Pruning Convolutional Neural Networks with Decision Trees: An Experiment

Joseph Du Toit

April 2024

Abstract

In this project, I seek to understand the relative importance of different convolutional filters in a convolutional neural network (CNN) for making classifications. Inspired by the lottery ticket hypothesis, I take the activations from the filters and use them to train decision trees and gradient-boosted decision trees. Using importance metrics, I selectively prune filters and observe how the neural-backed decision trees perform. My results seem to corroborate those of other researchers that the salient weights are sparse in neural networks. Finally, I suggest research possibilities for furthering this initial exploration.

1 Introduction

Over the last decade, neural networks (NNs) have proven to perform extremely well on difficult problems such as computer vision and natural language processing. NNs have become ubiquitous in important industries including healthcare, defense, and information technology to solve sensitive problems with high-consequence outcomes. However, as NN architectures have grown more complex, the operations determining predictions of NNs have become harder to understand. Furthermore, advances in computational capability have enabled NNs to reach unprecedented sizes (GPT-4 reportedly has around 1.76 trillion parameters). These trends in NN development motivate research into interpreting and simplifying the models to prevent opaque "black-box" behavior.

In this project, I explore how filters in a pre-trained convolutional neural network (CNN) influence its predictions by using decision trees (DTs). DTs are more easily understood than NNs because they learn clear binary splittings of data, so that each classification can be traced through a series of learned splittings. The predictions of a NN, on the other hand, result from repeated multiplication with learned weights followed by nonlinear activations which is much more difficult to understand. By combining the capabilities of NNs with the intelligible predictions of decision trees, I hope to find ways to simplify and elucidate the structure of NNs. My intuition is that even in a small CNN, we can find convolutional filters that do not contribute significantly to the final predictions, and by identifying and removing these filters, we can produce a smaller, more interpretable model.

2 Literature Review

The relationship between DTs and NNs has been explored for some time, although it has not always been made explicit. Many studies investigate the properties of neural networks with piecewise linear activation functions such as ReLU as piecewise linear function approximations. Since DTs also learn piecewise linear functions, the results imply a deep connection between NNs and DTs. Researchers have shown that piecewise linear activation functions such as ReLU give deep NNs powerful abilities to fit complex nonlinear functions. After ReLU became popular around 2010 [Vin10], [Mon+14] explained that one reason piecewise linear activations are effective is that they help NNs learn symmetries. [BB18] furthered this work by showing that deep NNs learn a composition of max-affine spline operations. The equivalence between neural networks with piecewise linear activations and DTs is formalized by [Ayt22] which also gives an algorithm for decomposing

max-affine splines into DTs. Unfortunately, Aytekin's algorithm is not practical because it creates a DT with 2^d branches for a network with d filters. This is far too large to be computationally feasible. However, Aytekin claims: "there may occur violating and redundant rules that would provide lossless pruning of the NN-equivalent tree", suggesting that finding a good DT representation is possible with the right pruning method.

Pruning, or removing ineffective weights from a neural network, is another area of NN research that has seen extensive interest. Many approaches successfully reduce the size and energy consumption of neural networks [Bla+20]. My project draws inspiration from the "lottery ticket hypothesis" coined by Frankle and Carbin [FC19]. They show that NNs consistently have randomly formed subnetworks due to random initialization of the weights that, after training, perform just as well if not better than the original network. These "lottery ticket" subnetworks may be as small as ten to twenty percent of the size of the original network. However, beyond attributing the existence of lottery tickets to fortuitous initialization, I have not yet found a satisfactory explanation in the literature for why this phenomenon happens. In this project I use decision trees to find a lottery-ticket hypothesis after training to shed light on which weights in the NN are important and why.

3 Methods

The experiment I conducted investigated the effects of pruning filters from a pre-trained CNN using feature-importance metrics from a decision tree. The experiment can be summarized in three steps:

1. Obtain activation maps for each filter from a pre-trained convolutional layer.
2. Train a DT on the activation maps and compute feature-importance metrics.
3. Remove filters with low feature-importance scores and observe accuracy.

