Reproducibility report of Deep Neural Decision Trees

Karim Gorgy
Department of Engineering
McGill University

Bassel Sharif
Department of Engineering
McGill University

Daniel Wassef
Department of Computer Science
McGill University

Reproducibility Summary

Scope of Reproducibility

The main claim of the original paper produced by Yongxin Yang [2], Irene Garcia Morillo, and Timothy M. Hospedales is that Deep Neural Decision Trees (DNDT) can effectively combine the interpretability of traditional decision trees with the representational capacity of neural networks. The paper argues that DNDT performs comparably or better than traditional decision trees on tabular data while maintaining interpretability and benefiting from neural network properties like differentiability and GPU acceleration.

Methodology

In our reproduction attempt, the official code provided as the baseline, ensuring compatibility with the original implementation. The experiments were conducted on a standard workstation with an Intel(R) Iris(R) Xe GPU. The original implementation was written in PyTorch which we utilized to ensure consistency in the results.

Results

The reproduction results closely aligned with those reported in the original paper. Specifically, the DNDT model demonstrated comparable accuracy on most datasets, with marginal differences within the range of 1-2% for some. These differences could be attributed to the stochastic nature of neural network training. The qualitative assessment of model interpretability remained consistent, as the tree structures produced by DNDT were straightforward to analyze and understand.

What was easy

The reproducibility of results was facilitated by the clear documentation and the availability of the official provided source code. The use of common machine learning frameworks like PyTorch also simplified the process of setting up the experiments and conducting them. The datasets we used were available and did not require extensive preprocessing and data cleaning.

What was difficult

One of the challenging aspects was ensuring the exact replication of the training environment. The libraries used were old and compatibility issues arose when initially setting up the environment. The torch library used in the original code was substituted with a newer version for instance, which led to some modifications in the process, as even minor differences in the versions coupled with differences in hardware could potentially lead to unexpected variations. Additionally, as the computational resources we possess are limited, extensive experimentation of hyper parameters was summarized.

Communication with original authors

There was no direct communication with the original authors during this reproducibility study. However, the comprehensive documentation and published code were sufficient to address the questions that arose during the re-implementation and reproduction of the results.

1 Introduction

This report documents the reproducibility study of a model known as Deep Neural Decision Trees (DNDTs). DNDTs aim to merge the interpretability of traditional decision trees with the representational power of deep learning. This study seeks to replicate the original findings using updated software libraries and to explore the robustness of the model under different conditions.

2 Scope of reproducibility

We aim to validate the following claims we want to prove from the original DNDT paper:

- Claim 1: We anticipate slightly similar results due to the evolution of the libraries used.
- Claim 2: Decision Trees can be more accurate in some cases but DNDTs offer an efficient alternative with easier interpretability.

These claims will be tested by replicating the experiments conducted in the original paper and noting any deviations in results or methods needed by the updated software environment.

3 Methodology

The reproducibility study utilized the original author's code with necessary modifications to account for the updated library versions. The experiments were conducted using Python 3.11.3 within a virtual environment. The Torch library version used was 2.1.1, contrasting with the original implementation's use of Torch 0.4.0. The original code lacked the functionality for accuracy computation and dataset splitting, requiring adaptations during replication. A subset of datasets was chosen in order to ensure rigorous experimentation.

3.1 Model descriptions

In the innovative domain of Deep Neural Decision Trees, our reproduced approach combines the interpretability of traditional decision trees with the powerful data processing capabilities of neural networks. The essence of DNDTs lies in creating decision boundaries similar to those in decision trees but within a neural network architecture. This is achieved by discretizing input features into distinct bins, similar to the way decision trees split data at various nodes[1]. These bins are then used to guide the decision-making process, with each decision path within the tree represented as a unique pattern in the neural network. The training process involves the iterative refinement of these decision paths, optimizing them to accurately classify data. By using optimization techniques commonly employed in neural network training, the model learns to make decisions that are not only accurate but also interpretable, mirroring the clear, rule-based logic of decision trees. This fusion of decision tree simplicity with neural network complexity allows DNDTs to tackle a wide array of tasks, offering both the depth of understanding and adaptability required in complex data environments. The DNDT model was re-implemented with the updated versions of the Torch library which required minor changes in the methods implemented. We implemented the Decision Tree model using Scikit-learn, which was not present in the GitHub repository provided by the author. The DNDT model uses the Kronecker product, cross-entropy loss and Adam optimizer.

