Is Pruning Compression?: Investigating Pruning Via Network Layer Similarity

Cody Blakeney Yan Yan Ziliang Zong
Texas State University
cjb92@txtstate.edu

Abstract

Unstructured neural network pruning is an effective technique that can significantly reduce theoretical model size, computation demand and energy consumption of large neural networks without compromising accuracy. However, a number of fundamental questions about pruning are not answered yet. For example, do the pruned neural networks contain the same representations as the original network? Is pruning a compression or evolution process? Does pruning only work on trained neural networks? What is the role and value of the uncovered sparsity structure? In this paper, we strive to answer these questions by analyzing three unstructured pruning methods (magnitude based pruning, post-pruning re-initialization, and random sparse initialization). We conduct extensive experiments using the Singular Vector Canonical Correlation Analysis (SVCCA) tool to study and contrast layer representations of pruned and original ResNet, VGG, and ConvNet models. We have several interesting observations: 1) Pruned neural network models evolve to substantially different representations while still maintaining similar accuracy. 2) Initialized sparse models can achieve reasonably good accuracy compared to well-engineered pruning methods. 3) Sparsity structures discovered by pruning models are not inherently important or useful.

1. Introduction

Nowadays, deep artificial neural networks have undoubtedly become the most promising method in solving many challenging computer vision problems [20, 6]. However, the model size and parameter space of successful deep neural networks are typically massive, which prevents them from being deployed on edge-devices (*e.g.,* mobile phones) with limited resources.

To address this problem, pruning has been studied extensively in the literature [9, 5, 2, 23, 17, 13] as an effective technique that can significantly reduce theoretical model size, computation demand and energy consumption of large neural networks without compromising accuracy. The key idea of pruning is to eliminate or mask non-essential components (*e.g.,* less important neurons or negligible weight values) of a deep neural network. Exemplary pruning methods include the early work presented by [10] and a more recent work by Han *et al.* [5]. Since then, a variety of pruning methods, such as parameter pruning and sharing [5, 2, 9], low-rank factorization [16, 7, 21], and compact convolutional filters [22, 18], have been published (ref. Related Work for details). Despite all this progress, our fundamental understanding about pruning is still in its infancy. For example, existing pruning theories and techniques tend to agree with the following hypotheses:

- Hypothesis 1: Pruning is an iterative compressing process. It compresses the original model to a subnet and the representation of the original network remains similar, which is why a pruned network can achieve similar accuracy as the original network.
- Hypothesis 2: A complex model needs to be trained first before it can be pruned. Models with sparsity at initialization are unlikely to succeed.
- Hypothesis 3: Once a deep neural network is pruned and recovered to good accuracy (after retraining), its pruned model structure and weights carry important information and should help improve models trained from scratch.

As more pruning methods being developed, it is time to rethink if these hypotheses that are derived from previous research and practices still hold true and ask the following fundamental questions about pruning. Do the pruned neural networks contain the same representations as the original network? Is pruning truly a compression process? Does pruning only work on trained neural networks? Do we really need sophisticated pruning strategies? What happens if sparsity is chosen randomly?

This paper strives to answer these questions. Specifically, we analyze three fine-grained pruning methods (magnitude based pruning, post-pruning re-initialization, and random sparse initialization). In our experiments, the Singular Vector Canonical Correlation Analysis (SVCCA) tool

