LEARNING A TREE OF NEURAL NETS

Arman Zharmagambetov *Miguel Á. Carreira-Perpiñán*

Dept. Computer Science & Engineering, University of California, Merced. Merced, CA, USA
Email: {azharmagambetov,mcarreira-perpinan}@ucmerced.edu

ABSTRACT

Much of the success of deep learning is due to choosing good neural net architectures and being able to train them effectively. A type of architecture that has been long sought is one that combines decision trees and neural nets. This is straightforward if the tree makes soft decisions (i.e., an input instance follows all paths in the tree with different probabilities), because the model is differentiable. However, the optimization is much harder if the tree makes hard decisions, but this produces an architecture that is much faster at inference, since an instance follows a single path in the tree. We show that it is possible to train such architectures, with guaranteed monotonic decrease of the loss, and demonstrate it by learning trees with linear decision nodes and deep nets at the leaves. The resulting architecture improves state-of-the-art deep nets, by achieving comparable or lower classification error but with fewer parameters and faster inference time. In particular, we show that, rather than improving a ResNet by making it deeper, it is better to construct a tree of small ResNets. The resulting tree-net hybrid is also more interpretable.

Index Terms— decision trees, deep neural nets, neural trees, fast inference, interpretability

1. INTRODUCTION

In recent years, deep learning have been successfully applied in various machine learning problems such as image, audio and natural language processing. One possible explanation of success and popularity of NNs (neural nets) is in their representation learning ability. Specifically, the set of nonlinear transformations performed by neural net layers helps to obtain features that can capture important properties of the input. Moreover, this type of models are usually trained using SGD optimization in a GPU which allows scaling to large datasets.

On the other hand, decision trees are also among one of the most popular machine learning model which is widely used in practice. Despite being pretty simple model, decision trees have number of practical benefits. Prediction of particular instance follows a single path which makes it very fast during inference time. Moreover, that path usually can be formulated as "IF-THEN" rules which allows to interpret a model. But training decision tree is a difficult optimization problem, involving a search over a large set of tree structures, and over the parameters at each node. A tree architecture is not differentiable since each internal node of a tree defines hard splitting (based on thresholding). The most successful algorithms are based on a greedy growth of the tree structure by recursively splitting nodes. The most popular metohds include CART [1] and C4.5 [2]. In practice, it works well with axis-aligned trees only (a decision node tests for a single feature) which limits input feature utilization.

These advantages and limitations of both decision trees and neural nets motivate for incorporating both of them to obtain a better model. Specifically, trees would become more powerful by employing neural nets inside a node which results to better performance. Moreover, the inference time of the whole architecture will be fast due to hierarchical structure of the tree and the new model potentially would be more interpretable than the regular NNs.

The problem is that this hybrid model inherits the same issues as in regular trees regarding optimization, i.e. it is not straightforward to train such model. Majority of the existing works on training hybrids of decision trees and neural nets either rely on transforming a tree into so called soft decision tree [3, 4, 5] (i.e., an input instance follows all paths in the tree with different probabilities) or take inspiration from CART-style algorithms by recursive greedy splitting based on some criteria (e.g. impurity measure) [6, 7, 8]. Both of these approaches have some pros and cons. For instance, the soft decision trees can be easy to optimize due to differentiability of the entire architecture, however, during inference, an instance follows every path and assigns every leaf a certain probability which considerably slows down inference time and affects negatively to the interpretability. Though it is possible to apply some relaxation and choose the path with the highest probability instead, but such approximations usually do not have any guarantees in terms of performance. The problem with the second approach (i.e. generic greedy tree growing) is that they have no guarantees concerning the optimization of the desired loss (e.g. classification loss) and shown to be highly suboptimal [9].

