What Happens To Uncertainty In Neural Networks Upon Pruning?

Ksheer Sagar Agrawal
IIT Gandhinagar

Lipika Rajpal
IIT Gandhinagar

Kanishk Singhal
IIT Gandhinagar

Abstract

Training large neural networks can be costly. Sometimes large neural networks learn unnecessary features which leads to poor performance on test data set. To tackle these issues, it is best to remove the unwanted edges or the entire neuron that is affecting the accuracy of the model. Pruning the network is one such solution that improves the accuracy, speed and uncertainty in the model.

1 INTRODUCTION

Pruning is a technique aimed at reducing the size or complexity of the model to improve its generalised performance and prevent overfitting. Overfitting occurs when a model becomes too complex and captures noise in the training data, leading to poor performance on unseen data. Pruning involves removing parts of the model that do not contribute significantly to the predictions. A comprehensive study of pruning methods on neural networks by Davis Blalock [2] suggests that pruning imposes a trade off between efficiency and accuracy. However it is also found that models performs better in accuracy for small amounts of pruning.

Pruning can be classified as structured and unstructured based on how the network's connection or parameters are removed. Structured pruning involves removing entire neurons, channels or other structured group of parameters from a neural network. This type of pruning maintains the original architecture of the network to some extent. Unstructured pruning, on the other hand, involves removing individual weights or connections from the neural network without regard for the networks original structure. It doesn't constrain the removal of parameters to structured groups. Unstructured pruning can be more fine-grained but may require more complex bookkeeping to manage the pruned connections. Both structured and unstructured pruning techniques have their advantages and trade-offs. In this paper, we have experimented with a structured pruning approach owing to its ease of implementation and hardware-friendliness.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. Copyright 2023 by the author(s).

2 Aleatoric Linear Regression

We have conducted experiments with linear regression models that incorporate aleatoric uncertainty. In these models, we focus on scenarios where the input consists of a single feature, and the output is a single scalar value. The output is determined by a linear relationship with the input, but we also introduce some level of noise.

There are two main types of models we have explored: homoskedastic and heteroskedastic. These models differ in how they introduce noise into the true function that generates the output.

2.1 Aleatoric Homoskedastic Model

The aleatoric homoskedastic model considers data uncertainty, which remains consistent and independent of the input. In this model, for each input point, the output is a linear function of the input value with the addition of Gaussian noise. This noise is characterized by having a mean of zero and a standard deviation denoted as σ. Importantly, the standard deviation σ does not depend on the input values. Therefore, at each input point, the noise is drawn from an identical Gaussian distribution with parameters $N(0, \sigma)$. This modeling approach allows us to account for consistent and uniform uncertainty in the data throughout the entire input space.

$$Y = X^T\Theta + N(0, \sigma)$$

2.2 Aleatoric Heteroskedastic Model:

The aleatoric heteroskedastic model incorporates varying levels of noise into the data, and this noise is influenced by the linear relationship with the input, offering a more flexible approach to modeling uncertainty compared to the homoskedastic model. The noise introduced into the true function of the output is not a fixed standard deviation σ but is instead a linear function of the input. At each point of the input, the amount of noise added to the output is determined by the linear relationship with the input value. This means that as the input values change, the level of uncertainty in the data also varies proportionally.

$$Y = X^T\Theta + N(0, \sigma)$$

$$\sigma = m * X + c$$

where, m and c are learnable parameters.

2.3 Reliability Diagrams

Reliability diagrams depicts the calibration and confidence of machine learning models. The x-axis represents predicted probabilities, while the y-axis shows the observed frequencies of actual outcomes. Ideally, points should align with the diagonal line (y = x), indicating perfect calibration. Deviations from this line reveal overconfidence (points above) or underconfidence (points below) in model predictions. Reliability diagrams help gauge the accuracy of a model's confidence estimates and identify areas for potential calibration improvements [3].

1. X-Axis (Confidence Range): The X-axis of the plot represents the range of predicted confidences, typically ranging from 0 to 1. This axis serves as a visual reference for the confidence thresholds.