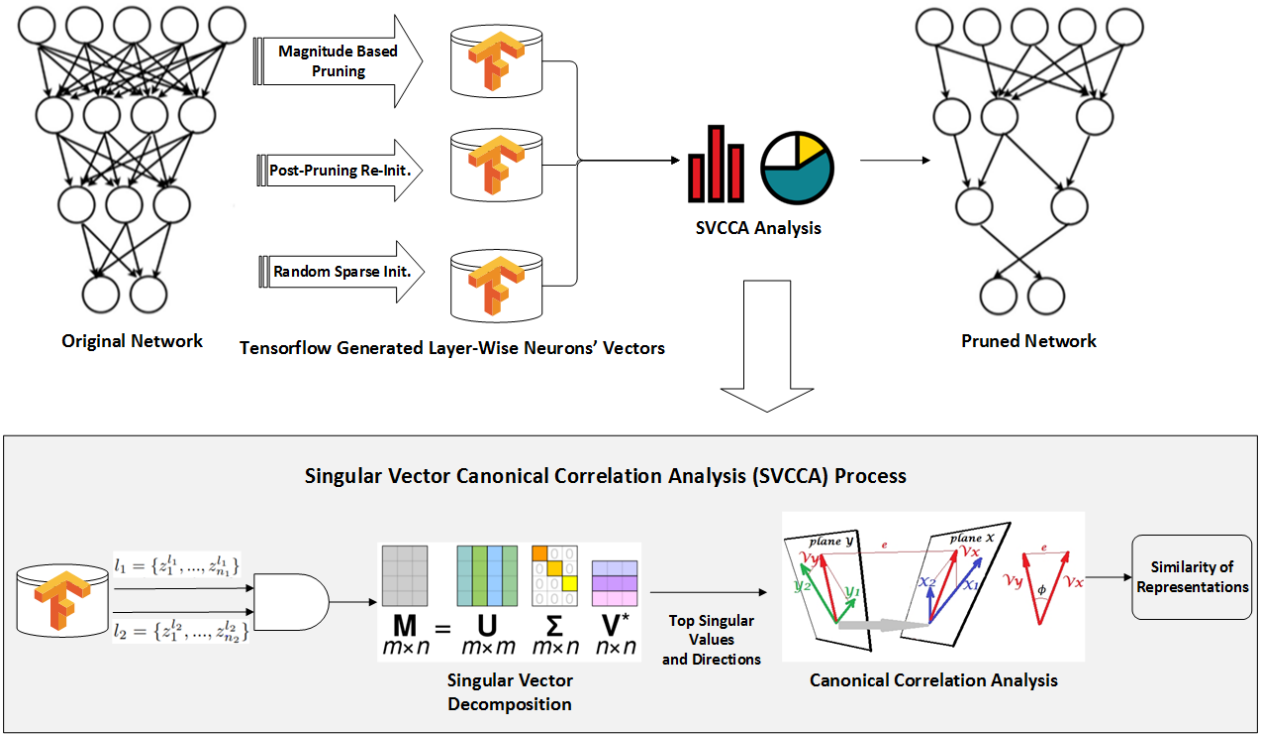

Figure 1: Overview of proposed approach.

[15] is utilized to study and contrast layer representations of pruned and original ResNet [6], VGG, and ConvNet [8] models. We find that: 1) Pruning is not a passive compression process without learning new knowledge. Rather, the pruned model is capable of evolving proactively to survive in a dramatically changed environment, which is done by learning and transforming to more effective representations when aggressive pruning is occurring. 2) Models initialized with sparsity structures can achieve reasonably good accuracy compared to well-engineered pruning methods. 3) Sparsity structures discovered by performing unstructured pruning on models are not inherently important or useful.

Figure 1 illustrates the overview of our proposed approach. We first take an original network (ResNet, VGG, or ConvNet) and prune it using magnitude based pruning, post-pruning re-initialization, or random sparse initialization (ref. section 4 for details) at different sparsities respectively. The neurons' vectors at each layer, which are generated by Tensorflow during the training and pruning process, are then stored and processed by the SVCCA analysis tool created by Google [15] (ref. section 3 for details). Lastly, the accuracy of different pruning methods and the similarity of different representations are analyzed and presented in section 5.

2. Related Work

Deep neural networks have become extremely popular and been successfully used in different applications recently. However, most designed neural networks in machine learning and computer vision field [20, 6] focused on accuracy rather than efficiency. There has been some work on reducing the storage and computation cost by model compression. For example, Lecun Yann had done early work about pruning network which has been investigated in the optimal brain damage work [10]. The basic idea is that different neurons contribute differently in the network. The low ranking neurons can be removed, which results in a smaller and faster network. Recently, Mariet et al. [12] proposed to identify a subset of diverse neurons that do not require retraining to reduce redundancy of the network. In this section, we first review the model compression [3] from three aspects, *i.e.*, parameter pruning and sharing, low-rank factorization, and compact convolutional filters. Afterwards, we briefly discuss the Lottery Ticket Hypotheses and the existing work to explore the nature of network pruning.

Parameter Pruning and Sharing reduces redundant parameters which are not sensitive to the performance. It can be used in both convolutional layers and fully connected layers. Han *et al.* [5] proposed to reduce the total number of parameters and operations in the entire neural network. Chen *et al.* [2] introduced a HashedNet model that used a

low-cost hash function to group weights into hash buckets for parameter sharing. Lebedev *et al.* [9] imposed group sparsity constraint on the convolutional filters to achieve structured brain damage. Zhou *et al.* [23] proposed a group-sparse regularizer on neurons during the training stage to learn compact CNNs with reduced filters. Magnitude-based weight pruning methods are computationally efficient and scalable to large networks and datasets, which makes it become a popular approach for network pruning. See *et al.* [17] showed that weight pruning with retraining was a highly effective method of compression and regularization on a state-of-the-art NMT system, compressing the model to 20% of its size with no loss of performance. Narang *et al.* [13] proposed a technique to reduce the parameters of a network by pruning weights during the initial training of the network. At the end of training, the parameters of the network were sparse while accuracy was still close to the original dense neural network. The network size was reduced by 8x and the time required to train the model remained constant. Anwar *et al.* [1] introduced a three-level pruning of the weights and locate the pruning candidates using particle filtering, which selected the best combination from a number of random generated masks. Polyak *et al.* [14] detected the less frequently activated feature maps with sample input data for face detection applications.

