Exploring the Structure Resulting from Unstructured Neural Network Pruning

Jamil Gafur, Max Milkert, Kevin Patrick Griffin, Nicholas T. Wimer, Charles Tripp, Steve Goddard
Computational Science Center, NLR, USA[1,2,3,4]
Computer Science Dept, Univ. of Iowa, USA[1,6]
Computer Science Dept, Vanderbilt Univ., USA[2]
Zazzle, USA[5]

***Abstract*—Iterative Magnitude Pruning (IMP) is a widely used technique for compressing neural networks by progressively removing low-magnitude weights while maintaining predictive accuracy. Despite its widespread application and simplicity, the underlying reasons for its effectiveness remain underexplored. In this work, the pruning dynamics and emergent structural traits of IMP are empirically examined across three benchmark convolutional architectures (ResNet20, Vgg16, and RegNetX) and evaluated on the CIFAR10 and Tiny ImageNet datasets. A comprehensive analysis reveals that IMP preferentially removes weights from deeper layers, preserving early feature extractors until a critical network sparsity threshold of 96% is reached. Up to this 96% threshold, test accuracy remains remarkably stable across all evaluated networks before experiencing significant degradation. Quantitative evidence is provided showing that later stages are pruned more heavily, taking advantage of the fact that deeper layers contain more low-magnitude weights. Furthermore, neuron-level investigations reveal that IMP does not produce any completely pruned (100% sparse) neurons, even at extreme sparsity levels. Sensitivity analysis via neuron zeroing demonstrates that individual neurons maintain a stable range of functional importance rather than narrowing as pruning progresses. Finally, activation similarity metrics indicate that feature representations are preserved throughout the pruning process, keeping pruned and unpruned networks in close representational alignment. These findings highlight the surprising degree of structure generated by unstructured pruning, offering new insights into network compression and potential pathways for hardware-efficient sparse matrix adaptations.**

***Keywords*—Machine learning; convolutional neural networks; multi-layer perceptrons; artificial intelligence**

I. Introduction

As neural networks grow in both complexity and size, so too does the need for additional computing capacity and energy. This creates a challenge in hosting neural network-based applications, particularly on resource-constrained devices. A common solution to this challenge is to prune (i.e., remove) excess parameters or components of the neural networks, thereby reducing the memory and/or energy requirements of the application while retaining its accuracy [1], [2], [3], [4].

A key challenge in pruning is identifying parameters that significantly contribute to model performance. The process determines which parameters can be removed without substantially affecting accuracy. Denil et al. [5] show that a neural network can be effectively reconstructed using only a subset of its parameters. This result demonstrates the potential for model compression while preserving performance, highlighting the importance of careful parameter selection in pruning. Neural network pruning is generally classified into two categories: unstructured and structured pruning, each with distinct trade-offs in sparsity, computational cost, and resulting hardware compatibility of the pruned network.

Unstructured pruning achieves higher compression rates and retains accuracy more effectively at higher sparsity levels. However, Han et al. [6] state that the resulting irregular sparsity patterns are difficult to exploit on modern hardware. In contrast, structured pruning aligns better with hardware accelerators but may significantly degrade accuracy if not applied carefully [7], [8].

Among unstructured methods, IMP [9] has gained popularity for reducing the size and computational demand of neural networks. IMP works by repeatedly training the network and then removing the surviving weights with the smallest magnitude. It then "rewinds" the values of the remaining weights back to a point near initialization and repeats the process (see Fig. 1). Despite its simplicity, IMP achieves comparable or better accuracy retention at high compression ratios when compared to more sophisticated techniques [10]. As identified in Han et al. [6], it is common to use a sparse matrix to represent the parameters of the pruned matrix, which results in less efficient use of hardware accelerators (diminishing the possible benefits of pruning).

This work empirically investigates whether IMP indeed yields an unstructured neural network, or whether sufficient structure is present to achieve the desired memory and energy savings in the pruned network. The pruning experiments (see Fig. 2) reveal the following structure and processing traits:

- Later layers are pruned much more than early layers at all levels of sparsity (until accuracy falls below acceptable levels).
- No neurons are completely removed. That is, no neurons have all of their input weights set to zero.
- Stable neuron-level sensitivity distributions. That is, no single neuron can be removed without impacting accuracy.
- Feature representations (as measured via activation patterns) are preserved throughout the pruning process.

Thus, the results are mixed. We have discovered that IMP does create a surprising amount of structure in the pruned

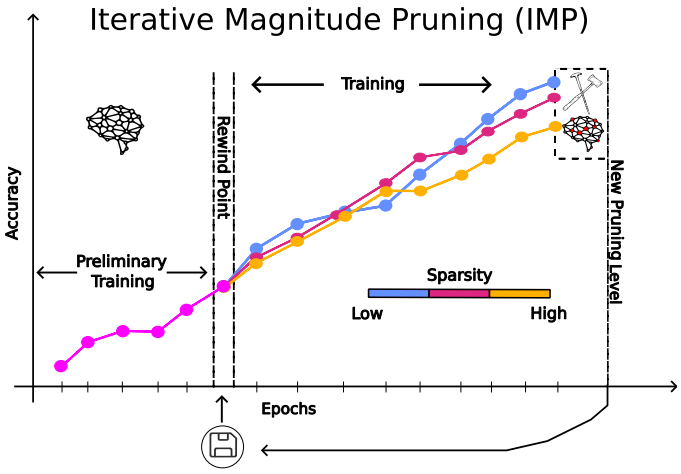

Fig. 1. An illustration of the IMP implementation, based on the approach described in [9], is as follows: Initially, an unpruned "Baseline" model undergoes preliminary training for a specified number of epochs. At the end of this phase, the model is saved as a "rewind point". From there, the network is further trained for a set number of epochs and saved again. After saving, the model is pruned to a new sparsity level, and its weights and biases are reset to those from the rewind point. The pruned weights are then set to zero and made untrainable. This process is repeated until the desired pruning level is achieved.

