Oldies but Goldies Project Report
Decision Trees and Neural Networks

Leander Girrbach girrbach@cl.uni-heidelberg.de
Institut für Computerlinguistik
Universität Heidelberg

Abstract

This project explores different ideas for combining neural networks (mostly MLPs) and decision trees (or random forests). In the first part, I implement and evaluate the model proposed by Frosst and Hinton (2017), including a slight generalisation (to n-ary trees). My experiments show that the model is fast to train, yields good performance, and ist robust wrt. to changes in hyperparamters. In the second part, I distill a MLP and a LSTM model into Decision Trees and evaluate the performance. My experiments show that this does not work very well: The drop in performance is huge, and tree models trained independently on the same data yield better results than the distilled models.

1. Differentiable Decision Trees

1.1 Introduction & Motivation

The aim of of this first part of the project is to implement and evaluate differentiable decision trees that can be trained by backpropagation. Of course, we want to compare differentiable decision trees both to standard decision trees and to regular neural networks (MLPs in this case). The reference paper which I reimplement is Frosst and Hinton (2017).

Neural networks show good performance, for example in learning to solve classification tasks. However, neural networks are not interpretable by humans. It would be extremely hard or impossible to figure out what the learned transformations mean in comprehensible terms. Decision Trees also show good performance, but more importantly, their classification is easy to understand by humans. Therefore, this project aims at combining techniques from deep learning (neural networks) and the interpretability of decision trees.

This part of the project consists of 2 parts:

1. First, I will develop the implementation of differentiable decision trees
2. I will evaluate the implementation on text classification tasks and compare the performance to the performance of other models

1.2 Method

Model architecture Frosst and Hinton (2017) propose the following approach: They represent the the differentiable decision tree as a binary tree. Each inner node computes the probability of descending to the right or left child node. The probability of descending to the right child node is given as

$$p_i(x) = \sigma(\beta \cdot (w \cdot x + b))$$

where $\sigma : \mathbb{R} \to (0,1), x \mapsto \frac{1}{1+\exp(-x)}$ is the sigmoid function, w is a learned weigth vector, b is a learned bias, β is a temperature hyperparameter, and x is the feature vector of the input data.

Accordingly, the probability of descending to the left child node is given as

$$1 - \sigma(\beta \cdot (w \cdot x + b))$$

Each leaf node stores a learned, constant distribution over the possible classes (called "bigot" in the paper). This means that while the distribution is learned by backpropagation, it is independent of the input.

Classification works as follows: The probabilities calculated for each node induce a probability for each leaf node. This probability is the path probability from the root to the respective leaf node. Frosst and Hinton (2017) propose two possibilities of calculating the final prediction:

1. Calculate a weighted linear combination of the constant leaf node distributions (weighted by the path probabilities)
2. Return the distribution with the highest probability (leaf node with highest path probability)

Only the first option allows straightforward end-to-end optimisation. However, for inference, both options are possible. While the first option more closely resembles random forests, the second option resembles (single) decision trees more closely. Frosst and Hinton (2017) report the first option to yield (slightly) better performance.

Regularisation Frosst and Hinton (2017) propose the following regularisation: For each node, they calculate $\alpha_i = \frac{\sum_x P^i(x) \cdot p_i(x)}{\sum_x P^i(x)}$ where $p_i(x)$ is the activation of node i and P^i is the path probability from the root node to node i. The sum is over all datapoints in a batch. Then, they propose to minimize the regularisation term:

$$C = -\lambda \cdot \sum_i \frac{1}{2} \log \alpha_i + \frac{1}{2} \log(1 - \alpha_i)$$

This is the negative cross-entropy between the (average) probabilities of descending to the two child nodes and the uniform distribution (equal probability of descending to any child node).

