Advanced Decision Making and Interpretability through Neural Shrubs

Abstract

Advanced decision making using machine learning should be both accurate and interpretable. Many standard machine learning techniques suffer from an inherent lack of transparency with regard to how the resulting decision was made. In the current work we aim to overcome this issue by introducing a hybrid learning approach using classical decision trees alongside artificial neural networks, dubbed a "*neural shrub*". The Neural Shrub methodology presented in this paper aims to maintain as much interpretability as possible without sacrificing either classification or regression accuracy. Experimental results are presented on several benchmark data sets to validate the proposed approach as well as provide insight into future research directions.

Keywords: Interpretable AI/ML, Decision trees, Neural networks

1 Introduction

Artificial intelligence (AI) and *neural networks* has been successfully applied to many fields. With the increase in the data volume and computation power, artificial intelligence focuses on making models to extract valuable information from large data to predict future behavior. Some of the popular applications of artificial intelligence and neural networks are seen in pattern recognition, system modeling and identification, signal processing, image processing, and stock market predictions to name but a few [1].

However, the advancement of artificial intelligence and neural networks has recently come under scrutiny, especially from the statistical community, due to lack of tools available to determine how or why a particular decision is being made [2]. Although powerful and highly effective at predicting behavior in non-linear data, lawyers and business practitioners are reluctant to use these tools because it is nearly impossible to explain the results and decisions made by these technologies [3]. In addition to their inherent lack of interpretability, quantifying the predictive capabilities and validity of such models often poses a significant challenge [4]. Therefore, this paper aims to answer this fundamental research question in artificial intelligence.

In attempt to provide more transparency of AI models this paper proposes a hybrid method that combines neural networks with binary decision trees. The advantage of our approach is that we maintain the interpretabilty of our predictions by partitioning the data space with a binary decision tree without sacrificing the accuracy of a standard feed forward neural network.

In the experiments section of this paper it will be shown that our approach improves the binary decision tree but is slightly less accurate than a standard neural network. We make the case that the small decrease in accuracy is well worth the benefit of having a partitioned data space which increases the interpretability of the results.

The remainder of this paper is organized as follows. We will discuss about binary decision trees and neural networks in the background. We will look into limits of some past works on hybrid methods similar to Neural Shrubs in related works. In methodology, will outline the neural shrubs algorithm, which will be followed by a comparison of its performance against a standard binary decision tree and neural a network. Finally, we will conclude by providing some interpretative remarks and future directions.

2 Background

2.1 Decision Trees

A *binary decision tree* organizes the available data by performing data partitions in response to simple yes/no questions based on predictor variables. They trace their initial roots back to social scientists in the early 60's [5]. With the development of the Automatic Interaction Detection (AID) program by Morgan and Sonquist [6], binary decision trees started to become a popular tool for organizing and interpreting data. Leo Breiman and Jerome Friedman began working on trees independently in the early 70's and then teamed up with Stone and Olshen to publish what most would consider the authoritative work on the subject, "Classification and Regression Trees (CART)" [7]. Since the 80's CART has become one of the more popular machine learning tools.

Tree based models or binary decision trees are generally used to solve two basic types of problems: (1) *Classification*, which has a categorical response variable, and (2) *Regression*, which involves a continuous response variable.

The CART algorithm is a greedy algorithm, meaning at each step it determines the partition that minimizes the impurity. For classification trees, this usually implies that it selects the partition that has the minimum number of misclassified cases. For regression trees, this usually implies that the selected partition has the smallest sum of squares of the target variable [8]. Trees are generally over grown by choosing locally optimum splits and then pruned back to the optimal depth using cross-validation in an effort to prevent over fitting.

To illustrate the interpretability of a decision tree, we provide an elementary example by considering the problem of predicting survivors for those who embarked on the titanic. The general idea here is that, based on factors such as sex, passenger class, sibling counts, etc., can we deduce information from these factors and use them for prediction? In other words, can we determine which factors (i.e. attributes) are most likely to lead to the passenger surviving the disaster [9]. Using the factors available in the dataset, as well as the known categorical response (survived/died), we used the standard CART methodology described above to build the tree illustrated in Figure 1. Reading the tree in a top down manner, we see that the most important factor is the sex of the passenger. From the root (top) node in the tree we see that 59% of the females survived as opposed to 41% of males. As you work your way down the tree you can see which other factors contribute to passenger survivability.

The decision tree partitions the data into two distinct classes (1 = survived and 0 = died) at each of the leaf nodes (i.e., the terminal nodes of the tree). Each branch of the tree is a decision that the tree makes to arrive to the value in the leaf node. Using some of the other factors, we see that the best chance of survival occurs when the passenger was a female and not a third class passenger. The far right leaf node in Figure 1 shows that 22% of the data consisted of females who were not 3rd class passengers. For those passengers, 93% survived and only 7% died.

