On the Similarity between Hidden Layers of Pruned and Unpruned Convolutional Neural Networks

Alessio Ansuini[2][a], Eric Medvet[1][b], Felice Andrea Pellegrino[1][c], and Marco Zullich[1][d]

[1]*Dipartimento di Ingegneria e Architettura, Università degli Studi di Trieste, Trieste, Italy*

[2]*International School for Advanced Studies, Trieste, Italy*

marco.zullich@phd.units.it,{fapellegrino,emedvet}@units.it, alessioansuini@gmail.com

Keywords: Machine Learning, Pruning, Convolutional Neural Networks, Lottery Ticket Hypothesis, Canonical Correlation Analysis, Explainable Knowledge

Abstract: During the last few decades, artificial neural networks (ANN) have achieved an enormous success in regression and classification tasks. The empirical success has not been matched with an equally strong theoretical understanding of such models, as some of their working principles (training dynamics, generalization properties, and the structure of inner representations) still remain largely unknown. It is, for example, particularly difficult to reconcile the well known fact that ANNs achieve remarkable levels of generalization also in conditions of severe over-parametrization. In our work, we explore a recent network compression technique, called Iterative Magnitude Pruning (IMP), and apply it to convolutional neural networks (CNN). The pruned and unpruned models are compared layer-wise with Canonical Correlation Analysis (CCA). Our results show a high similarity between layers of pruned and unpruned CNNs in the first convolutional layers and in the fully-connected layer, while for the intermediate convolutional layers the similarity is significantly lower. This suggests that, although in intermediate layers representation in pruned and unpruned networks is markedly different, in the last part the fully-connected layers act as *pivots*, producing not only similar performances but also similar representations of the data, despite the large difference in the number of parameters involved.

1 INTRODUCTION

The present paper aims at exploring and connecting two recent works on the topic of theoretical understanding on artificial neural networks (ANNs) working principles.

The first of these works (Frankle and Carbin, 2019) presents an iterative pruning technique, named Iterative Magnitude Pruning (IMP), of the parameters of a neural network allowing to obtain sparsity levels on the parameters space up to 1% of the original, while still matching the unpruned (*complete*) neural network test performance. The technique consists in iteratively pruning a fixed proportion of parameters having small magnitude, rewinding the network to initialization, and re-training only the smaller subnetwork identified by the non-pruned parameters.

The second work (Raghu et al., 2017) introduces Singular Vector Canonical Correlation Analysis (SVCCA), a technique based on Canonical Correlation Analysis (CCA) to compare two generic real-valued matrices sharing row dimension. SVCCA is applied to compare ANN layers by enabling a layer to be represented as the matrix of activations of its neurons in response to a dataset of fixed, finite size. The application of CCA yields a vector of correlations, which can be averaged to obtain a similarity measure between layers, called *Mean CCA Similarity*.

After training convolutional neural networks with *minimal* architecture on an ensemble of problems of incremental difficulty based on CIFAR10[1], we continue by pruning those models using IMP for a desired number of iterations, obtaining a set of pruned networks, one for each problem and iteration number. For each of those networks, we compare each layer with its unpruned counterpart using SVCCA (after evaluating the layer activations on a subset of the

[a] https://orcid.org/0000-0002-3117-3532
[b] https://orcid.org/0000-0001-5652-2113
[c] https://orcid.org/0000-0002-4423-1666
[d] https://orcid.org/0000-0002-9920-9095

[1] `https://www.cs.toronto.edu/~kriz/cifar.html`

original dataset). The obtained similarities are then analyzed relative to the pruning ratio, the location of the layer within the network, and the difficulty of the classification task.

The remainder of this paper is organized as follows. Section 2 describes the relevant related works. Section 3 briefly presents the two aforementioned works giving an outline of the methodologies and the underlying theory. Section 4 introduces the experimental work, describing the problems on which our networks have been trained, and explaining how the tools have been applied on these models. Section 5 presents the numerical results of both the test performance of the pruned networks, and the similarity between layers of pruned and unpruned networks, elaborating an interpretation of these ones; moreover, it formulates some prompts for future work.

2 RELATED WORK

The contribution of the present paper is the following: by combining two recent advances addressing over-parametrization (Frankle and Carbin, 2019) and hidden layers representation and comparison (Raghu et al., 2017) in ANNs, we aim at providing a layer-wise analysis of similarity in the representation of CNNs for vision tasks. We here review previous works that are somehow related to our aim.