Low-rank Factorization uses matrix or tensor decomposition to estimate the informative parameters. It can be used in both convolutional layers and fully connected layers. Rigamonti *et al.* [16] introduced learning separable 1D filter following the idea of dictionary learning. Jaderberg *et al.* [7] proposed using different tensor decomposition schemes to achieve double speed for a single convolutional layer with 1% drop in classification accuracy in text recognition. Tai *et al.* [21] proposed a new algorithm for computing the low-rank tensor decomposition for training low-rank constrained CNNs from scratch.

Compact Convolutional Filters is to design special structural convolutional filters to save parameters. The approaches can be only used for convolutional layers. Cohen *et al.* introduced Group equivariant Convolutional Neural Networks (G-CNNs), a natural generalization of convolutional neural networks that can reduce sample complexity by exploiting symmetries. G-CNNs use G-convolutions, a new type of layer that enjoys a substantially higher degree of weight sharing than regular convolution layers. Zhai *et al.* [22] proposed doubly convolutional neural networks (DCNNs), which significantly improved the performance of CNNs by further exploring this idea. Instead of allocating a set of convolutional filters that were independently learned, a DCNN maintained groups of filters where filters within each group were translated versions of each other. Shang *et al.* [18] integrated CRelu into several state-of-the-art CNN architectures and demonstrated improvement in their recognition performance on the CIFAR-10/100 and ImageNet datasets with fewer trainable parameters.

Lottery Ticket Hypotheses and The Nature of Network Pruning. After the renewed interest in pruning proposed by [5], many researchers have begun studying what is happening during the process of pruning and how it works. A notable recent work which focuses on unstructured iterative pruning [4] proposes the Lottery Ticket Hypothesis which states "Dense, randomly-initialized, feed-forward networks contain sub-networks (winning tickets) that - when trained in isolation – reach test accuracy comparable to the original network in a similar number of iterations." These subnets as the paper articulates have particularly lucky weights and connections such that they are able to converge through gradient descent towards accuracies as better than the model as a whole. The paper suggests that the effectiveness of deep learning is due in part to the high number of weights and layers which increases the odds of initializing some subnets within the model that can find some good local minima. While the authors claim that it is a combination of both weights and connections that make these "winning tickets", it is unclear to what role, the weights and connections respectively, play. Our work tries to address more thoroughly on how network topology affects accuracy and learned representations.

Another work [11] trying to explain the nature of pruning focuses primarily on structured pruning. They find the resulting model architectures from the structured pruning process are well suited for training from scratch. That is, the resulting architectures (not the found weights) are useful. Their work suggests that structured pruning can be considered as one type of neural network architecture search. The contradictions between their findings and that from [4] show that there is a fundamental difference in what structured and unstructured pruning algorithms do with the original model.

Nevertheless, none of the prior work has studied the fundamental questions that we asked previously. The lack of understanding about the nature of pruning, the learned representations during the pruning and retraining process, and the trend of developing more sophisticated pruning algorithms raises concerns to us. By exploring these questions, this study is distinct from all previous work on neural network pruning.

3. Singular Vector Canonical Correlation Analysis (SVCCA)

Most previous work has been focused on improving different pruning methods without deep understanding about how network pruning really works. In order to take a deep dive into the learning process, it is essential to understand what representations are learned at every stage of training and pruning. Additionally, it is critical to have a tool that

can quantitatively compare two representations and evaluate how similar or different they are. In this study, we leverage the Singular Vector Canonical Correlation Analysis (SVCCA) tool created by [15] to compare and analyze the learned representations of different layers. This section summarizes the SVCCA process and briefly discusses how it functions.

