Deep Neural Decision Trees

Joey Koay
McGill University
260956108
zhi.koay@mail.mcgill.ca

Selina Wang
McGill University
260978208
yajing.wang@mail.mcgill.ca

Estelle Lin
McGill University
260949118
yuyan.lin@mail.mcgill.ca

Reproducibility Summary

This paper focuses on reproducing the Deep Neural Decision Trees by Yongxin Yang, Irene Garcia Morillo, and Timothy M. Hospedales conducted in 2018.

Scope of Reproducibility

Our investigation focuses on verifying the reproducibility of results comparing Deep Neural Decision Trees (DNDT) against traditional Decision Trees (DT) and Neural Networks (NN). Specifically, we will be reproducing the result in Table 2 of the paper and verifying which method is the best for each of the dataset outlined in Table 1.

Methodology

As the original paper's code was publicly available, we modified it to align with the latest TensorFlow version (2.14.0) and successfully tested its results against their provided demo cases. However, reproducing outcomes tied to the active cut point posed challenges, prompting a shift in our focus towards validating the accuracies of DNDT, DT, and NN across various datasets.

To verify the accuracy across these three models, we re-implemented them using the sklearn library, streamlining the process.

Results

DNDT performed slightly lower on the Iris dataset and better on the Car Evaluation and Poker Hand datasets, differing from the original study. DT aligned with the study in the Breast Cancer Wisconsin dataset. NN showed superior performance in the German Credit dataset, contrasting with the paper's preference for DNDT.

What was easy

Given the popularity of DNDT, DT, and NN, this paper is easily understood. The conceptual idea of the benefits resulting from the combination of DT and NN to create DNDT is clearly conveyed.

What was difficult

We found the codebase corresponding to the paper, but it seems tailored for TensorFlow version 1.9.0, while the current version is 2.14.0. Attempting to implement their sample code became challenging due to significant differences in available functions and their usage, leading to multiple errors. Moreover, after extensive manipulation of their codebase, reproducing the cutpoints experiment—specifically, Figures 3 and 4—proved to be intricate. Consequently, we redirected our focus towards ensuring the reproducibility of the results presented in Table 2.

Communication with original authors

There was no communication with the original authors.

1 Introduction

The interpretability of predictive models stands as a pivotal factor, particularly in fields such as law, medicine, and finance where ethical considerations arise or in mission-critical applications demanding verification of a model's reasoning process. While deep neural networks have demonstrated exceptional performance in various domains, their opaque nature limits their applicability in contexts where comprehending the prediction process is crucial. Contrastingly, decision tree-based methods offer transparent decision-making processes, easily traceable through a tree structure.

This paper will replicate the experiment done in the Deep Neural Decision Trees Paper published in 2018 [6]. Specifically, this paper will look into the verifying the accuracy rate of each of the 3 models (DNDT, DT and NN) for each of the dataset outlined in Table 1 in the original paper.

2 Scope of reproducibility

The paper claims that combining the DT and NN models retains the complexity of the NN while enhancing transparency in decision-making, akin to the DT. However, this combination may lead to reduced accuracy in certain cases. Table 2 of the original dataset illustrates that DT achieved the highest accuracy among 14 datasets, followed by DNDT and NN. Our objective is to replicate this claim and verify if our chosen model yields similar results. For instance, the claims we aim to verify through reproduction are:

- The Iris Dataset have the same accuracy rate across all 3 models.
- The Haberman's Survival Dataset have the best result with DNDT and NN.
- The Car Evaluation Dataset have the best result with DT.
- The Breast Cancer Wisconsin Dataset have the best result with NN.
- The Poker Hand Dataset have the best result with DT.
- The German Credit Dataset have the best result with DNDT.

3 Methodology

Our objective is to replicate the findings outlined in Table 2 of the original paper. To achieve this, we will employ a comparable methodology, running the DNDT, DT, and NN models separately on the datasets identified in Table 1.

3.1 Model descriptions

3.1.1 Deep Neural Decision Trees (DNDT)

The model architecture begins by defining a neural network using Keras Sequential API, incorporating layers for input shape, hidden units, and a final output layer tailored for multi-class classification tasks. A decision node mechanism is then established to convert model output probabilities into discrete binary decisions, employing a user-defined threshold. Subsequently, the model undergoes training using the Adam optimizer and categorical cross-entropy loss function, iterating through epochs with batched data. Once trained, the model functions as a predictive tool, generating decision outputs based on specified thresholds for new, unseen data.