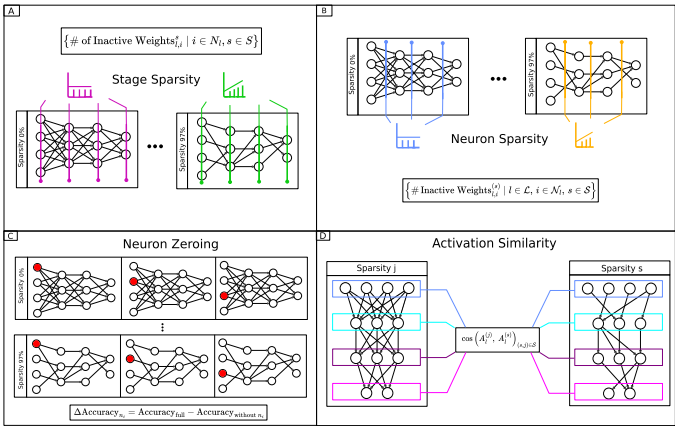

Fig. 2. Overview of the experimental procedures used to analyze sparse neural networks: (A) Stage Sparsity is a metric that calculates the fraction of pruned weights for each neuron across different stages and sparsity levels, providing insight into how pruning is distributed across the network's architecture. (B) Neuron Sparsity measures the percentage of pruned weights for each neuron within a specific layer at various sparsity levels, offering a characterization of the pruning distribution across stages. (C) Neuron Zeroing, where each neuron in a prunable layer is independently ablated by zeroing its incoming weights and bias. The resulting drop in accuracy quantifies the functional importance of each neuron. (D) Activation Similarity, where neuron activations are recorded on a held-out test set at different sparsity levels. These activations are compared using cosine similarity to evaluate how internal representations evolve with pruning.

model, but it is not yet clear how to take advantage of the resulting structure, though we do offer some insights for future exploration.

This study is organized as follows: Section II reviews prior work on pruning and internal network analysis. Section III describes the pruning algorithm, datasets, architectures, and evaluation methods. Section IV presents the experimental findings, and Section V draws conclusions and discusses future research directions.

II. Background and Related Work

Unstructured pruning of convolutional neural networks (CNNs) targets individual weights or connections between neurons for removal, with IMP being among the most widely studied of these techniques [2], [10]. The goal is to remove the "least important" weights, while maintaining the model's original level of accuracy. For IMP, this importance is based on the relative magnitude of the weight. IMP, as shown in Fig. 1, starts by pretraining a network, pruning a fraction of low-magnitude weights from layers with trainable parameters (i.e., pruneable layers), and resetting the remaining weights to an earlier training point, often referred to as the "rewind point". Training is then resumed from this rewound state for the new sparsity level. This process is repeated until the required sparsity level is achieved or a significant degradation in accuracy occurs, at which point the process ends.

Based on this concept, the Lottery Ticket Hypothesis (LTH), by Frankle and Carbin [9], posits that a randomly initialized, sparse sub-network of the original CNN exists. When trained independently from the beginning, it can achieve the performance of the original network with a fraction of the weights. Such sparse sub-networks are called "winning tickets". In this work, the experiments follow the methodology outlined by Frankle and Carbin in [11] for IMP of CNNs to identify winning tickets. Thus, the network is pretrained to allow its parameters to enter an "optimal" loss basin within a few epochs of training. By doing so, the network weights to encode useful information and enter a near-optimal loss basin prior to pruning. This methodology increases the likelihood of finding winning tickets during the IMP process.

Most work on pruning in the literature details the percentage of weights in the entire network that have been pruned. This is commonly referred to as network sparsity. This sparsity is measured across all "stages" in a network, where a stage is a single repeated set of consecutive layers in a network. Section IV-A explores how stage sparsity differs from network sparsity at various pruning levels. Fig. 2(A) illustrates this concept.

Han et al. [2] used a slightly different pruning methodology from IMP. Whereas IMP prunes using weights alone, their methodology first learns the most important weights *and* connections and then prunes "low-weight connections". In their work, a mask was used for each weight tensor to disregard pruned parameters during inference. Thus, their approach did not reduce memory usage. However, their pruning strategy resulted in neurons in which all input weights for a neuron, *and* the weights connected to the neuron's output, were zero. These neurons with "zero input connections or zero output connections" were then pruned from the network with a matching tensor mask.

In this study, the impact of IMP on neuron sparsity (i.e., the percentage of zeroed-out input weights to a given neuron) is examined. Unexpectedly, the results found that IMP yielded no 100% sparse neurons. Fig. 2(B) illustrates conceptually how neuron sparsity increases as the network is pruned. Section IV-B explores this in detail, including how close

neurons were to having all of their input weights be zero across different sparsity levels.

A growing line of research in pruning involves sensitivity analysis, where individual neurons or weights are zeroed out, and the resulting drop in accuracy is measured to estimate their importance to the overall model [3], [12]. Sensitivity analysis reveal that not all neurons contribute equally to a network's prediction; some neurons play a disproportionately important role in determining a network's performance. Section IV-C uses this approach to evaluate the impact of removing each neuron in a layer on the accuracy of the model, as illustrated in Fig. 2(C).

Pieterse et al. [13] implemented structured pruning effects on activation patterns across layers. They used cosine similarity [Eq. (1)] to "determine the importance of a connection based on the activations of the two neurons of that connection". Cosine similarity compares the geometric closeness of activation vectors between pruned and unpruned neurons, providing insights into the preservation of feature representations. Here, cosine similarity is defined as the dot product of two vectors, A and B, normalized by the product of their Euclidean norms.

$$\text{cosine similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

In Section IV-D, the experiments explore unstructured pruning effects on activation patterns across layers using a cosine similarity function. Finally, as shown in Fig. 2(D), pairwise activation cosine similarity matrices are computed for each prunable layer at different pruning levels.

Our use of activation similarity is related to the work of Roychowdhury et al. [14], who used cosine similarity between neuron weight vectors to quantify redundancy. However, such weight-based similarity may not capture functional similarity if upstream activations differ. In contrast, cosine similarity is computed directly on neuron outputs, providing a more complete picture of functional overlap.

III. Methodology

To evaluate the impact of IMP on CNN architectures, the experiment focused on a representative subset of benchmark networks: ResNet20 [15], RegNetX [16], and Vgg16 [17]. As shown in Fig. 3, the networks were trained and pruned using the CIFAR10 [18] and Tiny ImageNet datasets [19]. RegNetX, Vgg16TinyImageNet, Vgg16CIFAR10, and ResNet20 consist of approximately 11.6M, 40.7M, 33.6M, and 272k total number of parameters, respectively.

The experiment focuses on image classification models that can be pruned to fit on resource-constrained devices. CNNs are studied because they have been shown to be significantly compressed with IMP. These models were selected because they are widely used CNN architectures with distinct design philosophies (i.e., residual connections, block scaling, and deep sequential convolutional stacks). Together with the two aforementioned datasets, these models cover both the lightweight and larger scale CNN regimes, making them represent a broad class of widely used networks.