As this example illustrates, binary decision trees are exceedingly useful because interpretation is very straight forward. In addition, for a given data set, the topology of the tree will always be the same regardless of how many times the tree is built (i.e., the build process is repeatable). The disadvantage of decision trees is that other machine learning techniques (e.g. Neural Networks,

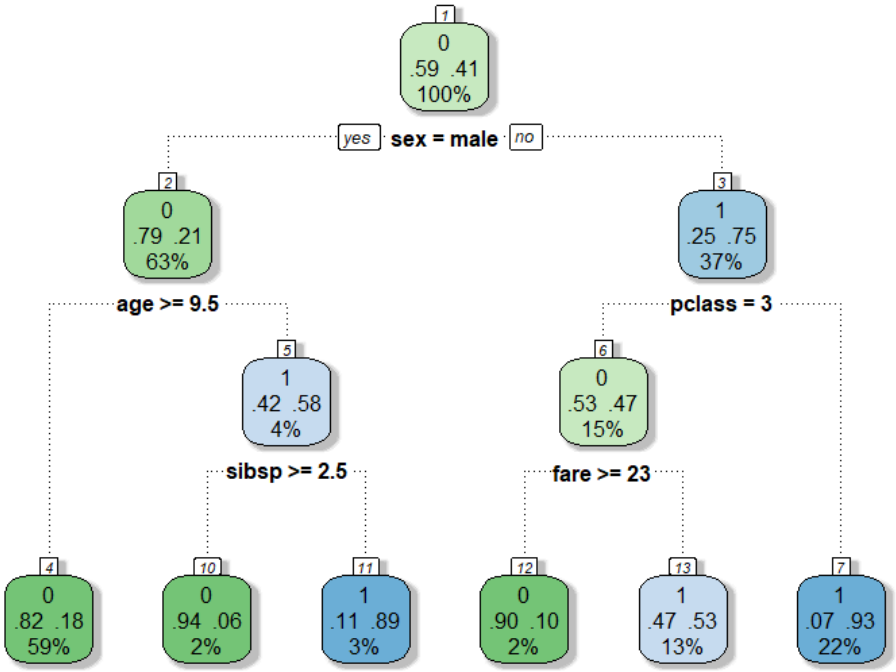

Figure 1: Example of a decision tree using the CART methodology for the Titanic dataset illustrating the simplicity of interpretable results. From the top (root node) we see that the most important predictor for survival is the sex of the passenger followed by either age (if you're a female) and passenger class (if you're a male).

Support Vector Machines) often outperform trees in terms of accuracy on independent data. As illustrated in Subsection 2.2, a relatively simple artificial neural networks (ANN) can achieve close to 84% prediction accuracy whereas the accuracy of the tree depicted in Figure 1 is closer to 80%.

2.2 Neural Networks

Although the idea of neural networks, also known as *Artificial Neural Networks* (ANNs) dates back to the late 1800's, the first computational model can be attributed to McCulloch and Pitts in 1943 [10]. However the research into computational neural networks stagnated when Minsky and Seymour presented two fundamental issues, 1) a single layer neural network could not model the exclusive-OR circuit and 2) the computational burden required for complex networks was unfeasible at the time [11]. Fast forward three decades, and due to increased computational power, and new application areas, neural networks have seen a major comeback.

ANNs have received considerable attention over the last several decades due to their ability to learn nonlinear mappings from one space to another [12–15]. While predominantly developed for their powerful classification capabilities, neural networks have branched out to solve both classification and regression problems, as well as a host of other problems in data science (e.g., time-series prediction, image/video annotation, dimensionality reduction, and more recently generation of new data) [16–24].

Neural networks represent a very powerful class of machine learning tools. As the name implies, ANNs are inspired from the brain and the nervous system. Like our brain, the neural network has the ability to *learn* and *adapt*. The basic building blocks of neural networks is a *neuron*. A neuron is a processing unit of a neural network. Depending on the type of connections between neurons, there are two main categories of network architectures, *feed-forward neural network* and *recurrent neural network*. If there is no *feedback* from the outputs of the neurons towards the inputs throughout the network, then the network is referred as a feed-forward neural network. Otherwise, it is called a recurrent neural network [1]. In this paper, we will focus on feed-forward neural networks.

