Mini-Project 4:

Reproduction of **Deep Neural Decision Trees**
by Yongxin Yang, Irene Garcia Morillo
and Timothy M. Hospedales

Lily Gostovic - 260958505
Mohammad Abdullah - 260980866
Justin Novick - 260965106

A report describing the findings of
Mini-Project 4 of COMP551

COMP551: Applied Machine Learning
McGill University
Montreal, Canada
December 7, 2023

1 Abstract

This paper reproduces the results of the paper (Yang, Morillo, and Hospedales 2018) in which a new model is created to bridge the gap between a conventional Neural Network and Decision Trees. We take the natural interpretability of tree based models and combine it with the deep learning techniques of training a neural network to develop a model that can take full advantage of modern GPU units to efficiently train the model using gradient descent. The model is designed for maximum interpretability for tabular data, data that conventional decision trees thrive on. The *neural* aspect for this model comes through utilizing back-propogation instead of the usual structural learning that use techniques like splitting and score matrices. Our main experiments included comparing the performance of the Deep Neural Decision Tree model (DNDT) with Nerual Networks (NN) and Decision Trees (DT). Due to resource constraints, we were unable to replicate the experiments on all 14 datasets that the paper used and instead used a subset of 5 datasets described below. The performance of different models was measured and compared on them, since each dataset has slightly different features, trends, and distributions. We further performed ablation studies on the main hyper parameter of the DNDT which is the number of cut-points assigned to each feature when training a model on it. Another set of experiments was performed on how DNDTs are able to perform "self-regularization" by systematically utilizing only the appropriate number of cut points needed. Due to the unavailibility of advanced computing units (`TITAN GPUS`) we were unable to perform time performance analysis for DNDTs which the latter half of the original paper focused on.

2 Introduction

This paper reproduces the findings of the Deep Neural Decision Trees paper (Yang, Morillo, and Hospedales 2018) and discusses challenges and compares results of doing so. The original paper defines a new machine learning model that is named a Deep Neural Decision Tree which is a combination of a Decision Tree and a Neural Network. The motivation for the new type of model came from the goal to create a machine learning model that has the excellent performance of a neural network, while possessing the interpretability of a decision tree. The original paper discusses that although neural networks have great performance, their use is often limited due to them being "black-box" models, meaning their predictions cannot be interpreted. This lack of interpretability prevents them from being used in medical or legal cases, or any case where the exact steps of how the result was achieved is crucial to be parsable. The implementation of Deep Neural Decision Trees found in (wOOL 2016) attempts to solve this exact problem.

The purpose of this paper was to implement the Deep Neural Decision Tree machine learning model found in (wOOL 2016) and to reproduce a subset of the experiments found in the original paper. In this paper, we define DNDT models (using the implementation from the original paper), DT models (implemented using sklearn), and NN models (implemented with tensor flow) to predict data from five datasets. We compare the accuracies achieved for each model on each dataset with the corresponding reported accuracies from the original paper. We found that we were able to achieve somewhat similar accuracies to those reported in the paper, but that they were still slightly off no matter what we tried. We attributed this to the fact that there were some specifics that were omitted from the paper regarding the exact details of their implementation. Next, we investigated the effects of tuning the hyper-parameters of the DNDT models on the model accuracies. The only hyper-parameter available for tuning is the number of cuts so we focused on this. We compared the trends in accuracies of each dataset when increasing the number of cut points from 1 to 10.

These results were significantly different from those reported in the original paper. These drastic differences were attributed to the changes that have been made to tensor flow's neural network model and the variation between consecutive runs due to multiple random variables effecting the output. Finally, we examined the number of active cut points of each DNDT model. We found that as the number of cut points increased, the percentage of active cut points steadily decreased. This is on par with what was reported in the original report.

Some challenges of the reproductions discussed in this paper included determining how to pre-process the data and the tuning of the hyper-parameters of the model. The lack of these details in the original paper forced us to make decisions on how to pre-process the data and left us tuning the hyper-parameters ourselves. Due to this lack of crucial information, the results we reproduced were not exactly the same as those presented in the original paper.