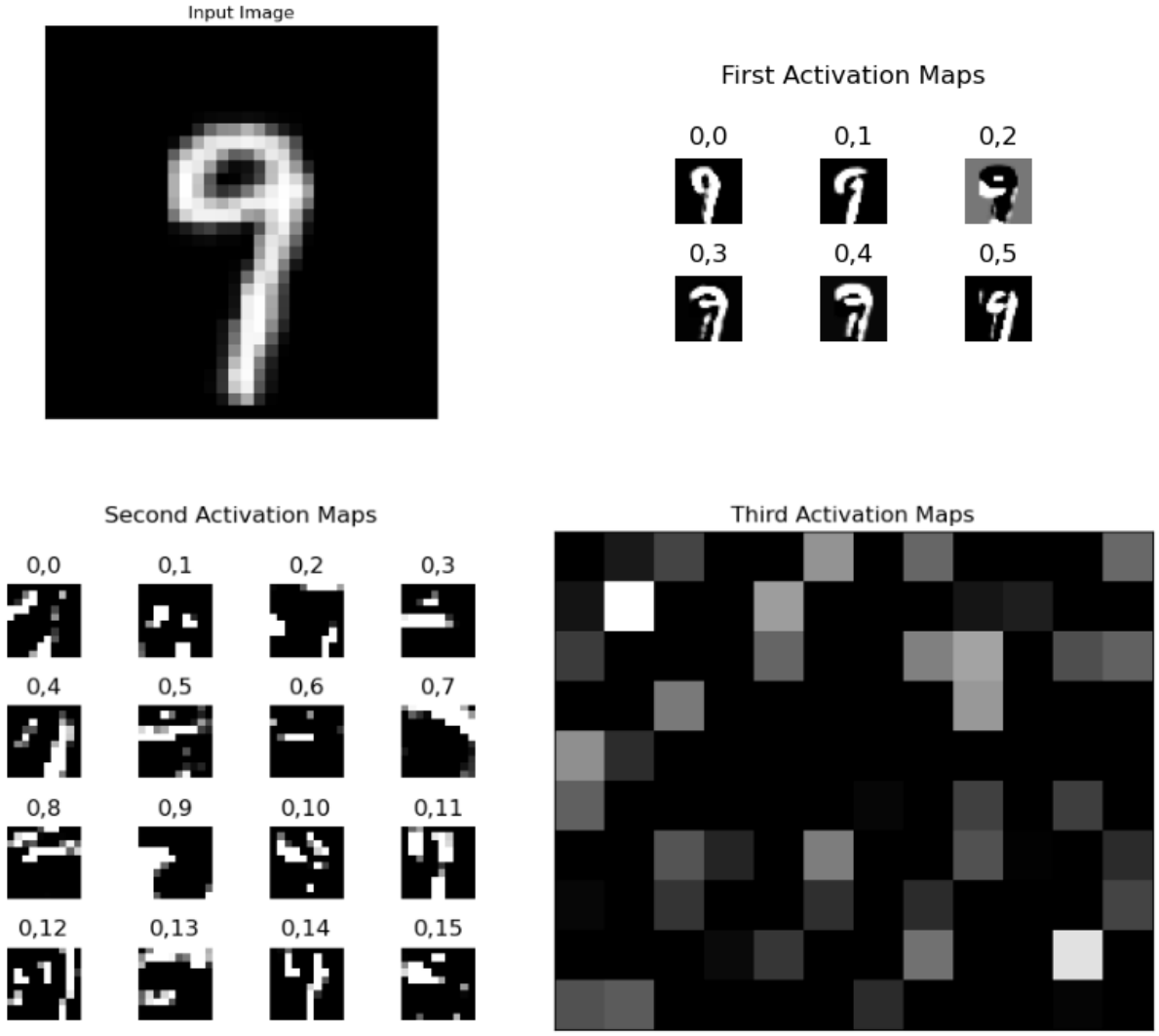
I chose to use the canonical LeNet5 CNN pre-trained on the MNIST handwritten digits dataset. This model has three convolutional layers followed by ReLU activations and maxpooling. The outputs are then passed through two fully connected layers with a ReLU on the first and softmax on the second. Although LeNet5 is comparably very small (only around 60,000 parameters), it achieves 0.9907 test accuracy. The full architecture for the model is:

```
LeNet5(
  (convnet): Sequential(
    (c1): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1))
    (relu1): ReLU()
    (s2): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1,
    (c3): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
    (relu3): ReLU()
    (s4): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1,
    (c5): Conv2d(16, 120, kernel_size=(5, 5), stride=(1, 1))
    (relu5): ReLU()
  )
  (fc): Sequential(
    (f6): Linear(in_features=120, out_features=84, bias=True)
    (relu6): ReLU()
    (f7): Linear(in_features=84, out_features=10, bias=True)
    (sig7): LogSoftmax(dim=-1)
  )
)
```