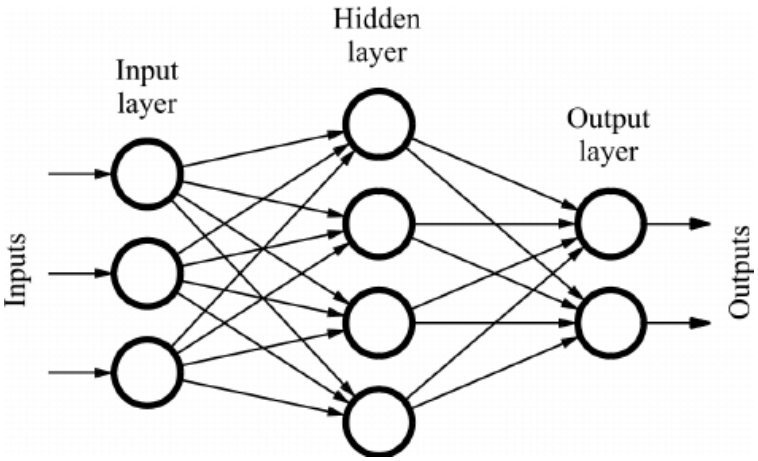

Figure 2: A multi-layer feed-forward artificial neural network consists of an input layer, which receives data; the hidden layer and output layer does the computation; and the result is transferred to the outside world through the output layer.

The structure of a feed-forward neural network is shown in Figure 2. It consists of an *input layer*, which receives the data. This layer does not do any computation. The data is processed in the *hidden layer* and *output layer* and it is transferred to the outside world through the output layer. *Activation functions* decide whether the neuron should be activated or not. This helps in adding non-linearity into the output [1].

As mentioned previously, neural networks have come under scrutiny because it is nearly impossible to determine how or why a particular decision is being made [2]. An example of this is illustrated in Figure 3, where one ANN topology might decide to separate class boundaries by artificially imposing a torus around the data sets (middle figure), whereas another topology may simply map the data to an entirely new space to simplify the class separation process using a line (right figure). Unfortunately, the ability to easily interpret which of these two approaches is being implemented is difficult if not impossible to determine in practice. Therefore, neural networks have been commonly referred to as “black box”.

If we consider the same dataset outlined in Subsection 2.1 where the goal is to predict survivors who embarked on the Titanic journey, a basic feed-forward ANN is both time consuming and tedious to interpret. For an input layer with 15 neurons, a single hidden layer with 14 neurons (both using rectified linear unit activation with external bias), and a single output layer using a soft-max activation, the network needs to learn 359 different parameters in order to achieve 84% prediction accuracy. If we compare this to the results obtained in Subsection 2.1 for the classical decision tree, only 5 parameters were required to achieve 80% classification accuracy (sex, age, passenger

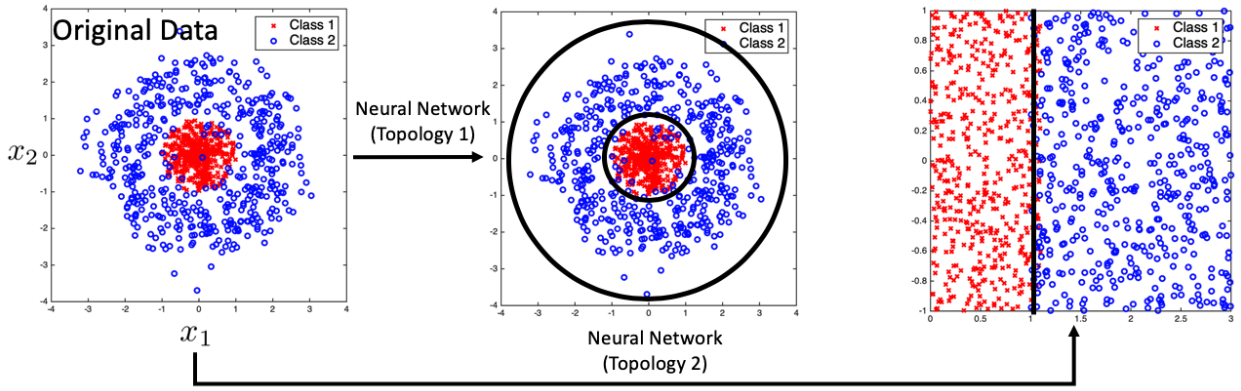

Figure 3: Graphical example of two potential ANN topologies creating drastically different results for the same two class data (left). The first ANN topology attempts to separate class boundaries by artificially imposing a torus around the data sets (middle), whereas another topology simply maps the data to an entirely new space to simplify the class separation process (right). Unfortunately, the ability to easily interpret which of these two approaches is being implemented is difficult if not impossible to determine in practice.

class, number of siblings, and passenger fare). Furthermore, while the tree is a deterministic process (i.e., the same tree will always result), the ANN can change during each training cycle due to gradient decent stagnation and/or convergence toward local minima. In addition, different ANN topology and hyper-parameters may produce drastically different prediction results and training rates.