Overall, our findings were very close to the findings of the original paper and followed the same trends in the data even though the numbers may not have been exactly the same for each.

3 Datasets

The following 14 datasets are used in the original paper

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
Covertype	581012	54	7

This report discusses the results of the following five datasets: the Haberman's Survival Dataset, the Titanic Dataset, the Pima Indian Diabetes Dataset, the Iris Dataset, and the Car Evaluation Dataset. The five datasets are briefly introduced below.

The **Haberman's Survival dataset** (Haberman 1999) comprises features such as age, year of surgery, and the number of positive axillary nodes, with the target variable indicating whether a patient survived for 5 years or more. The dataset is relatively small in size, has a potential lack of comprehensive patient information, and has an absence of detailed cancer characteristics, all of which can make working with this dataset difficult.

The **Titanic dataset** (Cukierski 2012) contains data points representing individuals who boarded the Titanic. Features include passenger class, gender, age, sibling/spouse count, parent/child count, ticket number, fare, and port of embarkment. The binary target variable denotes survival status, 1 for survived, 0 for perished. Challenges in working with this dataset include addressing missing values, particularly in the 'Age' column, and mitigating class imbalances. Rec-

ognizing the socio-economic implications inherent in passenger class and demographics adds contextual depth to survival dynamics.

The **Pima Indian Diabetes dataset** (Kahn n.d.) contains critical health indicators including pregnancies, glucose levels, blood pressure, skinfold thickness, insulin levels, BMI, a diabetes pedigree function, and age. The binary target variable represents the presence (1) or absence (0) of diabetes. Challenges in working with this dataset involve managing missing values, especially in features like insulin and skin thickness.

The **Iris dataset** (Fisher 1988) consists of botanical measurements for iris flowers. It has four features: sepal length, sepal width, petal length, and petal width. The target variable is a classification of iris species into Setosa, Versicolor, and Virginica. This dataset is known for its cleanliness, the dataset requires minimal pre-processing, yet optimal feature selection and model tuning contribute to accurate analyses.

The **Car Evaluation dataset** (Bohanec 1997) contains evaluations of varying car attributes. The dataset features include factors such as buying price, maintenance cost, number of doors, number of passengers, luggage capacity, and safety rating. The target variable is the classification of cars into categories ranging from "unacceptable" to "very good". Similar to the Iris dataset, the car evaluation dataset is notable for its relative cleanliness, and requires minimal pre-processing.

Note that the original paper discussed the results on 14 datasets but that we chose to use this subset of 5 since we did not have enough resources to compute all 14 and the five selected are those focused on in the experiments of the original paper.

4 Results

Through replication of the experiments and the implementation of the DNDT model, most of the original results were realized which are illustrated in the following sections and tables. Through tables 1 and 2, it is observed that most of the values obtained are indeed reproducible, with some slight deviations which could arise due to some software differences, including but not limited to different library (tensorflow) versions. Neural Networks performed remarkably better than the original results with substantial increase in accuracy across all 4 datasets. However, despite this anomaly, trends were approximately consistent across the experiments with respect to the original paper, with DNDT performing well on the Haberman dataset and Decision Trees working the best on the Pima dataset. The main deviation in our data from the original paper is the Neural Network's stark increase in performance on the Car Evaluation dataset. This would be an excellent point to further investigate in additional papers.

Dataset	Car Evaluation	Pima Indian Diabetes	Iris	Haberman's Survival	Titanic
NN	91.6	64.9	**100.0**	**70.9**	76.9
DT	**96.5**	**74.7**	**100.0**	66.1	79.0
DNDT	95.1	66.9	**100.0**	**70.9**	**80.4**

Table 1: Percent Accuracy from Original

4.1 Hyper-parameter Tuning and Ablation Studies

The paper states that the only hyper-parameter of a DNDT is the number of cut points. Therefore, we investigated the effects of changing the number of cut points to the accuracy of each model on each dataset.