At each ReLU layer (relu1, relu3, and relu5), I pulled the activation maps from the network for each training image in the dataset. The first activation layer yields 6 channels of 28x28 pixel activation maps, the second yields 16 channels of 10x10 pixel activation maps, and the third yields 120 channels of 1x1 pixel activation maps. Figure 1 contains visualizations of the activation maps from each layer for a sample image from MNIST. For each image, the activation scores for each map were calculated by taking the mean of all of the values in the activation map. This is a weakness in the method since important information about the magnitude of specific values is lost. An activation map with a single high value is scored equally with an activation map with all pixels near the average value. In further research, I think it would be insightful to experiment with using other aggregation scoring methods such as the maximum value or the average magnitude of the top k values in the activation map.

As a preliminary check of how the activation scores were distributed across classes and activation maps, I plotted a histogram for each activation map displaying the number of training images of each class that had a maximum score for that activation map. For example, if an image of class '2' achieved its highest score on map '3', then it would be

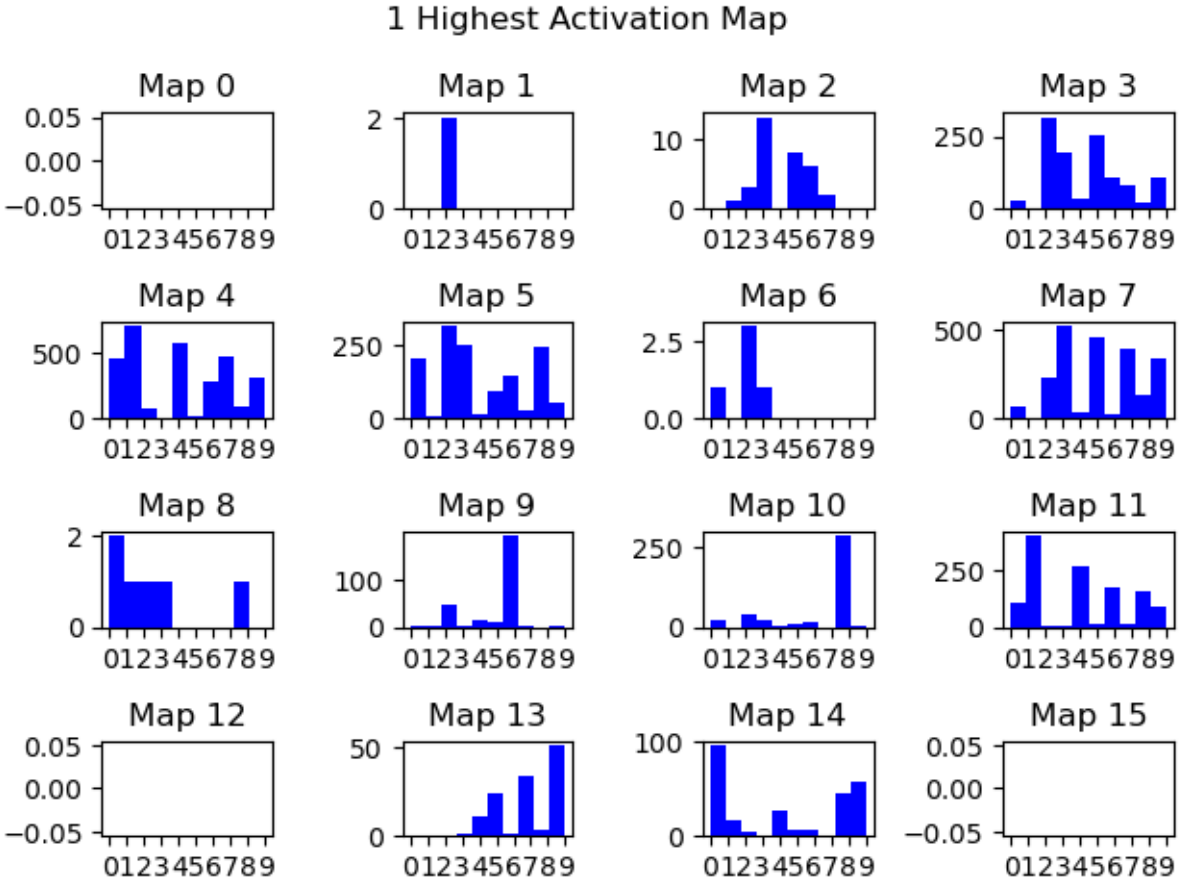
Figure 1: *Activation maps from the convolutional layers*