3.2 Datasets

The Iris, Haberman's Survival, and the Poker Hand datasets were used to test the model's robustness. The Iris dataset, a more commonly used dataset in the field of machine learning, consists of 150 samples from three species of Iris flowers. With four features representing the physical dimensions of the flowers, it is a prime choice for evaluating classification models.

Haberman's Survival dataset is utilized to predict patient survival rates following breast cancer surgery. It contains data from studies conducted between 1958 and 1970 at the University of Chicago's Billings Hospital and includes age, year of operation, and the number of positive additional nodes detected.

The Poker Hand dataset is composed of a combination of categorical and ordinal variables, each entry representing a hand drawn from a standard deck of cards. The task is to predict the poker hand's strength. With its complex pattern

recognition requirements, it serves as a rigorous test for any machine learning model's capability to manage complex pattern recognition tasks.

3.3 Hyperparameters

Cross-validation was implemented, and experiments were run with various learning rates and temperatures. The Adam optimizer was used to perform experimental analysis using different hyperparameters. The provided results are shown on the table below.

Batch Size	Learning Rate	Avg F1 Score	Avg Accuracy
Highest Values			
16	0.005	0.9597	96.00%
32	0.01	0.9598	96.00%
128	0.01	0.9600	96.00%
Lowest Values			
128	0.001	0.7158	76.00%
16	0.5	0.8825	88.67%
16	1	0.8737	88.67%

Table 1: The Three Highest and Lowest Average F1 Scores and Accuracies for the iris dataset

The values obtained from the experiments using five fold cross-validation and various learning rates and temperatures indicate the model's sensitivity to hyperparameter tuning, a common aspect in machine learning model training. The highest average F1 scores and accuracies were achieved with modest batch sizes (16 and 32) and a learning rate of 0.01. This suggests that a balanced approach to updating the model, not too slow as to miss capturing the data's patterns, nor too fast to overshoot optimal solutions, is ideal for the Iris dataset, which is relatively small yet aptly categorized into three classes.

Conversely, the lowest values were observed with extreme learning rates (either too low at 0.001 or too high at 1), combined with the smallest and largest batch sizes. A learning rate of 0.001 may be too conservative, failing to make significant model adjustments, leading to underfitting, particularly in the larger batch size of 128. This is reflected in the lowest average F1 score and accuracy. A high learning rate of 1, even with a small batch size of 16, likely results in too abrupt updates, causing the model to overlook the nuanced patterns within the data, thus not generalizing well.

These findings are consistent with the expectations for machine learning models, where the right balance of learning rate and batch size needs to be struck to achieve the best generalization from the training process. The Adam optimizer, known for its adaptive learning rate adjustments, seems to have performed well with the dataset at hand, as indicated by the high F1 scores and accuracies in the middle range of hyperparameter values. This demonstrates the optimizer's effectiveness in navigating the error landscape of a well-structured, balanced dataset such as the Iris dataset.

Learning Rate	0.001	0.005	0.01	0.05	0.1	0.5	1
Haberman's Survival Accuracy	67.74%	66.13%	70.97%	70.97%	62.9%	46.77%	66.13%
Iris Accuracy	73.33%	100%	100%	96.67%	96.67%	93.33%	93.33%

Table 2: Test accuracies for different learning rates

Table 2 displays test accuracies for the Haberman's Survival and Iris datasets over various learning rates. Optimal performance for the Haberman's Survival dataset is observed at a learning rate of 0.01, indicating a balance between learning efficiency and model stability. The Iris dataset excels with learning rates of 0.005 and 0.01, affirming its capacity for clear class distinction. However, both datasets show decreased accuracies at higher learning rates which suggests potential overfitting or loss of generalization due to overly aggressive model updates. Thus, fine-tuning the learning rate to match dataset complexity and structure possesses a critical role as highlighted previously.