Extension and Generalistation The method proposed by Frosst and Hinton (2017) can be easily extended to higher node-degrees than 2, that is, one node can have multiple children. In this case, the sigmoid activation is replaced by a softmax activation which gives the probabilities of descending to the respective child node. Furthermore, the regularisation can be replaced by a simple entropy regularisation. In order to avoid degenerate solutions, models with higher-entropy node distributions are preferred.

1.3 Experiments

In order to evaluate the differentiable decision tree implementations, I conduct the following experiments:

1. Hyperparameter study for depth, β, and regularisation. Here, I want to examine the effects of different value for β and γ. Also, I want to evaluate the relation between depth and number of training epochs, that is how fast training is in relation to the tree depth.
2. Comparison of performance of a text classification task between differentiable decision trees, decision tree classifiers, random forest classifiers, and MLPs.

All experiments are carried out using a binary differentiable decision tree and a ternary differentiable decision tree.

The goal of this project is not to optimise the performance of any model but rather to investigate the performance of differentiable decision trees. Therefore, it is methodologically unproblematic to report any evaluation scores on the test set and not use any validation set.

Data For conducting experiments, I use the 20 newsgroups dataset as provided by sklearn (Pedregosa et al., 2011). There are multiple reasons for using this particular dataset:

- It as an easily available NLP (text classification) dataset
- It is multiclass classification
- It is reasonably small so that we don't need too much compute to conduct experiments

For preprocessing, I lowercase all documents. Then, I use spacy to tokenize and lemmatize all documents. Finally, I remove stopwords.

In order to extract feature vectors, I construct the tf-idf document-term matrix. Tokens occuring in less than 10 documents are discarded. Furthermore, because high-dimensional data is unpractical with neural networks, I factorise the document-term matrix using SVD. Eventually, each document is represented as a 512-dimensional feature vector.

Training of Differentiable Decision Trees The model is implemented in PyTorch Paszke et al. (2019). I train the models by maximising the negative log-likelihood of predicting the correct class. As optimiser I choose AdamW (Loshchilov and Hutter, 2017) with default parameters, as AdamW shows much faster convergence and better results than SGD in this case. Batch size is 32 for all models and training continues for 12 epochs without early stopping. Also, I clip gradients with norm ≥ 1.0.

Test set accuracy is evaluated and saved after every epoch.

Hyperparameter Evaluation In this experiment, I want to study the effect of the β and γ parameters. Remember, β as a temperature parameter influences how skewed node distributions are. Higher values of β lead to more skewed distributions, while lower values increase entropy. γ is the weighting of the regulariser.

