Journal Title
XX(X):1–20
©The Author(s) 2016
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Neurosymbolic models based on hybrids of convolutional neural networks and decision trees

Rasul Kairgeldin and Miguel Á. Carreira-Perpiñán

Abstract
Building on previous work, we propose a specific form of neurosymbolic model consisting of the composition of convolutional neural network layers with a sparse oblique classification tree (having hyperplane splits using few features). This can be seen as a neural feature extraction that finds a more suitable representation of the input space followed by a form of rule-based reasoning to arrive at a decision that can be explained. We show how to control the sparsity across the different decision nodes of the tree and its effect on the explanations produced. We demonstrate this on image classification tasks and show, among other things, that relatively small subsets of neurons are entirely responsible for the classification into specific classes, and that the neurons' receptive fields focus on areas of the image that provide best discrimination.

Keywords
Class file, $\LaTeX\,2_\varepsilon$, *SAGE Publications*

Introduction

Neural AI systems and symbolic AI systems have both developed extensively since the mid-twentieth century and achieved impressive successes in different domains. Symbolic AI systems, which include many different models (rule-based, logic-based, etc.) are typically characterized by their transparent nature, in that a human can follow their chain of reasoning, at least to some extent. This brings control and trust in the results and often

Dept. of Computer Science & Engineering, University of California, Merced

some form of correctness guarantees (which is critical in, say, program verification or theorem proving). Neural AI systems, whose most powerful form at present are deep neural nets (NNs), have excelled at perceptual tasks, such as recognizing patterns in images or generating realistic text. This is due to their ability to learn from large labeled datasets by optimizing an objective function that measures the prediction error over the NN parameters. This, in turn, is made possible by the fact that NNs define differentiable functions, whose gradient can be used in effective optimization algorithms such as SGD. This also makes it possible to combine multiple NNs and train them jointly (end-to-end), thanks to the chain rule of derivatives. On the other hand, neural AI systems typically require large computational resources in terms of hardware, energy, and training and test time and memory. Also, their reasoning is impenetrable to humans: even though the structure and parameters can be inspected, their size and complexity is such that they behave like a black box. This makes it hard to understand why they make mistakes and how to correct them, among other things.

Thus, there has long existed an interest in neurosymbolic systems (Hitzler and Sarker 2021; Kautz 2022; d'Avila Garcez and Lamb 2023), which seek to combine the best of both worlds. In particular, there have been previous attempts to combine NNs and trees (see section). Here, building on the recent work of Hada et al. (2024) (HCZ24 for short), we will focus on a hybrid model consisting of specific forms of neural and symbolic AI systems: convolutional NNs (although other NNs could also be used) and decision trees, respectively. The trees can be turned into a set of decision rules if so desired. The hybrid can be seen as having the CNN learn some complex features that represent the input (say, an image) in a way that is more suitable for classification; while the classification is effected by the decision tree, whose structure and logic are interpretable by a human (to some extent, depending on the size of the tree). For example, one can follow the tree reasoning by tracing the path followed by the input instance from the root to a leaf, solve counterfactual explanations exactly (Carreira-Perpiñán and Hada 2021), etc.

Following HCZ24, their procedure starts from a trained CNN and replaces a part or module $\mathbf{M}$ of it with a decision tree that aims at being *approximately functionally equivalent (a.f.e.)*[*]. This is achieved by training the tree in a teacher-student way on a dataset with the inputs that $\mathbf{M}$ receives labeled with the corresponding outputs that $\mathbf{M}$ produces. For example, in a LeNet or VGG CNN, $\mathbf{M}$ could be all the fully-connected layers that follow the last convolutional layer (fig. 1). If the predictions of the tree are identical or very close to those of $\mathbf{M}$ on the training and test set, we deem it to be a.f.e. to $\mathbf{M}$. Then, by replacing $\mathbf{M}$ with the tree, we define a hybrid, neurosymbolic model that is a.f.e. to the original CNN. However, the symbolic module (the tree or set of rules) brings some amount of explainability into the hybrid model, which (since the tree represents $\mathbf{M}$ well) transfers to the original CNN. For example, among other things that HCZ24 explored is the fact that relatively small subsets of neurons are associated with specific

[*]By this we do not mean the tree represents exactly the same mathematical function as $\mathbf{M}$. This much stricter requirement would be overkill in practice because real data occupy a tiny, low-dimensional region of the input space. An estimate of this region is given (indirectly) by the observed data we have for training and test.

classes. This makes it possible to manipulate the CNN with surgical precision to force it to make certain classification decisions. The hybrid model can also replace the original CNN for actual prediction but with much faster inference.

Key to the success of this approach is the ability of the tree to match the predictive performance of the module $\mathbf{M}$, while remaining sufficiently interpretable. This may not always be possible, depending on the complexity of $\mathbf{M}$, but recent advances in decision tree learning have greatly enlarged the space of tree-type models we can train and improved the accuracy they can achieve and their scalability to large, high-dimensional datasets. In particular, we will use *sparse oblique trees* for classification (Carreira-Perpiñán and Tavallali 2018), which predict a constant label at each leaf, but use hyperplane splits with few features at each decision node. Such trees can be effectively trained with the Tree Alternating Optimization (TAO) algorithm (Carreira-Perpiñán and Tavallali 2018). Indeed, as shown in HCZ24's and our experiments, the resulting tree is very small (often having just one leaf per class), yet it produces an a.f.e. hybrid to the original CNN. This would not be possible with the traditional CART-style decision tree induction algorithms, which use a much more limited tree type (axis-aligned, using a single feature per split) and a much more suboptimal training algorithm (greedy recursive partitioning).

In our paper, a first contribution is to make this specific type of model (which would be a "type-3 or Neuro|Symbolic" system in the classification of Kautz (2022)) known to the neurosymbolic learning community. A second contribution is that we modify the TAO algorithm so that it learns sparse oblique trees with a controllable sparsity distribution over the nodes, thus increasing their interpretability while remaining highly accurate. A third contribution is that we take the interpretability of the hybrid model beyond what HCZ24 did, by showing the receptive fields of neurons that are responsible for the discrimination between specific classes. We display this as a density map that objectively shows where in the image those neurons are looking. For example, in the Fashion MNIST dataset we show how certain neurons act on specific image areas to detect the presence of specific object parts that are critical to tell one class from another (e.g. to tell a shoe from a bag, a critical part is the existence of a gap in the shoe front which is occupied by a corner in the bag, and a certain neuron detects precisely that). Also, using this knowledge, we are able to edit an image that the CNN misclassifies so it classifies it correctly.

Related work

We focus on work involving tree- or rule-based models and neural nets (NNs). Early on, the black-box nature of NNs was recognized and there were attempts to replace the entire, trained NN with a symbolic representation, specifically a decision tree or a set of rules, which provided an explainable system. Several approaches of this type were actively researched in the 1990s and 2000s (reviewed in (Andrews et al. 1995; Duch et al. 2004; Jacobsson 2005; McCormick et al. 2013; Guidotti et al. 2018)). One approach (Towell and Shavlik 1993; Fu 1994; Setiono and Liu 1996; Baesens et al. 2003; Duch et al. 2004) relied only on access to the architecture and weight values of a multilayer perceptron (MLP), although the neuron activations were assumed to be binary.

