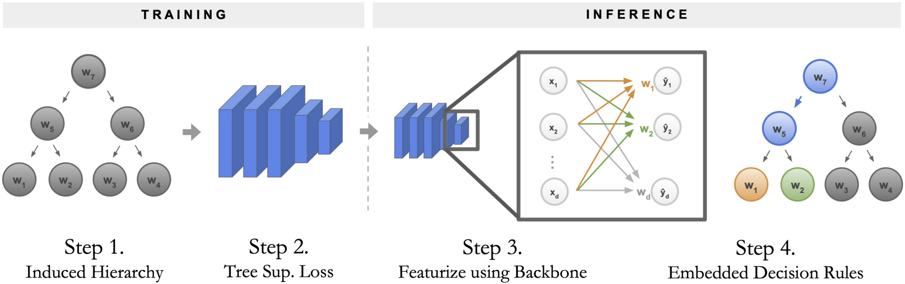
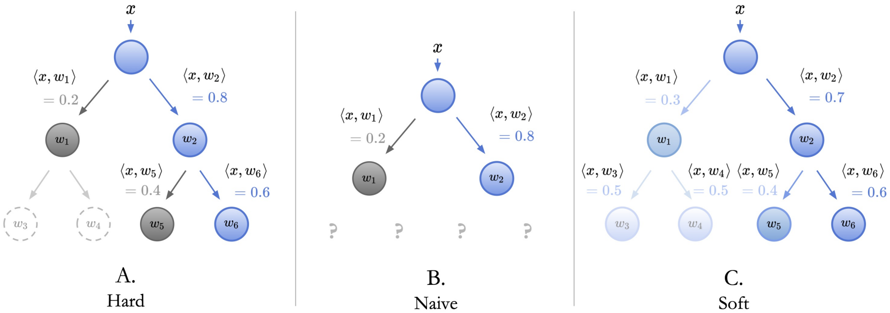


Figure:

Prediction

- get embedding of the sample $= x =$ output of the pre-last layer
- take the last weight matrix W of the net $=$ producing the prediction probabilities
- each column w_i corresponds to one output $=$ class
- each class $c =$ leaf node $c \implies w_c = r_c =$ representation vector of node c
- probability of node $n = < x, r_n > =$ for leaf node $c = < x, w_c >$
- representation vector of inner node $=$ average of repr. vectors of child nodes

Soft vs Hard prediction


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
A.
Hard
x
$\langle x, w_1 \rangle$ $= 0.2$
$\langle x, w_2 \rangle$ $= 0.8$
w_1
w_2
? ? ? ?
B.
Naive
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
C.
Soft

- Hard: select argmax of nodes at current level and continue only with the winners children nodes
- Soft: softmax of nodes at each level =¿ calculate probability of path from the leaf to the root and select the leaf with highest path probability
- Naive: does not support multiple levels

Training

- pre-train the model on the dataset
- construct the nodes by hierarchical clustering done from the w_i columns of the final weight matrix = get hierarchy of nodes = decision tree
- fine-tune the model on the fixed hierarchy to cluster the w_i of the parent nodes together better

Inducing = building hierarchy of nodes

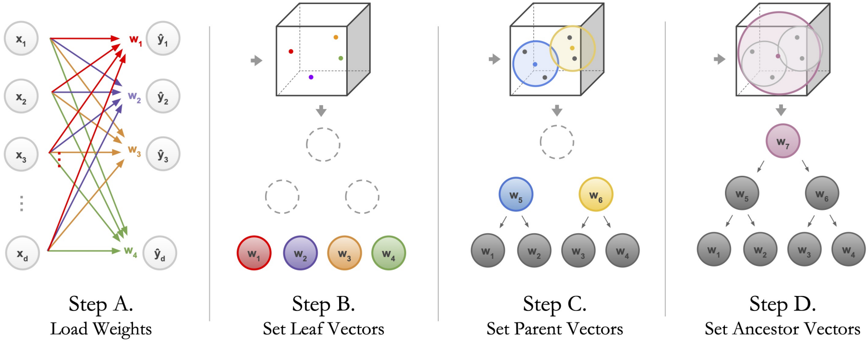


Figure: **A.** Load the weights of pre-trained neural network's final fully-connected layer = W. **B.** Use each column w_i of W as representative vectors for each leaf node. **C.** Use the average of each pair of leaves for the parents' representative vectors. **D.** For each ancestor, take the subtree it is the root for. Average representative vectors for all leaves in the subtree. That average is the ancestor's representative vector.

Fine-tuning is not essential

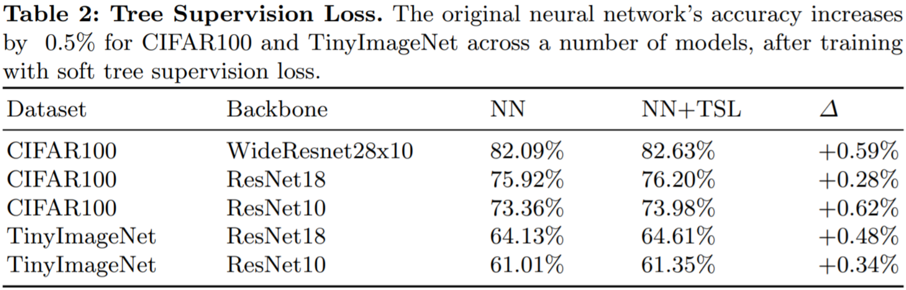
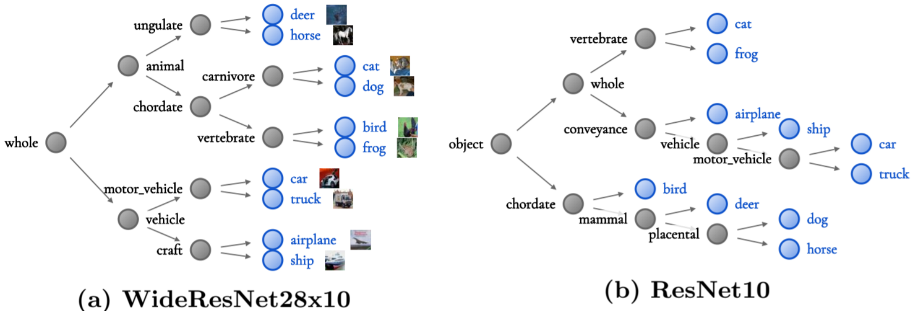


Table 2: Tree Supervision Loss. The original neural network's accuracy increases by 0.5% for CIFAR100 and TinyImageNet across a number of models, after training with soft tree supervision loss.

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

Interpretability



(a) WideResNet28x10

(b) ResNet10

Figure: The ResNet10 hierarchy makes less sense than the WideResNet hierarchy. In this hierarchy, Cat, Frog, and Airplane are placed under the same subtree. The WideResNet hierarchy cleanly splits Animals and Vehicles, on each side of the hierarchy.

Conclusion

- hierarchical clustering of the columns w_i of the last weight matrix W
- check if the clustering makes intuitive sense = animals / vehicles in different subtrees
- visualize tree traversal paths = find the most frequently traversed incorrect paths $\implies$ e.g. see dependency on the background: both fish and ships have sea

Sources

1. Wan, Alvin, et al. "NBDT: Neural-Backed Decision Trees." arXiv preprint arXiv:2004.00221 (2020).
https://arxiv.org/abs/2004.00221

2. Blog post with links to code.
https://bair.berkeley.edu/blog/2020/04/23/decisions/