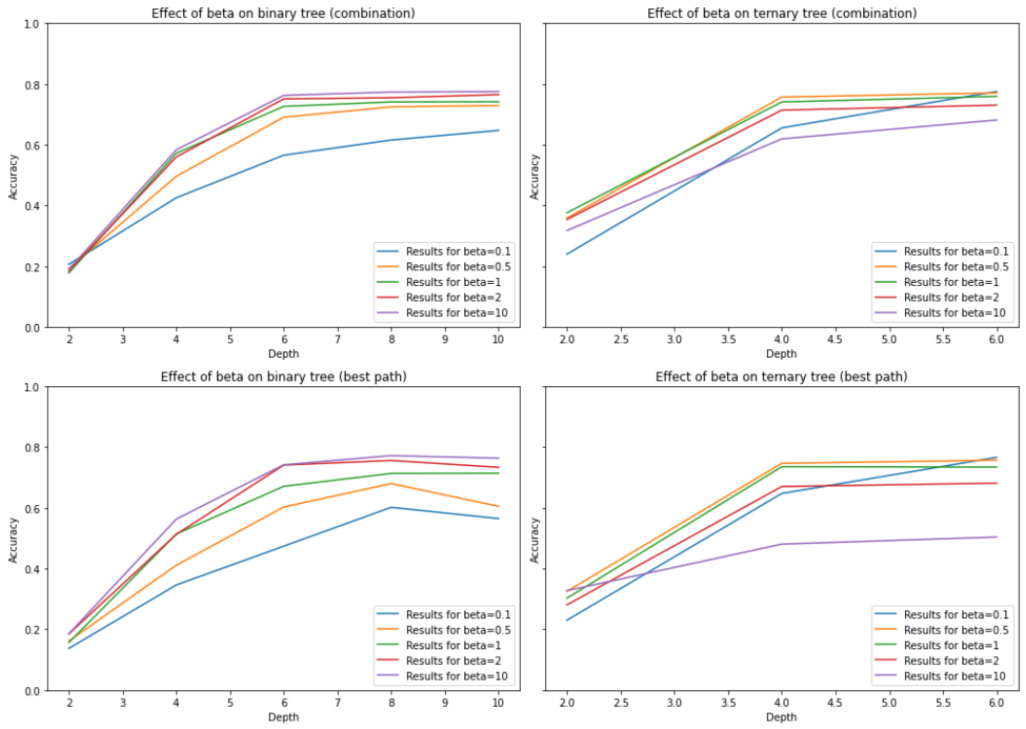

Figure 1: Hyperparameter study for β

The standard values are $\beta = 1$ and $\gamma = 0.01$. I am particularly interested in their effect on trees of varying depth. Therefore, I evaluate the following hyperparameter settings: depth $= \{2, 4, 6, 8, 10\}$, $\beta = \{0.1, 0.5, 1, 2, 10\}$ and $\gamma = \{0.001, 0.01, 0.1, 1\}$. For evaluating β, I vary the depth and β while keeping $\gamma = 0.01$. For evaluating γ, I vary the depth and γ while keeping $\beta = 1$. Results for β are in Fig. 1. Results for γ are in Fig. 2.

Also, I want to find out, how long trees of varying depth need to be trained and what their maximum performance is. Do we observe a depth and/or number of epochs where the performance does not increase further?

For this, experiment, I train binary trees of depth $\in \{2, 4, 6, 8, 10\}$ and ternary trees of depth $\in \{2, 4, 6\}$ for 30 epochs and report performance on the test set after every epoch. β and γ are set to their default values of $\beta = 1$ and $\gamma = 0.01$. Results are in Fig. 3.

Comparison to other models In this experiment, I compare the performance of differentiable decision trees to the performance of decision tree classifiers and random forest classifiers. In particular, I vary the depth of the trees. I want to find out how important the depth is for the different classifiers. Additionally, I want to compare the performance of differentiable decision trees to the performance of MLPs. However, it is impractical to vary the depth of an MLP. Therefore, as a proxy, I use MLPs with 1 hidden layer, and vary the width. If a tree has depth d, the corresponding MLP has width 2^d, that is the MLP has 2^d hidden units.