2. Y-Axis (Expected Accuracy): The Y-axis represents the expected accuracy for each bin. This is the ratio of accurate samples (correctly predicted) in a particular confidence range. Each point on the Y-axis corresponds to a specific bin.

3. Ideal Calibration Line (Y = X): On the plot, we draw a diagonal line with the equation Y = X. A perfectly calibrated model would follow this line, where the average confidence of each bin matches the expected accuracy for that bin. Deviations from this line indicate calibration issues.

4. Underconfident Bins: When the plot is below the Y = X line, it means that the model is underconfident. It's assigning lower probabilities to certain outcomes than they deserve based on their actual accuracy. These bins indicate areas where the model lacks confidence despite being correct more often.

5. Overconfident Bins: If the plot is below the Y = X line, the model is underconfident. It's assigning higher probabilities to certain outcomes than warranted by their actual accuracy. These bins represent areas where the model is too sure of itself but is often incorrect.

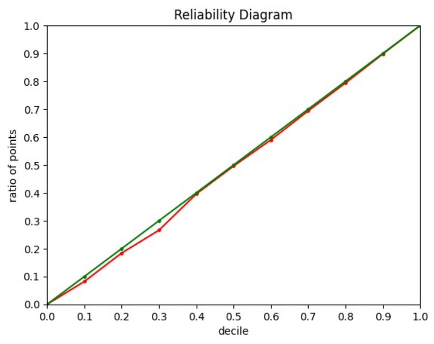

Figure 1: Reliability diagram for Heteroskedastic Regression

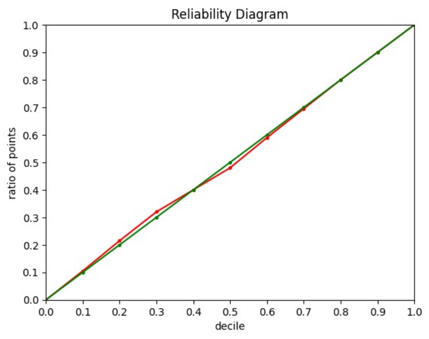

Figure 2: Reliability diagram for Homoskedastic Regression

3 OVERVIEW OF STRUCTURED PRUNING

Standard pruning techniques involves training a network and pruning smallest magnitude weights. The remaining unpruned network when re-initialized such that when trained in isolation matches the accuracy of the original network for similar number of iterations. These sub networks obtained via pruning are called winning tickets. The structure of the winning tickets alone does not guarantee success until properly re-initialised as regarded in the Lottery Ticket Hypothesis [1]. We aim to find these winning tickets to improve training performance and investigate Bayesian inferences.

3.1 Algorithms

Many papers employ various pruning strategies, often derived from Algorithm 1 (Han et al., 2015). These pruning strategies vary in their choices of sparsity structure, scoring, scheduling, and fine-tuning. In this paper, we will focus on pruning parameters in groups (*structured pruning*) and use L1-norm-based scoring.

ALGORITHM 1 Pruning

```
Input: $X$, the dataset on which to train the model
       $Pm$, ratio of nodes to be pruned in each layer
       $N$, the number of layers in Neural Network
1: W ← initialise()
2: W ← train To Convergence(f(X; W))
3: for i in 1 to N-1 do
4:     pruned layers ← prune Pm features from i$^{th}$ layer
5:     (i+1)$^{th}$ layer ← remove out edges from pruned layer
6: end for
7: return pruned model
```

Consider a large Multi-Layer Perceptron of size N. We initialize the model with certain weights and train it for a specified number of epochs. To initiate the pruning process, we rely on specific information, including the pruning ratio, P$_m$, which determines the proportion of features to be pruned in each layer. A for loop is employed to iterate through the model's layers, ranging from 0 to N-1. Pruning within the ith layer is rooted in the L1 norm, where the feature's score is determined by the sum of the absolute values of input weights for each feature. A fraction of Pm features with the lowest scores is subsequently removed from the layer and designated as pruned layers. As a subsequent step, the outgoing edges of the pruned layers within the (i+1)th layer are eliminated. The activation layers interspersed in between are excluded from the pruning process, and the final being vital for the classification process, remains untouched as well. In the end, the resulting pruned model is returned initialized with random values.