2.1 Pruning techniques for ANNs

Pruning techniques for ANNs have been proposed for decades. Early attempts include L1 regularization on the loss function (Goodfellow et al., 2016) in order to induce sparsity in the parameters, or operating pooling on the fully-connected layer(s) (Lin et al., 2013).

(Han et al., 2015) introduced a new technique based pruning parameter with small magnitude, on which IMP (Frankle and Carbin, 2019) is based upon.

More recently, a plethora of other techniques has been proposed, like ADMM (Zhang et al., 2018), or techniques for structured (block) pruning, summarised in (Crowley et al., 2018).

Our paper focuses solely on IMP, while considerations on other pruning techniques is left for the future.

2.2 Comparison of ANNs

Despite being a recent work, (Raghu et al., 2017) has already prompted a number of researches utilizing CCA in order to gain some knowledge on the similarities between neural networks: for instance, (Wang et al., 2018) use it to compare, layer-wise, the same network when initialized differently, finding that "surprisingly, representations learned by the same convolutional layer of networks trained from different initializations are not similar [...] at least in terms of subspace match".

(Morcos et al., 2018) argued about weaknesses of Mean CCA Similarity, instead proposing a new similarity metric for layers, called Projection Weighted Canonical Correlation Analysis.

On the other hand, other researches have introduced different methodologies to achieve the comparison: (Yu et al., 2018), for example, proposed a technique based upon the Riemann curvature information of the "manifolds composed of activation vectors in each fully-connected layer" of two deep neural networks. This technique is still at an early stage since it enables comparison on fully-connected layers only, and cannot be used for an analysis like ours.

(Kornblith et al., 2019), instead, offered some considerations on CCA as a tool for layers comparison in neural networks, arguing that it cannot "measure meaningful similarities between representations of higher dimensions than the number of data points", hence proposing yet another methodology called Centered Kernel Alignment.

2.3 Pruned vs. unpruned ANNs

To our knowledge, ours is the first work concerning an in-depth, layer-by-layer analysis of the *similarities* for pruned ANNs.

(Frankle and Bau, 2019) delved into the mechanics of IMP (and other related magnitude pruning techniques) by analyzing the *interpretability* (computed through the identification of "convolutional units that recognize particular human-interpretable concepts") of those networks, finding that pruning does not reduce it and prompting the conclusion that "parameters that pruning considers to be superfluous for accuracy are also superfluous for interpretability". This work does discuss the topic of pruned networks comparison, but it is rather a global analysis, not going into the detail of the single layers. This work may be thought of as an attempt, akin ours, to combine pruning techniques with other recent advances in order to gain additional insights on what may be called "pruning dynamics".

(Morcos et al., 2018), instead, use CCA to compare output layers of fully-trained, dense CNNs having different number of filters in their convolutional layers. The authors of the cited work attempt to corroborate the Lottery Ticket Hypothesis (see Section 3.1), but, in doing this, they do not actually operate any pruning, neither they compare hidden layers, focusing solely on the output representation.

3 TOOLS

3.1 Iterative Magnitude Pruning

Iterative Magnitude Pruning (IMP) is an algorithm first introduced in (Frankle and Carbin, 2019) to operate pruning on the parameters of a generic ANN. The authors start by formulating a hypothesis, called *Lottery Ticket Hypothesis*, which claims the following:

> dense, randomly-initialized, feed-forward networks contain subnetworks ("winning tickets") that—when trained in isolation—reach test accuracy comparable to the original network in a similar number of iterations

This substructure may be found via an iterative method in which the parameters with the lowest magnitude get progressively skimmed from the model until a target sparsity is reached.