Rules were extracted using some form of heuristic search. The other is a teacher-student approach, then called “pedagogical”, which needs a training set (actual or synthetic) on which a decision tree or a set of rules is trained to mimic the input-output behavior of the MLP (Craven and Shavlik 1994, 1996; Domingos 1998). Although the experiments in these papers were somewhat successful, they were limited to tiny two-layer MLPs and never scaled up to larger NNs (having more units and layers). This was due to the use of explainable models of limited power (such as axis-aligned trees) and/or to the heuristic nature of the procedure (a heuristic search or a suboptimal training with greedy recursive partitioning algorithms such as CART (Breiman et al. 1984) or C4.5 (Quinlan 1993)). In contrast, the work of Hada et al. (2024) did scale up to larger NNs (LeNet, VGG), without any assumption on the type of activations or NN architecture other than the ability of the tree to match the accuracy of the NN module replaced. This was achieved by using a more powerful type of trees (oblique) and a better optimization (TAO), and also by aiming to replace a part of a NN rather than all of it

Unlike CART and other greedy recursive partitioning algorithms, TAO (Carreira-Perpiñán and Tavallali 2018) optimizes a well-defined objective function jointly over the parameters of an entire tree whose structure is fixed. This has two advantages: one can train trees that are smaller yet much more accurate than those of traditional algorithms (Zharmagambetov et al. 2021b); and the types of tree-based models one can learn is greatly enlarged. For example, one can train trees of general form in the decision nodes (such as axis-aligned, bivariate, sparse oblique or others) and in the leaves (such as constant, linear, softmax and others) (Kairgeldin and Carreira-Perpiñán 2024; Zharmagambetov et al. 2021a). In terms of the objective function, one can use different types of losses suitable for different tasks, such as classification, regression, clustering, dimensionality reduction, semisupervised learning or imbalanced classification (Zharmagambetov and Carreira-Perpiñán 2020; Gabidolla and Carreira-Perpiñán 2022a; Carreira-Perpiñán and Gazizov 2024; Zharmagambetov and Carreira-Perpiñán 2022; Gabidolla et al. 2024), as well as to use various regularization terms (e.g. to promote sparsity in the decision node weight vectors). Finally, one can also apply TAO to train decision forests (Carreira-Perpiñán and Zharmagambetov 2020; Gabidolla and Carreira-Perpiñán 2022b; Carreira-Perpiñán et al. 2023), which are more accurate than state-of-the-art packages such as XGBoost (Chen and Guestrin 2016) or LightGBM (Ke et al. 2017) while using fewer, shallower trees.

Another related line of work is based on soft decision trees (SDTs) (Jordan and Jacobs 1994), which use a sigmoid instead of step function at each decision node. Thus, an input instance traverses all root-leaf paths rather than a single one as in a hard tree, and the SDT output is a weighted average of all the leaves’ labels. This defines a differentiable mapping which can be optimized via gradient-based methods, possibly end-to-end together with other modules (Kontschieder et al. 2015; Good et al. 2023; Hazimeh et al. 2020; Ibrahim et al. 2024; Borisov et al. 2024). Indeed, a SDT is much closer to (a specific type of) NNs than to trees. However, the fact than the input instance follows all paths, thus touching all parameters, makes the SDT a black box. For the same reason, training and inference are also much slower than with a hard tree. One can turn

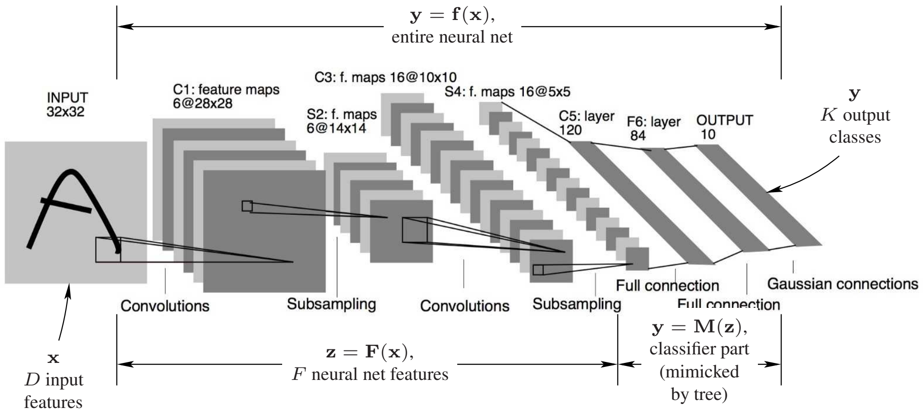

Figure 1. (Adapted from Hada et al. (2024).) A neurosymbolic model as a CNN-tree hybrid. The "neural feature" vector $\mathbf{z}$ consists of the activations (outputs) of the F neurons in the last convolutional layer. The fully-connected MLP layers $\mathbf{M}$ are replaced with a sparse oblique classification tree.

a SDT into a hard tree by replacing each sigmoid with a step function, but this degrades the accuracy considerably and is worse than training a hard tree directly.

The neural net / tree hybrid model: definition and training

We use the same basic model proposed in HCZ24 and illustrated in fig. 1. The starting point is a convolutional neural net (CNN), previously trained using a dataset (training and test) of labeled images (or other type of data) for a classification task. We regard the CNN as having the form $\mathbf{y} = \mathbf{M}(\mathbf{F}(\mathbf{x}))$, where $\mathbf{x}$ is the pixel image, $\mathbf{y}$ the predicted label (or label distribution), $\mathbf{F}$ the convolutional layers and $\mathbf{M}$ the fully-connected layers (i.e., an MLP). The partition of the original NN could be done in other ways, but this particular one makes sense in that the output of $\mathbf{F}$ (the output of the last convolutional layer, having F neurons) can be seen as having learned a feature representation[†] or embedding, while $\mathbf{M}$ acts as a classifier on them.

Next, the fully-connected layers $\mathbf{M}$ are replaced by another classifier, namely a sparse oblique decision tree $\mathbf{y} = \mathbf{T}(\mathbf{z})$. This is done in a teacher-student way, by constructing a new dataset[‡] having a pair $(\mathbf{z}_n, \mathbf{y}'_n)$ for every original data pair $(\mathbf{x}_n, \mathbf{y}_n)$, where $\mathbf{z}_n = \mathbf{F}(\mathbf{x}_n)$ (the output of the last convolutional layer) and $\mathbf{y}'_n = \mathbf{M}(\mathbf{z}_n)$ (the output of the CNN). If we can train a tree $\mathbf{T}$ such that the error $\|\mathbf{y}'_n - \mathbf{T}(\mathbf{z}_n)\|$ is very small

[†]One could also use non-adaptive features such as SIFT, constructed via a fixed formula, and interpretable by design.

[‡]If, instead of approximating the original NN, we seek the best possible hybrid model, we would use $\mathbf{y}'_n = \mathbf{y}_n$ instead (i.e., minimize the error on the ground-truth labels). In practice with NNs, especially if overparameterized, this makes little difference because the original NN usually has a very small error on the training set, so $\mathbf{y}'_n = \mathbf{y}_n$ for most training points.

over the training and test data, then $\mathbf{T}$ and $\mathbf{T} \circ \mathbf{F}$ are a.f.e. to $\mathbf{M}$ and $\mathbf{M} \circ \mathbf{F}$, respectively. In HCZ24's and our experiments this is the case. The reason is that sparse oblique trees trained with TAO are quite powerful classifiers.

Finally, we obtain our hybrid CNN-tree model $\mathbf{y} = \mathbf{T}(\mathbf{F}(\mathbf{x}))$, which can be used in place of the original CNN. Besides providing faster inference, the tree makes it possible to understand the workings of the CNN features and we explore this in our experiments.

At present, we do not have a way to train the hybrid model end-to-end (i.e., $\mathbf{T}$ and $\mathbf{F}$ jointly). This because the tree defines a piecewise constant function, so its gradient is zero nearly everywhere in parameter space and the chain rule cannot be used to update $\mathbf{F}$. However, this is not a limitation if our goal is to understand the meaning and effect on classification of the original CNN features $\mathbf{F}$.