3.1.1 Early Stopping Criteria

We are interested in measuring the speed at which networks learn. As a proxy for this quantity, we measure the iteration at which an early-stopping criterion would end training. The specific criterion we employ is the iteration of minimum validation loss with a patience of 3. Our findings co-match the lottery ticket hypothesis, as we find sub networks networks that learn at least as fast as their counter larger parts and achieves comparable accuracy.

3.1.2 One-Shot Pruning

One-shot pruning does indeed find winning tickets at 0.5 pruning ratio with comparable accuracy to the unpruned network. Since one-shot pruning does not require repetitive training, it is relatively cheaper compared to iterative pruning. The gap between the blue and red curves shows clearly that one-shot pruning performs poorly for a large pruning ratio.

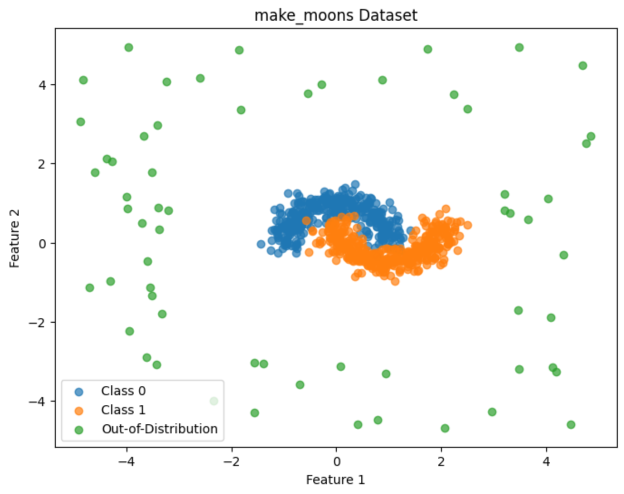

Figure 3: MakeMoons Dataset

3.1.3 Re-Initialised One-Shot Pruning

To measure the importance of the Re-initialization of a winning ticket, we retain the structure of pruned model. However, tweaking the algorithm by initializing it with the pre-training values. We find that with proper re initialisation, pruned models learn faster with similar test accuracy.These results support the lottery ticket hypothesis emphasis on re initialisation. [1].

3.1.4 Randomly Re-Initialised One-Shot Pruning

In order to better understand the lottery ticket hypothesis, which suggests re-initializing the weights to pre-training weights, we also implemented the technique of assigning random weights after pruning to measure the effects of re-initializing on pruning of models. The broader results of this experiments are the average random reinitialised winning tickets test accuracy drops off significantly compared to model initialised with pre-training values.

3.2 Iterative Pruning

There are two different ways of structuring the iterative pruning strategy - pruning with resetting and pruning with continued training. We employed the former technique, as the name suggest we recursively train the model with early stopping followed by pruning. This results in a trade off between computation power and test accuracy. Iterative pruning extracts smaller winning tickets and reach higher test accuracy for smaller network size. The red line in figure 3 and figure 4 depicts , that the test accuracy is retained above the base accuracy even when $Pm \geq 80\%$.

In Figure 5, the Accuracy of a simple MLP model is observed over the make moons dataset. Different kind of pruning techniques for low pruning ratio holds their ac-