3 Related Works

Substantial research has been published that combines decision trees and ANNs. The bulk of this research involves using ANNs to design a decision tree [25–28]. All of these methods have been shown to improve the decision tree and aid in interpretability. Others have used a ANNs to improve the splitting in a decision tree. Frosst *et al.* [29] use a distilling neural network to build what they call a “soft" decision tree. In a soft decision tree, the ANN is used to determine the splits. Bul and Kontschieder [29,30] use an ANN to determine splits for a random forest. Other seminal work include using a random forest to do prediction after designing an ANN [31] and Chakraborty *et al.* [32] use a decision tree to determine the salient features that should be used when designing an ANN.

Finally, in [33] the authors show that adding an ANN to the leaf nodes of a decision tree improves water state forecasts in river basins during typhoon events. The authors use the tree to partition the data space prior to attaching either a multi-layer perceptron (MLP) or radial basis network (RBN) to each leaf node of the tree in an effort to improve the overall forecasts.

The current research investigates a similar structure for both multi-class classification and regression problems by capitalizing on a hybrid methodology. However, rather than simply attaching an ANN to each leaf node of the decision tree, we illustrate that topologically similar data in the high dimensional data space can be grouped using the neural shrub architecture for improved classification/regression accuracy. We refer to this overall architecture as a neural shrubs because, not unlike the shrub (i.e. bush), it differs from a tree in that it has multiple stems and is generally shorter in height.

4 Methodology

All of these hybrid techniques are fundamentally different from the method proposed in this paper. The Neural Shrubs methodology is a 2-step process. First, we create a standard decision tree using the traditional CART methodology as described in Section 2.1 to both pre-partition the data space, and determine which predictor variables are the most useful [7]. Next, we find the terminal nodes that contain topologically similar data. In other words, those terminal nodes that result in the same class prediction. According to [7], this means the class that has the highest posterior probability. Using this data, we build ANNs.

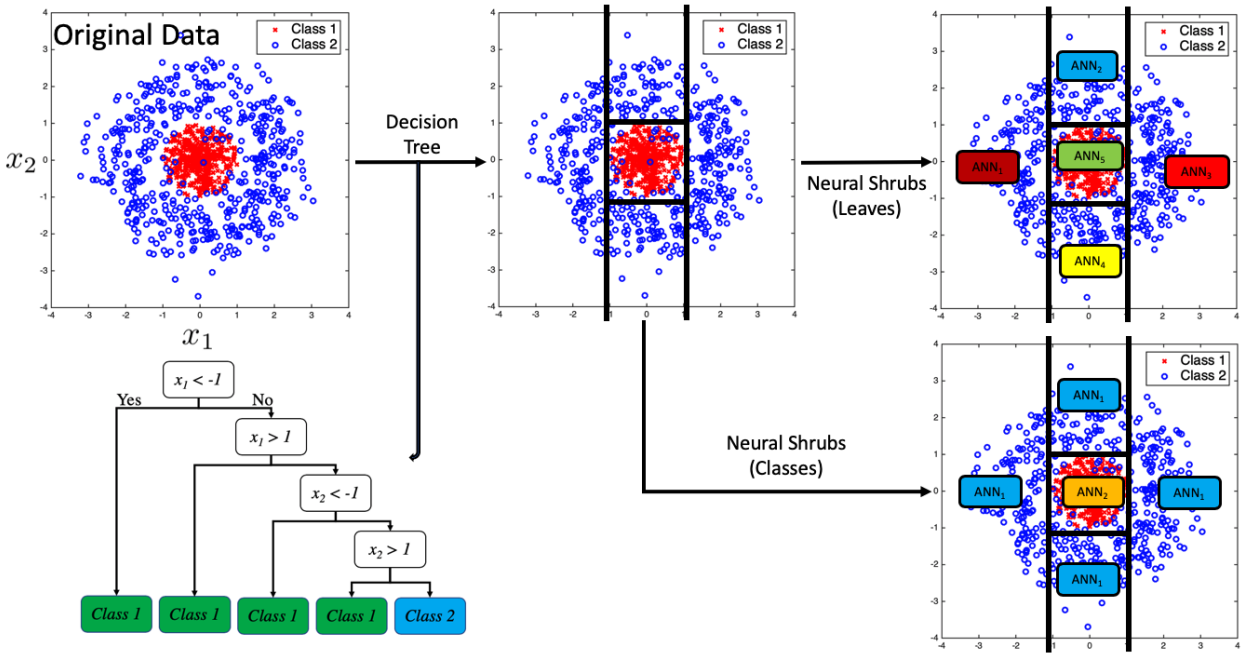

Figure 4: Graphical example of the neural shrubs approach outlined in the current paper applied to topologically similar data. The original data is shown in the upper left figure. Classical data space partitioning using CART is illustrated in the top middle figure with the resulting decision tree in the bottom left figure. The method proposed by [33] (applying an ANN to each leaf node) is shown in the upper right figure, and our neural shrubs approach (applying an ANN to topologically similar data) is illustrated in the bottom right figure.