Denoting by $\odot$ the element-by-element matrix multiplication, and defined the *pruning rate* as the proportion $p \in [0,1]$ of parameters we want to prune from the network, the algorithm is the following:

Algorithm 1 IMP.

```
1: Randomly initialize parameters in neural network, store them in structure $\Theta_0$;
2: Create trivial pruning mask $M$ with same structure as $\Theta_0$, initialize it at 1;
3: Train the network for $T$ iterations, store the parameters in $\Theta_T$;
4: Obtain the parameters of $\Theta_T$ whose magnitude falls below the $p$-th percentile; set the corresponding mask entries to 0;
5: Apply the mask to the initial parameters $\Theta_0$, obtaining a new initialization: $\Theta_0^{(1)} = \Theta_0 \odot M$;
6: Re-train the network for other $T$ iterations, obtaining a new final configuration $\Theta_T^{(1)}$;
7: Repeat 3–6, each time pruning only parameters having a corresponding entry of 1 in the previous mask, until a target sparsity rate is reached, or performance falls below a desired threshold.
```

The authors showed that this algorithm effectively finds winning tickets for shallow fully-connected and CNNs, but fails to do so on deeper architectures, such as VGG-19 (Simonyan and Zisserman, 2014) and ResNet 18 (He et al., 2016), unless a warm-up phase is employed. (Frankle et al., 2019) showed that, by rewinding to an early-training stage of the unpruned network (instead of rewinding at initialization), the algorithm enjoys a stabler parameters configuration and is able to converge to a solution whose performance is similar to (or better than) the complete neural network.

The method for determining the mask in IMP is referred to in (Zhou et al., 2019) as LF-Mask (*Large-Final Mask*). They produced a plethora of experiments using a different number of masks and rewinding policies (i.e., the scalar at which each parameter having a value of 0 within the mask gets rewound to) showing that, generally, the LF-Mask performs better than all of their other proposals.

3.2 SVCCA

SVCCA is a technique introduced in (Raghu et al., 2017), and later refined in (Morcos et al., 2018), which enables the comparison between two matrices sharing row size.

It can be applied to two generic layers of a fully-connected neural network when they are represented as the response of their neurons to the same fixed-size dataset. Taking a layer L_1 of m_1 neurons, L_2 of m_2 neurons, by feeding n distinct datapoints to their respective neural networks, we may store the neurons' response for each of those datapoints in matrices: L_1 will be represented by an $n \times m_1$ matrix, L_2 by an $n \times m_2$ matrix. The authors propose to compare those two representations using Canonical Correlation Analysis (CCA), which finds two linear transforms $W_1 \in \mathbb{R}^{m_1 \times \tilde{m}}, W_2 \in \mathbb{R}^{m_2 \times \tilde{m}}$, where $\tilde{m} = \min(m_1, m_2)$, which, applied to L_1, L_2, yield two sets of $\tilde{m}$ unit vectors Z_1, Z_2:

$$Z_1 = L_1 W_1 \quad (1)$$

$$Z_2 = L_2 W_2 \quad (2)$$

The two transformations W_1, W_2 are determined such that the components of Z_1, Z_2 are pairwise orthogonal and maximize the residual mutual Pearson correlation. This correlation is called *canonical correlation*: CCA hence yields $\tilde{m}$ values of canonical correlation: by averaging them, a measure of similarity between two layers is obtained, which the authors call *Mean CCA Similarity*.

Moreover, it is suggested that, by operating on the two layers a Singular Value Decomposition (SVD) for dimensionality reduction, with the aim of keeping only the singular values accounting for 99% of the variance, one may avoid some degenerate layers configurations which would have produced overestimations in the Mean CCA Similarity. This technique (SVD + CCA on the layers represented as matrices) has been named Singular Vector Canonical Correlation Analysis (SVCCA).

Table 1: Categories composing each dataset described in Section 4.1.

Dataset	Categories
Cifar2	Automobile, Truck
Cifar4	Cifar2 + Airplane, Ship
Cifar6	Cifar4 + Cat, Dog
Cifar8	Cifar6 + Deer, Horse
Cifar10	Cifar8 + Bird, Dog

4 METHODS

As stated before, we aim at investigating, using SVCCA, the similarities between pruned and unpruned layers in convolutional neural networks, the pruned models being obtained via IMP.

For achieving this goal, we:

1. train an ensemble of convolutional neural networks for image classification on subsets of the Cifar10 dataset;
2. prune those networks using IMP;
3. compare the pruned layers with their unpruned counterparts using CCA Mean Similarity.

All of the implementations supporting the results described in this paper were produced on Python 3.6.4, using libraries PyTorch 1.3.0, and NumPy 1.17.2; moreover, Google's own implementation for SVCCA[2] was used.