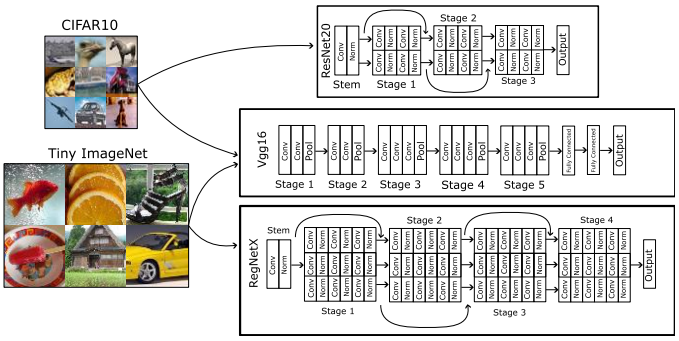

Fig. 3. An illustrative example of the evaluated CNN architectures. From top to bottom: ResNet20, Vgg16, and RegNetX, each shown in their CIFAR10 and Tiny ImageNet configurations. Architectures for Tiny ImageNet are scaled to account for the higher resolution and increased input complexity. Each multi-column layer in ResNet20 and RegNetX is referred to as a "block". This visualization highlights the relative depth, layer types, and structural differences across networks.

Each network underwent 1, 5, or 10 epochs of preliminary training, followed by an additional 30, 80, or 150 training epochs for each network sparsity level. After completing all training for a sparsity level, the network's weights and biases are saved for future analysis. A total of 18 sparsity levels were performed, achieving up to 99% sparsity, following an exponential decay schedule that progressively removed network parameters. This setup followed the work of [9] pruning only the weights of the network. All networks were trained with a batch size of 2048 to maximize hardware utilization. Optimization was carried out using stochastic gradient descent (SGD) with momentum 0.9 and weight decay set to 5×10^{-4} (or 1×10^{-5} for RegNetX). Learning rate schedules varied by architecture: Vgg16 and ResNet20 used a stage decay schedule, while RegNetX employed cosine annealing.

Models Vgg16 and ResNet20 used stage-wise decay schedules, while RegNetX employed cosine annealing. Vgg16's models reduced the learning rate by a factor of 10 every 30 epochs following a fixed interval step decay. On the other hand, ResNet20 had a constant learning rate for the first 50% of training for each pruning stage, then dropped by a factor of 10 until 75%, and was further reduced by another factor of 10 for the remaining epochs in the stage. Finally, RegNetX used cosine annealing during the pretraining phase, gradually reducing the learning rate from an initial value of 0.1 to a minimum of 10^{-6}.

The accuracy of the pruned networks was measured to validate the pruning process. Fig. 4 plots the accuracy of each of the four trained networks against their corresponding sparsity level. The accuracy of all four models is consistent with reported results in the literature for the corresponding model and dataset. He et al. [15] report approximately 91% accuracy on the CIFAR10 dataset using ResNet-20. Similarly, Giuste et al. [20] achieved 93% accuracy with VGG16 on CIFAR10 and approximately 60% accuracy on the ImageNet dataset using the same architecture. Radosavovic et al. [21] report a higher performance, reaching 70% accuracy on ImageNet. Given that the experiments were conducted on the Tiny ImageNet dataset, there was an expected drop in performance. This is because Tiny ImageNet has a lower resolution and

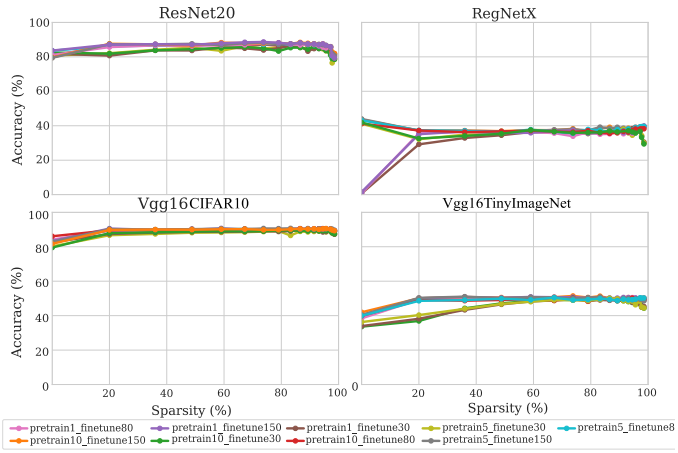

Fig. 4. Accuracy vs. Sparsity. Test accuracy as a function of global weight sparsity for four image classification networks: ResNet20, RegNetX, Vgg16-CIFAR10, and Vgg16-TinyImageNet. Each curve represents a different fine-tuning schedule, applied over 18 pruning iterations.

fewer data points than the full model. TinyImageNet contains 100,000 images across 200 classes with an image size of 64x64. ImageNet is 13 million images across 21841 classes with a resolution size of 256x256 [19].

Fig. 4 also illustrates the relationship between a network's sparsity levels and test accuracy across different architectures with various training schedules. Overall, the results indicate little sensitivity to the training schedules, except for a slight increase in accuracy when more preliminary training is done. This increase is more pronounced for Vgg16 with TinyImageNet. For the baseline RegNetX model (i.e., 0% sparsity), the networks with only one pretraining iteration have near-zero accuracies. One pretraining iteration is not sufficient for the baseline RegNetX model. Interestingly, the accuracy for the same network increases to match the other RegNetX networks after the first iteration of pruning.

The test accuracy remains stable across all networks until 96% sparsity. This suggests that the vast majority of initial parameters in each network are redundant or non-essential to retaining accurate predictions. However, at sparsity levels exceeding 96%, the remaining parameters are essential for representing the data effectively, and further pruning leads to underparameterization. The analysis in Section IV was restricted to sparsity levels below this point, which is called the critical sparsity threshold.

Given the similarity of the accuracy vs. sparsity levels results across various retraining and training configurations, this study focuses the pruned network structure exploration on those networks with 5 preliminary training epochs and 150 training epochs. A detailed analysis of the captured parameters to explore the remaining structure of those networks is detailed in Section IV.

IV. Exploring the Pruned Structure

This section presents four experiments, summarized in Fig. 2, which explore the sparsity patterns that result from IMP. The results provide insights into why IMP is effective and the extent to which structured sparsity emerges, which has implications for memory and energy savings. To this end, IMP is applied across different architectures, as shown in Fig. 3 to prune the "prunable layers", that is, layers with trainable parameter weights such as convolutional (Conv) layers and fully connected layers (FC). Rather than using a mask, the model's weights are directly updated.