In this paper, we propose a novel method of training hybrids of decision trees and neural nets which we call "a tree of neural nets". It trains a decision tree with hard split but without using recursive greedy approach. We build on top of recently proposed *Tree Alternating Optimization (TAO)* algorithm [10, 11]) since it shows promising results in training a tree of arbitrary complexity [12] and it has been successfully applied to train forests [13, 14]. Our final model is a single binary tree (see fig. 1) with hard decision nodes that has: *neural nets in the leaves* and *sparse linear internal nodes* (i.e. oblique nodes) which operate directly on input feature vector (e.g. raw image pixels). Training a truly hard tree allows a lightweight inference and more model interpretability.

2. TREE OPTIMIZATION USING THE TAO ALGORITHM

The TAO algorithm takes as input a decision tree with a predefined structure and optimizes that tree given the desired objective function such as misclassification error. In contrast with other methods, each TAO iteration guarantees monotonic decrease of the objective function and it can additionally penalize the complexity of the trees by applying various regularizations (we use ℓ_1-regularization in this paper).

Assuming a tree structure T is given (say, binary complete of some depth), consider the following optimization problem over its

parameters on a given training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$:

$$E(\boldsymbol{\Theta}) = \sum_{n=1}^{N} L(y_n, T(\mathbf{x}_n; \boldsymbol{\Theta})) + \lambda \sum_{i \in \mathcal{N}} \|\mathbf{w}_i\|_1 \qquad (1)$$

where $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}$ is the parameters of all nodes $i \in \mathcal{N}$ of the tree. The loss function $L(\mathbf{y}, \cdot)$ is 0/1 loss (although it is possible to use any other classification losses). The regularization term penalizes the parameters $\boldsymbol{\theta}_i$ of each node. Here, the tree output is denoted as $T(\cdot)$.

Separability condition, proposed in the original TAO paper, states that any set of non-descendant nodes of a tree (e.g. all nodes at the same depth) can be optimized independently. Here, we provide a high level overview (details can be found in [10, 11]). First, we assume that nodes do not share common parameters, and second, a non-descendant set of nodes have disjoint set of training points at each node. This is because decision tree defines hard partition, and thus each instance follows a unique root-to-leaf path. We also assume that the remaining part of the tree is fixed (due to the usage of alternating optimization). This allows the objective function defined in eq. (1) to separate over any non-descendant set of nodes:

$$E(\boldsymbol{\Theta}) = E_i(\boldsymbol{\theta}_i) + E_{\text{rest}}(\boldsymbol{\Theta}_{\text{rest}}) \qquad (2)$$

Note that $E_{\text{rest}}(\boldsymbol{\Theta}_{\text{rest}})$ can be treated as a constant since we fix $\boldsymbol{\Theta}_{\text{rest}}$.

Now, we can apply separability condition to train leaves and decision nodes. First, let us consider leaves. Clearly, leaves are non-descendant w.r.t each others. By fixing the remaining parameters of a tree, we can train each leaf independently. Note that instance follows root-to-leaf path and actual prediction is given by a leaf node. Therefore, tree output from eq. (1) can be replaced by the output of a leaf. But that output applies only to a subset of points reaching that particular leaf. Therefore, it solves much simpler problem which is a K-class classifier on that subset.

Similarly, separability condition can be applied to any subset of decision (internal) nodes that are non-descendants, for example: all nodes at the same depth. Below is the single decision node optimization problem:

$$\min_{\boldsymbol{\theta}_i} E_i(\boldsymbol{\theta}_i) = \sum_{n \in \mathcal{R}_i} l_{in}(f_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \alpha\, \phi_i(\boldsymbol{\theta}_i) \qquad (3)$$

where $f_i(\cdot)$ defines a decision function in that particular node. It can have only two possible outputs (namely right or left) because we consider only binary trees. Moreover, since we are using oblique trees, this thresholding function has a linear form with weight vector $\mathbf{w}_i \in \mathbb{R}^D$ and bias $b_i \in \mathbb{R}$. We also define a loss function l_{in} as the loss incurred if choosing the right or left subtree. So, we encourage $f_i(\cdot)$ to send the instance to the best child which can be formulated as the equivalent binary classification problem with pseudolabels $\overline{y}_n \in \{-1, +1\}$. To sum up, optimizing over internal nodes is equivalent to optimizing a linear binary classifier over $\{\mathbf{w}_i, b_i\}$ on the instances that currently reach node i.