4.1 The problems

Cifar10 is a publicly available collection of 60 000 labelled color images of size 32×32, divided into 10 classes of 6000 images each. The dataset is already split into train (50 000 images) and test (10 000 images) set.

In order to increase the number of observations, meanwhile generating correlated measurements, we built multiple subsets over the dataset so as to obtain a set of incrementally more difficult problems. Cifar10 was subset over 2, 4, 6, 8 select image classes, and the related dataset was called Cifar2, Cifar4, etc.; the selection of classes was not operated randomly in order not to create trivial problems, but the chosen categories had to show, to some extent, some similarities. The composition of all the categories is listed in Table 1.

[2] `https://github.com/google/svcca`

Table 2: Summary of the architectures for the CNNs for each of the aforementioned problems. MP stands for Max-Pooling; * indicates a layer with batch normalization; § indicates a layer with dropout. Concerning training epochs T, $^\#$ indicates deployment of early stopping with patience of 20 epochs and a validation dataset obtained on a stratified random sampling of 10% of the training set observations.

Problem	Conv. layers	Full layers	T
Cifar2	16^*, 16, MP	$256^\S$, 10	50
Cifar4	16^*, 16, MP, 32^*, 32, MP	$256^\S$, 10	50
Cifar6	64^*, 64, MP, 128^*, 128, MP, 256^*, 256, MP	$256^\S$, 10	100
Cifar8/10	64^*, 64^*, MP, 128^*, 128^*, MP, 256^*, 256^*, MP, 512^*, 512^*, MP	$1024^{\S *}$, 10	$200^\#$

4.2 Convolutional Neural Networks Architectures

The convolutional neural networks we designed were based off of VGGNet core (Simonyan and Zisserman, 2014) Namely, the architecture consists in stacking one or more convolutional blocks composed of 2, 3, or 4 convolutional layers followed by a max-pooling layer, to a fully-connected layer, eventually followed by the output layer, having softmax activation function, and number of neurons equal to the number of classes.

For our networks, we decided to employ batch normalization (Ioffe and Szegedy, 2015) on some hidden layers (both convolutional and fully-connected) and dropout (Srivastava et al., 2014) on fully-connected layers only. The architectures and dataset-specific hyperparameters are listed in Table 2.

All the networks were trained using Adam optimizer (Kingma and Ba, 2014) with mini-batch size of 128, learning rate equal to 0.001, and weight decay (L2 regularization) with parameter 0.0005. All of the layers use the Rectified Linear Unit activation function (except for the output layer, which uses the softmax activation function). Moreover, for each training mini-batch, a random data augmentation strategy was applied consisting in a composition of cropping, horizontal flipping, and roto-translation.

4.3 Application of IMP

IMP was applied with a strategy similar to (Zhou et al., 2019): we established two separate pruning rates, p_{conv} for the convolutional layers parameters,

Table 3: Choices for pruning rates for the convolutional layers and fully-connected layer used during the application of IMP.

Problem	p_{conv}	p_{fc}
Cifar2	0.1	0.2
Cifar4	0.1	0.2
Cifar6	0.1	0.2
Cifar8	0.2	0.2
Cifar10	0.2	0.2

and p_{fc} for the fully-connected layer parameters, and operated the pruning separately for convolutional layers (pooling together all the weights and biases pertaining to those layers and pruning the p_{conv}-th parameters with the smallest magnitude) and the fully-connected layer. The choices for pruning rates are shown in Table 3.

After training, IMP was applied for 20 iterations. At each iteration, the network was rewound at the third epoch, in order to stabilize the algorithm, as indicated in Section 3.1.

4.4 Layers Comparison

Since CCA, as described in Section 3.2, works on matrices, and the convolutional layers, if represented, as in Section 3.2, as the response of their neurons to a given dataset, are quadridimensional tensors (*datapoints* $\times$ *channels* $\times$ *vertical image size* $\times$ *horizontal image size*), (Raghu et al., 2017) proposed to collapse the dimensions corresponding to the image size into the datapoints dimension, thus reshaping the tensor in a matrix of shape (*datapoints* $\cdot$ *vertical image size* $\cdot$ *horizontal image size*) $\times$ *channels*.