This showed that groups of contiguous layers yield nearly the same results for each experiment. Thus, to simplify the discussion and presentation of the results, similar layers are grouped into stages. Each stage consists of multiple convolutional blocks, where each block contains several layers for feature extraction followed by downsampling. These stages represent higher-level groupings of blocks that share similar characteristics (e.g., kernel size, channels, or stride). Fig. 3 shows these layer groupings. For example, Vgg16 stages consist of simple stacks of convolutional layers, ResNet stages are organized with residual blocks, and RegNetX stages use a flexible design with dynamic channel widths.

In the remainder of this section, this study will refer to stages rather than layers. Stages are referred to as either earlier or later, referring to where they occur in the CNN processing pipeline. For example, Stage 1 occurs earlier than Stage 3, as illustrated in Fig. 3.

Section IV-A explores the distribution of weights in the network being pruned, how that distribution affects pruning, and the resulting structure of the network to sparsity levels at each stage (layer) of the network relative to the overall sparsity. Section IV-B analyzes neuron sparsity. We evaluate the significance of neurons at all sparsity levels in Section IV-C. Finally, in Section IV-D we explore cosine similarity between activations of the same network across consecutive sparsity levels and the original, unpruned model.

A. Weight Distributions and Stage Sparsity

The distribution of weights across stages in the network has a significant impact on how IMP prunes the network [Fig. 2(A)]. For example, if all weights in one stage are smaller in magnitude than any weights in another stage, IMP will effectively remove the entire layer. If this occurs during the pruning process, there may be opportunities to remove those layers structurally. This section explores these distributions and how they result in later stages being pruned more than early stages in the network. As noted previously, the stage-level results are consistent with the results for individual layers within said stage.

The probability density functions (PDFs) of these weight distributions can be found in Fig. 5 (for ResNet20 and RegNetX) and Fig. 6 (for Vgg16CIFAR10 and Vgg16TinyImageNet). To simplify the presentation, four representative sparsity levels were selected: 20%, 59%, 79%, and 89%. The PDFs are scaled such that the integrals of a curve indicate the total number of weights in that stage. The later stages (higher indices) have smaller weight magnitudes but larger integrals, which implies they have many small magnitude weights. Earlier stages have fewer and larger magnitude weights.

The PDFs in Fig. 5 and Fig. 6 show that IMP initially prunes from the later stages that contain smaller magnitude

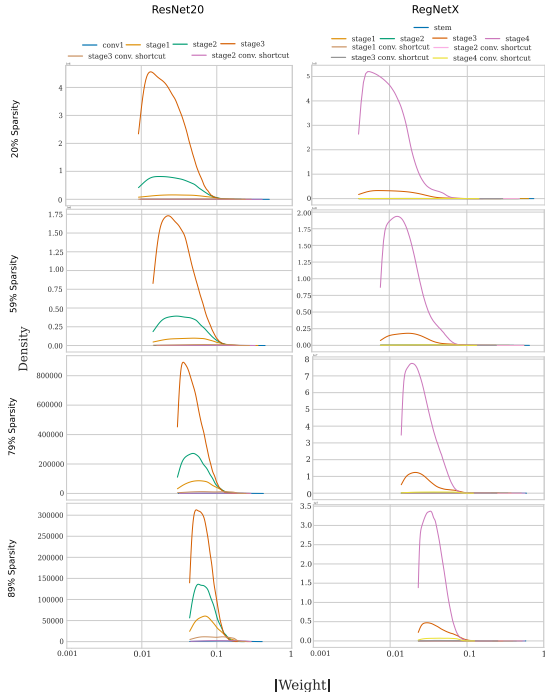

Fig. 5. The absolute value of the weight magnitude distribution across stages for varying sparsity levels for ResNet20 (Left) and RegNetX (Right).

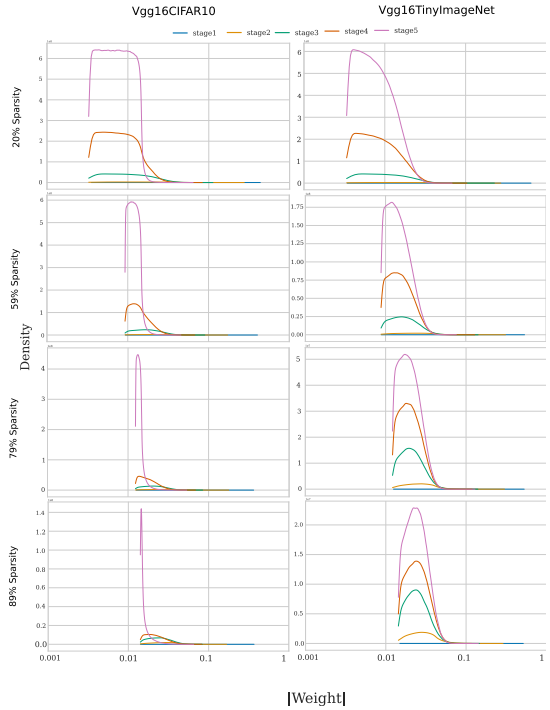

Fig. 6. The absolute value of the weight magnitude distribution across stages for varying sparsity levels for Vgg16CIFAR10 (Left) and Vgg16TinyImageNet (Right).

weights. While weights in earlier stages are still pruned, they experience less aggressive pruning, as illustrated in Fig. 7. Stage sparsity is defined as the percentage of weights pruned within a particular stage, and network sparsity as the overall percentage of pruned weights across the entire network.

We define stage sparsity as the percentage of weights in a stage that have been pruned. In contrast, the network sparsity is the percentage of weights in the *entire* network that have been pruned. In Fig. 7, the stage sparsity for each stage is plotted versus the total network sparsity. Data above or below the unit diagonal indicates that the sparsity of that stage is above or below the network average, respectively. At all network sparsities, the later stages have higher percentages of their weights pruned compared to earlier stages. This is consistent with the observations of the PDFs plotted in Fig. 5 and Fig. 6 that later stages have smaller values relative to earlier stages, resulting in later stages being more aggressively pruned. The figure illustrates these progressive sparsity dynamics up to the critical threshold, prior to the onset of accuracy degradation.

Although not plotted here, the differences in stage sparsity grow with rising network sparsity until the critical network sparsity, about 96% as mentioned previously. At that point, the sparsity of earlier stages increases rapidly. This is consistent with the degrading accuracy observed at this point in Fig. 4.