In SVCCA, a series of activations of a neuron is treated as a vector and the model's layers are treated as subspaces that those vectors will span. SVCCA is able to compare the learned representations of any two layers and tell whether or not they have learned the similar or different representations. It does this by combining Singular Vector Decomposition (SVD), which reduces the dimensions of layer subspaces, and Canonical Correlation Analysis (CCA), which maximizes a projection between the two layers that results in the highest correlation. This process allows layers with different configurations of weights, neurons, biases and activation functions to be analyzed. For a given dataset $X = \{x_1, ..., x_m\}$ and a neuron i on layer l the output of that neuron on the entire dataset is defined as the vector z_i^l. SVCCA takes input from two layers $l_1 = \{z_1^{l_1}, ..., z_{n_1}^{l_1}\}$ and $l_2 = \{z_1^{l_2}, ..., z_{n_2}^{l_2}\}$ where n_1 and n_2 are the number of neurons in their respective layers. SVD is performed on each layer to get new subspaces l_1' and l_2' where $l_1' \subset l_1$, $l_2' \subset l_2$. Next, l_1' and l_2' are linearly transformed to be as aligned as possible and correlation coefficients are calculated. SVCCA outputs aligned directions $(\tilde{z}_i^{l_1}, \tilde{z}_i^{l_2})$ and how well they correlate. The higher the correlation value ρ_i is, the more similar the two aligned directions are.

In this study, we primarily focus on the SVCCA similarity $\bar{\rho}$. $\bar{\rho}$ is the mean of the ρ_i values from the top CCA directions and essentially describes how similar the representations of two layers are with each other. The $\bar{\rho}$ values can be used to observe how the learned representations change overtime if similarties of layers are calculated at different time steps. In other words, we use layer similarities as the metric to evaluate the changes in learned representations as models undergo the pruning and retraining process.

4. Pruning Methods

Despite various pruning methods published in the literature, we only evaluate three unstructured pruning methods (magnitude based pruning, post-pruning re-initialization, and random sparse re-initialization) in this study. They are carefully selected to focus on answering the key questions about pruning: 1) do the representations remain identical during the pruning and retraining process? 2) How important is it to carefully design pruning methods? 3) How important is the structure learned from the pruning process

4.1. Magnitude Based Pruning

The magnitude based pruning presented by [5] serves as the baseline method, which first trains an original model to convergence, then prunes each layer by removing the weights with the smallest absolute value until the desired sparsity is reached, and finally retrains the pruned model to recover to a similar accuracy as the original model. The important distinction between our method and Han's approach [5] is that pruning is done layer by layer (not globally on the whole model). We use both an iterative and single shot pruning method, which aims to identify what knowledge is carried over from the original model. We also omit pruning the final fully connected layer as it represents only a tiny portion of the overall weights of a model. We speculate that magnitude based pruning would result in the most similar representations to the original model across layers.

4.2. Post-Pruning Re-initialization

The post-pruning re-initialization method makes copies of each magnitude based pruned model, reinitializes the all variables, but keeps the mask from the magnitude based method. In this way the topology of the pruned network is preserved and the network is trained from scratch for the same number of epochs as the baseline model. The post-pruning re-initialization method is specifically designed to find out: 1) how much capacity is needed for training; 2) if restricting weights to a learned topology will result in more similar representations to the original; and 3) if this topology uncovered by pruning encodes important information about the original model and helps improve models trained from scratch.

4.3. Random Sparse Initialization

To measure the efficacy of the post-pruning re-initialization method, we compare it with a model initialized to the same sparsity where the pruned weights are chosen at random. For each layer in the original model, we randomly prune an index from its weights until the desired sparsity is reached. We then allow those models to train for the same number of epochs as the baseline model. Random sparse initialization sheds lights on how models can be taught when their capacity is reduced in perhaps the least advantageous way possible. The random selection of weights ensures that it cannot take advantage of any architectural structure advantage, and the process may potentially remove important components.

5. Experimental Results

This section presents a series of experiments that are specifically designed to answer the following key questions: 1) Is pruning truly a compression process? 2) Does pruning work for untrained neural networks? 3) Do learned sparse

structures carry important information? For each question, we restate the hypothesis, explain the detailed experiments, discuss the results, and finally draw our conclusions.

5.1. Question 1: Is Unstructured Pruning Compression?

The conventional wisdom believes that pruning is merely a compression process, in which the redundant information is removed and the key network structure is preserved. In fact, almost all existing literature refer pruning as a compress technique for neural networks. The recently published "Lottery Ticket Hypothesis" [4] claimed that successfully trained large networks contain wining tickets from the beginning. The winning tickets refer to the sub-networks that have connections and initial weights that make training particularly effective. Therefore, the pruning process is just a lucky draw that helps to find the wining ticket. If pruning is merely a compression process, a neural network should learn no or minimal new knowledge while being pruned. As a result, the learned representations among all layers and models should have very high similarities (e.g. close to 100%). However, as the authors themselves noted in [4], the discovered lottery ticket winning sub-nets have weights that change the most during the retraining process.