3.4 Experimental setup and code

The replication of the experiments from the original paper involved setting up a controlled environment that mirrored the original study's conditions as closely as possible. This meant recreating the computational environment as closely as possible, including the software and libraries used in the original experiments. We used updated versions for some libraries, Python 3.11.3 and PyTorch 2.1.1 within a virtual environment to ensure that our experimental setup did not interfere with other projects or system dependencies. The codebase provided by the authors was the starting point for our experiments. However, necessary modifications were made to adjust for the updates in library versions and to add functionality for accuracy computation and dataset splitting, which were not present in the original code. The improvisations made were carefully documented to ensure transparency and reproducibility.

4 Results

Our replication of the experiments confirmed the main findings of the original paper, upholding the second claim that Deep Neural Decision Trees (DNDT) can indeed blend the interpretability of traditional decision trees with the representational power of neural networks. Our results mirror those of the initial study, with slight differences that fall within an acceptable range, thereby supporting the first claim. These minor differences in results are due to the stochastic nature of neural network training and the foreseeable evolution of software libraries.

4.1 Results reproducing original paper

4.1.1 Decision Tree

The performance of traditional decision trees on the datasets paralleled the results of the original work, with the iris dataset achieving 100% test accuracy (refer to Figure 2). For the Haberman's Survival dataset, the model achieved a test accuracy of 65%. Notably, an improved test accuracy of 71% was observed for the Haberman's Survival dataset upon the exclusion of one feature, surpassing the original paper's results. The decision trees were implemented using the Scikit-learn library, and the choice of the Gini impurity measure for node splitting was informed by its mention in the original paper. This underscores the critical role of detailed documentation in preserving the replicability of machine learning research outcomes. Figures 3 and 4 provide a visual presentation.

4.1.2 DNDT

The DNDT model, re-implemented with updated software libraries, demonstrated comparable accuracy to traditional decision trees, while also offering the added benefits of neural networks such as differentiability and GPU acceleration. This aligns with the original paper's assertions, confirming the efficacy of DNDTs as a viable machine learning model that does not compromise on interpretability. Testing the model on the Iris (refer to Figure 1) and Habermans's survival datasets, we achieved remarkable test accuracies of 100% and 70.9% respectively, the same values achieved in the paper.

4.2 Results beyond original paper

Additional explorations conducted as part of this study delved deeper into the impacts of various batch sizes and learning rates not specified in the original paper. Through these extended experiments that were performed, we observed that while the choice of batch size and learning rate greatly affects model performance, the DNDT maintained a robustness that showcases its efficiency. This additional analysis reinforces the claim that DNDTs not only match the performance of traditional decision trees but can also provide enhanced interpretability, making them an excellent tool for complex decision-making tasks.

4.2.1 Analysis of Optimizers

Upon reviewing the reference code of the paper, we observed that the Adam optimizer was utilized but without any explicit justification. In order to gain a more profound understanding, we thought it would be essential to investigate the correlation between the optimizer and learning rate, as these factors are very connected. As a preliminary step, it was necessary for us to ascertain the optimal learning rate. Knowing that the original code specified a learning rate of 0.01,

and after testing, we adopted this value assuming it is the optimal learning rate, as detailed in the Hyperparameters section of our report.

Subsequently, we embarked on an empirical analysis to evaluate the performance of various optimizers using the Iris dataset. The table below summarizes the test accuracies achieved with each optimizer, providing a clear comparison:

Optimizer	SGD	Momentum	Adagrad	RMSprop	Adam
Accuracy	96.67%	93.33%	63.33%	96.67%	100%

Table 3: Test Accuracy of Different Optimizers on the Iris Dataset

The obtained results indicate that Adam optimizer yields a perfect accuracy score, which corroborates its selection in the original paper. However, to ensure a comprehensive understanding, we also assessed the Stochastic Gradient Descent (SGD) and RMSprop optimizers, both of which demonstrated competitive accuracies. In contrast, the Adagrad optimizer significantly underperformed, suggesting its incompatibility with the dataset's characteristics or the model's configuration.

These findings higlight the importance of optimizer selection in the context of machine learning experiments. The optimal choice of an optimizer varies upon both the nature of the dataset and the specific learning task at hand. In future work, we advocate for a methodical approach to the selection and justification of both optimizers and learning rates, to bolster the reproducibility and interpretability of the experimental results.