The Tree Alternating Optimization (TAO) algorithm: review

Tree Alternating Optimization provides a unified framework for effectively training complex decision tree-based models. We will discuss it in the setting of training one oblique decision tree. Consider a training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N} \subset \mathbb{R}^D \times \{1, \dots, K\}$ with N samples, D-dimensional features, and K classes. We define an oblique decision tree $\mathbf{T}(\mathbf{x}; \mathbf{\Theta})$ as a rooted binary tree with decision nodes $\mathcal{D}$ and leaf nodes $\mathcal{L}$. Each decision node $i \in \mathcal{D}$ employs a linear decision function $g_i(\mathbf{x}; \boldsymbol{\theta}_i)$ to route an instance $\mathbf{x}$ to the left (if $\mathbf{w}_i^T\mathbf{x} + w_{i0} \geq 0$) or right child (otherwise), where $\boldsymbol{\theta}_i = \{\mathbf{w}_i, w_{i0}\}$ are learnable parameters. Note how the decision function makes hard decisions, unlike in soft trees, where an instance $\mathbf{x}$ is propagated to both children with a positive probability. Each leaf $j \in \mathcal{L}$ contains a constant label classifier that outputs a single class $c_j \in \{1, \dots, K\}$. We collectively define the parameters of all nodes as $\mathbf{\Theta} = \{(\mathbf{w}_i, w_{i0})\}_{i \in \mathcal{N}_{\text{dec}}} \cup \{c_j\}_{j \in \mathcal{N}_{\text{leaf}}}$. The predictive function of the whole tree $\mathbf{T}(\mathbf{x}; \mathbf{\Theta})$ then works by routing an instance $\mathbf{x}$ to exactly one leaf through a root-leaf path of (oblique) decision nodes and applying that leaf's predictor function.

Given a fixed-structure oblique decision tree $\mathbf{T}(\mathbf{x}; \mathbf{\Theta})$ (e.g., a complete tree of depth Δ or one from CART) with random initial parameters, TAO aims to minimize the following objective:

$$E(\mathbf{\Theta}) = \sum_{n=1}^{N} L(\mathbf{y}_n, \mathbf{T}(\mathbf{x}_n; \mathbf{\Theta})) + \lambda \sum_{i \in \mathcal{D}} \phi(\mathbf{w}_i) \tag{1}$$

where $L(\cdot, \cdot)$ is the loss (cross-entropy, MSE or $0-1$) and $\phi(\mathbf{w}_i)$ is a regularization term over the weight vectors. Typically, ℓ_1 is used to promote sparsity via hyperparameters $\lambda \geq 0$.

The TAO algorithm relies on two key theorems: *separability condition* and *reduced problem over nodes*. The *separability condition* ensures that the objective function separates over non-descendant nodes (e.g., all nodes at a given depth), allowing for independent and parallel optimization over parameters of each node.

Theorem 0.1. Separability condition. *Let $i, j \in \mathcal{N}$ be two nodes such that neither is a descendant of the other. Denote by $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ the parameters associated with nodes*

i and j, respectively. All remaining node parameters are collected in $\boldsymbol{\Theta} \setminus \{\boldsymbol{\theta}_i, \boldsymbol{\theta}_j\}$, and are assumed to be fixed. Then the function $E(\boldsymbol{\Theta})$ of eq. (1) can be equivalently rewritten as

$$E(\boldsymbol{\Theta}) = E_i(\boldsymbol{\theta}_i) + E_j(\boldsymbol{\theta}_j) + E_{rest}(\boldsymbol{\Theta}_{rest}) \tag{2}$$

here E_i depends on the parameters of node i together with those of other nodes, but not on node j. Likewise, E_j is independent of node j, and E_{rest} does not involve $\boldsymbol{\theta}_i$ or $\boldsymbol{\theta}_j$.

Proof. Since each training instance x_n follows a unique path from the root to a leaf, node i operates only on its local subset $\{(x_n, y_n) : n \in \mathcal{R}_i\}$. If nodes i and j are not descendants of one another, then $\mathcal{R}_i \cap \mathcal{R}_j = \varnothing$.

Because the global loss $L(\Theta)$ is additive over data points, it separates as:

$$L(\Theta) = \sum_{n\in\mathcal{R}_i} L(y_n, \mathbf{T}(x_n; \Theta)) \; + \sum_{n\in\mathcal{R}_j} L(y_n, \mathbf{T}(x_n; \Theta)) + \text{constant} \tag{3}$$

and the constant term aggregates contributions from all $n \notin (\mathcal{R}_i \cup \mathcal{R}_j)$.

By construction,the first term depends on θ_i (and possibly other nodes) but not on θ_j, and symmetrically for the second term. The constant term is independent of both. Thus, the separability condition holds.

The key implication of the separability condition is that the reduced sets and weights associated with non-descendant nodes are disjoint. This ensures that the parameters of any two such nodes can be optimized independently, based only on their respective reduced sets.

Theorem 0.2. Reduced problem over a node. *Consider the objective function $E(\boldsymbol{\Theta})$ from eq. (1) and a particular node $i \in \{\mathcal{D} \cup \mathcal{L}\}$. When the parameters of all other nodes are held fixed, the global optimization problem decomposes into a well-defined subproblem that depends only on the training instances that reach node (reduced set $\mathcal{R}_i \subset \{1, \dots, N\}$). We define the following reduced problems, which capture the parameter updates local to each node. The exact form of the reduced problem differs for leaves and for decision nodes:*

- *For a decision node $i \in \mathcal{D}$, the top-level problem of eq. (1) reduces to a* weighted 0/1 loss binary classification problem*:*

$$E_i(\mathbf{w}_i, w_{i0}) = \sum_{n\in\mathcal{R}_i} \overline{L}(\overline{y}_n, g_i(\mathbf{x}_n; \mathbf{w}_i, w_{i0})) + \lambda \left\|\mathbf{w}_i\right\|_1 \tag{4}$$

Here, $\overline{y}_n \in \{\texttt{left}_i, \texttt{right}_i\}$ is a pseudolabel indicating the optimal child for $\mathbf{x}_n$, minimizing the subtree loss. The weighted 0/1 loss $\overline{L}(\cdot,\cdot)$ is defined by the loss difference between the chosen and alternative child. While optimizing an oblique node is generally NP-hard, it can be effectively approximated using a surrogate loss like cross-entropy (i.e., logistic regression). The top-level objective (1) is guaranteed to decrease by accepting updates only if they improve (4), though this is often unnecessary in practice.

- *For leaf node $j \in \mathcal{L}$, the top-level problem of eq.* (1) *reduces to a form involving the original loss but only over the parameters of the leaf predictor function. It can be solved by finding the majority class (or mean value of the samples in the reduced set for regression)*

While these theorems do not prescribe the order in which the nodes should be optimized, we follow a reverse breadth-first search order: all the nodes at a given depth are optimized in parallel, starting from the deepest ones until the root. Each optimization subproblem involves solving either an ℓ_1-regularized logistic regression or an finding a majority class. By ensuring that the (approximate) solution of the reduced problem of a decision node improves upon the previous node parameter values, TAO is guaranteed to decrease the objective function (1) monotonically.

Finally, node pruning occurs automatically because the ℓ_1 penalty can drive a node's entire weight vector to zero. This makes the node redundant (it sends all instances to the same child) and it can be removed at the end.

Finer sparsity control with a modified TAO algorithm

The hyperparameter λ controls the overall sparsity in the tree, and by making it large enough we also achieve pruning (and thus a form of tree structure learning) automatically. However, it also has the effect that shallow nodes (e.g. the root) are much denser than deeper nodes (e.g. the leaf parents). This is seen in the trained trees and its cause is explained below. We address this here with a second hyperparameter α that controls *how sparse individual nodes are* in relation to the number of instances they handle.