binned in the histogram for map '3' as a count in the bin for class '2'. I plotted histograms for the highest score, second-highest score, and down to the lowest score to visualize how the activation maps split the data. The histograms for the highest activation scores from the second convolutional layer are plotted in Figure 2. Already we can see that certain activation maps are separating the data. Furthermore, a few activation maps are activated much more than the others, suggesting that there are filters that do not contribute to the final prediction and can be pruned.

Next, a DT was trained on the activation map scores. The DT was meant to be an interpretable approximation for the rest of the model after the convolutional layer from which its scores were taken. This model was inspired by the "neural-backed decision trees"

Figure 2: *Activation maps with highest scores by image class for the second convolutional layer. The x-axes are the class labels and the y-axes are the number of training data instances of that class that achieve maximum score on the corresponding map*



design presented by [Wan+21]. Truncating the model after activation layers allowed me to determine if the depth of the NN was important for attaining better classification accuracy, or if deeper layers could be dropped altogether. I experimented with the depth of the DT, and settled on using the depth d closest to $d = log_2(n)$ where n is the number of features (activation maps) that the DT was trained on. I also experimented with gradient-boosted forests rather than a single decision tree, which significantly improved test accuracy. However, for the final pruning, a single DT was used.

Once the DT was trained, the "gain" feature importance for each feature was computed following the algorithm put forward by Chen and Guestrin in their original paper on XGBoost [CG16]. The gain is defined as

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} + \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma$$

where G_L, H_L and G_R, L_R are the first and second derivatives, respectively, of the training loss with respect to the model prediction for a potential new left node and right node, λ is a regularization parameter, and γ is a minimum loss reduction parameter. See [CG16] for a more complete explanation. Essentially, the gain measures the performance of making a new split in the DT against the performance of the decision tree without making a split. Features with high gain are important for splitting the data while features with low gain do not have as much influence over final predictions. Intuitively, this means that features with low gain can be removed with the least consequence to model accuracy. Using gain feature importance for the activation maps, the activation maps were removed from the training data and a new DT was trained. The feature importances and accuracies of the pruned models are reported in the results section.

4 Results and Discussion

The most important results from this experiment are the model accuracies, the feature importance, and finally the change in model accuracy as features are removed. Although the model accuracies are lower than desired, the pruning algorithm shows promise for finding lottery ticket networks.

The test accuracies of the neural-backed decision tree models never reached the level of the original NN even when using gradient-boosted forests. Realistically, NNs have far more connections giving rise to powerful emergent properties that are not easily replicable in a small decision tree. The accuracies for the neural-backed decision trees are reported in Table 1.

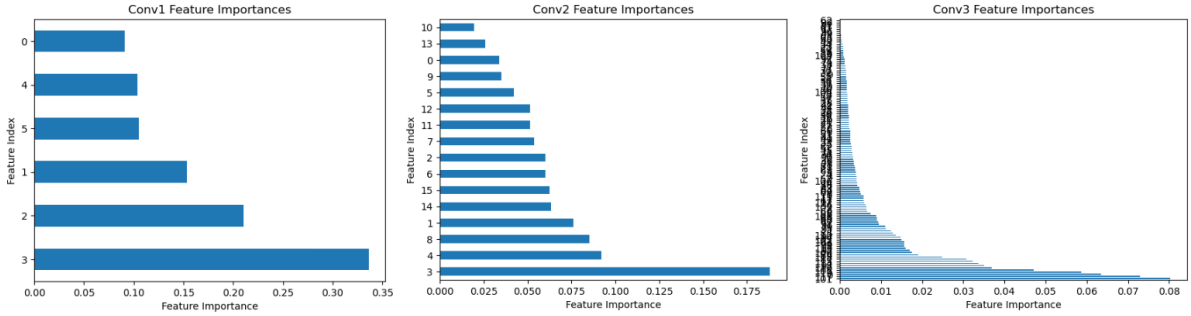
Table 1: *Accuracies from training a single DT are compared against the accuracies of XGBoost with 500 estimators. For reference, the original LeNet5 achieves 0.9907 test accuracy which is much better than most of these neural-backed decision trees*