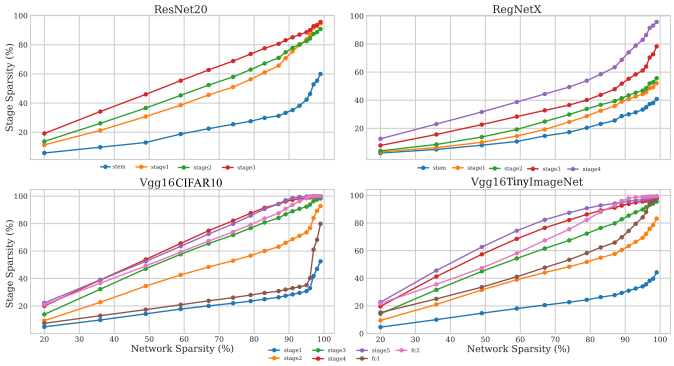

Fig. 7. **Stage-wise sparsity patterns.** Fraction of non-zero weights retained per stage and Fully Connected (FC) layers after the final sparsity level. Early stages are consistently more preserved across networks, while later stages are pruned more aggressively.

B. Neuron Sparsity

Building on insights into stage-wise pruning patterns, this study investigates the sparsity characteristics at the neuron level within these networks [Fig. 2(B)]. Given a highly sparse CNN, it is reasonable to expect that several neurons will exhibit a neuron sparsity of 100%, meaning all weights within the neuron are zeroed out. However, the experiments revealed that this was never observed. There are several reasons why this occurs. Given a constant-width layer, the probability that all the weights on an individual neuron are below the 99th percentile is given by (0.99^x), where x is the layer width. The probability that every neuron in the layer keeps at least one neuron is $(1 - 0.99^x)^x$, which tends towards 1 in an infinitely wide layer, meaning that it should not be expected to see any full neurons removed if pruning at initialization. Furthermore, it is reasonable to assume that after some weights are pruned the remaining weights in a neuron would undergo large gradient updates, further decreasing the likelihood of their removal after training. This is confirmed in Fig. 5 and Fig. 6.

This observation led to an exploration of neuron sparsity patterns within the network throughout the pruning process. The individual neuron sparsities were then grouped by layer to create layer-level neuron sparsity distributions. Tracking these distributions at each sparsity level attempts to uncover patterns in how neuron connectivity evolves. The procedure for doing this is outlined in Algorithm 1.

Algorithm 1 Layer-wise Neuron Sparsity Analysis

Require: Pruned model for all sparsity levels $\{M_s\}_{s=1}^{S}$, list of prunable layers $\mathcal{L}$

for each pruning level $s \in \{1, \ldots, S\}$ **do**
  Load network weights from checkpoint M_s
  for each layer $l \in \mathcal{L}$ **do**
    Extract weight tensor W_l of shape $[n_{\text{out}}, n_{\text{in}}]$
    Compute total weight count $T_l = n_{\text{out}} \times n_{\text{in}}$
    Compute zero weight count $Z_l = \text{count}(W_l == 0)$
    Compute layer sparsity $S_l = Z_l / T_l$
    for each output neuron $i \in \{1, \ldots, n_{\text{out}}\}$ **do**
      Compute neuron-level sparsity $s_{l,i} = \text{count}(W_l[i, :] == 0)/n_{\text{in}}$
    end for
    Store $\{S_l, \{s_{l,i}\}_{i=1}^{n_{\text{out}}}\}$
  end for
end for

Ensure: Per-layer and per-neuron sparsity statistics for all pruning sparsity levels

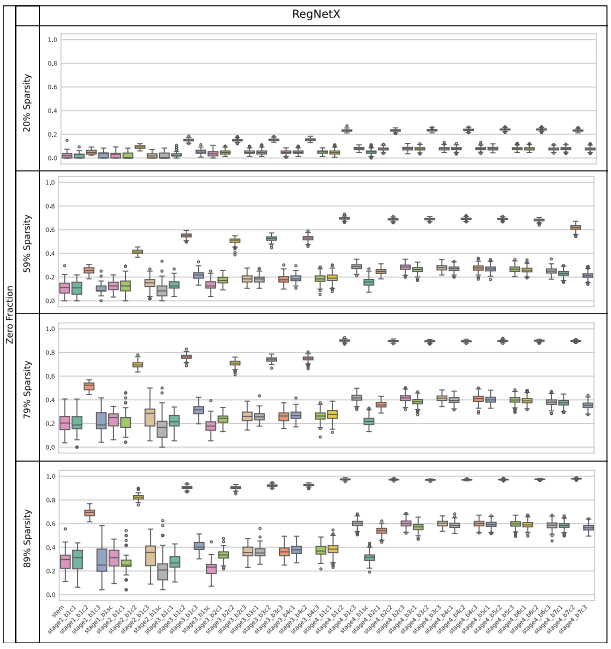

Fig. 8. Neuron sparsity distribution for each layer in RegNetX across different sparsity levels. Later layers show increased sparsity and clustering of inactive neurons, highlighting emergent structural sparsity.

Fig. 8, Fig. 9, and Fig. 10 present box-and-whisker plots of neuron sparsity across layers for each network at multiple sparsity levels. When the median neuron sparsity in a layer differs from the global network sparsity, it indicates that most neurons in that layer have been pruned more or less aggressively than average. This implies that these median values tend to increase with network depth, suggesting that neurons in later layers retain a smaller proportion of their incoming connections. This trend aligns with the stage-wise sparsity patterns shown in Fig. 7, where later layers consistently undergo heavier pruning.

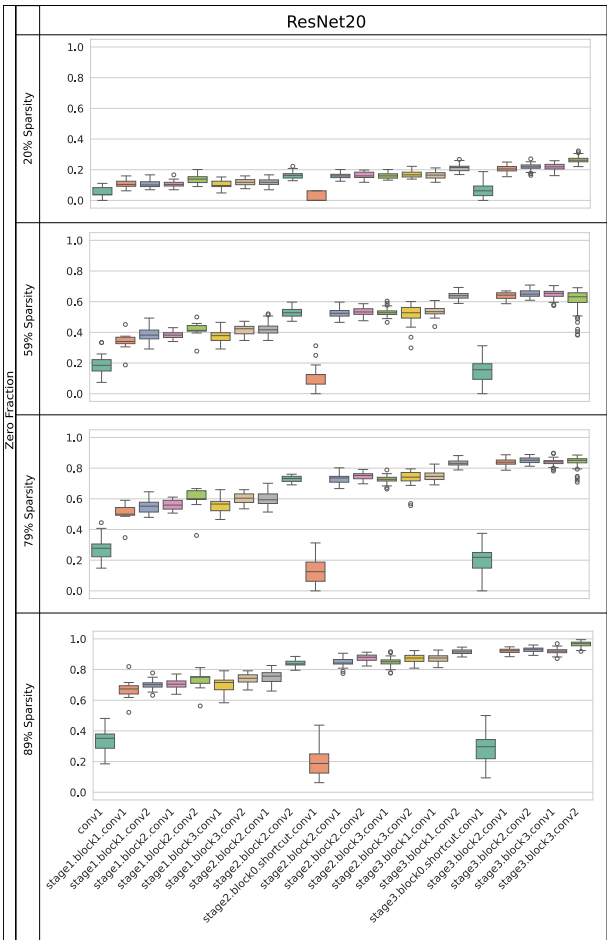

Fig. 9. Neuron sparsity distribution for each layer in ResNet20 across different sparsity levels. Later layers show increased sparsity and clustering of inactive neurons, highlighting emergent structural sparsity.