This concept is best illustrated through a practical example as depicted in Figure 4. In this figure, the original data contains two classes (red x's and blue o's) in the shape of a torus (top left). A classical decision tree using the CART algorithm described above would simply partition this particular data-space into five distinct regions based on the two predictor variables x_1 and x_2 (top middle) to produce the tree structure shown in the bottom left. From this partitioning, the approach outlined in [33] would simply train an ANN on each of the five leaf nodes for subsequent classification or regression (illustrated in the top right figure). However, in this particular example, we see that even though the resulting decision tree contains five leaf nodes, only one of those five would "predict" class 2 whereas the other four would "predict" class 1, i.e., the additional four leaf nodes contain topologically similar data in the data space. Because this information can be determined *a priori*, the current work proposes grouping such topologically similar data into a single data set and training an ANN on this data set only as illustrated in the bottom right figure.

Unlike the data space pre-partitioning a decision tree performs as outlined in Figure 4, a single ANN could be used for classification and/or regressions. The advantage is that the ANN has the ability to *learn* nonlinear boundary conditions in an attempt to perform class separation in the data space. The drawback of course is the way in which the ANN *learns* to do this partitioning is unpredictable, uninterpretable, and can change based on network topology (and/or change of hyper-parameters).

The neural shrubs method proposed in the current work aims to capitalize on both the decision tree and ANN with the goals of increased accuracy while maintaining interpretability. The advantage of our proposed neural shrub approach is that the data set is still pre-partitioned by a decision tree, so that most of the interpretatbility is maintained while the overall accuracy of either classification and/or regression is generally improved.

In addition, the level of interpretability is completely controllable in that the user can decide how deep to grow the tree prior to pruning and applying the ANN architecture. The neural shrub approach has the additional advantage of being tune-able so as to allow the user to trade-off accuracy for interpretability.

Indeed, for applications where accuracy is more important than interpretability the user should build a more shallow tree which increases the contribution of the ANN. Whereas, for applications where interpretation is paramount, the user should increase the overall depth of the tree thereby decreasing the contribution of the ANN. One can envision that as the tree gets smaller and smaller, both the accuracy and interpretability approach that of an ANN. However, as the tree gets larger the accuracy and interpretability approach that of a decision tree.

5 Experiments

5.1 Datasets

To compare neural shrubs to ANNs and decision trees, we use an ensemble of notable benchmark data sets commonly used for testing. These test sets vary in size (i.e. number of instances) and number of predictor variables (i.e. attributes). Table 2 summarizes these data sets. For the experiments, we chose three classification datasets and one regression dataset.

Table 1: Testing Data

Data Set	Training	Testing	Attributes	Type
Connect 4	67,557	NA	126	Classification
MNIST	60,000	10,000	780	Classification
SensIT	78,823	19,705	50	Classification
YearPredictionMSD	463,715	51,630	90	Regression

Each of the classification sets aim to separate the data in to several distinct classes. For the Connect-4 data, the goal is to classify an intermediate board condition that will eventually lead to either a win, a loss or a draw (three classes). For the MNIST dataset, the goal is to classify handwritten digits (0-9). The SensIT data uses 50 attributes to classify three different types of military vehicles. The only regression dataset (YearPredictionMSD) tries to determine the year a song was released based on 90 different attributes. For additional details regarding these data sets, the reader is referred to [34].

5.2 Models

For each dataset, we build four separate models: decision tree, neural network, neural shrub on leaves described in [33], and neural shrub on classes using the training dataset. We then use the performance metrics outlined in Section 5.3 on each model to evaluate either the classification accuracy (for classification problems) or the mean absolute error (for regression problems).

Table 2: Decision tree depths for testing data

Data Set	Depth
Connect 4	12
MNIST	10
SensIT	5
YearPredictionMSD	13

Table 3: Neural Network Architecture

Dataset	Layer	Number of Neurons	Activation Function
Connect-4	Input Layer	42	-
	Hidden Layer 1	8	relu
	Ouput Layer	3	softmax
MNIST	Input Layer	784	-
	Hidden Layer 1	512	relu
	Hidden Layer 2	512	relu
	Ouput Layer	10	softmax
SensIT	Input Layer	49	-
	Hidden Layer 1	30	relu
	Hidden Layer 2	10	relu
	Ouput Layer	3	softmax
YearPredictionMSD	Input Layer	90	-
	Hidden Layer 1	90	relu
	Hidden Layer 2	90	relu
	Hidden Layer 3	90	relu
	Hidden Layer 4	90	relu
	Ouput Layer	1	linear

The decision tree is built using the CART methodology described in Section 2.1. The optimal depths calculated for the selected datasets are shown in Table 2. Feed-forward neural networks were made for each dataset as shown in Table 3. For a fair comparison, the same architectures are used to train the decision tree and neural networks for the neural shrubs on leaves and classes. Neural shrubs on leaves are built by attaching the architecturally similar neural network on each leaf node, while neural shrubs on classes are built by combining the leaf nodes of the decision tree which are topologically similar, and training architecturally similar neural network on the new groups of data.