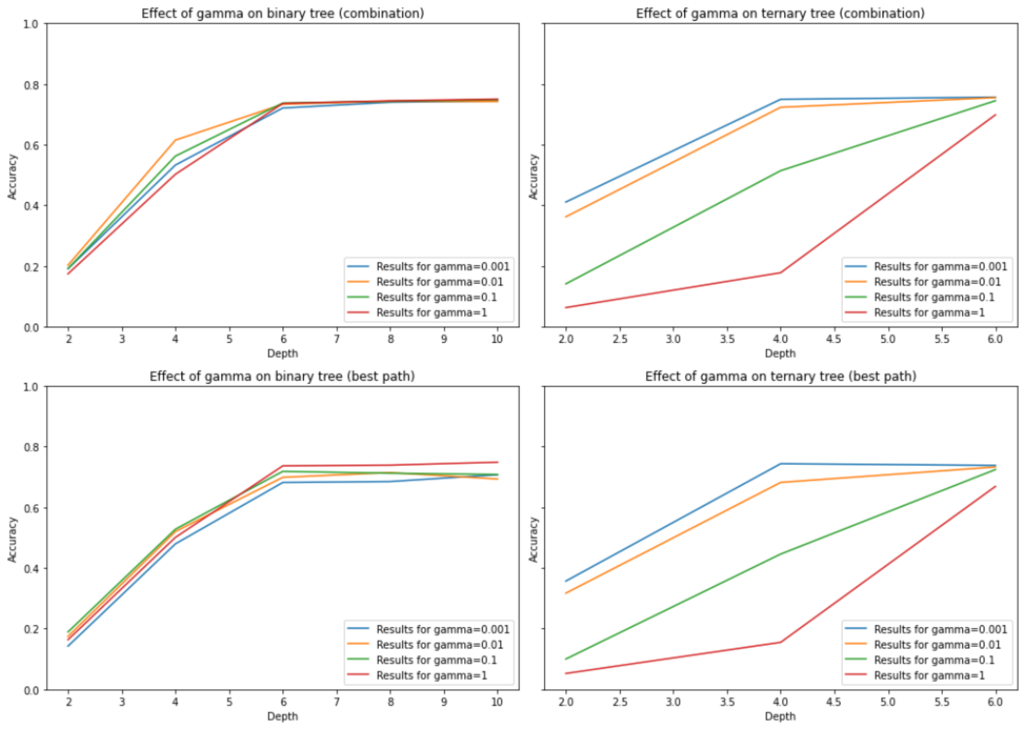

Figure 2: Hyperparameter study for γ

For the random forest classifier, I use 512 random decision trees. All other hyperparameters take their default value (provided by sklearn). The MLP is trained using the Adam optimiser and batch size 32. Results are in Fig. 4.

1.4 Analysis

From the conducted experiments, we the following conclusions can be drawn:

1. Both binary trees and ternary trees are relatively robust wrt. changes of hyperparameters (β and γ). Generally, lower-entropy solutions seem to yield better performance (high β for binary trees, low β for ternary trees). However, this can be achieved much better by adjusting β than by increasing γ. Too high γ has negative effects on the performance. This is most obvious for ternary trees.

2. The trees need relatively few epochs until the test set performance saturates. Also, the trees need a certain depth in order to achieve maximum performance, but increasing depth more doesn't increase performance further.

3. Differentiable Decision Trees (both using a weighted combination of distributions and using only the most probable path) perform better than Decision Trees and Random Forests but worse than MLPs. This suggests that they constitute an acceptable middle-ground between neural networks and random forests, both in terms of interpretability and performance.