Consider the following objective function, equal to (1) but with a modified regularization term (whose motivation is explained later) with hyperparameter $\alpha \in \mathbb{R}$:

$$
\begin{aligned}
E(\boldsymbol{\Theta}) &= \sum_{n=1}^{N} L(\mathbf{y}_n, \mathbf{T}(\mathbf{x}_n; \boldsymbol{\Theta})) + \lambda \sum_{i \in \mathcal{D}} h_\alpha(|\mathcal{R}_i|)\, \|\mathbf{w}_i\|_1 \\
h_\alpha(t) &= \begin{cases} 1, & t = 0 \\ t^\alpha, & t > 0 \end{cases}
\end{aligned}
\tag{5}
$$

where $\mathcal{R}_i$ is the RS of node i and $|\mathcal{R}_i|$ its cardinality. This seems difficult to optimize: the h_α term is a non-differentiable function of the tree parameters (specifically, the weight vectors of the decision nodes in the path ascending from i to the root), because it depends on $|\mathcal{R}_i|$ (an integer), which depends on the said parameters. However, the TAO theorems still apply, and the only change is in the RP over a decision node i, which now takes the following form:

$$
E_i(\mathbf{w}_i, w_{i0}) = \sum_{n \in \mathcal{R}_i} L(\mathbf{y}_n, \mathbf{T}(\mathbf{x}_n; \boldsymbol{\Theta})) + \lambda\, h_\alpha(|\mathcal{R}_i|)\, \|\mathbf{w}_i\|_1. \tag{6}
$$

However, $\mathcal{R}_i$ is constant if the nodes ascending from i are kept fixed (as they are in each TAO iteration). Thus, the RP is exactly as in TAO but with a reweighted hyperparameter "$\lambda\, h_\alpha(|\mathcal{R}_i|)$".

```
input training set; initial tree $\mathbf{T}(\cdot;\boldsymbol{\Theta})$ of depth $\Delta$
$\mathcal{N}_0,\dots,\mathcal{N}_\Delta \leftarrow$ nodes at depth $0,\dots,\Delta$, respectively
generate $\mathcal{R}_1 \leftarrow \{1,\dots,N\}$ using initial tree
repeat
  for $d = \Delta$ down to $0$
    parfor $i \in \mathcal{N}_d$
      if $i$ is a leaf then
        $\boldsymbol{\theta}_i \leftarrow$ fit a leaf predictor
          $\mathbf{g}_i$ on reduced set $\mathcal{R}_i$
      else
        generate pseudolabels $\overline{y}_n$ for each point $n \in \mathcal{R}_i$
        $\boldsymbol{\theta}_i \leftarrow$ minimizer of the reduced problem:
        $\sum_{n\in\mathcal{R}_i} \overline{L}(\overline{y}_n, g_i(\mathbf{x}_n;\boldsymbol{\theta}_i)) + \lambda\, h_\alpha(|\mathcal{R}_i|)\, \|\mathbf{w}_i\|_1$
  update $\mathcal{R}_i$ for each node
until stop
prune dead subtrees of $\mathbf{T}$
return $\mathbf{T}$
```

Figure 2. Pseudocode for the tree alternating optimization (TAO) algorithm, modified to handle our new sparsity regularization term with hyperparameter $\alpha \in \mathbb{R}$. Visiting each node in reverse breadth-first search (BFS) order means scanning depths from depth(T) down to 1, and at each depth processing (in parallel, if so desired) all nodes at that depth. "stop" occurs when either the objective function decreases less than a set tolerance or the number of iterations reaches a set limit.

The reason for this new algorithm can be seen by dividing[§] the RP objective function by the constant $N_i = |\mathcal{R}_i|$ (the number of points in i's RS) and rewriting it as "avg-loss + λ' reg", where we define avg-loss $= \frac{1}{N_i}\sum_{n\in\mathcal{R}_i} L(\mathbf{y}_n, \mathbf{T}(\mathbf{x}_n;\boldsymbol{\Theta}))$ (the loss per instance in node i), reg $= \|\mathbf{w}_i\|_1$, and $\lambda' = \lambda N_i^{\alpha-1}$ (an *effective sparsity* hyperparameter). Thus, avg-loss is the loss per instance in node i and λ' is an an effective sparsity hyperparameter. This makes it clear that $\alpha < 1$ (e.g. $\alpha = 0$ as in regular TAO) penalizes larger RSs *less* than smaller ones (thus the root weight vector is denser); $\alpha > 1$ penalizes larger RSs *more* than smaller ones; and $\alpha = 1$ penalizes all nodes equally, regardless of how many instances they receive. This gives us control on how the feature sparsity is distributed across the tree (clearly seen in fig. 3 vs fig. 4), which is useful for interpretability purposes. Further, it can actually find trees that are *both sparser overall and possibly even more accurate than those of the regular TAO algorithm*. Indeed, experimentally we observe that the trees with best generalization error occupy a relatively wide region in (λ, α) space, from which we can pick the sparsest tree.

[§]This assumes $|\mathcal{R}_i| > 0$. If $\mathcal{R}_i = \varnothing$ then $h_\alpha(|\mathcal{R}_i|) = 1$ and the RP solution is to set $\mathbf{w}_i = \mathbf{0}$, as in TAO.

Experiments

In this section, we experimentally demonstrate that our approach learns trees with higher node sparsity while maintaining, or even improving, accuracy. The resulting tree performs comparably to neural network. This, along with other experiments, suggests that insights gained from the tree also apply to the neural network. Importantly, our findings are guaranteed to be correct for hybrid model and verified to hold well empirically for the original NN; unlike those of other interpretability attempts based on saliency maps or Shapley values.

Predictive error and tree size

We train LeNet-5 on the Fashion MNIST dataset using PyTorch 1.10. Full hyperparameters and TAO implementation details are provided in the Appendix. Our trees are trained on embeddings from the last convolutional layer ($F = 400$) to closely match NN performance. To maintain interpretability, we restrict tree depth to 5. The best-performing tree that uses only 1 298 non-zero parameters, achieving $E_{\text{train}} = 5.4\%$ and $E_{\text{test}} = 11.7\%$ is achieved with uniform sparsity distribution ($\lambda = 0.001, \alpha = 1$). In comparison, the NN's fully connected layers contain 59 134 parameters—nearly 50 times more—while improving error rates by only about 1% on both train and test sets.

We also trained axis-aligned trees using CART and TAO. The best CART tree achieved a training error of 8.7% and a test error of 23.4% with a depth of 31 and 5 567 nodes. The best TAO univariate tree had a test error of 21.8% with 4 400 nodes. These results highlight the limitations of axis-aligned trees in capturing the complex feature interactions learned by neural networks while also losing interpretability due to their excessive size.

Regularization path

We further analyze the impact of sparsity parameters λ and α. Figure 5 illustrates the regularization path by fixing λ to 100, 10, 1, and 0.1, while gradually increasing α. As predicted by Eq. 6, increasing α enhances relative sparsity in decision nodes closer to the root. Beyond a certain threshold, the root becomes too sparse to sustain an oblique split, causing the tree to collapse. Lowering λ shifts this threshold, allowing the tree to maintain its structure over a broader range of α. This effect is evident in the figure: for $\lambda = 100$, $\alpha = 0.5$ leads to collapse, whereas for $\lambda = 0.1$, $\alpha = 0.5$ still preserves competitive performance. We find that there is a range of λ, α values that achieve best trees but with different sparsity distribution, from which we can pick the best one. The values of λ between 10 and 0.1 with corresponding α between -1 and 0.5 generally produce similar test error, however the number of non-zero parameters can be reduced $\times 10$ from around 5000 ($\lambda = 1, \alpha = -1$) to 220 ($\lambda = 1, \alpha = 0.6$). For example, comparing 3 trees shows how sparsity distribution changes from uniform in Fig. 3 to closer to leaves in Fig. 6 and Fig. 4.