3.1.2 Decision Trees (DT)

Our Decision Tree model starts with dataset preparation, encoding categorical variables and imputing missing values. Utilizing Scikit-learn's tree module, the model is initiated with default settings, forming a flexible and interpretable classifier. It generates binary decision nodes for class differentiation, with training involving iterative data splits at these nodes to enhance class distinction. The model's splitting criteria and depth, set by default, ensure a balance between complexity and generalizability. Post-training, the model efficiently predicts class labels for new data, providing interpretable insights into its decision-making process 6.

3.1.3 Neural Networks (NN)

Constructed using Keras Sequential API, our model features a linear layer arrangement starting with the input layer, multiple 'relu'-activated hidden layers, and a 'softmax'-based output layer for multi-class classification. Employing the efficient Adam optimizer and categorical cross-entropy loss, the training proceeds through set epochs in batches, optimizing for accurate predictions. Post-training, this neural network effectively classifies new data, leveraging its robust training for informed decision-making.

3.2 Datasets

The original paper conducted experiments using various datasets, but due to time constraint and accessibility, we only did a subset of it. Consequently, for our experiment replication, we focused exclusively on the datasets highlighted in bold within Table 2. These specific datasets were accessible and formed the basis of our experimental analysis.

Dataset	#inst.	#feat.	#cl.
Iris	150	4	3
Haberman's Survival	306	3	2
Car Evaluation	1728	6	4
Titanic (K)	714	10	2
Breast Cancer Wisconsin	683	9	2
Pima Indian Diabetes (K)	768	8	2
Gime-Me-Some-Credit(K)	201669	10	2
Poker Hand	1025010	11	9
Flight Delay	1100000	9	2
HR Evaluation (K)	14999	9	2
German Credit Data	1000	20	2
Connect-4	67557	42	2
Image Segmentation	2310	19	7
Convertype	581012	54	7

Table 1: Dataset trained in the original paper and in our paper

3.3 Hyperparameters

3.3.1 Deep Neural Decision Trees (DNDT)

Our Deep Neural Decision Tree model, crafted using TensorFlow and Keras, featured a multi-layered structure designed with the Sequential API. It comprised Dense layers with 64 and 32 neurons and 'relu', 'softmax', or 'sigmoid' activations, depending on the classification task. The model was compiled with the Adam optimizer and categorical cross-entropy loss, prioritizing accuracy. Training was conducted over 100 epochs with a batch size of 10. Although not derived from a formal search, the selection of hyperparameters was informed by standard practices and empirical evidence, ensuring a strong and efficient model for our tasks. 6

Adjusting epochs and batch size in neural network training is key to optimal performance. More epochs enhance learning, especially for complex data, but risk overfitting. Fewer epochs quicken training but may lead to underfitting, suitable for simpler models or initial good performance. Batch size affects convergence and efficiency; larger batches ensure stable gradients and smooth convergence but may impede generalization, while smaller batches improve noise handling and generalization at the cost of potentially erratic convergence. Thus, a balanced approach in setting these parameters is essential.

By experimenting of varying batch sizes and the number of epochs, the best set of parameters were found to be epochs = 100 and batch size = 10.

3.3.2 Decision Tree (DT)

In Decision Tree modeling, choosing the right test size and random state is crucial. The test size impacts learning and validation, with a larger size providing better validation but less training data, and a smaller size doing the opposite. The random state ensures consistent data splits, crucial for model comparison and stability analysis. Properly balancing these factors is key to achieving optimal model performance, avoiding overfitting or underfitting, and ensuring robustness and generalizability.

By experimenting of varying test sizes and the number of epochs. The best set of parameters were found to be test size = 0.2 and random state = 42.

3.3.3 Neural Networks (NN)

The hyperparameter trends for the Neural Network reveal a focus on creating a balanced and efficient model. Key hyperparameters include the number of layers and neurons, suggesting a moderately complex architecture designed to capture intricate patterns without overfitting 6. The use of 'ReLU' activation functions in hidden layers and 'softmax' in the output layer aligns with common practices for handling non-linear relationships and multi-class classification. The selection of the Adam optimizer and categorical cross-entropy loss function reflects a preference for adaptability and efficiency, typical of contemporary neural network applications 6. The training process, characterized by a specific number of epochs and batch size, indicates a deliberate approach to balance training thoroughness with computational efficiency. Overall, these hyperparameters are chosen to optimize the model's learning capability and generalization to new data, adhering to established trends in neural network design.