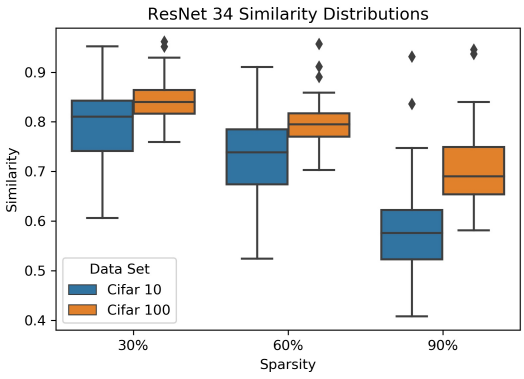

Figure 2: Similarity distributions for Pruned Magnitude ResNet-34 as compared to the original model.

To verify if pruning is truly compression or not, we design and conduct an experiment as follows. We train the original ResNet and VGG models using CIFAR-10 and CIFAR-100 datasets to serve as the baseline. The representations of each layer are recorded and analyzed using the SVCCA tool. Once the baseline model has been trained, we prune it in two ways: 1) iteratively using the magnitude based pruning method, and 2) at incremental sparsity levels of 30%, 45%, 60%, 75% and 90% using magnitude based pruning, post-pruning re-initialization, and random sparse re-initialization. This allows us to observe how models react from moderate pruning towards intensive pruning where

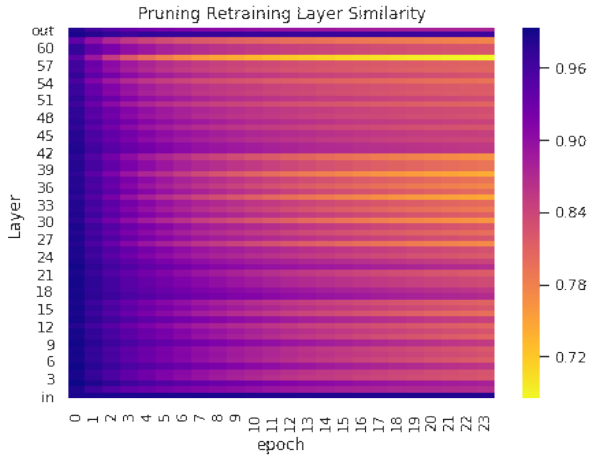

Figure 3: Similarity of 90% sparse ResNet during retraining. Similarities are calculated with respect to the initial state of the model before retraining (note: additional parts of the computation graph like batch normalization and skip layer additions are also calculated resulting in a layer count greater than 50).

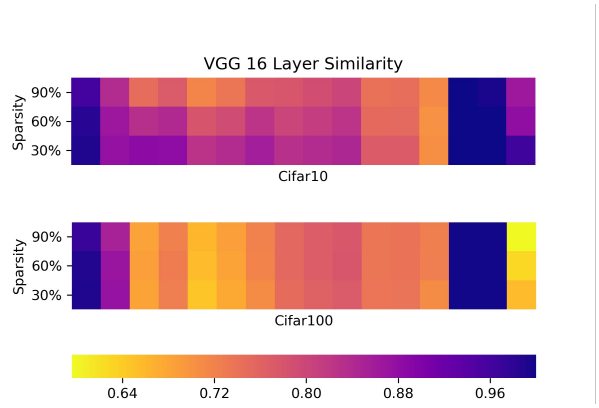

Figure 4: Similarities of VGG layer by layer at different sparsities resulting from iterative pruning. Layers go from left to right with the left most squares being the first convolutional layer and the last three being the fully connected and softmax layers.

they could no longer maintain the desired accuracy. When all pruned models have finished training, we evaluate their accuracy, record their representations, and analyze the similarities using the SVCCA tool. We construct the input function in a way that grantees each model to see the exact same images in the same order, which is critical for preserving the consistency of learned neuron vectors. Lastly, we calculate the SVCCA similarity value $\bar{\rho}$ for each layer and use it as the metric to fairly compare the similarities of different representations.

Figure 2 shows the SVCCA similarity value distributions

for iteratively pruned magnitude ResNet-34 as compared to their original models. We can observe that the learned representations of moderate pruning (*e.g.* 30% of sparsity) remain relatively similar with the original model. However, the more aggressively (*i.e.* the higher the sparsity) a model is pruned, the less similar its learned representations are to the original model it is derived from.