	Conv 1	Conv 2	Conv 3
Single DT	0.4495	0.763	0.7805
XGBoost	0.5365	0.902	0.9295

Although the results for accuracies are disappointing, the results for feature importance and pruning are much more interesting. The feature importances (shown in Figure 3) show that the gain measurement successfully begins to distinguish the relevance of the

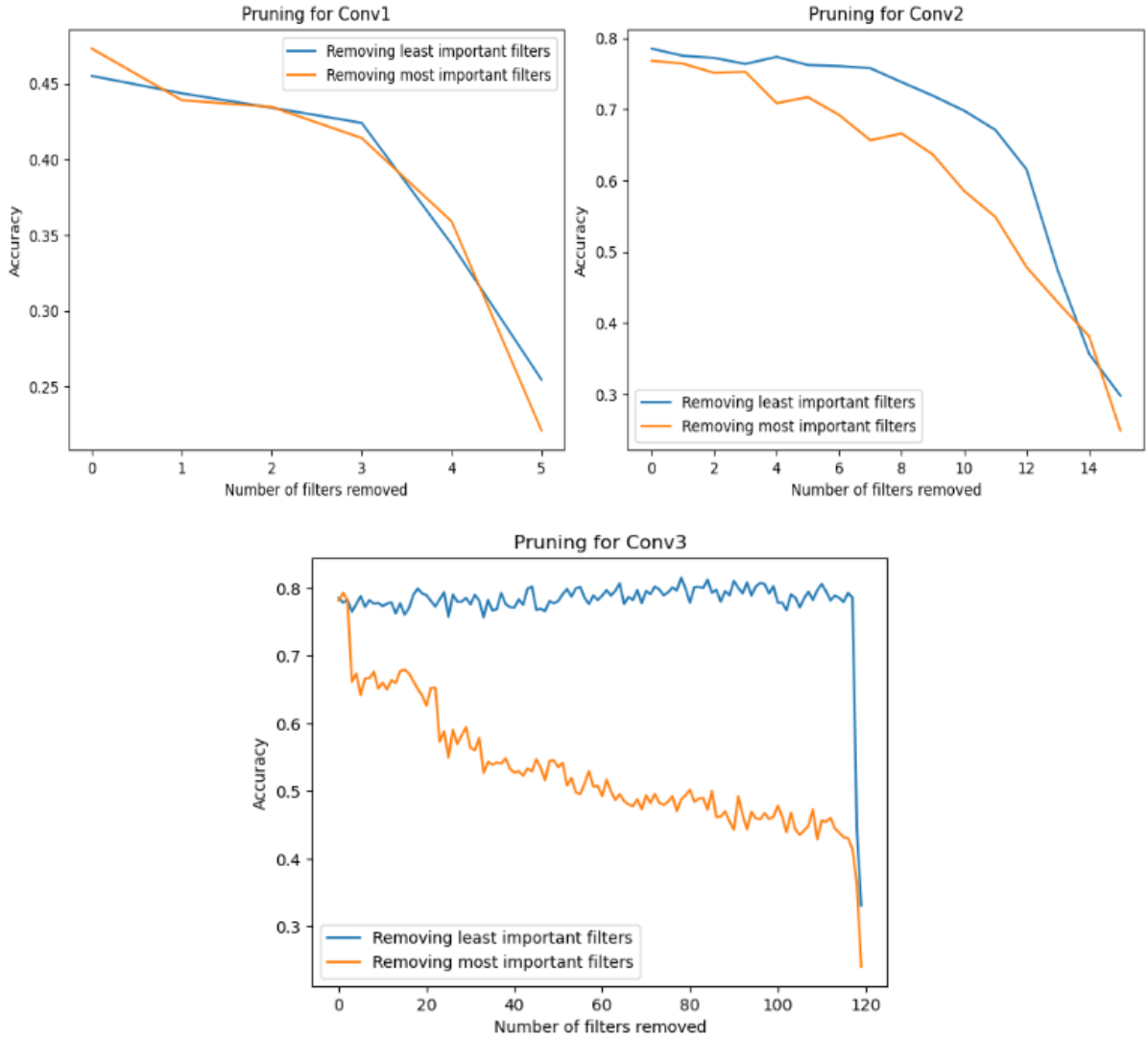
activation maps to the final classification. The difference is especially pronounced as the activation maps are taken from deeper convolutions. This suggests that the embedding produced by the convolutional layers concentrates relevant information into fewer activations maps deeper into the network. By modern standards, LeNet5 is not a very deep network compared to models like ResNets. As NNs trend toward being deeper and deeper, it would be interesting to explore if this is even more apparent. I suspect that dropout and residual connections help mitigate the concentration of signal into a few activations.

