

Feature Learning in Soft Decision Trees

Abel Seyoum
aseyoum@ucsd.edu

Ethan Shapiro
enshapir@ucsd.edu

Carlos Monterosa
cmonterosa@ucsd.edu

Mikhail Belkin
mbelkin@ucsd.edu

Yi-An Ma
yianma@ucsd.edu

Abstract

In modern machine learning, deep neural networks (DNNs) have been very popular in recent study, especially with the recent emergence of large language models (LLMs). Common methods to train said models include gradient based methods including (Stochastic) Gradient Descent for example. However, not much is known about the layers of said models, especially in the context of what occurs to these features during training. However, what is known is that DNNs are very capable in learning the true underlying nature of these features. However, recent progress in the field as a result of work produced by Prof. Belkin and members of his lab at UC San Diego, has shed light into this feature learning problem, and they have produced what they call the Deep Neural Feature Ansatz (DNFA). As such, we have previously validated the DNFA both through replication study and other empirical experiments in our quarter 1 project. Now however in this work, we continue to in similar line to showcase that soft decision trees learn relevant features in a very similar fashion.

Website:

<https://ethan-shapiro.github.io/Soft-Decision-Tree-Feature-Learning/>

Code:

<https://github.com/Ethan-Shapiro/Soft-Decision-Tree-Feature-Learning>

1	Introduction	2
2	Methods	3
3	Results	5
4	Summary, Discussion, and Future Work	8
5	Acknowledgements	9
	References	10

1 Introduction

As of recent, major contributions to machine learning have been made as a result of deep neural networks (DNNs). We have witnessed the emergence of large language models such as Open-AI’s GPT-4 and Anthropic’s Claude models, major improvements to self-driving vehicles, as well as a plethora of others. A large reason as to why DNNs are able to perform so well in a myriad of tasks is in large part due to how well they are able to learn features during training. Although this may seem like a very important task to understand, there is still a large lack of understanding among the community as to why DNNs are able to learn features so well. By being able to understand how these models learn their features, we will be better able to improve the performance, albeit they are already quite good, as well as improve model transparency.

Prior to the work produced in [Radhakrishnan et al. \(2022\)](#), not much was known about neural feature learning in deep fully connected neural networks (understanding feature learning in this setting is important for these are what are being used in modern application). However, lots of work has been able to highlight the importance and benefits of introducing feature learning into neural networks vs. non-feature learning models [Abbe, Adsera and Misiakiewicz \(2022\)](#) [Ba et al. \(2022\)](#).

As mentioned previously, prior to the work in [Radhakrishnan et al. \(2022\)](#), not much was known about feature learning, however in their paper, they were able to provide some insight into this problem, namely through their Deep Neural Feature Ansatz. Essentially what their ansatz claims is that important features of deep fully connected neural networks can be determined by looking at the average gradient outer product across all samples taken with respect to the input of the layer for any given layer of the network. In mathematical terms, the Deep Neural Feature Ansatz can be defined as:

$$W_i^T W_i \propto \frac{1}{n} \sum_{p=1}^n \nabla f_i(h_i(x_p)) \nabla f_i(h_i(x_p))^T \quad (\text{Deep Neural Feature Ansatz}) \quad (1)$$

What this means in non-mathematical terms is that these types of neural networks learn important features through a mechanism that relies on the average gradient outer product across the network’s layers. This mechanism emphasizes the learning of simpler, more predictive features out of potentially many available ones by up-weighting features strongly related to model output.

With this, we previously successfully replicated some of the empirical experiments performed in [Radhakrishnan et al. \(2022\)](#), and verified that the Deep Neural Feature Ansatz does indeed hold true in practice. However, we also understand that many other models outside of DNNs are also popular in practice, such as Decision Trees. As such, we are curious to understand if Decision Trees learn features in a similar fashion as DNNs do. As such, in the following sections, we will showcase that (Soft) Decision Trees do in fact learn important features as posited in the Deep Neural Feature Ansatz.