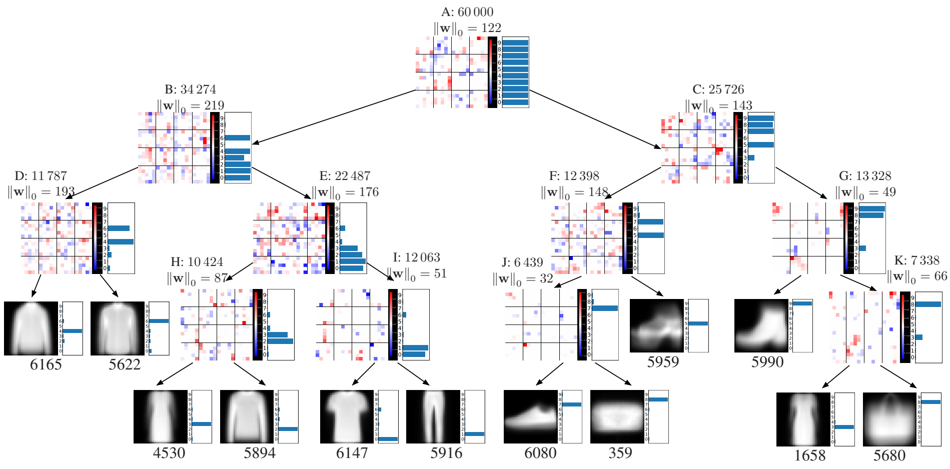

Figure 3. Tree trained on LeNet embeddings on Fashion MNIST dataset $\lambda = 0.001, \alpha = 1$. $E_{\text{train}} = 5.4\%$, $E_{\text{test}} = 11.7\%$ and # non-zero params is 1298

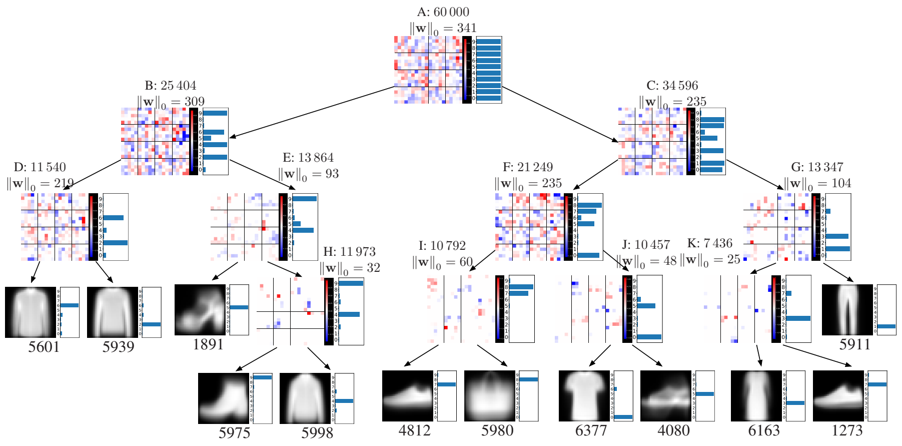

Figure 4. Tree trained on LeNet embeddings on Fashion MNIST dataset, $\lambda = 100, \alpha = -0.25$. $E_{\text{train}} = 5.2\%$, $E_{\text{test}} = 13.0\%$ and # non-zero params is 1701

Global structure

Figure 3 visualizes the structure of the best-performing tree, showing the learned sparse weights at decision nodes. Each node displays weights in a 4×4 grid, where each cell represents a 5×5 activation from the last convolutional layer of LeNet ($16 \times 5 \times 5$). Color coding indicates weight values: red for positive, blue for negative, and white for zero. At the top of each decision node and the bottom of each leaf, we display the number of non-zero weights and the size of the reduced set. Additionally, class histograms are shown on the right side of each node to illustrate the label distribution.

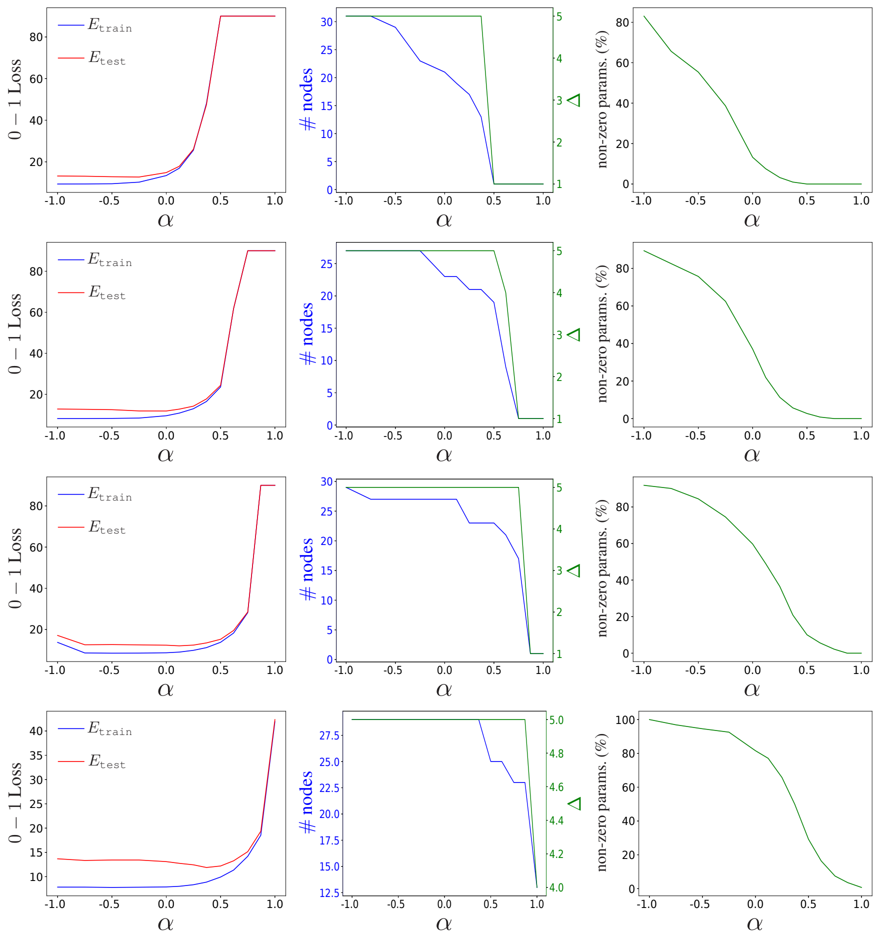

Figure 5. Regularization path over α for different λ (top to bottom $\{100, 10, 1, 0.1\}$).

The tree has just 12 leaves, nearly one per class, significantly enhancing interpretability. Analyzing the leaf nodes reveals a clear hierarchical structure among the classes. The root node A utilizes only 122 activations to almost perfectly separate classes $\{9, 8, 7, 5\}$ from $\{6, 4, 2, 1, 0\}$, with minor misclassifications in class 3. Similarly, decision node C maintains comparable sparsity to the root while effectively distinguishing between $\{7, 5\}$ and $\{9, 8, 3\}$. Subtree at F specializes in distinguishing shoe types, and 359 samples (2.89% of the whole reduced set) of class 8 ("bag"). Interestingly, it uses 148 activations to distinguish class 5 ("sandal") from both "bags" and "sneakers", while differentiating between "sneakers" and "bags" uses relatively small subset of neurons (only 32 out of 400). Similarly, subtree G focuses on distinguishing classes $\{9, 8, 3\}$. Unlike F, subtree G) uses significantly fewer activations in decision