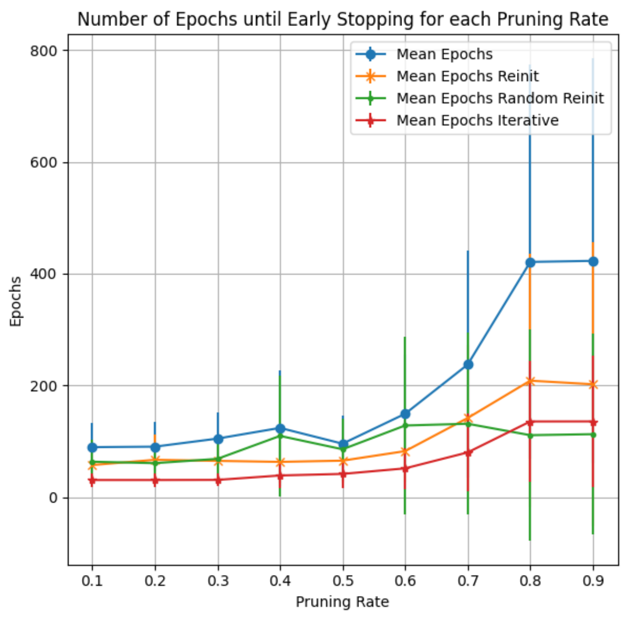

Figure 4: Early-Stop Iteration (Val.) vs. Pruning Ratio for MakeMoons Dataset

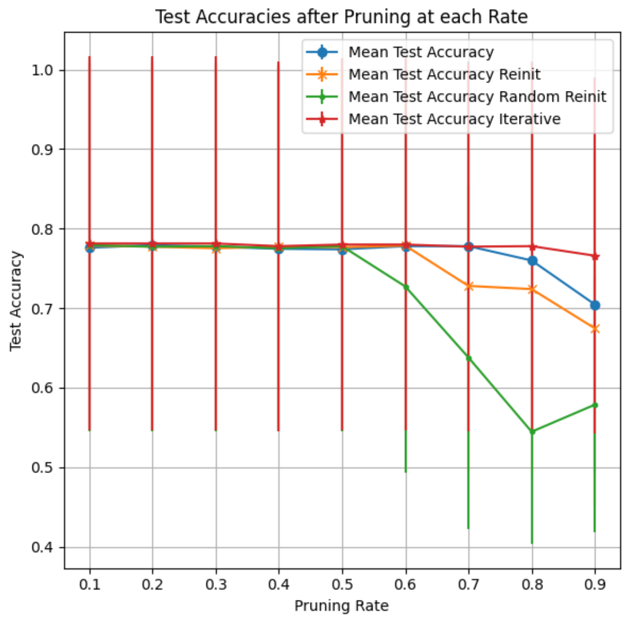

Figure 5: Accuracy vs. Pruning Ratio for MakeMoons Dataset

curacy as the problem is simple. But as we increase the pruning there is a clear drop in accuracy of Oneshot Random Reinit pruning. This is also intuitive as in Oneshot random Re-init the model is randomly initialized and behaves as good as taking a new smaller model, which will perform poorly compared to a larger model. An important note here is that different pruning techniques retain accuracy better than Oneshot random reinit, which shows the promising aspects of pruning a model, which can make a model smaller while retaining the accuracy.

4 PRUNING MODELS

We now discuss the step-by-step implementation of processes before the actual pruning of neural networks.

4.1 Define Model

As we wanted to see the effects of pruning, we described a very large Multi-Layer Perceptron (MLP), with many layers and neurons. Taking a large MLP is better for the following reasons:

4.1.1 Layer-Specific Analysis

A large MLP allows us to perform a layer-specific analysis of the pruning effects. Different layers in a neural network often capture different levels of abstraction and complexity in the data. By using a large MLP, we can investigate how pruning influences each layer individually. This provides insights into whether certain layers are more resilient to pruning or whether specific layers contribute significantly to the network's overall accuracy.

4.1.2 Pruning Technique Variability

A diverse set of pruning techniques can be applied to different layers of the large MLP. Each pruning method may exhibit varying effects on different layers, which could reveal the strengths and weaknesses of each technique. For example, one-shot pruning with reinitialized weights, one-shot pruning with post-training weights, and iterative pruning may have different impacts on early, middle, and late layers of the network.

4.1.3 Realistic Pruning Scenarios

In practical applications, deep neural networks are often deployed with a substantial number of parameters. By using a large MLP, our study aims to emulate realistic pruning scenarios where significant model compression is needed while maintaining acceptable accuracy. This aligns our research with the challenges faced by practitioners in deploying deep learning models in resource-constrained environments.

4.2 Defining Metrics

In the context of evaluating and comparing the effectiveness of neural network pruning techniques, we have selected accuracy as the primary metric. The choice of this metric is grounded in its fundamental significance in assessing the overall performance of a machine-learning model. We have chosen this metric because of its overall simplicity and intuitiveness. Also, this allows us to perform comparisons across models and techniques.