2 Methods

2.1 Further Motivating Context into Feature Learning

To provide further motivation and context into the idea and problem of feature learning, let us begin by understanding it in a simpler scenario, namely in the context of single index models. Single index models are models that predict some label y based solely off one feature of some input matrix X . Thus to motivate feature learning, we can simulate a controlled environment in which we know the features of importance, and can test how well a single index model learns the important features.

With this, imagine we are given some dataset in d dimensions: \mathbf{X} in \mathbb{R}^d and their corresponding labels, y_i in \mathbb{R} , and we knew that the only relevant feature to predict y_i was, for example, x_1^2 , this would be an example of a single index model. Now in regards to feature learning, it would be trivial to state that the learned parameter vector, \mathbf{v} , should simply be $(1, 0, \dots, 0)$, where there is only one 1 in the position of x_1 and the other $d - 1$ entries are zeros. However, if you attempt to generalize this, it quickly begins to become a nontrivial problem as to what the learned parameter vector \mathbf{v} is. However, neural networks are able to do this quite well.

We can show this empirically as well. If we were to again create a dataset X in 10 dimensions, where each datapoint x_i was generated off some normal distribution, $x_i \sim \mathcal{N}(0, I)$, as well as have some output y_i in \mathbb{R} , where y_i is solely made from x_1^2 , we can assume that the learned parameter vector \mathbf{v} , should simply be $(1, 0, \dots, 0)$. As can be seen in Figure 1, a trained DNN was able to learn this parameter vector \mathbf{v} quite well.

As mentioned previously, it is quite trivial to do this in the context of understanding how both our data and outputs were generated, however in real world scenarios, this problem gets much more complex. This, in essence, was the goal of [Radhakrishnan et al. \(2022\)](#): to understand how the weight assignments of a neural network evolve during training, and why do neural networks estimate this vector \mathbf{v} well, and what they came up with was their Deep Neural Feature Ansatz.

2.2 Why Soft Decision Trees vs. Traditional Decision Trees?

As mentioned previously, for our project we will be investigating how Decision Trees learn, and if they follow a similar framework as posited in the DNFA. However, in order to study this under the DNFA, studying normal decision trees will not work given their hard decision boundaries. In the case of standard decision trees, items are predicted given binary decision(s). For example, one branch of a binary decision tree may be to consider whether or not $w_1 > 10$. However, this would present real problems when attempting to take the gradient of said tree as you can not take the gradient of an edge. As such, we have decided to use soft decision trees.

Essentially, the difference between standard decision trees and soft decision trees is that when soft decision enter a branch, rather than making a binary decision at each split, soft