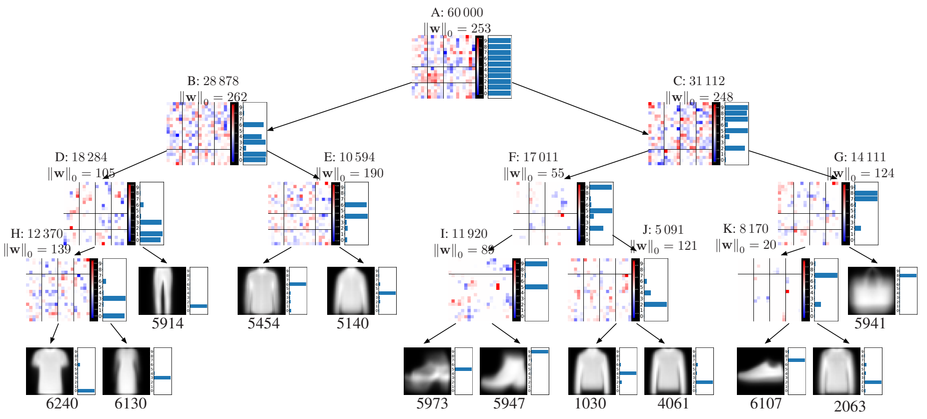

Figure 6. Tree learned on LeNet embeddings on Fashion MNIST dataset $\lambda = 1, \alpha = 0.25$. $E_{\text{train}} = 4.4\%$, $E_{\text{test}} = 12.4\%$ and # non-zero params is 1606

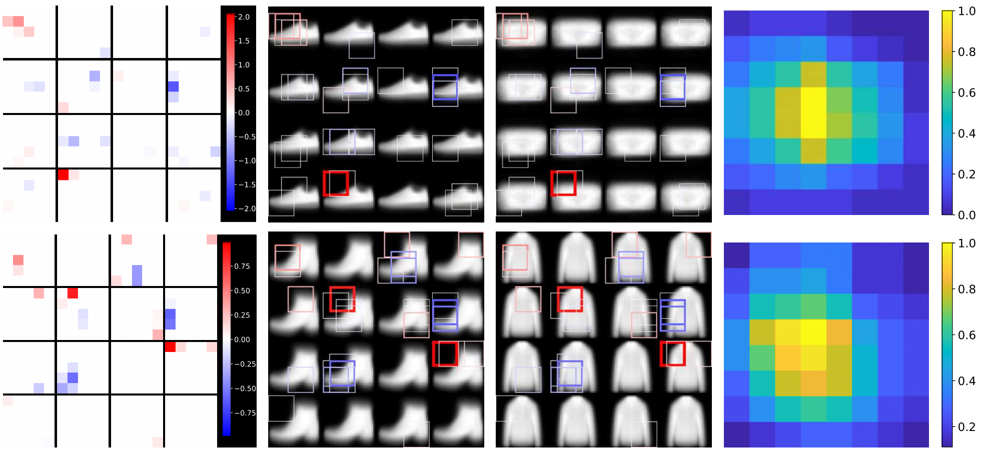

Figure 7. Where is decision node looking in the image? Col. 1 shows sparse weights of decision node J (top) and H (bottom) for each CNN feature map in the last layer of Lenet ($16 \times 5 \times 5$). Col. 2-3 show receptive field produced of neurons with non-zero decision node weights on the mean image from left and right leaf. Receptive field follow the order of CNN outputs left to right and top to bottom. Color and thickness of receptive fields correspond to weights of decision node. Col. 4 is heatmap of the "density" of the receptive fields. Tree hyperparameters are $\lambda = 0.001, \alpha = 1$ (top) and $\lambda = 100, \alpha = -0.25$ (bottom).

nodes G (49) and K (66). These examples highlight how certain neurons specialize in recognizing specific patterns, allowing the sparse tree to retain only essential activations for classification. In contrast, decision node D requires 193 activations to differentiate class 4 ("coat") from class 6 ("shirt"). Subtree E is similar to subtree F as it uses more information to separate sets of classes ($\{3, 2\}$ and $\{0, 1\}$), but distinguishing within these

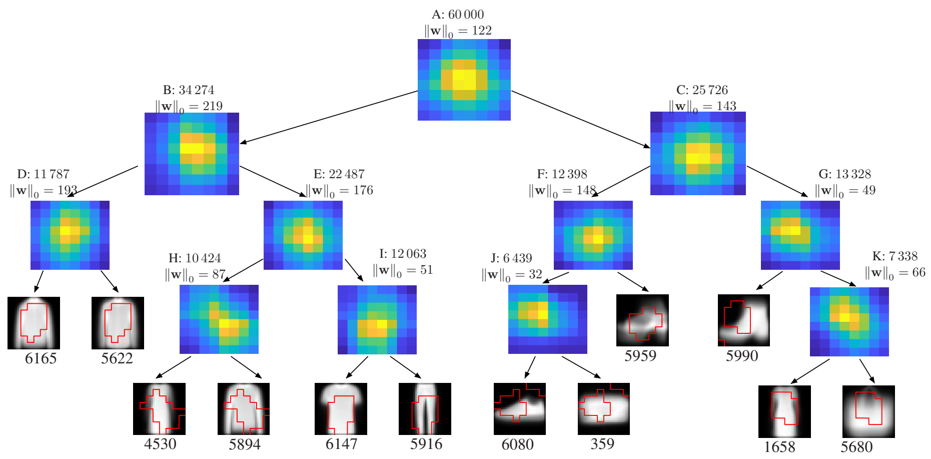

Figure 8. Tree trained on LeNet embeddings on Fashion MNIST dataset $\lambda = 0.001, \alpha = 1$. Similar to fig. 7 each decision node contains "density" map of the receptive fields. Each leaf node contains contour produced by the parent decision node weights and density map.

sets requires far fewer information. Visual inspection suggests that the high similarity between these classes forces the tree to use more activations to maintain accuracy.

Where is a specific neuron, and a specific decision node, looking at?

Although understanding the precise meaning of a neural feature is difficult, due to the complexity of the function of the pixels it defines, what is possible is to construct its *receptive field (RF)*. This is the region of the image that a neuron in a convolutional layer is looking at. It occurs by design: a neuron in convolutional layer i only receives input from a small subset of neurons in layer $i - 1$. Also, since those neurons are organized spatially in a systematic way over the image grid, so are their RFs. We can construct the RFs for all F neurons and then look at how they participate in a decision node. Fig. 7 shows the RFs for a selected node, with a thickness proportional to their weight in the node, overlaid on an input image. Interestingly, RFs associated with the highest positive weights are concentrated in the top left region of the image. This suggests that neurons identifying ink presence in this area play a crucial role in distinguishing between a bag and a shoe, as the shoe image typically lacks content in that specific RF location.

Also, for any decision node, we can construct a *RF density map* over the image as a linear combination of *all* the F RFs (considered as 0/1 indicator functions over the image) using the magnitude of their weights in the node. This objectively indicates the region of the image that is being used in that node: zero density means it is not used at all, and the larger the density the more it is used (because many neurons look there and/or their weights are large). In the example in fig. 7, this clearly shows that that node is focusing on the region where typically we find either the hollow in the front of a shoe or the top-left corner of a bag, which (at that point in the tree) is sufficient to discriminate those two classes.

Figure 8 illustrates the receptive field (RF) density map at each decision node of the tree. For the leaf nodes, we display a single contour line positioned at the midpoint of the density range. This visualization effectively captures the mid-density boundary, which marks the transition between low- and high-density regions. The interior of the contour represents the subset of the receptive field where the density exceeds the midpoint, or the region the model "attends to" more strongly. From the figure it is clear that decision nodes immediately above the leaves concentrate their receptive fields on semantically informative parts of the image. For example, decision node I localizes the cut between the legs of trousers in the central area, a highly discriminative cue for separating pants from t-shirts. Similarly, node G highlights the mid and upper-left regions, corresponding to the tongue and shoelaces of shoes. By leveraging these localized features, the model can clearly separate shoe instances from visually similar categories such as dresses paired with bags. They show how local decision boundaries align with human-interpretable visual structures.