3. LEARNING A TREE OF NEURAL NETS

We propose the following method to optimize hybrids of decision trees and neural nets. Suppose we have $\{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset \mathbb{R}^D \times \{1, \dots, K\}$ is a training set of D-dimensional real-valued instances and their ground truth labels (in K classes). The architecture of a tree of neural nets is depicted in fig. 1 and it consists of linear internal nodes and NNs at the leaves. Neural net architecture is fixed and defined ahead of time as well as the depth of a tree. Initially, internal nodes have random parameters. Initial parameters for leaves can be either random or from the pretrained neural net. Each type of nodes (leaf or internal) are optimized as follows: *Internal node* training can be solved by fitting a linear binary classifier over the "reduced set" as described in section 2. A linear classifier is directly trained on input feature vector (e.g. raw image pixels). We force linear classifier to be sparse to achieve high interpretability and fast inference; Optimization over a *Leaf* is done by training a neural net in a traditional way by applying SGD optimization over the subset of training instances $\{(\mathbf{x}_n, y_n)\}$ that currently reach leaf i.

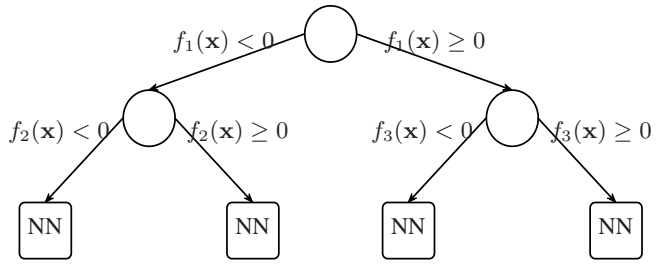

Fig. 1. Architecture of a tree of neural nets with depth=2. Here $f_i(x) = \mathbf{w}_i^T \mathbf{x} + b_i$.

input training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset \mathbb{R}^D \times \{1, \dots, K\}$
initial tree T (with random parameters)
repeat
  for $i \in$ nodes of T, visited in reverse BFS
    if i is a leaf **then**
      $y_i \leftarrow K$ class classifier (e.g. neural net) on the training points that reach i
    else
      $\boldsymbol{\theta}_i \leftarrow$ binary classification on the reduced set
until stop
postprocess T: remove dead branches & pure subtrees
return T

Fig. 2. Pseudocode for the TAO algorithm. Visiting each node in reverse BFS order means scanning depths from depth(T) down to 0, and at each depth processing (in parallel, if so desired) all nodes at that depth. "Stop" occurs when either the objective function changes less than a set tolerance or the number of iterations reaches a set limit.

This leads to the simple and efficient alternating optimization algorithm depicted in fig. 2 which repeatedly trains one subset of nodes and fixes all the rest.

3.1. Computational complexity

Training time can be computed similarly as in [10, 11]. Consider some depth of a tree. Each node i at that depth solves a reduced problem on a subset of points (of size N_i) that currently reach node i. Since the subsets N_i are disjoint, the union of all N_i gives us the total set of points of size N. For internal nodes, each node solves SVM problem on a subset of size at most N_i in time $\mathcal{O}(DN_i^\alpha)$ for $\alpha \geq 1$ (α depends on the SVM solver; section 4.2 in [15]). Hence, since $\sum_i N_i^\alpha \leq \left(\sum_i N_i\right)^\alpha \leq N^\alpha$ if $\alpha \geq 1$, solving all the SVMs at the same depth is at most as costly as solving a single SVM on

Table 1. Structure of the neural nets at the leaves for MNIST, Fashion-MNIST (FMNIST) and CIFAR10. "conv5-x" denotes x convolutional kernels of size 5×5, "MP" stands for 2×2 max-pooling, "FC" fully connected layer followed by softmax. For all ResNets we use the same structure as in the original paper [20].