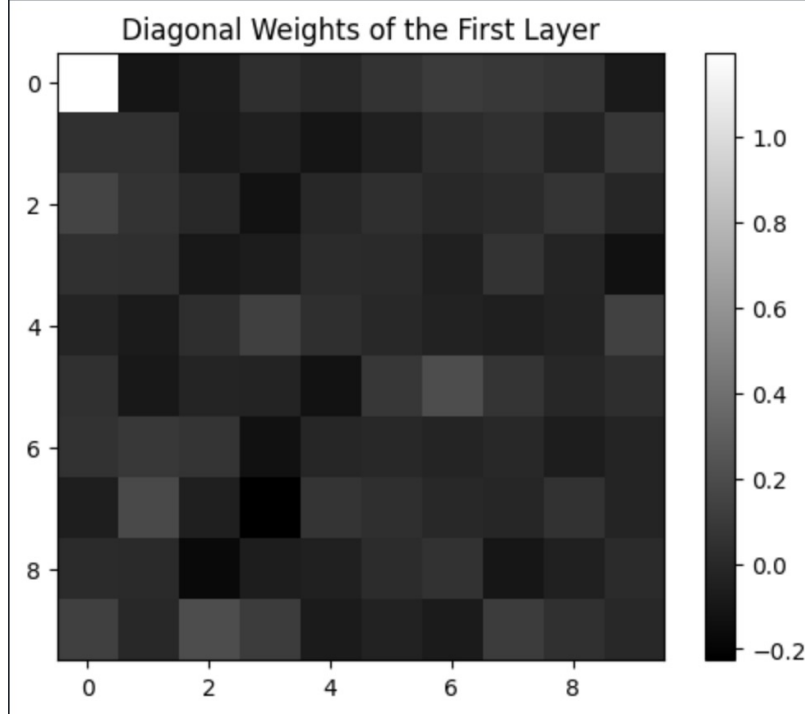


Figure 1: Feature importances across all 10 dimensions for a single index model.

decision trees calculate the probability of a data point lying across both the left and right branches and is oftentimes calculated using some sort of function i.e. sigmoid functions. What this does for us is allow us to take the gradient because said probabilities are calculated using some function at each split, rather than it being a binary decision.

2.3 Soft Decision Tree Architecture

As mentioned previously, soft decision trees differ from standard decision trees in that rather than have binary splits at each non leaf node, a soft decision tree replaces this hard split between the two leaves, with a soft probabilistic decision at each node. With this, there are many ways to introduce "softness" at each split for example using a sigmoid function at each node. However for our use case, we decided to use a soft decision tree introduced by [Frosst and Hinton \(2017\)](#).

To quickly summarize the soft decision tree they introduced in their paper, each inner node in the tree has a learned filter w_i and a bias b_i . Instead of making a hard decision at each of the inner nodes, the inner nodes calculate the probability of an input following a certain branch. This is done by applying a sigmoid logistic function (this is what makes the tree "soft" and allows us to take the Average Gradient Outer Product (AGOP) for the DNFA) to the linear combination of input features x and the learned filter, plus the bias. Thus, the probability that we go right down the tree is $p_i(x) = \sigma(xw_i + b_i)$, and thus going left would simply be $1 - p_i(x)$ where x is the set of input features, w_i is the learned filter, b_i is the bias, and $p_i(x)$ is the outputted probability.

As for the leaf nodes, each leaf node has a learned probability distribution Q_l . When an input reaches a leaf node, it is assigned a distribution over possible classes based on the parameters learned during training. In the paper, they propose two ways of outputting the predictive distribution, one using the distribution from the leaf with the greatest path probability, or averaging the distributions over all the leaves, weighted by their respective path probabilities. For our project, we chose to simply use the distribution from the leaf with the greatest path probability. In the paper they also introduce some other things about the tree such as regularizers and others, so please feel free to look into their paper if interested.

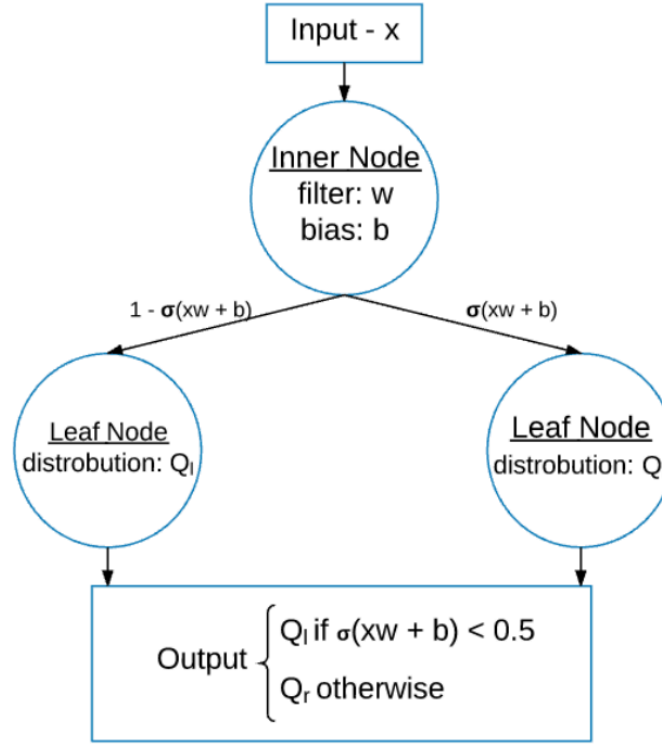


Figure 2: Diagram of soft binary decision tree. Image is taken from [Frosst and Hinton \(2017\)](#).