4.2.1 Expected Calibration Error

In the context of machine learning and predictive modeling, ensuring not only accurate but also well-calibrated predictions is crucial. The Expected Calibration Error (ECE) is a widely used metric designed to quantify the degree of calibration in machine learning models, particularly in classification tasks. It provides valuable insights into how well a model's predicted probabilities align with the true probabilities of outcomes.

Regression For regression tasks, calibration can be calculated as the measure of percentage of data points falling in a confidence interval against the supposed number of data points which should lie in the interval. To calculate the calibration error we assume a normal distribution with mean as the prediction and standard deviation can be learnt with the model.

1. Bin Creation: The first step is to find the confidence interval in terms of standard deviation in which the given percentage of data points should ideally lie. We use the 'ppf' function from 'scipy.stats' to calculate the above.

2. Bin Assignment: Assign the percentage of data points which actually lie in the confidence interval. We can plot this data for different confidence intervals which gives us the reliability diagram for regression models.

Classification Calibration, in the context of a classification model, assesses the agreement between the model's predicted probabilities (confidences) and the actual outcomes. This process ensures that the predicted probabilities represent accurate measures of the likelihood of an event occurring. The function takes as input the true labels, predicted labels, and confidences, along with the number of bins to divide the predicted confidences into. The procedure is as follows

1. Bin Creation: The function divides the range of predicted confidences (usually between 0 and 1) into a specified number of equally sized bins(we can also do a non-uniform division of the confidence range in some cases). This step creates thresholds for confidence levels.

2. Bin Assignment: Each prediction in the dataset is assigned to one of the bins based on its confidence level. The function digitizes the confidence values and determines which bin each prediction falls into.

The function then computes the average accuracy and average confidence over the entire dataset. It also calculates the Expected Calibration Error (ECE), which is a weighted average of the calibration gaps for each bin. The calibration gap for a bin is the absolute difference between the bin's accuracy and the average confidence in that bin.

A well-calibrated model should have predicted probabilities that closely match the actual outcomes, resulting in a low ECE. By analyzing these metrics, we can gain insights into the reliability of your model's confidence scores and potentially make adjustments to improve its performance.

5 Results and Analysis

After identifying winning tickets in a fully-connected architecture for make moons dataset, we analyse pruning on convolutional architectures for CIFAR10 dataset. We have plotted their accuracies and ECE against the pruning ratios for iterative pruning strategy. This has given various insights into the trade-offs of pruning neural networks.

5.1 Accuracy

As the pruning ratio increases, the initial impact on model accuracy is generally predictable: accuracy decreases. This is because pruning indiscriminately removes connections, leading to a loss of information and potentially critical features. Higher pruning ratios result in more aggressive compression but also a more significant accuracy drop.

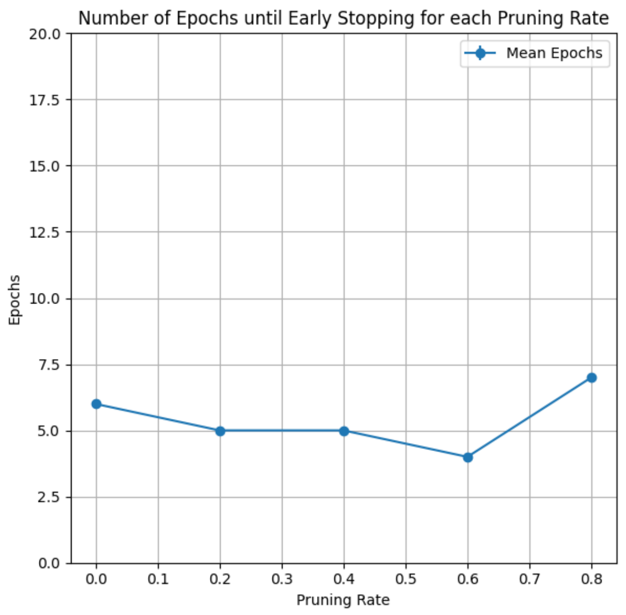

Figure 6: Early-Stop Iteration (Val.) vs. Pruning Ratio for Cifar10 Dataset

In this experiment, we have aimed at attaining the "winning ticket" for iterative pruning on the cipher-10 dataset. The