Model	Structure
tao-mnist-lin	FC
tao-mnist-cnn1	conv5-1 + MP + FC
tao-mnist-cnn2	conv5-5 + MP + conv5-10 + MP + FC
tao-mnist-cnn3	conv5-6 + MP + conv5-16 + MP + FC
tao-fmnist-lin	FC
tao-fmnist-cnn	conv5-32 + MP + conv5-32 + MP + FC
tao-cifar-resnet*	see [20]

all N points. Moreover, we need to compute reduced set for each node optimization which introduces some overhead. This overhead is negligible at the internal nodes since propagating points through the tree follows a single path and internal nodes are sparse linear functions. But the noticeable overhead arises in the leaves due to neural nets. Let us denote M as the computation time for passing a single point through the neural net which is proportional to the neural net complexity. Since the total number of points at some depth is N then the total time for propagating them is proportional to MN. So, the overall training time for the internal nodes at some depth is $\mathcal{O}(DN_i^{\alpha} + MN)$ for $\alpha \geq 1$.

Each leaf is a K-class classifier and training them involves a neural net optimization. Similarly, each leaf i obtains a subset of points (of size N_i) and union of all N_i for all leaves gives us the total set of points N. Therefore, the training time of all leaves is at most as costly as training a single neural net on all N points.

4. EXPERIMENTS

We evaluate our trees of neural nets on MNIST [16], Fashion-MNIST [17] and CIFAR-10 [18] classification benchmarks since most of the competing neural tree models evaluate their performance on them (see [5, 4, 8]). We compare our method with DT/NN based models and related works which combine them.

We take as initial tree a deep enough, complete binary tree with random parameters. Internal nodes are linear and trained using ℓ_1-regularized logistic regression implemented inside LIBLINEAR [19] package. Leaves are neural nets of different complexity which varies depending on dataset (see table 1). We pretrain neural net on the whole training set for the fixed number of iterations and use it as initialization for all leaves. All of our code is in Python and for neural net training we use Tensorflow. We run TAO algorithm for a fixed (most of the time 20) number of iterations to train our trees.

4.1. Model performance

Table 2 summarizes our results. In general, DT based models such as Random Forest [21] are compact (i.e. has fewer parameters and very fast during prediction time) but show poor performance on the given classification tasks. Whereas state-of-the-art deep nets (e.g. DenseNet [22]) show the lowest error but has many parameters and can be heavy during inference. *Our produced trees of neural nets have comparable performance w.r.t. deep nets and they are shallower trees with less model size and fast inference time. We also show that the produced trees are in general more interpretable* (see section 4.2).

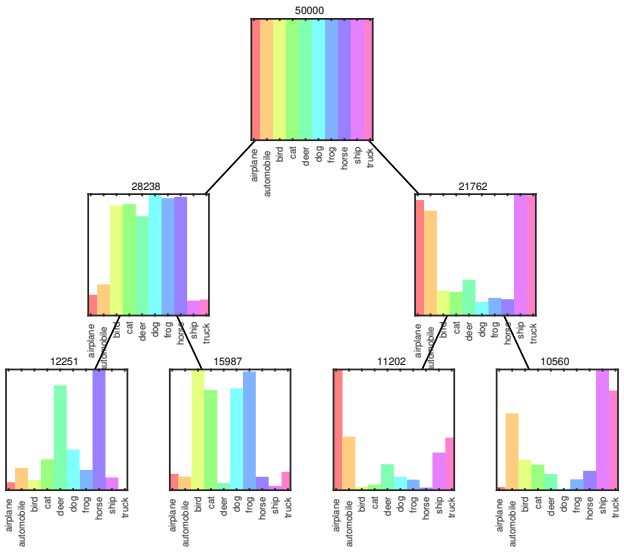

Fig. 3. Visualization of class distributions of a tree of neural nets obtained by our algorithm for tao-cifar-resnet56. Clear semantic separation of classes shows that each leaf specializes on some group of classes (zoom in for a better view).