Figure 3: *Feature importances for DTs fit to activation maps from the first, second, and third convolutional layers. Notice how the feature importances from deeper layers is skewed toward the most important features.*



The results from pruning features by importance extend these results and are plotted in Figure 4. Compared to worst-case pruning (pruning the most important features first), the DT trained on activations from the first convolutional layer shows no improvement. In the second layer, there is marginal improvement in best-case pruning over worst-case pruning. Finally, by the third layer, there is a drastic difference between worst and best-case pruning. Surprisingly, the DT can maintain around 0.80 accuracy with over ninety percent of the activation map features removed. Again, this confirms that the signal is concentrated into a few important channels while the rest of the activations have little effect on outputs. These results may also demonstrate the powerful capabilities of convolutional layers to turn spatial information in images into salient information for classification. The limitation in the experimental design of taking the average activation of a filter may contribute to the poor results for the first and second convolutional layers. Taking the average activation destroys spatial information that is utilized by the next convolutional layer and in theory could help a DT as well.

Figure 4: *Accuracies for DTs trained on activation maps from the first, second, and third convolutional layer. The orange line is best-case pruning accuracy while the blue line is worst-case pruning accuracy*



From the perspective of the referenced analysis on piecewise linear function approximation, it seems that many of the splines learned by the NN are either redundant or scaled so that they have little influence on the final output. Instead, the NN takes advantage of only a few splines to approximate the classification function. This was surprising to me, but it seems to agree with the lottery-ticket hypothesis in that random initialization induces a subnetwork structure in which only a small percentage of the weights are important. The few splines that have random initialization conducive to becoming a

separatrix are reinforced by the neural network while others are rendered consequential.

5 Conclusion

The results of this experiment corroborate the lottery-ticket hypothesis by demonstrating that NNs concentrate signal into only a few channels. Furthermore, using a DT to compute "gain" filter importance and then prune corresponding filters from a NN is a promising strategy. This project also suggests several avenues for future research. For example, it would be interesting to reconstruct a CNN with the pruned features and observe its performance against the original CNN. Since the depth of the network also seems to have an important effect on finding filter importance, I suspect that applying these methods to a deeper network such as a VGG or ResNet model could yield even better results. Finally, analyzing the decision tree to determine why filters are more relevant than others could be insightful for developing even better pruning algorithms and network architectures.

References

[Mon+14] Guido Montúfar et al. "On the Number of Linear Regions of Deep Neural Networks". In: *arXiv:1402.1869 [cs, stat]* (June 2014). URL: https://arxiv.org/abs/1402.1869.

[CG16] Tianqi Chen and Carlos Guestrin. "XGBoost: a Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16* (2016), pp. 785–794. DOI: 10.1145/2939672.2939785. URL: https://dl.acm.org/doi/10.1145/2939672.2939785.

[BB18] Randall Balestriero and Richard Baraniuk. *Mad Max: Affine Spline Insights into Deep Learning*. arXiv.org, May 2018. URL: https://arxiv.org/abs/1805.06576 (visited on 04/25/2024).

[FC19] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *arXiv:1803.03635 [cs]* (Mar. 2019). URL: https://arxiv.org/abs/1803.03635.

[Bla+20] Davis Blalock et al. "What is the State of Neural Network Pruning?" In: *arXiv:2003.03033 [cs, stat]* (Mar. 2020). URL: https://arxiv.org/abs/2003.03033.

[Wan+21] Alvin Wan et al. "NBDT: Neural-Backed Decision Trees". In: *arXiv:2004.00221 [cs]* (Jan. 2021). URL: https://arxiv.org/abs/2004.00221.

[Ayt22] Caglar Aytekin. "Neural Networks are Decision Trees". In: (Oct. 2022). DOI: 10.48550/arxiv.2210.05189.

[Vin10] Hinton Geoffrey E., ed. *Rectified linear units improve restricted boltzmann machines*. Twenty-Seventh International Conference on Machine Learning. International Machine Learning Society, 2910, pp. 807–814.