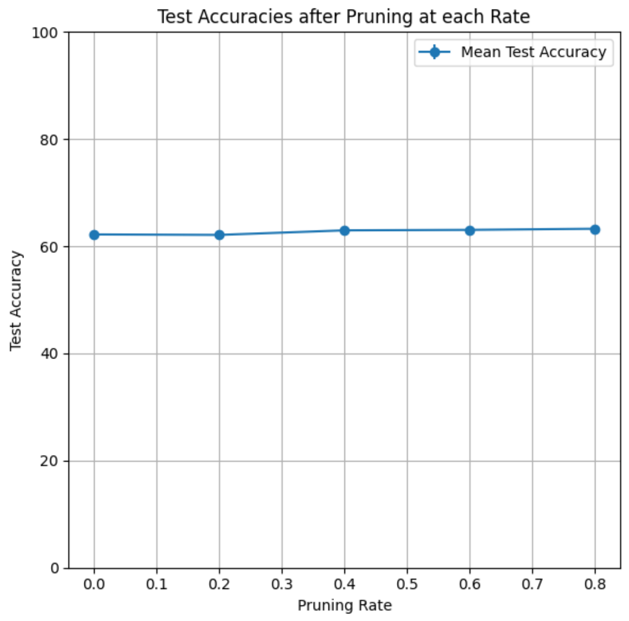

Figure 7: Test Accuracy vs Pruning Ratios for iterative pruning on CIFAR10

pruning technique applied is iterative pruning with reinitializing the weights of the pruned model with that of the corresponding nodes in the initial unpruned model. Furthermore, we have re-trained the model on the training dataset until we achieve the minimum validation loss. As it can be observed from the figures, we have attained the winning ticket at a pruning ratio of 0.6. At this point, the model was able to retain the actual accuracy on the test dataset, consuming a minimum number of training epochs.

5.2 Expected Calibration Error

To evaluate the performance of a deep learning model, accuracy is often insufficient, as not only the decision but the confidence of a model's decision also has to be evaluated. Reliability diagrams are usually employed to measure the deviation between the model's confidence and measured accuracy. The identity function would represent a perfect calibration (shown as a dashed line in the ECE figures below). Any deviation from this identity function means a miscalibration of the model, and the model is considered accordingly either overconfident or underconfident.

1. From Figure 8 & 9 , it can be seen that pruning has a positive effect on the calibration of a network. Often, at meager compression rates, accuracy even increases, giving an additional advantage through pruning. At higher compression rates, there is a trade-off between accuracy and calibration that must be considered individually.
2. Compared to the baseline, the accuracy of the 80% pruned model is reduced only by 2%. However, the ECE has improved by 5% from the baseline. Further pruning as seen in Figure 3.2, yields a drop in accuracy but improves model calibration. It thus improves the model's awareness of its low accuracy.