Once the pruned networks were obtained, we proceeded as follows. For each problem:

1. We randomly subset the (non augmented) training dataset (problem specific) on 5000 datapoints.
2. We evaluated each network (pruned and unpruned) on this subset, storing the representation of each of layer for each network. We name $n_i = \{L_i^{(0)}, \dots, L_i^{(K)}\}$ the network pruned at i-th iteration of IMP, represented as the set of its layers, $L_i^{(0)}$ being the input layer, $L_i^{(K)}$ being the output layer; henceforth, n_0 represent the unpruned network. Note that K is problem specific.
3. For each nework, we compared, using SVCCA, each convolutional, pooling, or fully-connected layer of each pruned network with its unpruned counterpart, i.e., $L_0^{(j)}$ with $L_i^{(j)}$, $\forall i \in \{1, \dots, 20\}, \forall j \in \{1, \dots, K-1\}$. The similarity was then summarized using Mean CCA Similarity (see Section 3.2).

5 RESULTS

5.1 Test performance

Recalling from Section 4.3, we trained 5 unpruned convolutional neural networks on problems directly obtained from Cifar10; each of those networks were pruned using IMP for 20 iterations, with pruning rates as of Table 3.

We repeated the runs of IMP 20 times, each time starting from the same unpruned network, but using a different seed for the optimizer.

Median test accuracy of the aforementioned models, for each of iteration of IMP, are reported in Figure 1.

5.2 Layers similarity

The results in terms of Mean CCA Similarity are shown in Figure 2; again, since the pruning was repeated 20 times to increase robustness, the results, for each layer and number of IMP iteration, are summarised in the figure as median values.

We notice the presence of two different trends. First, the neural networks for the datasets Cifar2 and Cifar4 exhibit a decreasing similarity: (a) as the iteration number increases and (b) as we progress along the network, from input to output. If we do not consider what would seem a slight incremental effect on similarity which we can notice in almost all pooling layers (also present in the other problems), the networks for Cifar2 and Cifar4 show a progressive detachment from the input dataset;

Second, on the other hand, the networks for the larger problems, Cifar6, Cifar8, and Cifar10, exhibit a different behavior. The similarity starts high for the first convolutional layer, it decreases until it bottoms around the 3rd convolutional block, finally it grows again with a spike on fully-connected layer. Moreover, while before we noted an increasing dissimilarity w.r.t. the unpruned network layers as the IMP iteration increased, now the range of similarity, layer per layer, is much smaller.

These observations, especially on the second group of problems indicated in the list above, may hint at some insights on the roles of the hidden layers in convolutional neural networks as the network gets pruned.

First of all, from all of the graphs in Figure 2, we can notice how pruning, even at small rates, produces