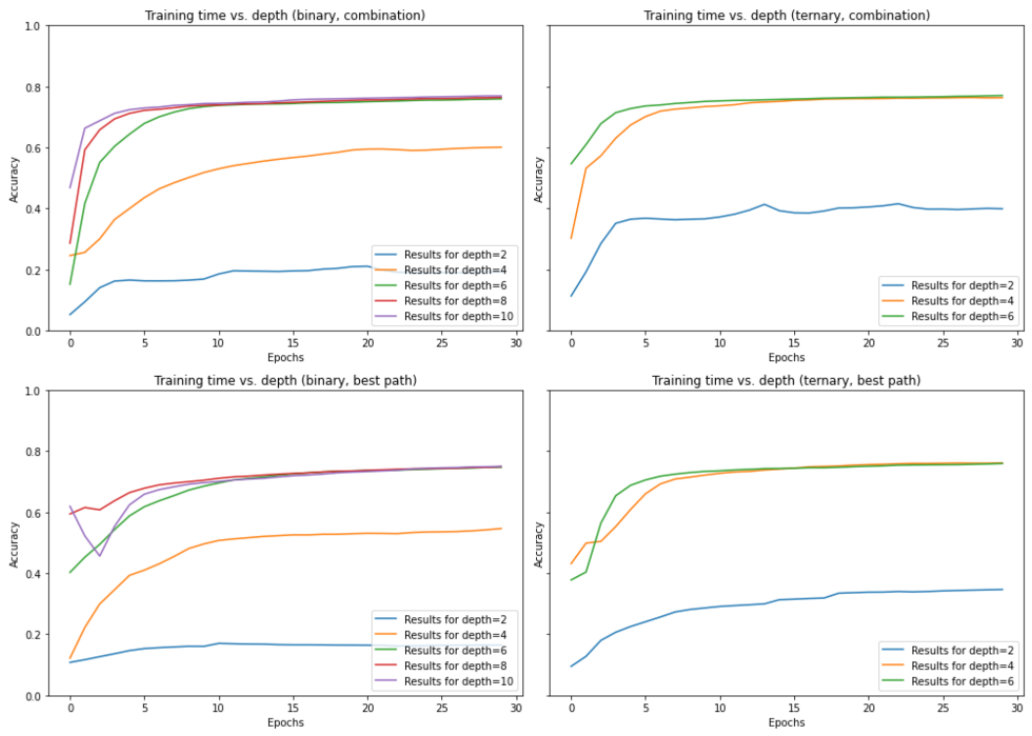

Figure 3: Hyperparameter study for epochs vs. tree depths.

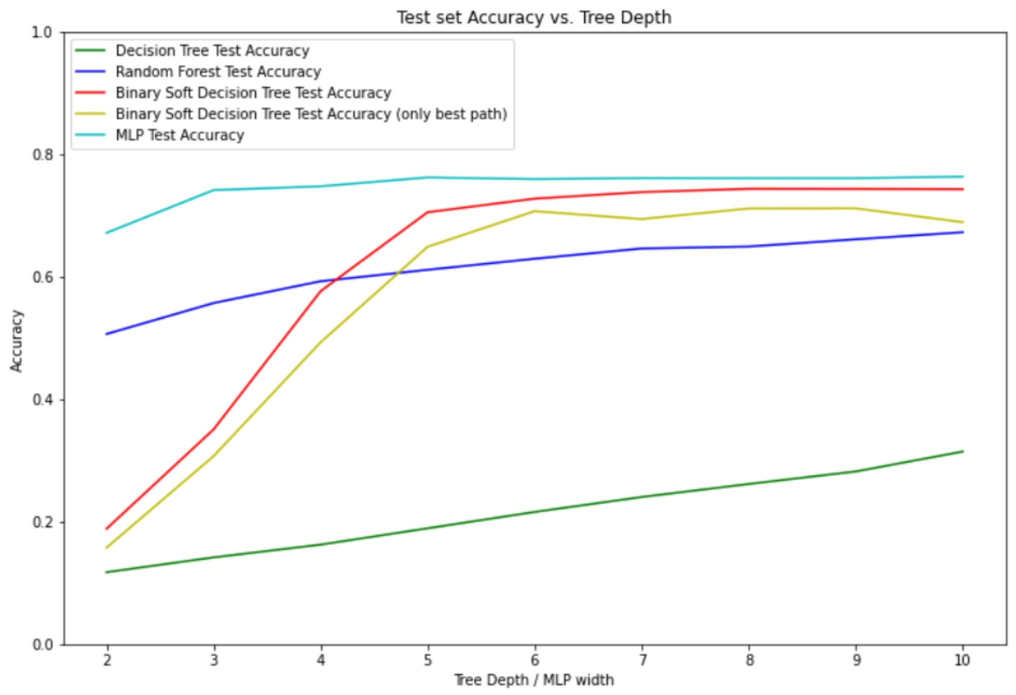

Figure 4: Comparison to other models wrt. tree depth/MLP width

2. Distilling Neural Networks into Decision Trees

2.1 Idea

When training a neural network on a classification task, we receive an uninterpretable classifier. A subfield of deep learning is distillation, which seeks to make a smaller neural network behave like a larger neural network. One of the main reasons for distillation is to reduce the computational power needed to solve a certain problem.

The same idea can be used to a learn a completely different type of classifier to imitate the calculations of the neural network. To this end, we view the neural network as a multivariate function mapping input vectors to probability distributions over the labels $\mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{\#\text{labels}}$.