Since regression models do not have classes, we use only one neural shrubs method (leaves). Separate ANNs are affixed to every leaf node since there is not a reasonable way to combine leaf nodes in the regression setting.

5.3 Performance Evaluation

After training the models, we pass the test dataset through each model and analyse the performance of neural shrubs against other models. If the goal is classification, we use the model and the test data to build a *confusion matrix* which groups the data into four categories as shown in Figure 5. Some popular confusion matrix terminologies are described below:

- *Positive*: the observation belongs to the selected class.
- *Negative*: the observation does not belong to the selected class.

- *True Positive*: the number of observations that are positive, and predicted as positive.
- *False Negative*: the number of observations that are positive, but predicted as negative.
- *True Negative*: the number of observations that are negative, and predicted as negative.
- *False Positive*: the number of observations that are negative, but predicted as positive.

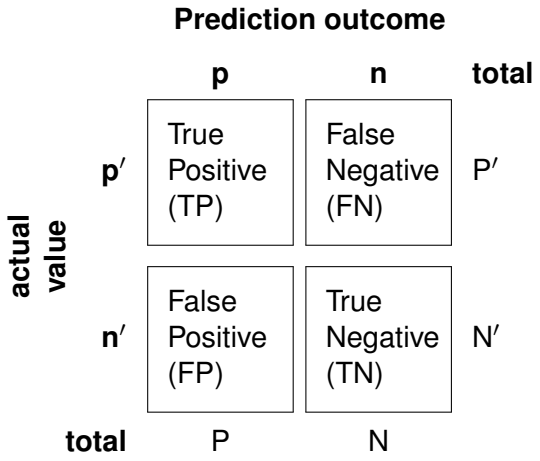

Figure 5: A Confusion matrix groups the predictions for a classification problem into four groups: true positions, false negative, false positive, and true negative. These values are used to calculate performance metrics such as accuracy, precision, recall, and f-measure.

The confusion matrix is used to calculate performance metrics for a classification dataset. *Classification accuracy* is the ratio of correct predictions among all predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset. Therefore, apart from accuracy, we calculate the *Precision*, *Recall*, and *F-Measure* from the confusion matrix.

Precision is the ratio of predictions correctly classified out of all the predicted values.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall represents the ratio of the correctly classified values out of all the actual values.

$$\text{Recall} = \frac{TP}{TP + FN}$$

As discussed previously, accuracy will not always be the metrics to select the best model. As you can see from the formula, accuracy can be largely contributed by a large number of true negatives. In most circumstances, we do not focus much on false negatives and false positives. *F-measure* (F1 Score) might be a better measure to use if there is an imbalance between Precision and Recall and there is an uneven class distribution.

$$\text{F-Measure (F1)} = 2\left(\frac{Precision \times Recall}{Precision + Recall}\right)$$

Thus, in situations where the number of false positives and false negatives are about the same, accuracy is generally the best measure of performance. If this is not the case, than F-measure is more informative than accuracy [35].

Since all 3 classifications sets have multiple classes, the Precision, recall and F-Measure values are averaged over all classes. The accuracy values are for the entire test set.

For regression datasets, we measure the performance using the Mean Absolute Error (MAE) which is the average of the absolute differences. We use MAE instead of Mean Squared Error (MSE) which is the average of the square of difference, because we do not want to penalties the large errors more than the small errors. MAE will ensure that all errors are treated the same.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

For each model, we also look at the training time. We assume that building the neural network for a neural shrub (leaves or classes) can be done in parallel. Thus, the time for the neural shrub method equals the time to create the decision tree plus the time to train the largest ANN.

5.4 Results

The classification results are shown in Tables 4-6. We present the results for both shrubs methods (classes/leaves). With the exception of Connect-4 (Leaves), we see that the neural shrub method improves the decision tree and comes close to the accuracy of the ANN. In the case of MNIST, the improvement is fairly substantial. The slight reduction in the Connect-4 leaves accuracy is a result of not having enough data in the leaf nodes to produce an accurate ANN. This obstacle is overcome by combining topologically similar nodes (neural shrubs on classes).