For instance, our neural trees with extremely simple neural nets on the leaves (e.g. "tao-mnist-cnn2") outperforms various ensemble methods like Random Forest or Alternating Decision Forests [23]. As for the neural nets, "tao-mnist-cnn3" produced by the algorithm was able to achieve the same test error as LeNet5 [16] with far less model size and inference time. Similar situation happens with other datasets as well.

4.2. Model interpretability

Hierarchical structure of neural trees obtained by TAO algorithm allows us to interpret a model to some extent. By "to some extent" we mean that the resulting model will not be as interpretable as traditional decision trees but it is possible to extract some useful information out of it. Here, we need to compromise between model interpretability and performance. Decision trees (axis-aligned) are easy to interpret since root-to-leaf path usually can be formulated as "IF-THEN" rules. Therefore, one can easily understand and justify a certain decision made by the model. But they perform poorly on number of tasks and sometimes one may need to use more powerful model. Moreover, interpretability starts to vanish once we use ensemble of trees (e.g. random forest) or very big trees. On the other hand, deep neural nets are powerful and show state-of-the-art results on various problems, but they are hard to interpret. Models between DT and NN form some kind of path. We believe that our trees of neural nets stand in the middle of this path finding a good balance between high interpretability and high performance. For instance, we visualize class distributions at each node for one of the obtained trees (see fig. 3). We can clearly see that each leaf achieves some form of specialization on subset of classes rather than classifying all of them. This specialization potentially makes training a leaf easier by focusing more on majority classes. Moreover, we observe that this separation is not random but due to semantic similarity of the objects. For example, human made objects were separated from ani-

Table 2. Performance comparison of different models on MNIST, Fashion-MNIST and CIFAR-10. "Params" - the total number of parameters in the model. "Inference (flops)" - inference runtime in number of operations (scalar multiplications and additions). "Ensemble Size" indicates the number of trees used. "Tree depth" shows the depth of a single tree in an ensemble. The numbers in brackets indicates our approximate estimation of number of parameters and flops in the worst case.