We emphasize that 1) this would not be possible without the tree and the weight vectors, and 2) it is much more objective than saliency maps and other NN visualization techniques that have been shown to be often misleading (Fong and Vedaldi 2017; Adebayo et al. 2018; Ghorbani et al. 2019).

Changing classification of the neural net

The above information can be used to understand why an image is (mis)classified as a certain class, and even to alter this by editing the image. Fig. 9 shows this with the image of a bag that the NN (and the tree) misclassifies as a shoe. From the RF density map discussed above it seems like the bag, which has an odd shape, is missing the typical left corner that bags have. When we edit the image to add that, both the NN and the tree classify it correctly.

Fig. 10 was produced similar to HCZ24 by extracting a mask of non-zero entries from the sparse weight vector, then multiplying it element-wise with the activations from the last CNN layers. The result is passed through fully connected layer to produce classification. The results show that the tree is able to capture specialized neurons, by keeping which, NN is only able to classify some specific classes.

Node mask serves as the core mechanism for feature-level manipulation: given a decision node with weight vector w, the method constructs a binary multiplicative mask $\mu_\times \in \{0,1\}^F$ (optionally complemented by a small additive mask μ_+) that deterministically diverts all instances reaching that node to a desired child. Concretely, the weight vector is partitioned into indices with positive, negative, and zero coefficients, and the mask selectively zeros out features so that the decision rule $w^\top z \geq 0$ always evaluates in favor of the chosen branch, regardless of the input feature vector z. This enables "cutting" entire subtrees, effectively collapsing the classifier's behavior to a prescribed path toward a target leaf, and hence enforcing controlled outcomes such as forcing a class, suppressing a class, or redirecting between sibling classes. Importantly, since sparse oblique trees use only a few active features per node, the resulting masks are

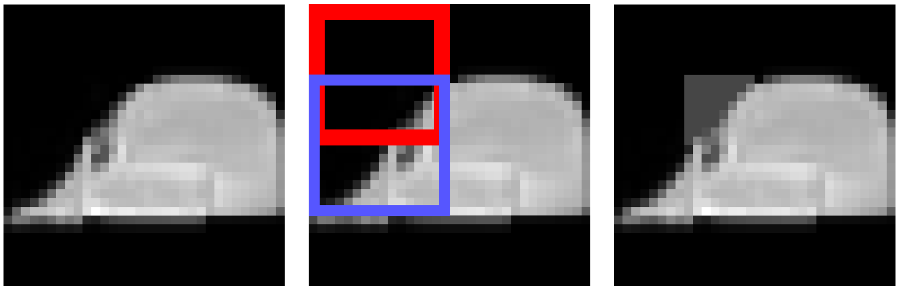

Figure 9. Sample of class 8 misclassified as 7 (Left). Receptive field of neurons from the last convolutional layer of Lenet with largest positive (red) and negative (blue) weights in oblique decision node J (Middle). Small changes in the intersection of two regions fixed the misclassification error (Right).

both compact and interpretable, and they highlight which internal neural representations are causally responsible for specific predictions.

In fig. 10 top 3 figures and bottom left one were produced by creating node masks from the tree in fig. 4 decision nodes G, H, I and J. The masks were designed to navigate the samples to this nodes and predict classes that appear in the leaves. Similarly, next two figures were produced from the decision node K in the tree from fig. 6 and node G from fig. 3. From all 6 results it is clear reveals that internal features (neurons/activations) are functionally critical for classification and how manipulating them alters outcomes. By applying a Node-Mask derived from the sparse oblique tree, one can systematically suppress or activate subsets of features to redirect predictions, showing that only a small fraction of the learned representation drives the network's decisions.

Conclusion

We have revisited, and introduced to the neurosymbolic learning community, a hybrid model consisting of a convolutional feature extraction module composed with a sparse oblique decision tree. By introducing a new type of regularization and suitably modifying the Tree Alternating Optimization (TAO) algorithm, we have further developed this model to control how the feature sparsity is distributed over the tree structure. We train it in two stages: first, we train a regular CNN using SGD; then, we train the tree to replace its fully-connected layers using a teacher-student approach and our modified TAO algorithm. This produces an accurate hybrid model that benefits from the ability of convolutional layers to learn a better representation of an image, and from the ability of trees to explain the reasoning used to classify the image based on those neural features. This makes it possible to understand to some extent which neurons affect which classes, where in the image those neurons are looking at, why a specific image is (mis)classified as a certain class, and how to edit an image to alter its classification in desired ways.

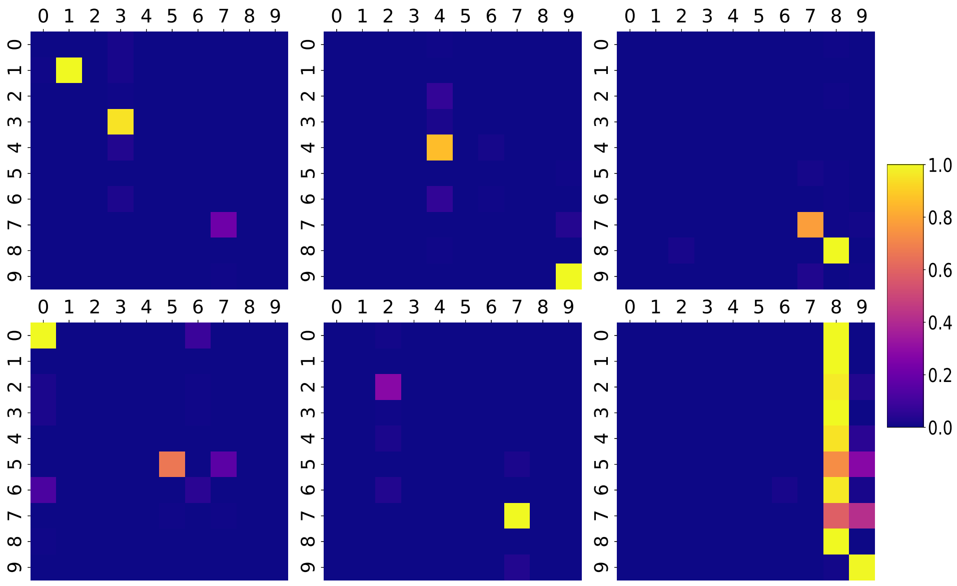

Figure 10. Confusion matrix of neural network predictions modified by tree mask. The masks are produced by analysing sparse weights of decision nodes. From left to right (top to bottom) first 4 are decision nodes G, H, I, J of tree with $\lambda = 100, \alpha = -0.25$. Next, decision node K of tree with $\lambda = 1, \alpha = 0.25$. Lastly, positive weights from subtree G in tree with $\lambda = 0.001, \alpha = 1$. The values are normalized.

References

Adebayo J, Gilmer J, Muelly M, Goodfellow I, Hardt M and Kim B (2018) Sanity checks for saliency maps. In: Bengio et al. (2018), pp. 9505–9515.

Andrews R, Diederich J and Tickle AB (1995) Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems* 8(6): 373–389.

Baesens B, Setiono R, Mues C and Vanthienen J (2003) Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science* 49(3): 255–350.

Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N and Garnett R (eds.) (2018) *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31. MIT Press, Cambridge, MA.

Borisov V, Leemann T, Seßler K, Haug J, Pawelczyk M and Kasneci G (2024) Deep neural networks and tabular data: A survey. *IEEE Trans. Neural Networks and Learning Systems* 35(6): 7499–7519.

Breiman LJ, Friedman JH, Olshen RA and Stone CJ (1984) *Classification and Regression Trees*. Belmont, Calif.: Wadsworth.

Carreira-Perpiñán MÁ, Gabidolla M and Zharmagambetov A (2023) Towards better decision forests: Forest Alternating Optimization. In: *Proc. of the 2023 IEEE Computer Society Conf.*

Computer Vision and Pattern Recognition (CVPR'23). Vancouver, Canada, pp. 7589–7598.