3 Results

Now, with a strong background and motivation for the problem of feature learning established, as well as the model architecture of the soft decision tree and the concept of the Deep Neural Feature Ansatz (DNFA), this section will attempt to verify whether soft decision trees indeed learn features as suggested by the DNFA. To this end, we have trained our soft decision tree on two distinct datasets and generated what are known as Neural Feature Matrices (NFM). While NFMs are described by the Gram matrix $W_i^T W_i$ for neural networks, where W_i denotes the weights at the i th layer of the network, we adapt this concept for

soft decision trees, considering their probabilistic nature. Here, the NFM represents the sensitivity of the model’s output to its input features, quantifying how slight changes in each input feature would influence the model’s predictions. This analysis is invaluable for understanding the features deemed important by the model for decision-making, thereby enhancing our insights into the model’s interpretability.

3.1 CelebA Dataset

The CelebFaces Attributes (CelebA) dataset is a large-scale face attributes dataset with more than 200,000 celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter, making it a challenging dataset for developing facial attribute recognition, face detection, and landmark (or facial part) localization technologies.

We trained the soft decision tree on this dataset, and generated the NFMs for three distinct classes: rosy cheeks, smiling, and glasses. Additionally, we trained a DNN on the same three classes, and showcased their NFMs as well. The results can be seen in figure 3 below:

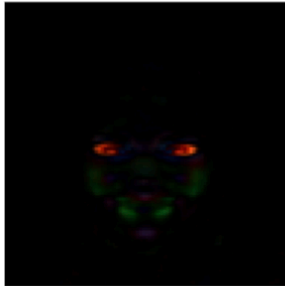
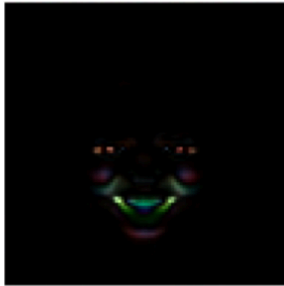
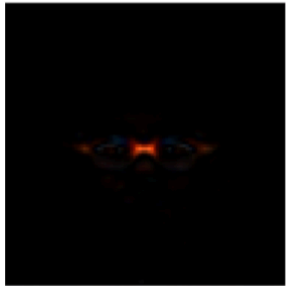
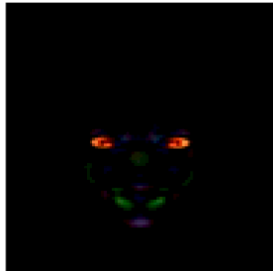
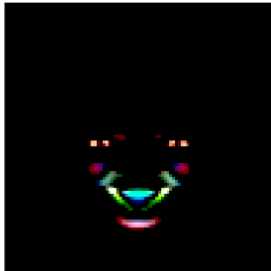
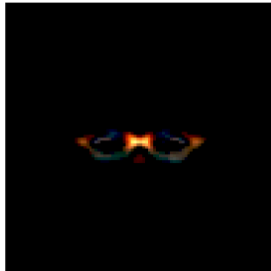
Model Choice NFM	Rosy Cheeks	Smiling	Glasses
Deep Neural Network			
Soft Decision Tree			

Figure 3: Table showcasing NFMs generated between soft decision tree and DNN.