	Method	E_{test} (%)	Number of parameters	Inference (FLOPS)	Ensemble size	Maximum depth
MNIST	CART axis-aligned [1]	12.50	(4,096)	(12)	1	12
	CART oblique [1]	11.00	(3,210,480)	(9,408)	1	12
	Linear Classifier	7.81	7,840	15,680	-	-
	tao-mnist-lin	**4.11**	127,515	19,108	1	4
	tao-mnist-cnn1	**4.10**	47,184	33,000	1	5
	Random Forests [23]	3.21	(3,600,000)	(2,500)	100	25
	Shallow NDF (sNDF) [4]	2.80	(18M)	(18M)	80	10
	Alternating Decision Forests [23]	2.71	(3,600,000)	(2,500)	100	25
	Neural Decision Tree (NDT) [8]	2.10	(1,773,130)	(502,170)	1	3
	tao-mnist-cnn2	**0.91**	24,403	305,000	1	3
	LeNet5-pruned [24]	0.82	12,915	-	N/A	N/A
	Deep NDF (dNDF) [4]	0.70	(544,600)	(4.3M)	10	5
	Adaptive Neural Trees (ANT) [5]	0.69	100,596	-	1	2
	LeNet5 [16]	0.67	431,000	4.2M	N/A	N/A
	tao-mnist-cnn3	**0.67**	21,029	481,000	1	2
Fashion-MNIST	Logistic Regression [17]	16.00	7,840	0.02M	-	-
	Linear Classifier [17]	15.75	7,840	0.02M	-	-
	KNN [17]	14.00	60,000	60,000	-	-
	tao-fmnist-lin	**13.15**	62,720	0.02M	1	3
	Random Forests [17]	12.10	-	5,000	100	50
	MLP(1 hidden) [17]	12.30	79,400	0.16M	N/A	N/A
	2 Conv+pooling [25]	8.84	3.2M	17M	N/A	N/A
	tao-fmnist-cnn	**7.84**	252,353	4.2M	1	3
	2 Conv+3 FC [25]	7.77	1.8M	10.67M	N/A	N/A
CIFAR-10	gcForest [26]	38.22	-	-	500	
	ResNet20 [20]	8.51	0.27M	(58.42M)	N/A	N/A
	tao-cifar-resnet20	**7.81**	1.07M	(58.42M)	1	2
	ResNet32 [20]	7.42	0.46M	(99.98M)	N/A	N/A
	tao-cifar-resnet32	**6.98**	1.85M	(99.98M)	1	2
	ResNet56 [20]	6.73	0.85M	(183.11M)	N/A	N/A
	Adaptive Neural Trees (ANT) [5]	6.72	1.30M	-	1	2
	tao-cifar-resnet56	**6.51**	1.70M	(183.11M)	1	1
	ResNet110 [20]	6.43	1.70M	(370.15M)	N/A	N/A
	DenseNet-BC(k=24) [22]	3.74	27.2M	-	N/A	N/A

mals at depth 1. Such distribution of classes came out automatically as a result of training a tree, and we did not impose any constraints. Similar observations were found in [5].

4.3. Inference runtime and model size

During prediction time, each point follows a single root-to-leaf path. Computations on internal nodes in this path are extremely fast because each decision function is sparse oblique node. The main computational intense part corresponds to the leaf which is a single NN. Therefore, the total prediction time is proportional to the NN complexity at the leaves. Our final trees also have a compact model size since each leaf uses smaller neural net instead of having a single but large deep net. Results on table 2 agree with above statements. Both of the number of parameters and inference time are less than the comparable NNs or neural tree models. For instance, our best tree for MNIST shows the same test error as LeNet5 but has about 20 times fewer parameters and 10 times faster (in FLOPS).

5. CONCLUSION

We address an issue of optimizing a tree of neural nets. Such combination of neural nets and trees can potentially result to better models which show a good balance between error, model size and interpretability. We have presented our approach to train such hybrids with complex structures. The proposed tree optimization method is efficient and can be scaled to large data. The method produces compact trees (fewer parameters and fast inference) which shows comparable or better performance against SoA neural nets or other models. Moreover, resulting trees make it possible to interpret.

6. ACKNOWLEDGMENTS

Work partially funded by NSF award IIS–2007147. The authors gratefully acknowledge computing time on the Multi-Environment Computer for Exploration and Discovery (MERCED) cluster at UC Merced, which was funded by NSF Grant No. ACI–1429783.

7. REFERENCES

[1] Leo J. Breiman, Jerome H. Friedman, R. A. Olshen, and Charles J. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, Calif., 1984.

[2] J. Ross Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[3] Michael I. Jordan and Robert A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, vol. 6, no. 2, pp. 181–214, Mar. 1994.

[4] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Buló, "Deep neural decision forests," in *Proc. 15th Int. Conf. Computer Vision (ICCV'15)*, Santiago, Chile, Dec. 11–18 2015, pp. 1467–1475.

[5] Ryutaro Tanno, Kai Arulkumaran, Daniel C. Alexander, Antonio Criminisi, and Aditya Nori, "Adaptive neural trees," in *Proc. of the 36th Int. Conf. Machine Learning (ICML 2019)*, Kamalika Chaudhuri and Ruslan Salakhutdinov, Eds., Long Beach, CA, June 9–15 2019, pp. 6166–6175.