Layers that belong to the same stage, as indicated by their layer names, typically have similar neuron sparsity distributions. A notable exception to this pattern is the second convolutional layers in RegNetX, which have relatively high sparsities indicating that IMP preferentially prunes these layers. The vertical extent of the plots indicates the width of the distributions. The vertical extents are generally small compared to the variation between stages, especially for layers with high neuron sparsities. This suggests that, generally, in heavily pruned layers, all neurons are similarly sparse. However, even for the latest layers, at the highest network sparsities, no neurons exhibit exactly zero neuron sparsity.

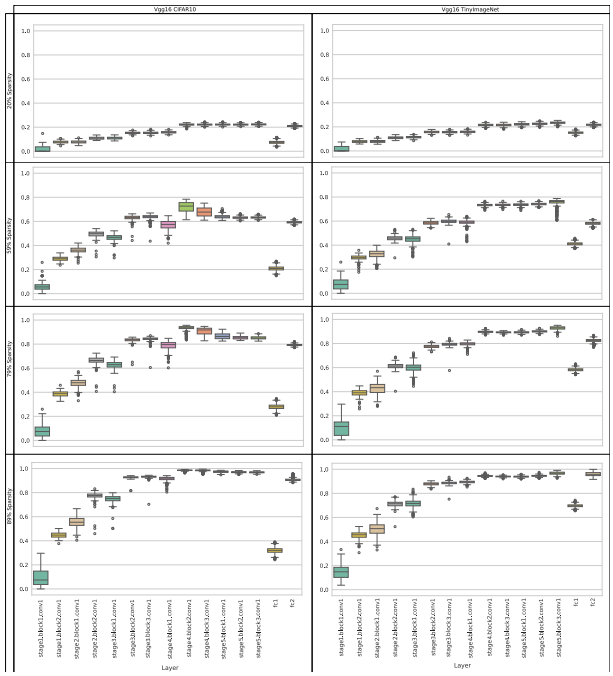

Fig. 10. Neuron sparsity distribution for each layer in Vgg16CIFAR10 and Vgg16TinyImageNet, respectively. Later layers exhibit increased sparsity and grouping of inactive neurons, indicating the presence of emergent structural sparsity.

Our neuron zeroing experiments (discussed in Section IV-C) show that removing neurons from these later layers typically results in a substantial decrease in prediction accuracy. This suggests that naive structured pruning, which removes neurons with the highest sparsity from the pruned network, would not be effective. However, this conclusion overlooks the fact that the neuron zeroing experiments do not involve retraining the networks after zeroing. Retraining following neuron removal could significantly reduce accuracy losses, potentially providing a pathway for structured pruning after IMP, but such an approach is prohibitively expensive for this investigation.

C. Neuron Zeroing

To further understand the contribution of individual neurons, a sensitivity analysis that measures the impact of neuron zeroing on prediction accuracy is conducted [Fig. 2(C)]. Neuron zeroing refers to the process of setting all weights for a specific neuron to zero. Algorithm 2 was applied to each sparsity level, with the results shown in Fig. 11 and Fig. 12.

In this work, accuracy and loss provided similar insights; since accuracy is more commonly plotted, only accuracy is reported.

Fig. 11 and Fig. 12 use box and whisker plots to show the change in accuracy distributions across all neurons. Although only the results for the first layer, two intermediate layers, and the final layer of each network are shown, all other layers show similar results.

These results show that neuron zeroing has a negligible impact on accuracy, with the highest variation being at 1.1% increase in accuracy for the unpruned Vgg16CIFAR10. For the other three cases, the sensitivity to neuron zeroing increases initially with pruning, which may indicate that as unnecessary weights are removed, each neuron is more likely to contribute favorably to the network's accuracy. This is in line with Fig. 5 and Fig. 6, where the weight magnitudes increase with each sparsity level. Then, with further pruning, rather than continuing to increase, the sensitivities remain fairly stable up to the critical sparsity.

Algorithm 2 Neuron Zeroing Experiment

Require: Trained network M, test Dataset TD, prunable layer list $\mathcal{L}$, baseline accuracy $(\text{Acc}_{\text{base}})$

```
for each layer l ∈ L do
    for each neuron i in layer l do
        Save the original weights and biases of neuron i
        Zero out weights and biases of neuron i
        Evaluate the network on TD to obtain Acc_i
        Compute ΔAcc_i = Acc_i − Acc_base
        Restore the original parameters of neuron i
    end for
end for
```

Ensure: Per-neuron sensitivity metrics for all layers

The observation that Vgg16CIFAR10 does not follow this trend may be related to the network's substantial overparameterization for the CIFAR10 dataset. This network contains 33.6 million parameters applied to a relatively low resolution dataset (32×32 image resolution). From the experimental setup, as described in Section III, the network struggles with generalization, as seen in the lower test accuracy of the unpruned network. Generalization refers to the model's ability to extract patterns from the training data and apply these patterns to unseen data, rather than memorizing specific examples. After the initial pruning step, the network shows very low sensitivity to the removal of neurons, suggesting a high degree of neuron redundancy. This behavior persists until 97% sparsity level, at which point, the model exhibits a sharp increase in neuron sensitivity. This occurs when the pruned network retains approximately 1 million parameters, which is comparable to the total parameter count of the unpruned ResNet20 (0.27 million parameters), which shows higher sensitivity from the outset of pruning. The trend is consistent with the hypothesis that, in the experiments, moderate pruning acts as a form of regularization by reducing excess capacity, thereby improving test performance before extreme pruning levels began to degrade accuracy (Fig. 4).

While retraining the network after a neuron removal could reduce accuracy loss, such an approach is computationally intractable for this scale of investigation. Furthermore, the one-shot zeroing experiments isolates the impact of each parameter, providing a strict lower-bound measure of the neuron's contribution to the network's overall accuracy.