We can use a trained teacher model and some data to train another student classifier to behave like the teacher neural network. We can either require the student classifier to only output the same labels as the teacher neural network or also require the student classifier to output the same probability distribution over labels as the teacher neural network.

2.2 Method

Given a dataset of paired inputs and labels $(\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}})$, I train a neural networks on $\mathcal{X}_{\text{train}}$ to predict the corresponding labels. This model serves as the teacher. After the training process, I use another set of inputs $\mathcal{X}_{\text{distill}}$ to calculate the label distribution induced by the teacher model. By taking the $\arg\max$, I can get the predicted labels.

For $\mathcal{X}_{\text{distill}}$, I evaluate 2 options:

- The train dataset $\mathcal{X}_{\text{train}}$
- A larger dataset containing documents from the same domain

Using this information, I train Decision Tree classifiers / Random Forest classifiers to predict the same labels on $\mathcal{X}_{\text{distill}}$ as the teacher model. Furthermore, I train Decision Tree regressors / Random Forest regressors to predict the same probability distributions on $\mathcal{X}_{\text{distill}}$ as the teacher model.

2.3 Data

For $\mathcal{X}_{\text{train}}$, I reuse the 20 newsgroups dataset as in Part 1. For $\mathcal{X}_{\text{distill}}$, I add documents from the AG NEWS-dataset as provided by torchtext. Preprocessing is the same as in Part 1, namely lowercasing, tokenising, lemmatising, and filtering stopwords.

2.4 Models

I evaluate the distilling neural networks on 2 types of neural networks (trained on the same data):

1. A MLP feedforward neural network. Here, I represent documents by SVD truncated tf-idf weighted bag-of-words features.

2. A Bidirectional LSTM classifier. Here, tokens are represented by pretrained word2vec embeddings provided by gensim (Řehůřek and Sojka, 2010)

The MLP has 2 hidden layers with 128 units each. The LSTM has also has 2 layers with 128 units each (both directions). The LSTM is implemented in keras (Chollet et al., 2015). Both models are trained using the Adam optimiser with default parameters (as specified by sklearn/keras) by minimising the cross-entropy of the predicted label probabilities and the real one-hot-encoded labels. Batch size is 32 in both cases. The LSTM is trained for 20 epochs.

2.5 Evaluation

I report the following metrics:

- Test set accuracy (on the 20 newsgroups test set)
- Reference accuracy (on the 20 newsgroups test set): Here, the predictions of the teacher model are treated as true labels
- Train set accuracy (on the 20 newsgroups train set)
- R2 coefficient of determination between probabilities predicted by student and teacher models
- KL-Divergence between probabilities predicted by student and teacher models. For decision trees, this makes only sense for the regressors, because classification trees return only one label (one-hot distribution)

With these metrics, both a good impression of the overall performance of the models (wrt. the data) and the approximation performance (wrt. approximating the teacher model) can be evaluated.

2.6 Results & Analysis

Results for distilling an MLP are in Table 1. Results for distilling an LSTM are in Table 2.

This experiment, too, shows that Random Forests are superior to single Decision Trees, which is not surprising. Regressors are also better than classifiers in this case.

Comparing the different metrics proves that accuracy is not a good suitable metric for measuring how similar the calculations are, because the non-distilled ("reference") models achieve strong accuracy results, while distilled models are visibly better when comparing the coefficient of determination (R2) and KL-Divergence.

This comparison also shows that at least Regression Forests are to some extend able to simulate the calculations of neural networks. This becomes especially clear from looking at the KL-Divergence. However, this ability remains rather limited, which is visible from the overall performance and the exact scores.