Dataset	Car Evaluation	Pima Indian Diabetes	Iris	Haberman's Survival	Titanic
NN	**100.0**	74.0	**100.0**	71.0	**80.9**
DT	95.7	**74.7**	**100.0**	66.1	79.2
DNDT	92.4	65.6	**100.0**	**78.1**	75.3

Table 2: Percent Accuracy Reproduced

Figures 1-4 show how the accuracy of a DNDT model changes as the number of cut points change. The respective graphs from the original paper are included in the appendix. It can be

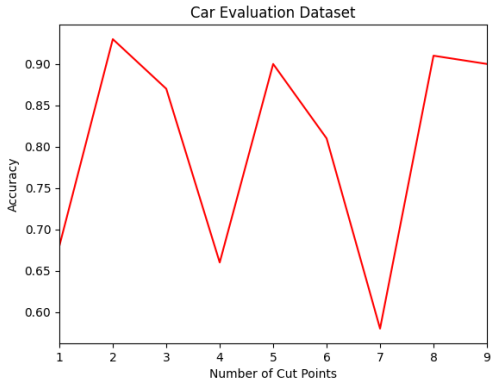

Figure 1: DNDT on Car Evaluation Dataset

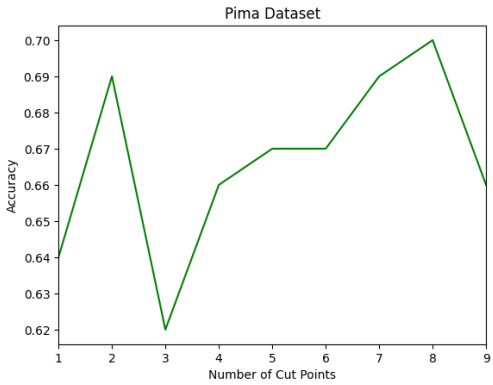

Figure 2: DNDT on Pima Dataset

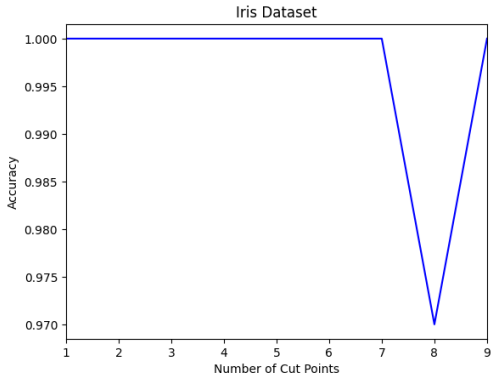

Figure 3: DNDT on Iris Dataset

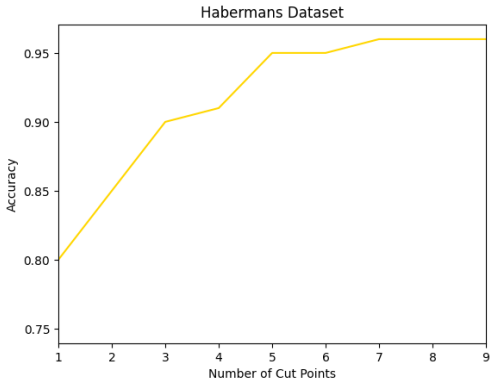

Figure 4: DNDT on Haberman's Dataset

seen that there are quite a few differences in the two sets of graphs. First, the original paper's graphs all experience a plateaued performance after a certain number of cut points, while only the Haberman's dataset begins to experience a plateau in our implementations.

It is possible that this could be the result of the changes that have been made to tensorflow in the past 5 years since the original paper was published. Since the performance of the DNDT is dependent on tensorflow's implementation of neural networks, any changes to the tensorflow neural network model will cause changes in the performance of the DNDT. Therefore, the changes made to the tensorflow neural network that have been made in the past 5 years are a possible causation

of the differences in these graphs.