We also investigate in what ways the models are changing while they undergo pruning. Figure 3 shows the similarities using a heatmap for the 90% sparsity model's layers in different epochs while retraining ResNet. This allows us to obtain visual insights about the evolutionary process the model undergoes to regain its accuracy. We observe that the filters that are closer to the output, where presumably high level class features are located, change at the fastest pace. The high level layers reach a steady state in the earliest time and propagate backward to the earlier layers in the model. This is different from the observations reported in [15] for models that are training from scratch, where the lower level layers converge to their final representations first. We believe our results complement [15] well with Google's results by showing how pruned models must repair their high level representations first during the retraining process. In addition, an interesting stripping pattern is observed in the similarities, which is the result from the residual block of ResNet. While further investigation is needed to understand the phenomena, we speculate it is easier for the model to discard the information between the skip connections than to repair the damage to the layers caused by pruning. Figure 4 contains a similar visualization for layer similarities of VGG at different sparsities compared to its original model. Figure 4 shows that the first two convolution layers maintain high similarities in both data sets similar to ResNet's first few layers. The CIFAR-10 VGG model has a pattern of propagating change similar to the ResNet CIFAR-10 model, while the CIFAR-100 model has noticeably more disruption in the middle layers at all sparsity values.

These results clearly demonstrate that pruning does not always result in the same representation. Actually, the aggressive pruning seems to force the network to adapt to the dramatically changed environment. As a result, the learned representations evolve to a new form, probably a more effective one than the originally learned representation. These observations are controversial to the traditional compression theory which treats pruning as a passive process without much of new learning.

5.2. Question 2: Does Pruning Work for Untrained Neural Networks?

It is the common belief that a complex model should be trained with full capacity first before it can be pruned for better efficiency and initialized sparsity in an untrained neural network is unlikely to succeed. To test this hypothesis, we conduct an experiment with sparsly intialized models to see if pruning before training is a viable option, and if so how much model capacity is needed for training. In this experiment, we use the post-pruning re-initialization method. First a baseline mode is trained using TensorFlow and either the CIFAR-10, or CIFAR-100 dataset. Once the baseline model is finished training, the model is pruned to incremental sparsity levels of 30%, 45%, 60%, 75% and 90%. We take those pruned models and reinitialize all of their values, but leave their sparse structures, and let them train for the same number of epochs as the baseline model. Table 1 and 2 show the results of the post-pruning re-initialization accuracy as it compares to the original model, from which we can observe that full capacity is not always needed to train a model from scratch. For all but the most extreme level of pruning (e.g. 90% sparsity), the models can be trained to very similar accuracy as the baseline model. Furthermore, as demonstrated in Table 1, the sparsely initialized models perform equally well and sometimes even better than the original magnitude based pruning strategy for ResNet and ConvNet models.

VGG, however, is an exception. The accuracy of VGG (with initialized sparsity pruning) drops significantly when the sparsity goes beyond 45% (e.g. 10% for CIFAR-10 and 1% for CIFAR-100) and even the magnitude based method drops to 1.1% at 90% sparsity (see Tables 1 and 2). This indicates that sparsities over 45% prevents the VGG model from learning. The recorded test accuracies become only as good as random guessing for both datasets. We tried to adjust the learning rates and reduce or remove the drop out layers, but none of these changes helped the model to learn anything meaningful.

This is interesting because the ConvNet is similar in design philosophy to the VGG model. It has 2 convolutional layers, 2 fully connected layers, and a softmax layer. If redundancy of a model is measured only in number of weights or layers, then the VGG model should be able to survive from more aggressive pruning and still learn. These results agree with the observation by [11] that redundency in VGG is not evenly distributed. We believe ResNet is able to survive these catastrophic collapses during training with initialized sparsity because of its skip connections in the residual blocks. If a single convolution layer is broken or problematic, its deep network is capable of bypassing the layer.

Comparing the accuracies of the iterative pruning methods in Table 3 to that of the single pass magnitude pruning and pre-initialized sparsity methods in Tables 1 and 2, we observe that the real advantage iterative methods have is increasingly better initialization for the model weights and substantially larger training budgets. The original VGG paper [19] explicitly explained how sensitive VGG is to initialization, going as far as to train a smaller model to warm start the larger VGG-16 and VGG-19. This may also explain