Table 4: Connect4 Results

Method	Precision	Recall	F1	Accuracy	Time (sec)
Decision Tree	0.56	0.49	0.48	0.73	0.23
Neural Network	0.64	0.54	0.52	0.78	192.61
Neural Shrub-C	0.64	0.55	0.54	0.78	84.82
Neural Shrub-L	0.48	0.48	0.47	0.69	15.55

For Connect-4 the results are slightly unbalanced, as the percentage of false positives to false negatives can differ by as much as 10%. Here there are no dire consequences of false positives and negatives unless you are gambling. In which case, we argue you might want to consider false positives and negatives equally. Thus, F-measure might be the best choice when determining

which model to use. Here we see that in terms of F-measure, the neural shrub-C is the best performer of the methods.

Table 5: MNIST Results

Method	Precision	Recall	F1	Accuracy	Time (sec)
Decision Tree	0.84	0.85	0.85	0.78	10.52
Neural Network	0.98	0.98	0.98	0.98	147.73
Neural Shrub-C	0.96	0.96	0.96	0.97	38.74
Neural Shrub-L	0.96	0.96	0.96	0.95	23.32

For MNIST, the number of false positives and false negatives are rather low ($< 1\%$). Thus, we probably want to focus on accuracy. Here we see that both neural shrubs methods substantially increase the accuracy over a standard decision tree and are very close to the accuracy of the neural network.

Table 6: SensIT Results

Method	Precision	Recall	F1	Accuracy	Time (sec)
Decision Tree	0.66	0.67	0.66	0.68	2.92
Neural Network	0.68	0.69	0.68	0.71	12.66
Neural Shrub-C	0.68	0.68	0.68	0.71	12.57
Neural Shrub-L	0.66	0.67	0.66	0.68	5.76

The SensIT dataset is also fairly balanced so again we focus on accuracy. Here we see more of the same. The neural shrub method on classes increase the accuracy of the decision tree and provides an accuracy equal to the ANN.

From Tables 4-6 we see that training a neural shrub less than training a full ANN. The neural shrub-C method takes longer to train than the leaves method because the data is aggregated, thus more data to train on.

Table 7: YearPredictionMSD Results

Method	MAE	Time (sec)
Decision Tree	7.09	32.77
Neural Network	6.08	209.6
Neural Shrub	6.80	47.96

For the regression dataset, we are predicting a continuous target value. The goal is to see on average how much we are off. To asses this, we determine how far the the actual target (y_i) is from the predicted target ($\hat{y}_i$) using Mean Absolute Error. Models with lower MAE is more accurate. Table 7 shows results that are comparable to the classification sets in that the ANN is the most accurate closely followed by the neural shrub with the decision tree being the worst performer.

6 Conclusions

The current paper presented a new approach to advanced decision making by capitalizing on the interpretability of decision trees and the accuracy of ANNs. The approach, deemed *neural shrubs*, proceeded by using a decision tree architecture to pre-partition the data space, and grouping topologically similar data through a feedforward ANN. Experimental results were presented for classification and regression using several standard benchmark data sets and it was shown that the accuracy of the neural shrub was slightly less than that of a ANN and higher than a standard decision tree, while maintaining much of the interpretability of the decision tree.

Furthermore, the time required for training the neural shrubs was minimal as compared to training times associated with ANNs. The reduction in training time will help in rapid evaluation of different neural network architectures which will result in a more thorough hyper-parameter optimization.

Comparing neural shrubs on leaves and neural shrubs on classes, we found that neural shrubs on classes outperform in accuracy neural shrubs on leaves. However, the time taken to train a neural shrub on class was more. This is because there are more data to train the neural networks in the neural shrubs on classes.

We found that the level of interpretability is completely controlled by the depth of the tree which can be controlled by the user. When the depth of the tree increases, the neural shrubs behave more like a decision tree. Similarly, when the depth of the tree is reduced, the neural shrub behaves like a neural network. The challenge is to find the right depth of the tree which makes the neural shrub interpretable, without loosing accuracy of the neural network.

Future work will focus on further examination of the leaf node topology to determine bounds on total data needed for accurate training of the ANN in the leaf nodes of the neural shrubs. In addition, we aim to focus future work on investigating automatic pruning of the tree depth based on cross-validation of the overall neural shrub architecture as opposed to the traditional cross-validation using CART alone. Another interesting study will be to look at different neural network architecture for each topological similar points instead of using the same neural network architecture.

References

[1] M. Sazli, "A brief review of feed-forward neural networks," *Communications, Faculty Of Science, University of Ankara*, vol. 50, pp. 11–17, 01 2006.