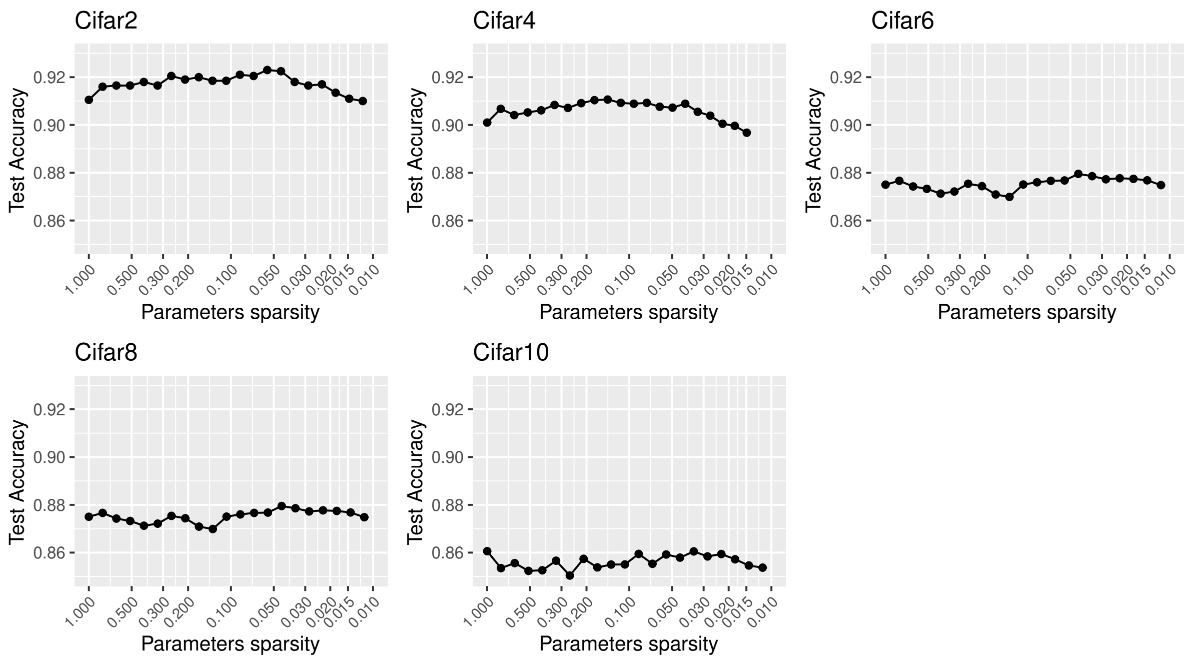

Figure 1: Median test accuracy for the models from Section 4.3. The point corresponding to Parameters sparsity equal to 1.000 refers to the unpruned model.

layers representation which are different from the unpruned network ones. All the graphs seem to exhibit trends, which, especially considering that they're the result of multiple trials, allow us to rule out the possibility of the results being purely the product of randomness in the path of the optimizers.

Moreover, we can notice that, in harder problems, pruning (with IMP) seems to have a stronger effect on the convolutional layers, forcing them to produce different representations; as we get closer to the fully-connected layer, the network seems to be forced to lead the representations produced by the intermediate convolutional blocks towards a common representation for the fully-connected layer, in order to produce a similar output, and thus getting a comparable, if not better, test accuracy. Summarising, it would seem that the fully-connected layer acts as a *pivot* during the pruning, allowing for the network to produce similar performance despite different representations being learnt in the previous layers.

5.3 Limitations and future works

We remark that, as highlighted in Section 2, we are aware of the existence of other metrics and methodologies for networks comparison, and we are aware of the potential limitations of CCA raised by (Kornblith et al., 2019) as well. Our next step will be hence devoted towards the incorporation of these works into our research.

Since our work was carried out purely on a set of convolutional neural networks for category-level recognition, based on VGG, and trained on Cifar10, a future goal would be to extend the analysis to other, deeper networks (like VGG19, or Resnet 18), or to other, more difficult problems—like ImageNet (Deng et al., 2009)—to inspect whether the same results are observable in these networks.

Moreover, as noted by (Han et al., 2015), since the effectiveness of pruning techniques is essentially a consequence of the over-parameterization of ANNs, the present work may be potentially linked to other papers addressing the issue of over-parameterization. For example, in (Ansuini et al., 2019), the intrinsic dimensionality (ID) of a layer in a deep neural network (i.e., the "minimal number of coordinates which are necessary to describe its points without significant information loss") is analyzed. The authors computed the ID of layers for an ensemble of convolutional neural networks for image classification observing that the ID was exhibiting a bell-shaped trend, somewhat complementary to what we observed in our research in Figure 2 for the three largest models: the ID started low, increased, spiked at around 30–40% of the network depth, then decreased, bottoming at the output layer. A direction of future work is studying whether the shape for layer-wise similarity in our research may somewhat be connected to these observations on ID,