[6] Heng Guo and Saul B. Gelfand, "Classification trees with neural network feature extraction," *IEEE Trans. Neural Networks*, vol. 3, no. 6, pp. 923–933, Nov. 1992.

[7] Samuel Rota Buló and Peter Kontschieder, "Neural decision forests for semantic image labelling," in *Proc. of the 2014 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'14)*, Columbus, OH, June 23–28 2014, pp. 81–88.

[8] Han Xiao, "NDT: Neual Decision Tree towards fully functioned neural graph," arXiv:1712.05934, Dec. 16 2017.

[9] Trevor J. Hastie, Robert J. Tibshirani, and Jerome H. Friedman, *The Elements of Statistical Learning—Data Mining, Inference and Prediction*, Springer Series in Statistics. Springer-Verlag, second edition, 2009.

[10] Miguel Á. Carreira-Perpiñán and Pooya Tavallali, "Alternating optimization of decision trees, with application to learning sparse oblique trees," in *Advances in Neural Information Processing Systems (NEURIPS)*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. 2018, vol. 31, pp. 1211–1221, MIT Press, Cambridge, MA.

[11] Miguel Á. Carreira-Perpiñán, "The Tree Alternating Optimization (TAO) algorithm: A new way to learn decision trees and tree-based models," arXiv, 2021.

[12] Arman Zharmagambetov, Suryabhan Singh Hada, Miguel Á. Carreira-Perpiñán, and Magzhan Gabidolla, "An experimental comparison of old and new decision tree algorithms," arXiv:1911.03054, Mar. 20 2020.

[13] Arman Zharmagambetov and Miguel Á. Carreira-Perpiñán, "Smaller, more accurate regression forests using tree alternating optimization," in *Proc. of the 37th Int. Conf. Machine Learning (ICML 2020)*, Hal Daumé III and Aarti Singh, Eds., Online, July 13–18 2020, pp. 11398–11408.

[14] Miguel Á. Carreira-Perpiñán and Arman Zharmagambetov, "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, Oct. 19–20 2020, pp. 35–46.

[15] Leon Bottou and Chih-Jen Lin, "Support vector machine solvers," in *Large Scale Kernel Machines*, Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, Eds., Neural Information Processing Series, pp. 1–28. MIT Press, 2007.

[16] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[17] Han Xiao, Kashif Rasul, and Roland Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," arXiv:1708.07747, Sept. 15 2017.

[18] Alex Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. of Computer Science, University of Toronto, Apr. 8 2009.

[19] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, "LIBLINEAR: A library for large linear classification," *J. Machine Learning Research*, vol. 9, pp. 1871–1874, Aug. 2008.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proc. of the 2016 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'16)*, Las Vegas, NV, June 26 – July 1 2016, pp. 770–778.

[21] Leo Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[22] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger, "Densely connected convolutional networks," in *Proc. of the 2017 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'17)*, Honolulu, HI, July 21–26 2017, pp. 2261–2269.

[23] Samuel Schulter, Paul Wohlhart, Christian Leistner, Amir Saffari, Peter M. Roth, and Horst Bischof, "Alternating decision forests," in *Proc. of the 2013 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'13)*, Portland, OR, June 23–28 2013, pp. 508–515.

[24] Miguel Á. Carreira-Perpiñán and Yerlan Idelbayev, ""Learning-compression" algorithms for neural net pruning," in *Proc. of the 2018 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'18)*, Salt Lake City, UT, June 18–22 2018, pp. 8532–8541.

[25] "Official web page of the Fashion-MNIST dataset," https://github.com/zalandoresearch/fashion-mnist.

[26] Zhi-Hua Zhou and Ji Feng, "Deep forest: Towards an alternative to deep neural networks," in *Proc. of the 17th Int. Joint Conf. Artificial Intelligence (IJCAI'01)*, Seattle, Washington, USA, Aug. 4–10 2001, pp. 3553–3559.