Accuracy							
Sparsity		Ref (0%)	30%	45%	60%	75%	90%
ResNet-34	Mag.	92.6%	92.4%	92.5%	91.9%	91.2%	87.2%
	Re-Init.	92.6%	92.1%	92.0%	90.8%	89.9%	88.4%
	Rand.	92.6%	92.4%	91.8%	91.1%	90.2%	87.2%
VGG-16	Mag.	88.7%	88.7%	88.6%	88.2%	87.4%	10.0%
	Re-Init.	88.7%	87.3%	86.9%	10.0%	10.0%	10.0%
	Rand.	88.7%	87.0%	87.0%	10.0%	10.0%	10.0%
ConvNet	Mag.	86.3%	86.1%	85.8	86.0%	86.0%	83.6%
	Re-Init.	86.3%	85.7%	85.6%	84.6%	84.3%	81.0%
	Rand.	86.3%	86.0%	86.2%	85.2%	84.1%	80.1%

Table 1: Accuracy of single pass Pruned Models on CIFAR-10

Accuracy							
Sparsity		Ref (0%)	30%	45%	60%	75%	90%
ResNet-34	Mag.	68.8%	68.2%	68.5%	67.6%	65.8%	54.4%
	Re-Init.	68.8%	68.3%	67.5%	66.1%	65.4%	60.8%
	Rand.	68.8%	68.2%	67.2%	66.4%	64.6%	60.5%
VGG-16	Mag.	56.7%	57.5%	57.25%	56.4%	50.1%	1.1%
	Re-Init.	56.7%	53.1%	56.6%	1.0%	1.0%	1.0%
	Rand.	56.7%	57.4%	56.2%	1.0%	1.0%	1.0%
ConvNet	Magn.	58.5%	58.5%	58.3	57.6%	58.1%	53.4%
	Re-Init.	58.5%	57.9%	57.9%	58.0%	58.4%	58.4%
	Rand.	58.5%	58.3%	58.0%	58.8%	58.0%	58.%

Table 2: Accuracy of single pass Pruned Models on CIFAR-100

why iterative and single pass magnitude pruning are able to succeed at higher sparsities while the initialized sparsity methods can not.

These results overturn the conventional wisdom that models must be first trained in full capacity before they undergo pruning. When attempting to design efficient models, new methodologies thus can be developed to speed up the pruning/training process by training models with some amount of sparsity from the beginning.

Accuracy						
Sparsity		Ref (0%)	60%	70%	80%	90%
CIFAR-10	ResNet-34	92.6%	92.9%	93.0%	92.6%	91.7%
	VGG-16	88.7%	89.2%	89.9%	88.5%	87.9%
CIFAR-100	ResNet-34	68.8%	69.3%	69.3%	68.5%	60.7%
	VGG-16	56.7%	63.0%	63.4%	63.7%	63.4%

Table 3: Accuracy of Iteratively Pruned Models

5.3. Question 3: Do Learned Sparse Structures Carry Important Information?

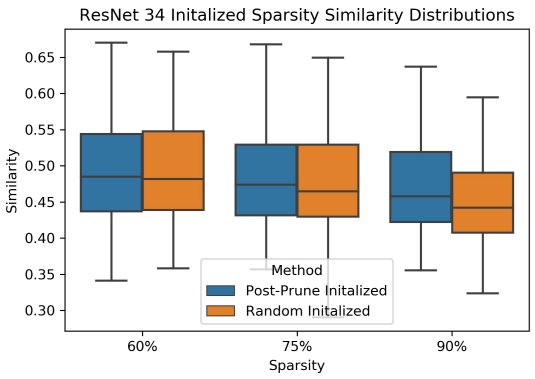

Figure 5: Similarity distributions for Post-Pruning Re-initialization and Random Sparse Initaial ResNet-34 as compared to the original model.

Sparse structures resulting from structured pruning as in [11] has been demonstrated to have significant value. It is less clear what if any value network topology that is a result of unstructured pruning has. We design an experiment in this subsection to verify if the learned structure of the sparsity is important and if random sparsity can yield good accuracy.

In this experiment, we take the baseline model and randomly select weights in each layer to prune until the desired sparsity is reached. We prune to the same sparsities described in the previous experiment, and allow the model to train for the same number of epochs as our baseline model. Tables 1 and 2 compares the results of the learned pre-pruning method to the random pre-pruning method. In contrast to [11]'s findings for structured sparsity, there is virtually no distinction in performance between our random sparse initalized and post-pruning re-initialization methods. Our random initialized sparsity method does nearly as well or better than the post-pruning initialized method at all sparsities. ResNet-34 model preforms only 1% less in accuracy compared with the time-consuming magnitude based pruning method at the different sparsity levels. It also performs as well as magnitude based pruning methods at the 90% sparsity level for CIFAR-10 and outperforms magnitude based pruning for CIFAR-100. We also show that models that have prior knowledge of good sparse structures do not necessarily preform better. The only requirement is not to be over-aggressive for capacity (e.g. use 90% or higher sparsity). As long as moderate amounts of weights are available, the model will find a way to adapt and overcome its lower capacity and still learn. This indicates that not only can pre-pruning be effective, but it can be easily implemented using randomness.