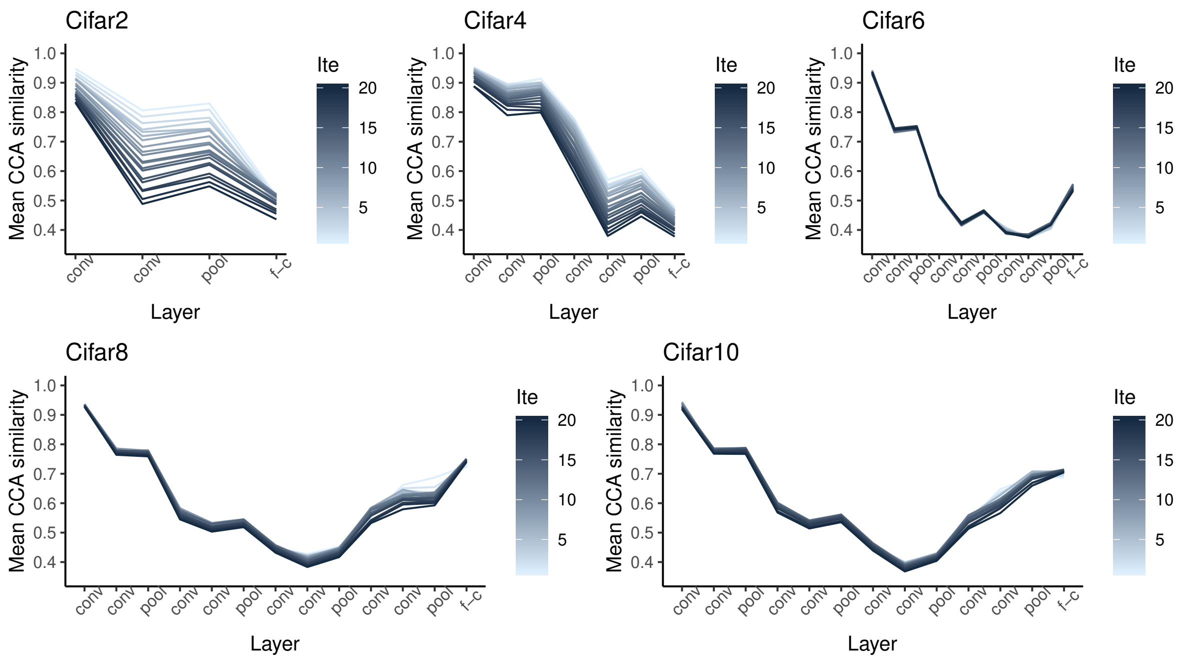

Figure 2: Median values of Mean CCA Similarity between pruned layers and their unpruned counterparts for models, and IMP iteration number, from Section 4.4. Layers are ordered, left to right, from closer to input to closer to output. Line color shade is related to iteration number of IMP.

i.e., that the decrease in similarity in the mid-ranked convolutional layers is low because ID of these layers is high, and, on the other hand, a low ID in the early and later layers is what causes the representations to be similar as far as Mean CCA Similarity is concerned.

6 CONCLUSIONS

In this paper, we applied IMP to CNNs based on VGG, trained on an set of increasingly difficult problems derived from Cifar10. We then inspected the layer-wise similarities between unpruned and pruned networks using SVCCA. For the more difficult problems, as we got farther from the input layer, we observed a decreasing similarity, bottoming at around half of the network depth, and then an increase, as the fully-connected layer is approached. That behavior may indicate that the fully connected layer plays a role of *pivot* for leading the differences produced by convolutional layers back to somewhat similar representations.

Future work includes exploiting some very recent advances in the similarity measures for network layers and exploring the connection to existing results on the over-parameterization in artificial neural networks.

REFERENCES

Ansuini, A., Laio, A., Macke, J. H., and Zoccolan, D. (2019). Intrinsic dimension of data representations in deep neural networks. In *NIPS 2019*.

Crowley, E. J., Turner, J., Storkey, A., and O'Boyle, M. (2018). Pruning neural networks: is it time to nip it in the bud? *arXiv preprint arXiv:1810.04622*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Frankle, J. and Bau, D. (2019). Dissecting pruned neural networks. *arXiv preprint arXiv:1907.00262*.

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

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. (2019). Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*, chapter 7.1.2 - L1 Regularization. MIT press.

Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 1135–1143. Curran Associates, Inc.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of* *the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. E. (2019). Similarity of neural network representations revisited. In *ICML*, pages 3519–3529.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Morcos, A., Raghu, M., and Bengio, S. (2018). Insights on representational similarity in neural networks with canonical correlation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 5732–5741. Curran Associates, Inc.

Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. (2017). Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6076–6085. Curran Associates, Inc.

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

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Wang, L., Hu, L., Gu, J., Hu, Z., Wu, Y., He, K., and Hopcroft, J. (2018). Towards understanding learning representations: To what extent do different neural networks learn the same representation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 9584–9593. Curran Associates, Inc.

Yu, T., Long, H., and Hopcroft, J. E. (2018). Curvature-based comparison of two neural networks. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 441–447. IEEE.

Zhang, T., Ye, S., Zhang, K., Tang, J., Wen, W., Fardad, M., and Wang, Y. (2018). A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. (2019). Deconstructing lottery tickets: Zeros, signs, and the supermask. *arXiv preprint arXiv:1905.01067*.