The Adam optimizer often outperforms other optimizers since it the way it functions combines the advantages of two other popular optimizers: AdaGrad and RMSProp. Its efficacy is attributed to its sophisticated mechanisms that adaptively adjust learning rates for each parameter, guided by the magnitude of their gradients. This dynamic adjustment is based on both the immediate and past gradient information, allowing for more nuanced and efficient learning steps. Adam's computational efficiency is further enhanced by its invariance to the scale of gradients and small memory requirements, which makes it particularly suitable for large-scale problems.

Moreover, Adam's design includes a bias correction feature, which ensures that learning accelerates from the start, leading to faster convergence. This is especially useful in practical scenarios involving non-stationary objectives or data with high variability. Adam's resilience to hyperparameter tuning is another benefit, as it displays a certain resilience to the selection of hyperparameters, unlike other optimizers such as SGD with momentum or RMSProp.

While Adam is generally robust and performs well across various tasks and was the preferred option in our context, its dominance is not absolute. There may be instances where alternative optimizers outperform Adam, showing the importance of context-specific optimizer selection. It's prudent not to default to Adam unconditionally but to consider a range of optimizers, particularly when faced with unique or challenging problem domains.

4.2.2 Kronecker Product

In our experiment, we sought to enhance the original DNDT model by introducing a modification to the default Kronecker product operation. The motivation behind this choice was to explore the impact of introducing non-linearity into the DNDT model. The Kronecker product, a linear operation, had already demonstrated its effectiveness in capturing complex data relationships and achieving a high accuracy of 0.95 in F1 score. However, we wanted to investigate whether a custom product method, which introduces non-linearity through a non-linear transformation, could offer an alternative approach. This modification aimed to strike a balance between linearity and non-linearity within the DNDTs to potentially improve model performance. However, it led to a lower accuracy, of 0.61 in F1 Score.The decrease in accuracy can be attributed to the complexity introduced by non-linearity, making it harder for the model to capture the underlying data relationships effectively. While the custom product method offers flexibility, it appears that the original Kronecker product's linearity is better suited for this specific DNDT application, providing superior accuracy.

5 Discussion

The efforts to reproduce the findings of the original paper on Deep Neural Decision Trees were largely successful, affirming the core assertions of the authors. Our experiments affirm the claim that DNDTs can integrate the interpretability of traditional decision trees with the expressive power of neural networks, as evidenced by our results

which closely matched the original findings.
An essential aspect of DNDTs that merits further discussion is their interpretability. The architecture of DNDTs facilitates an intuitive understanding of the decision-making process, similar to that provided by traditional decision trees. Each decision node within a DNDT corresponds to a decision boundary in the input feature space, which can be explicitly visualized and analyzed. This transparency allows researchers and practitioners to trace the path of any given decision, understand the model's rationale, and, crucially, to validate the integrity of the decisions made by the model. Such interpretability is paramount in fields where explainability is as critical as accuracy, such as healthcare or finance.

However, the evolution of software libraries necessitated methodological adjustments. These changes, while seemingly minor, underscore the fluid nature of machine learning research where updates to libraries can influence not only the performance but also the reproducibility of results. Despite these challenges, the inherent robustness and interpretability of the DNDTs were confirmed through our systematic and rigorous replication process. In conclusion, our reproduction effort emphasizes the significance of maintaining up-to-date documentation and code compatibility with the current software ecosystem. It highlights the need for a dynamic approach to the reproducibility of machine learning models, one that evolves alongside the tools and libraries that underpin these models.

5.1 Contributions

Karim Gorgy performed extensive research, experimentation on the datasets and analysis regarding Decision Trees and DNDTs.

Bassel Sharif handled the report documentation alongside experimental analysis on hyperparameters.

Daniel Wassef worked on the analysis of different optimizers and researched code setup and computation power aspects.

References

[1] Randall Balestriero. Neural decision trees, 2017.

[2] Yongxin Yang, Irene Garcia Morillo, and Timothy M. Hospedales. Deep neural decision trees, 2018.