As illustrated in Figure 3, the soft decision tree appears to learn features consistent with the Deep Neural Feature Ansatz (DNFA), as evidenced by the Neural Feature Matrices (NFM) where key features pertinent to the desired output are accentuated. In some instances,

Table 1: Test accuracy of the soft decision tree model across different categories on CelebA dataset.

Category	Test Accuracy (%)
Rosy Cheeks	84.21
Smiling	87.75
Glasses	87.00

the NFM associated with the soft decision tree exhibits more pronounced activations compared to those of the DNN, suggesting a potential for more distinct feature representation. However, further investigation is necessary to ascertain whether this implies superior feature learning. The robust accuracy scores across different tasks also suggest that the soft decision tree model generalizes effectively, indicating successful feature learning.

3.2 STL Star Dataset

The STL-10 dataset is designed for developing unsupervised feature learning, deep learning, and self-taught learning algorithms. It contains 5,000 labeled training images and 8,000 labeled test images from 10 classes, as well as 100,000 unlabeled images, with each image being a color image of size 96x96 pixels. For our use case, we modified the STL-10 dataset slightly to add an image of a star to the top left or bottom right. This changed the learning goal from determining one of ten classes, to determining the class based off the star’s color (black or white).

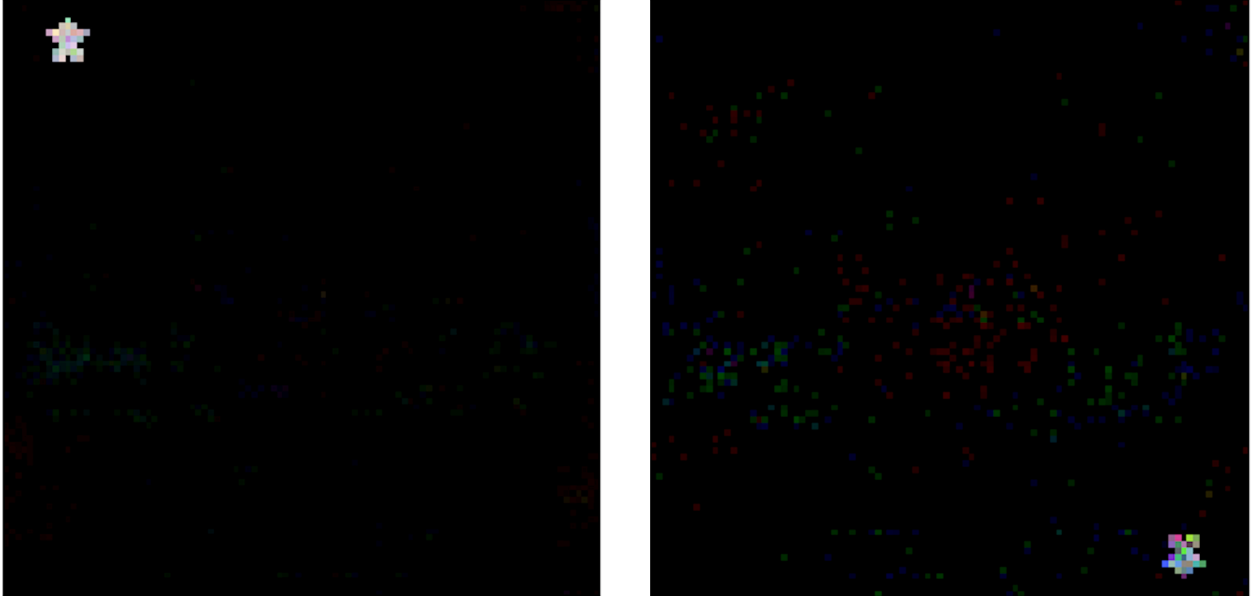


Figure 4: Figures showcasing generated NFMs of the stars in STL Star dataset.