[2] P. Norvig, "On chomsky and the two cultures of statistical learning," in *Berechenbarkeit der Welt?* Springer, 2017, pp. 61–83.

[3] M. Brundage, S. Avin, J. Clark, H. Toner, P. Eckersley, B. Garfinkel, A. Dafoe, P. Scharre, T. Zeitzoff, B. Filar *et al.*, "The malicious use of artificial intelligence: Forecasting, prevention, and mitigation," *arXiv preprint arXiv:1802.07228*, 2018.

[4] P. Hall and N. Gill, *Introduction to Machine Learning Interpretability*. O'Reilly Media, Incorporated, 2018.

[5] E. B. Hunt, "Concept learning: An information processing problem." 1962.

[6] J. A. Sonquist, "Finding variables that work," *Public Opinion Quarterly*, vol. 33, no. 1, pp. 83–95, 1969.

[7] L. Brieman, J. Friedman, R. Olshen, and C. Stone, "Classification and regression trees. belmont (ca): Wadsworth," *Google Scholar*, 1984.

[8] C.-C. Tsai, M.-C. Lu, and C.-C. Wei, "Decision tree–based classifier combined with neural-based predictor for water-stage forecasts in a river basin during typhoons: a case study in taiwan," *Environmental engineering science*, vol. 29, no. 2, pp. 108–116, 2012.

[9] M. A. Whitley, "Using statistical learning to predict survival of passengers on the rms titanic," 2015.

[10] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec 1943.

[11] M. Minsky and S. A. Papert, *An Introduction to Computational Geometry*. MIT Press, 1969.

[12] D. F. Specht, "Probabilistic neural networks for classification, mapping, or associative memory," in *IEEE International Conference on Neural Networks*, July 1988, pp. 525–532 vol.1.

[13] G. P. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 4, pp. 451–462, Nov 2000.

[14] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *ArXiv*, vol. abs/1901.06032, 2019.

[15] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991.

[16] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[17] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 240–254, March 1994.

[18] K. Chakraborty, K. Mehrotra, C. K. Mohan, and S. Ranka, "Forecasting the behavior of multivariate time series using neural networks," *Neural Networks*, vol. 5, no. 6, pp. 961 – 970, 1992.

[19] G. Dorffner, "Neural networks for time series processing," *Neural Network World*, vol. 6, pp. 447–468, 1996.

[20] H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu, "Video paragraph captioning using hierarchical recurrent neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[21] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, "Describing videos by exploiting temporal structure," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[22] Y. Jiang, Z. Wu, J. Wang, X. Xue, and S. Chang, "Exploiting feature and class relationships in video categorization with regularized deep neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 2, pp. 352–364, Feb 2018.

[23] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang, "Hierarchical recurrent neural encoder for video representation with application to captioning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 2672–2680.

[25] J. M. Jerez-Aragonés, J. A. Gómez-Ruiz, G. Ramos-Jiménez, J. Muñoz-Pérez, and E. Alba-Conejo, "A combined neural network and decision trees model for prognosis of breast cancer relapse," *Artificial intelligence in medicine*, vol. 27, no. 1, pp. 45–63, 2003.

[26] Y. Yang, I. G. Morillo, and T. M. Hospedales, "Deep neural decision trees," *arXiv preprint arXiv:1806.06988*, 2018.

[27] J. Sirat and J. Nadal, "Neural trees: a new tool for classification," *Network: computation in neural systems*, vol. 1, no. 4, pp. 423–438, 1990.

[28] I. K. Sethi, "Entropy nets: from decision trees to neural networks," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1605–1613, 1990.

[29] N. Frosst and G. Hinton, "Distilling a neural network into a soft decision tree," *arXiv preprint arXiv:1711.09784*, 2017.

[30] S. Rota Bulo and P. Kontschieder, "Neural decision forests for semantic image labelling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 81–88.

[31] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo, "Deep neural decision forests," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1467–1475.

[32] T. Chakraborty, S. Chattopadhyay, and A. K. Chakraborty, "A novel hybridization of classification trees and artificial neural networks for selection of students in a business school," *Opsearch*, vol. 55, no. 2, pp. 434–446, 2018.

[33] C.-C. Tsai, M.-C. Lu, and C.-C. Wei, “Decision tree–based classifier combined with neural-based predictor for water-stage forecasts in a river basin during typhoons: a case study in taiwan,” *Environmental engineering science*, vol. 29, no. 2, pp. 108–116, 2012.

[34] C.-C. Chang and C.-J. Lin, “Libsvm: a library for support vector machines," 2001. software available at http://www. csie. ntu. edu. tw/˜ cjlin/libsvm,” 2012.

[35] J. Davis and M. Goadrich, “The relationship between precision-recall and roc curves,” in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 233–240.