D. Activation Similarity Across Sparsity Levels

To understand how a network's activations evolve across different levels of network sparsity using IMP, activation similarity, a measure of layer-wise activation on the same data, was analyzed across sparsity levels [Fig. 2(D)]. Specifically,

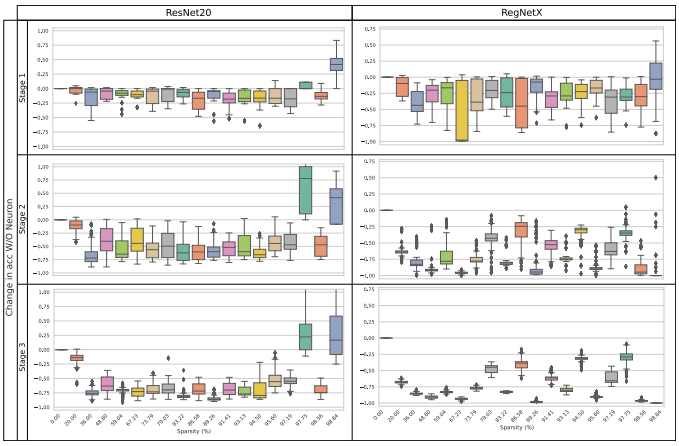

Fig. 11. The average accuracy change per stage (y-axis) from ablating individual neurons in (a) ResNet20 and (b) RegNetX networks, plotted as a function of sparsity levels (x-axis). These plots show the distribution of accuracy changes. As pruning increases, the variance and magnitude of neuron influence grow, indicating increasing functional specialization and reduced redundancy.

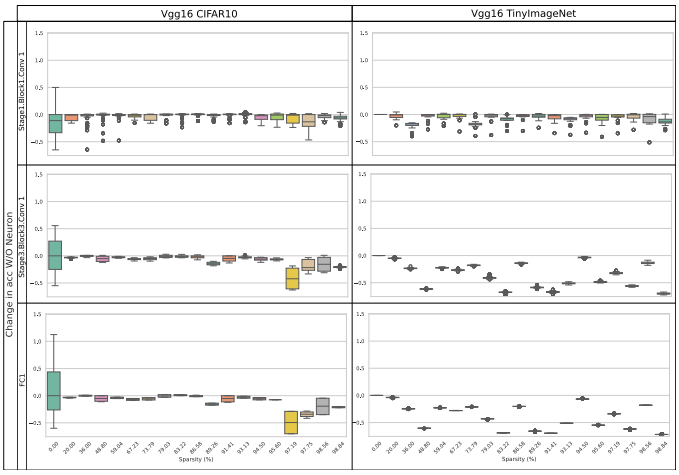

Fig. 12. Neuron zeroing impact across sparsity levels. Accuracy change (y-axis) from ablating individual neurons in (a) Vgg16CIFAR10 and (b) Vgg16TinyImageNet networks, plotted as a function of sparsity levels (x-axis). These plots show the distribution of accuracy changes. As pruning increases, the variance and magnitude of neuron influence grow, indicating increasing functional specialization and reduced redundancy.

the activation outputs of every neuron are recorded on the test set during inference at each sparsity level (after retraining the pruned network). Pairwise activation similarity matrices are then computed for the same layer across different sparsity levels using cosine similarity [Eq. (1)] to quantify changes in representation.

To compute the similarity matrices for each layer, neuron outputs are reshaped into two-dimensional matrices of shape $[\text{samples}, \text{neurons}]$. In convolutional layers, where each "neuron" corresponds to a filter, activations are first flattened to $[\text{samples} \times \text{width} \times \text{height}, \text{filters}]$ before computing cosine similarity. Algorithm 3 details this process.

The computed cosine similarity of layer-wise activation from a given sparsity level to the baseline network is shown

Algorithm 3 Activation Similarity Calculation

Require: Trained network M, test Dataset TD, prunable layer list $\mathcal{L}$, sparsity level s

1: Load network weights at sparsity level s
2: Run inference of M_s on TD, collecting activations A_l for each layer
3: Run inference of M_{s-1} or M_0 on TD, collecting activations B_l for each layer.
4: **for** each layer l **do**
5: Reshape A_l and B_l to $[\text{samples}, \text{neurons}]$
6: Compute similarity score S_l using Eq. (1).
7: **end for**

Ensure: $\{S_l\}_{l \in \mathcal{L}}$

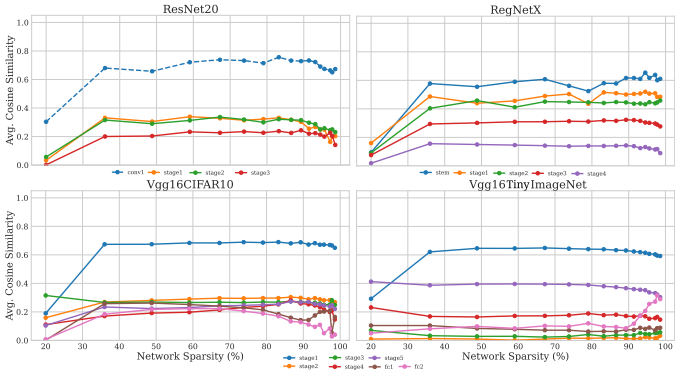

Fig. 13. Activation similarity to baseline network. The cosine similarity between layer-wise activations in the pruned network and the baseline network. This figure shows that early layers retain high similarity even at high sparsity levels, while later layers share little similarity.

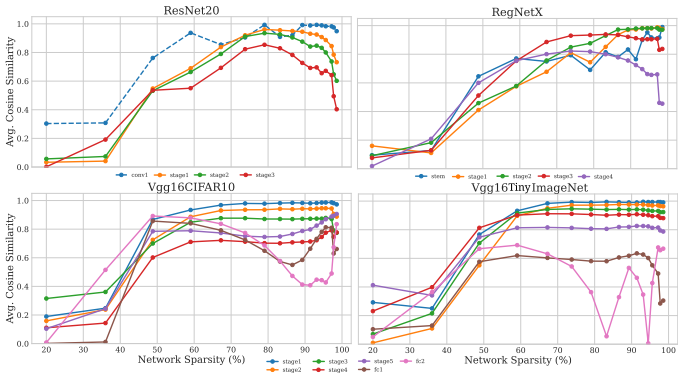

Fig. 14. Step-wise activation similarity across pruning iterations. The cosine similarity between activations of the same network across sparsity levels. The similarity declines gradually with pruning, but drops sharply near the critical sparsity level, signaling large degeneration in the network's representation.

in Fig. 13. It is observed that activation similarities between the sparsity level and the baseline decrease as the stage index increases. That is, later stages are less similar to the baseline than early stages. This is likely because the activations of later stages depend on those of earlier stages, thereby compounding the dissimilarity in subsequent stages.