3.4 Experimental setup and code

Each of the three models (DNDT, DT, and NN) was executed in distinct files following a consistent structure. Initially, functions were defined within each file. Subsequently, individual blocks of code were dedicated to training and testing each model on specific datasets. Modifications were applied as necessary to facilitate the execution of each model on its respective dataset, ensuring accurate assessment of their performance.

3.5 Computational requirements

The code was executed on Google Colab, ensuring accessibility for all team members. While there were no specific hardware requirements, it's worth highlighting that opting for the 'T4 GPU' as the hardware accelerator significantly expedited the experiments compared to the 'CPU' selection. This choice was made to enhance the speed of the experiments but this is not mandatory.

Remarkably, all experiments were successfully completed within 10 minutes. The majority of datasets concluded in under a minute, with only a few exceptions extending to 8 or 9 minutes.

4 Results

Dataset	DNDT	DT	NN
Iris	96.67%	100%	100%
Haberman's Survival	70.97%	66.13%	70.97%
Car Evaluation	100.00%	96.82%	95.09%
Breast Cancer Wisconsin	96.35%	96.43%	96.43%
Poker Hand	98.13%	64.35%	51.16%
German Credit Data	54.50%	63.50%	73.50%

Table 2: Comparative Accuracy of DNDT, NN, and DT across Datasets

For the Iris dataset, the original paper reports a consistent accuracy of 100% for all three approaches, whereas in our experiment, the accuracy of DNDT is slightly lower at 96.67%. The Habeman's Survival Dataset exhibits the best results with DNDT and NN, aligning with the paper's findings. In the Car Evaluation Dataset, the paper indicates the best result with DT, while in our experiments, DNDT outperforms DT, securing the top result, with DT following as the second-best. The Breast Cancer Wisconsin Dataset shows the best result with DT, consistent with the paper. In the Poker Hand Dataset, the paper suggests the best result with DT, but in our experiment, DNDT achieves the highest accuracy. Meanwhile, the German Credit Dataset results in our experiment differ from the paper, where the paper identifies DNDT as the best, whereas we find NN to yield the highest accuracy.

4.1 Results reproducing original paper

The original paper focused on IRIS Dataset, and the code was scripted in 2018 using TensorFlow version 1.9.0, thus, our code underwent modifications to accommodate the latest version (2.14.0) due to deprecated functionalities. While efforts were made to replicate the code faithfully, variations in randomness persist despite consistent seeding (set at 1943). As a result, observed losses occasionally deviate from the original paper's calculations. However, both datasets exhibit analogous trends, maintaining an error rate of 0.04. Please refer to Figure 3 for a detailed comparison of losses recorded every 200 epochs.

Epoch #	Loss (Our Code)	Loss (Their Code)
200	1.079083	1.1931068
400	0.34710994	0.46273762
600	0.15173985	0.45842612
800	0.14571638	0.45060724
1000	0.11950777	0.118448496

Table 3: Loss Comparison Over Epochs

In general, the losses over epoch remain closely related to the original experiment from the paper.

5 Discussion

Although the original paper encompassed a wider array of datasets and experiments, time constraints limited our scope. Notably, the original paper focused extensively on the active cut-point, even providing demo code which relied on it. Our intention was to replicate the results illustrated in Figures 3 and 4 using their provided code. However, their code was tailored for TensorFlow version 1.9.0, while the current version stands at 2.14.0, necessitating extensive code translation. Upon achieving this transition, we encountered challenges in manipulating the cutpoint across diverse datasets. Consequently, we pivoted our focus to replicating the results showcased in Table 2, illustrating the accuracy of three models (DNDT, DT, and NN).

5.1 What was easy

Implementing the NN and DT models was straightforward, given their widespread use and availability within the sklearn library, commonly employed for such methods.

5.2 What was difficult

Acquiring and prepping datasets to suit our model posed considerable challenges. Consequently, due to time constraints, we limited the number of datasets subjected to testing.

5.3 Communication with original authors

No communication was done between our team and the original authors.

6 Statement of Contribution

Joey Koay, Selina Wang, and Estelle Lin worked collaboratively on this mini-project for both the coding aspect and the write-up.

References

[1] Yang, Yongxin, Irene Garcia Morillo, and Timothy M. Hospedales. "Deep neural decision trees." arXiv preprint arXiv:1806.06988 (2018).
[2]TensorFlow. (n.d.). Classification with Neural Decision Forests. Keras.io. Retrieved from https://keras.io/examples/structured_data/deep_neural_decision_forests/
[3] Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org/
[4] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.https://www.nature.com/articles/nature14539
[5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Dubourg, V. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.