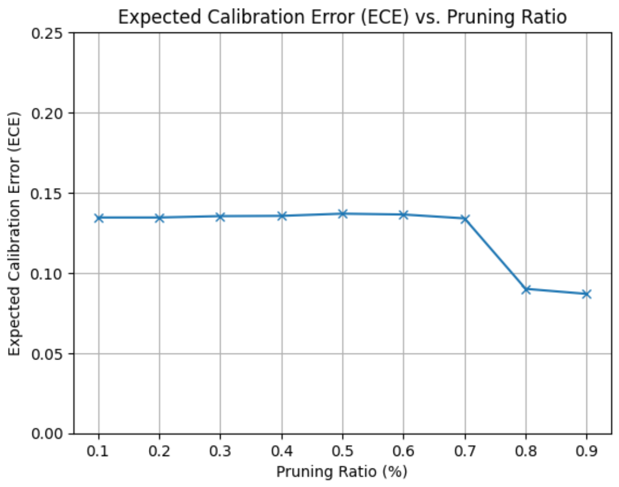

Figure 8: ECE vs Pruning Ratios for Make Moons Dataset

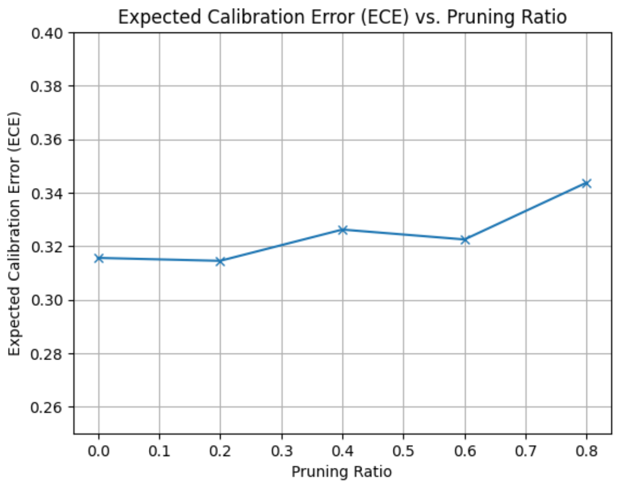

Figure 9: ECE vs Pruning Ratios for CIFAR 10 Dataset

5.3 Out of Distribution (OOD) Detection

For a better understanding of calibration and the value of ECE, we performed an Out-of-Distribution (OOD) Detection on CIFAR 100 dataset. OOD is the task of identifying instances that do not belong to the distribution on which the classifier has been trained. Compared to the baseline, pruned models are considerably better in calibration, even without using a pruning method specifically designed for this purpose. This is an unambiguous indication that pruning can impact and improve a network's decision and therefore making it safer and more robust.

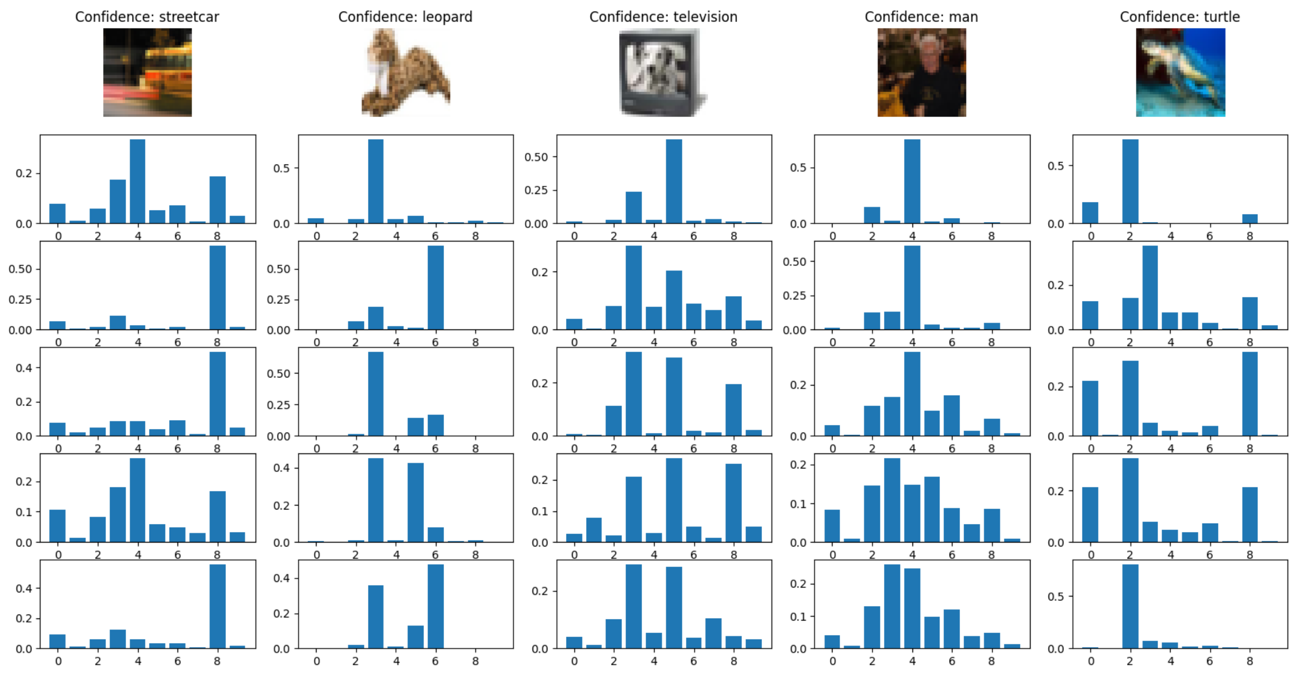

Figure 10:

5.4 Pruning Best Practices

Most Pruning Research papers perform analysis on very few combination of data sets and model architecture. Moreover, due to very few pruning libraries and absence of evaluation metrics , it becomes difficult to reproduce results. State of Neural Network[1] has compiled results of all pruning based research papers, observing which we adhered to some of its good practices.

1. Reported and plotted means and sample standard deviations, instead of one-off measurements, whenever feasible.
2. Used modern large scale datasets like CIFAR 10, instead of toy data sets like MNIST.
3. Identified the exact sets of architectures, datasets, and metrics from pytorch library, ideally in a structured way that is not scattered throughout the results section.
4. For any given pruned model, reported both compression ratio and theoretical speedup.

6 Conclusion

Our results show that pruning may considerably improve the model's calibration without being specifically designed for this purpose. A well calibrated model excels at estimating the reliability of its own decisions. Pruning may thus have a positive effect on reliability and robustness. This result complements literature reports pointing out a positive contribution of the pruning to adversarial robustness.

For future work we have planned,

1. Better analysis on (dataset, architecture) combinations while following best practices of pruning.
2. Introduce model uncertainty with MC Dropout and Laplace.

References

Jonathan Frankle, Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. `https://arxiv.org/pdf/1803.03635.pdf`

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, John Guttag. What is the state of Neural Network Pruning? `https://arxiv.org/pdf/2003.03033.pdf`

Vibhuti Bansal, Rohit Khoiwal, Hetvi Shastri, Haikoo Khandor, Nipun Batra. "I do not know": Quantifying Uncertainty in Neural Network Based Approaches for Non-Intrusive Load Monitoring `https://nipunbatra.github.io/papers/2022/buildsys22-nilm.pdf`

A Visualization of Neural Network

We have used the python library 'networkx' to visualize the process of structured pruning on a neural network. The aim of this activity is to verify the accuracy of the structured pruning algorithm defined above. So, we begin by creating a fully connected neural network and assigning weights to each of the input edges in each layer manually. *Visualization Blog*

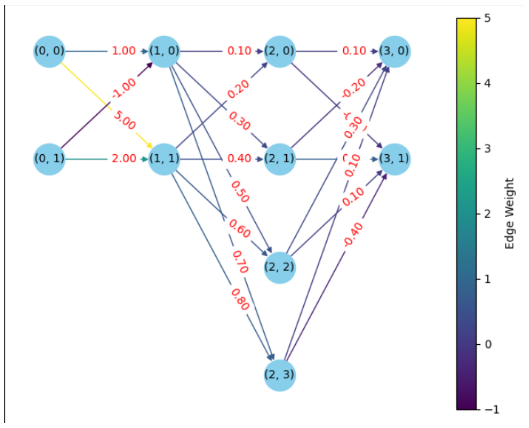

Figure 11: Network before Pruning

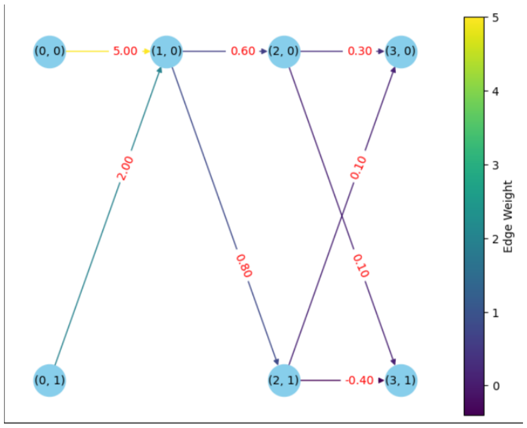

Figure 12: Network after Pruning

B Github Link

We are including the *Github link* of our repository with this paper.