In Figure 4, we explore the adaptability and feature retention capabilities of the soft decision tree. The initial task was to classify images with a star located in the top left, as

indicated by the NFM on the left, which shows clear activation where the star is present. After achieving satisfactory classification, the task was altered by moving the star to the bottom right. The model was then trained to recognize this new pattern, and the learned features were collected. Subsequently, we applied a mask to these learned features to challenge the model’s retention and relearning abilities. The model was trained once more, and the resulting NFM on the right demonstrates that, despite the masking intervention, the model managed to adapt and focus on the new location of the star. This process and the resulting NFM highlight the model’s robust feature learning — it successfully shifted its attention to the new critical feature region even after the application of the mask, showcasing the persistence and flexibility of its learned features and showcases that it does learn feature per the DNFA.

4 Summary, Discussion, and Future Work

4.1 Summary and Discussion

Summary of our findings: All in all, we were able to verify the main idea formulated in [Radhakrishnan et al. \(2022\)](#) known as the Deep Neural Feature Ansatz in soft decision trees. Essentially, it claims that feature learning occurs by prioritizing the features that are the most influential on the output. In math notation, this can be characterized by taking the average gradient outer product with respect to the input of the layer for any given layer of the network. We validated their theory by running empirical experiments on the CelebA dataset, training various models on different facial features, and comparing the generated Neural Feature Matrices generated from the soft decision tree to those generated from the Deep Neural Network. What we found was that across various facial features, the Neural Feature Matrices generated across all tasks look quite similar to those generated by the Deep Neural Network, thus showcasing that feature learning does indeed occur by up-weighting relevant features as posited in the Deep Neural Feature Ansatz in soft decision trees. In addition to this, in order to test whether the relevant features were truly being learned we masked the learned features in STL star, retrained the soft decision tree, and then generated the Neural Feature Matrices, and the model did in fact truly learn the most relevant features. With this, we can confidently say that the results showcased in the theory of their paper, also follow suit in practice for soft decision trees.

4.2 Future Work

For future work, there is still a lot of work and experimentation to be done to validate our results. Immediate next steps should be to continue to test this on more data, as well as include tabular datasets to our results. Assuming it still continues to hold true, for work past this, it would be interesting to feed the learned features into a neural network classifier and see what the generated NFMs and accuracies would be. We have performed some preliminary investigations into this and have received promising results (89.05% accuracy on glasses

from CelebA with only 10 epochs of training). In addition to this, the [Frosst and Hinton \(2017\)](#) paper mentions distilling a neural network into the soft decision tree, and it would be interesting to do the same in our use case and generate more NFMs. Finally, we have done preliminary tests using gradient boosted trees, and have actually gotten slightly better accuracy metrics singular soft decision trees. Thus, to follow up on this, we were curious to see what the Neural Feature Matrices generated would look like using ensemble methods, as well as if seeing if training would be faster using say XGBoost.

5 Acknowledgements

We would like to thank our advisors Proferssors Yi-An Ma and Mikhail Belkin for all of their help throughout the year, as well as Suraj Rampure and HDSI for having a fantastic capstone program.

References

- Abbe, Emmanuel, Enric Boix Adsera, and Theodor Misiakiewicz. 2022. “The merged-staircase property: a necessary and nearly sufficient condition for sgd learning of sparse functions on two-layer neural networks.” In *Conference on Learning Theory*. PMLR
- Ba, Jimmy, Murat A Erdogdu, Taiji Suzuki, Zhichao Wang, Denny Wu, and Greg Yang. 2022. “High-dimensional asymptotics of feature learning: How one gradient step improves the representation.” *Advances in Neural Information Processing Systems* 35: 37932–37946
- Frosst, Nicholas, and Geoffrey E. Hinton. 2017. “Distilling a Neural Network Into a Soft Decision Tree.” *CoRR* abs/1711.09784. [\[Link\]](#)
- Radhakrishnan, Adityanarayanan, Daniel Beaglehole, Parthe Pandit, and Mikhail Belkin. 2022. “Feature learning in neural networks and kernel machines that recursively learn features.” *arXiv preprint arXiv:2212.13881*