Model	Test Acc.	Ref. Acc.	Train Acc.	R2	KL-Divergence
Ref. MLP	0.752	1.000	0.997	1.000	0.000
Ref. LSTM	0.773	0.708	0.988	0.501	1.661
Ref. DTC	0.424	0.436	0.999	-0.228	-
Ref. RFC	0.714	0.725	0.999	0.380	-
(T/o) Dist. DTC	0.431	0.440	0.997	-0.198	15.220
(T/o) Dist. RFC	0.658	0.668	0.997	0.488	1.223
(Syn.) Dist. DTC	0.344	0.351	0.997	-0.417	-
(Syn.) Dist. RFC	0.620	0.638	0.997	0.275	-
(Syn.) Dist. DTR	0.346	0.357	0.997	-0.273	11.982
(Syn.) Dist. RFR	0.579	0.594	0.997	0.385	1.499

Table 1: Results for distilling a trained MLP into decision tree /random forest models. Metrics are described in Sec. 2.5. Note that Decision Tree Classifiers directly return labels (≈ discrete distribution), therefore KL-Divergence is ill defined for Decision Tree classifiers. Explanation of acronyms: “Ref.” = “Reference”, “Acc.” = “Accuracy”, “DT” = Decision Tree, “RF” = Random Forest. “(T/o)” means distilled only using $\mathcal{X}_{\text{train}}$. “(Syn.)” means distilled using $\mathcal{X}_{\text{synthetic}}$. The closing “C” or “R” as in “RFC” mean classifier or regressor. “Ref.” in models means trained on $\mathcal{X}_{\text{train}}$, not distilled. “Ref. Accuracy” means accuracy wrt. the reference MLP. R2 means R2 coefficient of determination.

Model	Test Acc.	Ref. Acc.	Train Acc.	R2	KL-Divergence
Ref. LSTM	0.774	1.000	0.988	1.000	0.000
Ref. MLP	0.752	0.708	0.997	0.460	3.903
Ref. DTC	0.424	0.412	0.999	-0.333	-
Ref. RFC	0.714	0.672	0.999	0.366	-
(T/o) Dist. DTC	0.430	0.412	0.988	-0.270	6.046
(T/o) Dist. RFC	0.661	0.630	0.986	0.456	1.244
(Syn.) Dist. DTC	0.326	0.321	0.988	-0.537	-
(Syn.) Dist. RFC	0.547	0.537	0.988	0.240	-
(Syn.) Dist. DTR	0.348	0.341	0.988	-0.252	4.958
(Syn.) Dist. RFR	0.560	0.542	0.986	0.357	1.499

Table 2: Results for distilling a trained LSTM into decision tree /random forest models. Metrics are described in Sec. 2.5. Note that Decision Tree Classifiers directly return labels (≈ discrete distribution), therefore KL-Divergence is ill defined for Decision Tree classifiers. Explanation of acronyms: “Ref.” = “Reference”, “Acc.” = “Accuracy”, “DT” = Decision Tree, “RF” = Random Forest. “(T/o)” means distilled only using $\mathcal{X}_{\text{train}}$. “(Syn.)” means distilled using $\mathcal{X}_{\text{synthetic}}$. The closing “C” or “R” as in “RFC” mean classifier or regressor. “Ref.” in models means trained on $\mathcal{X}_{\text{train}}$, not distilled. “Ref. Accuracy” means accuracy wrt. the reference LSTM. R2 means R2 coefficient of determination.

Two surprising findings are that the additional data doesn't increase or even decreases the performance of the distilled models, and that results for distilling the LSTM are very similar to results for distilling the MLP. Possible consequences are either that decision trees/random forests do not benefit very much from additional data, or that in this case, there is a domain mismatch between $\mathcal{X}_{\text{test}}$ and $\mathcal{X}_{\text{distill}}$. Another possible consequence is that the performance of distilled trees may be rather independent of the teacher model complexity.

Summing up, these experiments have shown that distilling neural networks into decision trees/random forests yield only very limited success. The drop in performance is huge, and training an independent classifier on the original data always yields better results. Also bear in mind that trees cannot really process sequence data, which makes them an unpractical tool in NLP in general.

References

François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.