Appendix

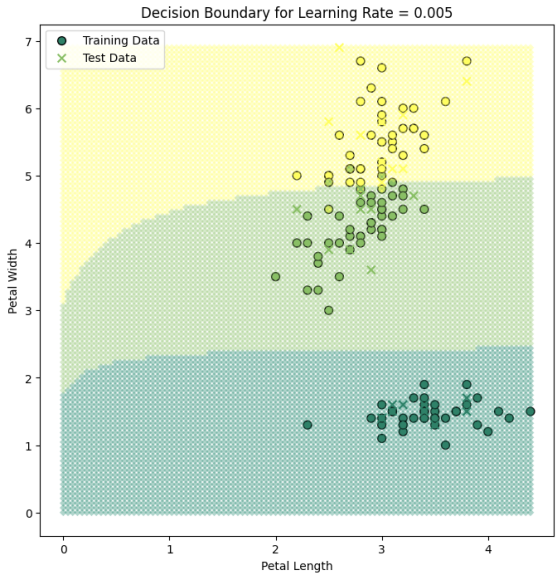

Figure 1: Decision boundary of the DNDT on the Iris dataset.

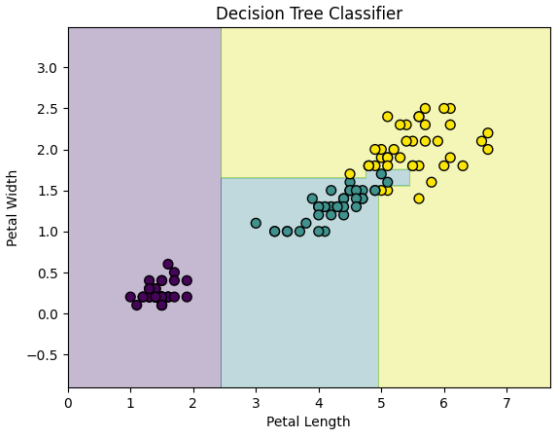

Figure 2: Decision Tree Classifier on the Iris dataset.

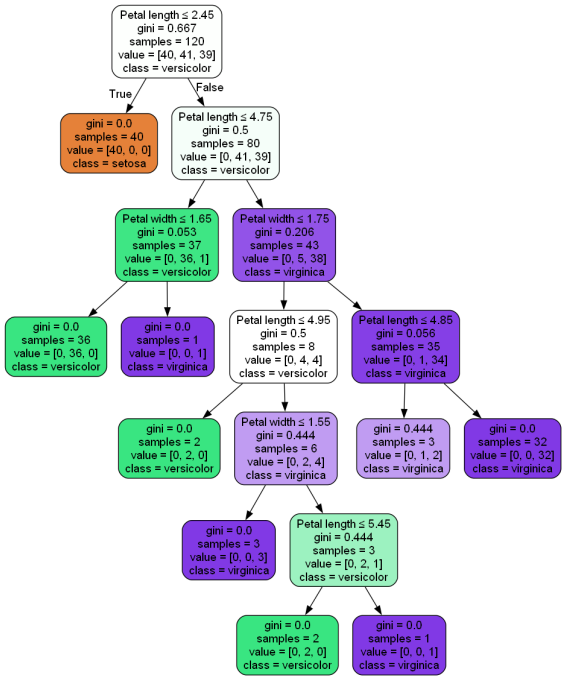

Figure 3: Iris Decision Tree

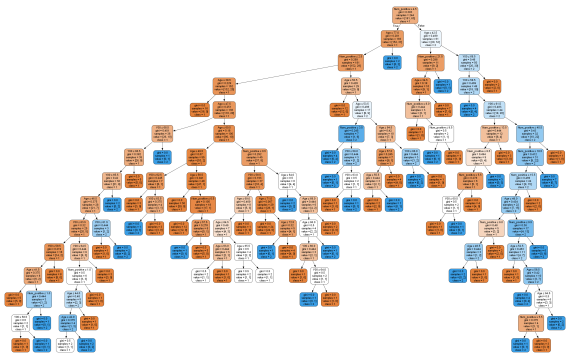

Figure 4: Haberman's Survival Decision Tree