This persistent similarity of activations to their values in the baseline network may imply that each pruned network consistently finds a similar basin of attraction in the loss landscape. In fact, pruning may increase the likelihood of finding the same basin of attraction by further constraining the solution as weights are set to zero.

The main exception to this activation similarity observation is at a network sparsity of 20%, where the similarities are substantially lower than at other points. This drop in similarity is observed (but not included in the manuscript) even with various pretraining and training epochs. As shown in Fig. 4, the accuracies at 20% network sparsity with 150 training epochs are not particularly low, so this suggests an alternate local minimum in the loss landscape has been identified. More investigation would be required to determine why such alternate minima are not identified at other network sparsities.

Next, similarities are considered between activations of the same network across sparsity levels, as shown in Fig. 14. The first two pruning iterations (20% and 36% sparsity) show that the activation function outputs change drastically between iterations. This is consistent with Fig. 13, where the first pruning iteration has low similarity to the baseline and the second has high similarity. Thus, the 20% sparsity level network is stage-wise dissimilar to the baseline, and then the next iteration is similarly dissimilar to the first iteration (and more similar to the baseline).

Subsequent stage-wise similarities are much higher, as they are similar to the second iteration (36% sparsity). Many of the similarities are even above 60% or even 80%, which is remarkable considering the high dimensionality of the vectors considered. This indicates that the neuron activations become highly similar between pruning iterations for all layers, with the highest similarities observed for the earliest convolutional layers. The stage-wise similarities of the convolutional layers remain high until the critical network sparsity. For the fully connected layers in Vgg, the stage-wise similarities begin to decrease well before the critical sparsity. This may be because these layers are the latest in the architecture and must adapt to changes in earlier layers.

V. Conclusion

This study explores the structure resulting from using IMP to prune a series of CNNs. A detailed analysis of IMP of both neurons and layers across three CNN architectures (ResNet20, RegNetX, and Vgg16) and two datasets (CIFAR10 and TinyImageNet) was conducted. We arrive at four key observations:

- Preferential pruning of deep layers Because CNNs concentrate more parameters in deeper layers and weight magnitudes typically scale inversely with layer width, IMP naturally removes more weights from late stages. This preserves early convolutions, critical for capturing fundamental image features, and explains why accuracy remains high up to extreme sparsities, only degrading as early layers are pruned substantially. The preferential pruning of deep layers suggests that inference performance may be improved by using differing data structures in various parts of the network (e.g., dense matrices in early stages and sparse matrices in later stages).
- Persistence of partially sparse neurons While many neurons retain high sparsity late into pruning, none reach full zeroing. The effect of these "last few weights" limits potential matrix-size reduction and, hence, hardware efficiency. The findings motivate IMP variants that explicitly target the removal of residual weights within neurons.
- Stable neuron-level sensitivity distributions Contrary to expectations, the variance of single-neuron importance does not narrow as pruning progresses: remaining neurons continue to exhibit a broad range of sensitivities. It is observed that one-shot zeroing experiments (without retraining) may overestimate individual neuron importance; future work could apply iterative zeroing with fine-tuning to refine these insights.
- Preservation of feature representations Cosine-similarity analysis show that IMP minimally perturbs activation patterns throughout pruning. This suggests pruned and unpruned networks occupy nearby regions of parameter space, maintaining the representational capacity that underlies accuracy preservation.

Overall, these observations clarify why IMP achieves very high unstructured sparsities without sacrificing accuracy, and they highlight why magnitude-based pruning alone is unlikely to yield structured sparsity patterns conducive to hardware acceleration. Future experiments hope to: 1) develop IMP extensions that incentivize fully zeroing neurons, 2) evaluate the runtime and energy impacts on real hardware, and 3) extend the analysis to other model families (e.g., transformers and graph networks).

Conflicts of Interest

The authors have no conflicts of interest to declare.

Data Availability Statement

The software developed for this research is available at the following link:

https://github.com/NatLabRockies/network-pruner

The data presented in this manuscript can be made available upon reasonable request.

Acknowledgment

This work was authored in part by the National Laboratory of the Rockies (NLR) for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. This work was supported in part by the Laboratory Directed Research and Development (LDRD) Program at NLR. The views expressed

in the study do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the study for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes. A portion of the research was performed using computational resources sponsored by the Department of Energy's Office of Critical Minerals and Energy Innovation and located at the National Laboratory of the Rockies.

REFERENCES

[1] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," *Advances in neural information processing systems*, vol. 2, 1989.

[2] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.

[3] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," *Advances in neural information processing systems*, vol. 1, 1988.

[4] Sietsma and Dow, "Neural net pruning-why and how," in *IEEE 1988 international conference on neural networks*. IEEE, 1988, pp. 325–333.

[5] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. De Freitas, "Predicting parameters in deep learning," *Advances in neural information processing systems*, vol. 26, 2013.

[6] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: efficient inference engine on compressed deep neural network," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. IEEE Press, 2016, p. 243–254. [Online]. Available: https://doi.org/10.1109/ISCA.2016.30

[7] Z. Yang and H. Zhang, "Comparative analysis of structured pruning and unstructured pruning," in *International Conference on Frontier Computing*. Springer, 2021, pp. 882–889.

[8] A. Bragagnolo, E. Tartaglione, A. Fiandrotti, and M. Grangetto, "On the role of structured pruning for neural network compression," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 3527–3531.

[9] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[10] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," *arXiv preprint arXiv:1902.09574*, 2019.

[11] J. Frankle, G. K. Dziugaite, D. Roy, and M. Carbin, "Linear mode connectivity and the lottery ticket hypothesis," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 3259–3269. [Online]. Available: https://proceedings.mlr.press/v119/frankle20a.html

[12] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient transfer learning. arxiv 2016," *arXiv preprint arXiv:1611.06440*, 2016.

[13] J. Pieterse and D. Mocanu, "Evolving and understanding sparse deep neural networks using cosine similarity," *ArXIV*, vol. 2019, pp. 1903–07 138, 2019.

[14] A. RoyChowdhury, P. Sharma, E. Learned-Miller, and A. Roy, "Reducing duplicate filters in deep neural networks," in *NIPS workshop on deep learning: Bridging theory and practice*, vol. 1, no. 2, 2017, p. 6.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 428–10 436.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[18] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[19] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.

[20] F. O. Giuste and J. C. Vizcarra, "Cifar-10 image classification using feature ensembles," 2020. [Online]. Available: https://arxiv.org/abs/2002.03846

[21] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," 2020. [Online]. Available: https://arxiv.org/abs/2003.13678