4.2 Results Comparison

The accuracies of the different models on each dataset can be seen in Table 1 and Table 2. The second experiment we chose to replicate was the analysis of active cut-points. These points describe how the "splits" are created in the `DNDTs` and is one the main complexity parameter for the model. Its important to note that the values of these cut points are not bound and some of them may continue to be inactive. This leads to an interesting consequence we dub "Self Regularization". Despite increasing the complexity of the model, we notice that the percentage of the number of active cut points decrease in proportion. We see the results of this emulated in the original paper as well, with the conclusion that a `DNDT` does not utilize all the parameters available to it, unlike a Neural Network. We saw how performance changed by manipulating the number of cut points available to each feature in the corresponding data set above. In the images that follow below, we observe how the usage of the cut points varies as we increase the complexity of the model by incrementing the number of cut points. We can clearly see how the % use decreases in proportion to the increase in the number of cut points.

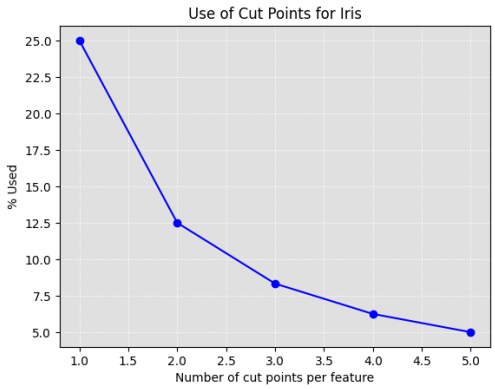

Fig 9: Iris Cut-Point Analysis

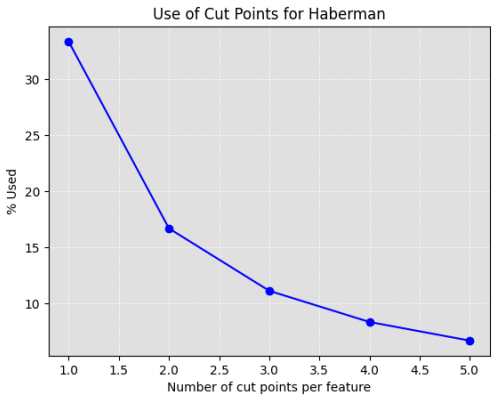

Fig 10: Haberman Cut-Point Analysis

5 Reproduction Notes

There was some missing information that was inferred in order to complete the reproductions. These missing details include many specifics regarding the manner in which the data was pre-processed, and details on the hyper-parameters used for the DNDT implementation.

First, there were many details about the pre-processing methods used that were missing from the original paper. Some datasets had lots of missing data and it was unclear in the paper how this was dealt with. It was unclear whether features with missing data points were removed altogether or if there was a default value that was set for points missing that data point. In all cases, we chose to remove the feature altogether if there were missing data points to ensure equal training to all data points. This is one factor that might have influenced the difference in our results from those presented in the paper.

It was also not disclosed what percentage of the datasets were used for training and what percentage were used for testing. Changing these percentages can drastically change the performance

of a model, so the lack of this detail presented us with another challenge on reproducing the results exactly. For future reproducibility, note that we used a training/test set split of 80/20 for all datasets. We used sklearn's `train_test_split` function with a random state of 42 to divide the datasets into training and test sets.

There were also details missing regarding the hyper-parameters used for the DNDT implementation. For example, throughout the paper it is mentioned that you can use tensor flow or pytorch implementations of neural networks for the neural networks in the DNDT implementation, but it is unclear which of the two was used to calculate the results presented in the paper. We experimented with using both implementations to see how the results would change. In the end, we chose to use the tensor flow neural network implementation since it gave closer results to what the paper reported.

Including random seeds that were used for numpy and tensor flow would greatly increase the ease of reproducing the results to obtain exactly consistent values. For future reproducibility, we used a random seed of 1943 for numpy and tensor flow.

The cumulation of all these missing details contributed to our results being slightly off from those reported in the paper. In future reports, it would be very helpful to include these details to avoid so much inference in important decisions when trying to reproduce the paper.