We have already demonstrated that post-pruned re-initialized models do not have any significant difference in performance to randomly intialized sparse models. Purhaps as [4] suggest the data intensive process of unstructured pruning will create a sparsity structure that contains a certain bias. To further investigate if the learned sparsity structures carry valuable information, we analyze the SVCCA similarity value distributions for the post-pruning initialized and random initialized ResNet-34 models compared with the original ResNet-34 model (see Figure 5). It can be ob-

served that the median similarity values of the representations are less than 50% at all sparsities and the distinction between the randomly initialized and post-prune initialized methods is negligible. We also observe that as the level of sparsity increases, the similarities of post-prune initialization actually decreased (it should increase if the sparse structures really carry important information), which indicates that the learned sparse structures do not carry valuable information.

6. Conclusion

As an effective technique to reduce the size of modern neural networks, pruning has been extensively studied in recent years. However, the current understanding about pruning is still in its early stage with numerous misconceptions and inappropriate hypotheses. This paper explores several fundamental questions about pruning and strives to find out 1) if pruning is truly compression; 2) if pruning can work on untrained neural networks; and 3) if sparsity structures from unstructured pruning provide valuable information. The following conclusions can be drawn from our experiments and observations.

First, after analyzing the similarities of various learned representations using the SVCCA tool, we find that pruning is not a passive compression process without learning new knowledge. Rather, the pruned model is capable of evolving proactively to survive in a dramatically changed environment, which is done by learning and transforming to more effective representations when aggressive pruning is occurring.

Second, our results indicate that initialized sparsity can work for untrained neural networks with certain architectures or capacities. Taking advanced knowledge from one model (e.g. which weights are important) and using it to train another model will not result in any significant performance gains when training.

Third, we observe that the sparsity structure from the post-pruning re-initialization is not inherently useful or meaningful. Random initialized sparse models perform equally well compared to post-pruning re-initialized sparse models. The similarity of their layer representations share almost no connection or valuable information to the original models where they derive from. This is in contrast to structured pruning where the discovered structures can be used to train efficient models from scratch.

ACKNOWLEDGMENT

The work reported in this paper is supported by the U.S. National Science Foundation under Grant No. CNS-1908658.

References

[1] S. Anwar, K. Hwang, and W. Sung. Structured pruning of deep convolutional neural networks. In *arXiv preprint arXiv:1512.08571*, 2015.

[2] W. Chen, J. Wilson, T. S., W. K. Q., and C. Y. Compressing neural networks with the hashing trick. In *JMLR workshop*, 2015.

[3] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. A survey of model compression and acceleration for deep neural networks. *IEEE Signal Processing Magazine*, 2018.

[4] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[5] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural networks. In *NIPS*, 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2015.

[7] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolitional neural networks with low rank expansions. In *BMVC*, 2014.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

[9] V. Lebedev and V. Lempisky. Fast convnets using group-wise brain damage. In *CVPR*, 2016.

[10] Y. Lecun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *NIPS*, 1990.

[11] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019.

[12] Z. Mariet and S. Sra. Diversity networks. In *ICLR*, 2016.

[13] S. Narang, G. F. Diamos, S. Sengupta, and E. Elsen. Exploring sparsity in recurrent neural networks. In *arXiv preprint arXiv: 1704.05119*, 2017.

[14] A. Polyak and L. Wolf. Channel-level acceleration of deep face representations. In *IEEE Access*, 2015.

[15] M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *NIPS*, pages 6076–6085, 2017.

[16] R. Rigamonti, A. Sironi, V. Lepetit, and F. P. Learning separable filters. In *CVPR*, 2013.

[17] A. See, M.-T. Luong, and C. D. Manning. Compression of neural machine translation models via pruning. In *CoNLL*, 2016.

[18] W. Shang, K. Sohn, D. Almeida, and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *ICML*, 2016.

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

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[21] C. Tai, T. Xiao, and X. Wang. Convolutional neural networks with low-rank regularization. In *arXiv preprint arXiv:1511.06067*, 2015.

[22] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang. Doubly convolutional neural networks. In *NIPS*, 2016.

[23] H. Zhou, A. J. M., and P. F. Less is more: Towards compact cnns. In *ECCV*, 2016.