Carreira-Perpiñán MÁ and Gazizov K (2024) The tree autoencoder model, with application to hierarchical data visualization. In: Globerson A, Mackey L, Belgrave D, Fan A, Paquet U, Tomczak J and Zhang C (eds.) *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37. MIT Press, Cambridge, MA, pp. 32281–32306.

Carreira-Perpiñán MÁ and Hada SS (2021) Counterfactual explanations for oblique decision trees: Exact, efficient algorithms. In: *Proc. of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*. Online, pp. 6903–6911.

Carreira-Perpiñán MÁ and Tavallali P (2018) Alternating optimization of decision trees, with application to learning sparse oblique trees. In: Bengio et al. (2018), pp. 1211–1221.

Carreira-Perpiñán MÁ and Zharmagambetov A (2020) Ensembles of bagged TAO trees consistently improve over random forests, AdaBoost and gradient boosting. In: *Proc. of the 2020 ACM-IMS Foundations of Data Science Conference (FODS 2020)*. Seattle, WA, pp. 35–46.

Chen T and Guestrin C (2016) XGBoost: A scalable tree boosting system. In: *Proc. of the 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (SIGKDD 2016)*. San Francisco, CA, pp. 785–794.

Craven M and Shavlik JW (1994) Using sampling and queries to extract rules from trained neural networks. In: *Proc. of the 11th Int. Conf. Machine Learning (ICML'94)*. pp. 37–45.

Craven M and Shavlik JW (1996) Extracting tree-structured representations of trained networks. In: Touretzky DS, Mozer MC and Hasselmo ME (eds.) *Advances in Neural Information Processing Systems (NIPS)*, volume 8. MIT Press, Cambridge, MA, pp. 24–30.

Daumé III H and Singh A (eds.) (2020) *Proc. of the 37th Int. Conf. Machine Learning (ICML 2020)*. Online.

d'Avila Garcez A and Lamb LC (2023) Neurosymbolic AI: The 3rd wave. *Artificial Intelligence Review* 56: 12387–12406.

Domingos P (1998) Knowledge discovery via multiple models. *Intelligent Data Analysis* 2(1–4): 187–202.

Duch W, Setiono R and Żurada JM (2004) Computational intelligence methods for rule-based data understanding. *Proc. IEEE* 92(5): 771–805.

Fong RC and Vedaldi A (2017) Interpretable explanations of black boxes by meaningful perturbation. In: *Proc. 16th Int. Conf. Computer Vision (ICCV'17)*. Venice, Italy, pp. 3449–3457.

Fu L (1994) Rule generation from neural networks. *IEEE Trans. Systems, Man, and Cybernetics* 24(8): 1114–1124.

Gabidolla M and Carreira-Perpiñán MÁ (2022a) Optimal interpretable clustering using oblique decision trees. In: *Proc. of the 28th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (SIGKDD 2022)*. Washington, DC, pp. 400–410.

Gabidolla M and Carreira-Perpiñán MÁ (2022b) Pushing the envelope of gradient boosting forests via globally-optimized oblique trees. In: *Proc. of the 2022 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'22)*. New Orleans, LA, pp. 285–294.

Gabidolla M, Zharmagambetov A and Carreira-Perpiñán MÁ (2024) Beyond the ROC curve: Classification trees using Cost-Optimal Curves, with application to imbalanced datasets. In: Salakhutdinov R, Kolter Z, Heller K, Weller A, Oliver N, Scarlett J and Berkenkamp F (eds.) *Proc. of the 41th Int. Conf. Machine Learning (ICML 2024)*. Vienna, Austria, pp. 14343–14364.

Ghorbani A, Abid A and Zou J (2019) Interpretation of neural network is fragile. In: *Proc. of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*. Honolulu, HI, pp. 3681–3688.

Good J, Kovach T, Miller K and Dubrawski A (2023) Feature learning for interpretable, performant decision trees. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M and Levine S (eds.) *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36. MIT Press, Cambridge, MA, pp. 66571–66582.

Guidotti R, Monreale A, Ruggieri S, Turini F, Giannotti F and Pedreschi D (2018) A survey of methods for explaining black box models. *ACM Computing Surveys* 51(5): 93.

Hada SS, Carreira-Perpiñán MÁ and Zharmagambetov A (2024) Sparse oblique decision trees: A tool to understand and manipulate neural net features. *Data Mining and Knowledge Discovery* 38: 2863–2902.

Hazimeh H, Ponomareva N, Mol P, Tan Z and Mazumder R (2020) The tree ensemble layer: Differentiability meets conditional computation. In: Daumé III and Singh (2020), pp. 4138–4148.

Hitzler P and Sarker MK (eds.) (2021) *Neuro-Symbolic Artificial Intelligence: The State of the Art*. Number 342 in Frontiers in Artificial Intelligence and Applications. IOS Press.

Ibrahim S, Behdin K and Mazumder R (2024) End-to-end feature selection approach for learning skinny trees. In: Dasgupta S, Mandt S and Li Y (eds.) *Proc. of the 27th Int. Conf. Artificial Intelligence and Statistics (AISTATS 2024)*. Valencia, Spain, pp. 2863–2871.

Jacobsson H (2005) Rule extraction from recurrent neural networks: A taxonomy and review. *Neural Computation* 17(6): 1223–1263.

Jordan MI and Jacobs RA (1994) Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* 6(2): 181–214.

Kairgeldin R and Carreira-Perpiñán MÁ (2024) Bivariate decision trees: Smaller, interpretable, more accurate. In: *Proc. of the 30th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (SIGKDD 2024)*. Barcelona, Spain, pp. 1336–1347.

Kautz H (2022) The third AI summer. *AI Magazine* 43(1): 105–125.

Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q and Liu TY (2017) LightGBM: A highly efficient gradient boosting decision tree. In: Guyon I, v Luxburg U, Bengio S, Wallach H, Fergus R, Vishwanathan S and Garnett R (eds.) *Advances in Neural Information Processing Systems (NIPS)*, volume 30. MIT Press, Cambridge, MA, pp. 3146–3154.

Kontschieder P, Fiterau M, Criminisi A and Rota Buló S (2015) Deep neural decision forests. In: *Proc. 15th Int. Conf. Computer Vision (ICCV'15)*. Santiago, Chile, pp. 1467–1475.

McCormick K, Abbott D, Brown MS, Khabaza T and Mutchler SR (2013) *IBM SPSS Modeler Cookbook*. Packt Publishing.

Quinlan JR (1993) *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Setiono R and Liu H (1996) Symbolic representation of neural networks. *IEEE Computer* 29(3): 71–77.

Towell GG and Shavlik JW (1993) Extracting refined rules from knowledge-based neural networks. *Machine Learning* 13(1): 71–101.

Zharmagambetov A and Carreira-Perpiñán MÁ (2020) Smaller, more accurate regression forests using tree alternating optimization. In: Daumé III and Singh (2020), pp. 11398–11408.

Zharmagambetov A and Carreira-Perpiñán MÁ (2022) Semi-supervised learning with decision trees: Graph Laplacian tree alternating optimization. In: Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K and Oh A (eds.) *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35. MIT Press, Cambridge, MA, pp. 2392–2405.

Zharmagambetov A, Gabidolla M and Carreira-Perpiñán MÁ (2021a) Softmax tree: An accurate, fast classifier when the number of classes is large. In: Moens MF, Huang X, Specia L and Yih SWt (eds.) *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP 2021)*. Online, pp. 10730–10745.

Zharmagambetov A, Hada SS, Gabidolla M and Carreira-Perpiñán MÁ (2021b) Non-greedy algorithms for decision tree optimization: An experimental comparison. In: *Int. J. Conf. Neural Networks (IJCNN'21)*. Virtual event.