6 Challenges

A large challenge we faced was interpreting the code provided for the Deep Neural Decision Trees. There is a demo file provided which shows how to implement a DNDT on the Iris dataset which was very useful in implementing our own instance of the DNDT. However, the code provided hard codes a lot of values which makes it very difficult to interpret where exactly these hard coded values came from. A lot of time was spent in the beginning of this project parsing the code that was published to understand what exactly each line was doing to be able to manipulate the values and extend it to other datasets. A large challenge was adapting the DNDT implementation to classify datasets with a different number of features than the provided example.

We approached this problem by playing around with the provided code and seeing how our changes would adjust the output. Eventually, we fully understood each line of code in the demo file and were able to write implementations of DNDTs to model the four other datasets, all with varying types of data and numbers of features.

7 Key Takeaways and Future Investigation

The findings of this paper have suggested that the DNDT is a more interpretable structure than a traditional Neural Network, without sacrificing performance. The interpretability, arising from the tree-like architecture, can give more information as to what features are influencing the decisions being made concerning classification. In the DNDT discussed in this paper, NN architecture was understood to be standard, given that there was only tabular data. The tabular nature of the data also explained why the regular decision trees were able to perform very well on the datasets. However, if the data sets were representing images, instead, the tree structures may fail to produce meaningful classifications. Therefore, it is worth investigating if CNNs could be used as a base, with a tree-structured implementation for interpretability. Complex CNNs are often thought to be black boxes in the way they capture the significance of objects, edges, and other visual characteristics. If when working with CNN models there was a way to discern which neurons were at play in abstract

recognition, quality architectural choices could be made much more efficiently in finetuning to a dataset.

8 Statement of Contributions

All group members contributed equally on coding and writing the report.

References

Bohanec, Marko (1997). *Car Evaluation*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5JP4

Cukierski, Will (2012). *Titanic - Machine Learning from Disaster*. URL: https://kaggle.com/competitions/titanic.

Fisher, R. A. (1988). *Iris*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56C76.

Haberman, S. (1999). *Haberman's Survival*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5X

Kahn, Michael (n.d.). *Diabetes*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5T59G.

wOOL (2016). *DNDT*. https://github.com/wOOL/DNDT.

Yang, Yongxin, Irene Garcia Morillo, and Timothy M. Hospedales (2018). "Deep Neural Decision Trees". In: *CoRR* abs/1806.06988. arXiv: 1806.06988. URL: http://arxiv.org/abs/1806.06988.

Appendix

Figures 5-8: Number of cut points effect on DNDT performance from original paper

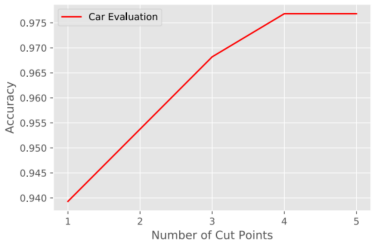

Figure 5: DNDT on Car Evaluation Dataset

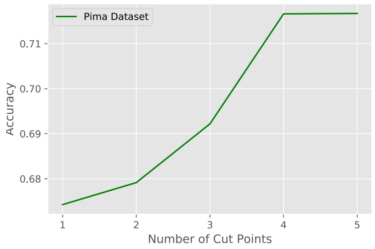

Figure 6: DNDT on Pima Dataset

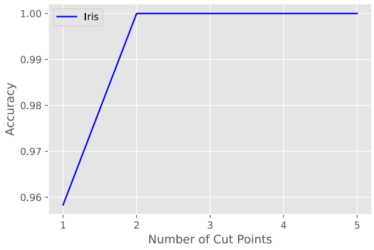

Figure 7: DNDT on Iris Dataset

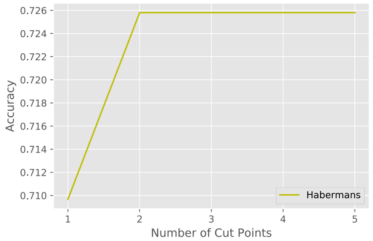

Figure 8: DNDT on Haberman's Dataset

Figure 11: Percentage (%) of active cut points used by `DNDT`.