Reducing Duplicate Filters in Deep Neural Networks

Aruni RoyChowdhury [1] Prakhar Sharma [1] Erik Learned-Miller [1]

Abstract

This paper investigates the presence of duplicate neurons or filters in neural networks. This phenomenon is prevalent in networks and increases with the number of filters in a layer. We observe the emergence of duplicate filters over training iterations, study the factors that affect their concentration and compare existing network reducing operations. We validate our findings using convolutional and fully-connected networks on the CIFAR-10 dataset.

1. Introduction

In this paper, we focus on two ways in which neurons or filters in neural networks may be redundant – (1) if they have negligibly small values (i.e. the filters have weight vectors with low norm), or (2) if their functionality is mimicked by another filter. The latter may be quantified as high cosine similarity between the weight vectors of two filters. The presence of near-duplicate filters in deep neural networks (Zeiler & Fergus, 2014; Rodríguez et al., 2016) is an interesting phenomenon which to our knowledge has not been explored empirically for recent neural network architectures. The contributions of this paper are:

(i) empirically shows that fully-connected networks have a significantly larger number of near-duplicate parameters with increasing layer size and contrasts this to the behaviour of convolutional networks;

(ii) visualizes the evolution and convergence of duplicate filters in networks over training iterations;

(iii) draws connections between low-rankedness of weight matrices and increasing redundancy;

(iv) provide a network-equivalent reduction operation to eliminate duplicates, and empirically compares existing norm-based approaches with the duplicate-based approach to redundancy-removal.

[1]College of Information and Computer Sciences, University of Massachusetts, Amherst, USA. Correspondence to: Aruni RoyChowdhury <arunirc@cs.umass.edu>.

2. Related Work

Singularities and irreducible neural networks. Several papers in the early neural network literature have studied the phenomenon of identical parameters and their effect on optimization in feed-forward neural networks. Chen & Hecht-Nielsen (1991) analyzes a class of "*equi-error*" weight space transformations for multi-layer perceptrons that leaves the network input-output map unchanged. Sussmann (1992) introduces the concept of an *irreducible* net – one that does not have zero-weight nodes, nor a pair of nodes that could be collapsed into a single node without altering the network input-output map. Fukumizu (1996) show that a neural network with such redundancies will have a singular Fisher information matrix. Amari et al. (2006) and Wei et al. (2008) analyze the learning dynamics near singularities of the Hessian and identify two types of plateaus in the optimization landscape that arise due to identical or redundant model parameters. Recently, Orhan & Pitkow (2017) show that residual networks overcome the singularity problem due to the presence of skip connections between layers.

Convergent learning. Li et al. (2015) pose the question: do different neural networks (with the same architecture) learn the same things? Based on their experiments, different networks *do* converge to a similar set of weights, up to a permutation symmetry (e.g. unit 1 in network 1 may be a red color detector, while in network 2 the red detector is in unit 47), with a few unique weights that are not commonly learned across networks. Our work is similar is spirit but focuses on the different situation of duplicate filters in the *same* layer of a *single* network.

Decorrelated network weights. A line of work has focused on loss functions that penalize the correlation of neural network parameters, such as the DeCov loss (Cogswell et al., 2015) and OrthoReg (Rodríguez et al., 2016), thus ensuring diversity in the neurons of a layer.

Network pruning. Li et al. (2017) discuss the advantages of *structured pruning* of network parameters, wherein entire filters with low L1-norm are dropped, as opposed to approaches that zero out redundant weights in an unstructured fashion and then rely on sparse libraries to take advantage of the sparsity structure. Mariet & Sra (2016) select a subset of diverse neurons using a Determinantal Point Process

(DPP) based on correlations between their *activations*. They also observe that simply dropping neurons causes a large change in the input-output map and propose to re-weigh the selected neurons to account for this using a least squares solution. The only work to our knowledge that addresses the redundancy due to near-identical neurons in deep networks is from Srinivas & Babu (2015), which uses an iterative procedure to eliminate near-duplicates a single pair at a time, but does not focus on analysing duplicate filter concentration. Molchanov et al. (2016) use a Taylor expansion of the network function w.r.t. activations, which results in removing both low-activation and low-gradient neurons.

3. Duplicate filters in neural networks

We use two simple network architectures — a fully-connected multi-layer perceptron (MLP) and a convolutional neural network (CNN). We use *filter* to denote both the channels of a convolutional layer and also indicate an individual neuron's weights in the weight matrix of a fully connected layer.

The MLP has two fully-connected (fc) layers followed by a 10-way softmax. The size of the first layer (`fc1`) is varied across 100, 500 and 1000 units. The size of the second layer (`fc2`) is fixed at 100 units. The CNN consists of two convolutional layers (`conv1` and `conv2`), each followed by a ReLU non-linearity and 2×2 max-pooling, two fully-connected layers (`fc1` and `fc2`) , both followed by a ReLU and finally a 10-way classifier with a softmax. The first conv-layer's size is varied across 100, 500, 1000, similar to what we do for the MLP experiments. The second conv-layer is fixed to have 50 filters. The fc-layers have sizes 120 and 84. Both networks are trained till convergence on CIFAR-10 (Krizhevsky & Hinton, 2009). Visualizations of the first layer filters are in the Appendix.

3.1. Distribution of redundant filters

Each of the MLP networks are trained from 100 different random initializations. The distribution of *intra-network* filter similarity (measured in terms of cosine similarity, $\langle \frac{\mathbf{w}_i}{||\mathbf{w}_i||}, \frac{\mathbf{w}_j}{||\mathbf{w}_j||} \rangle$ for filters i and j, with $\langle .,. \rangle$ denoting the inner-product between two vectors) among the first layer weights is plotted in subplots *(a-c)* of Figure 1. The right tail shows a small but significant heaviness with increasing layer size, indicating that there are more filters with a high cosine similarity between them as we keep increasing the network width. This trend was not evident in similar plots for CNNs.

The distributions of the norms of filter weights for the MLP networks are plotted in Figure 1(d-f). We use the 2-norm, $||\mathbf{w}_i||$, for filter i. With increasing layer size there is a corresponding increase in the the frequency of low-norm

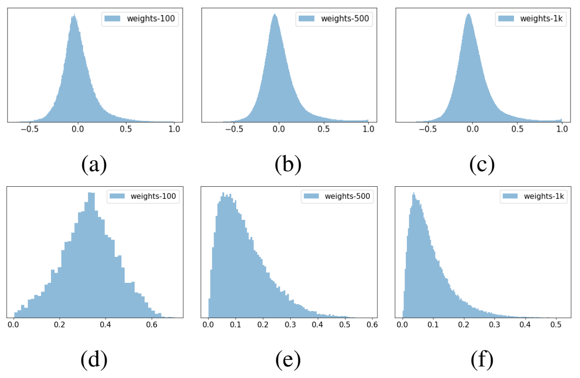


Figure 1. **(a-c)** distribution of cosine similarity between filters with increasing layer size. The right tail gets heavier – kurtosis values are 3.367 (`fc:100`), 4.152 (`fc:500`), 4.149 (`fc:1000`); **(d-f)** distribution of filter norms for `fc:100`, `fc:500` and `fc:1000`.

filters. These plots give a visual indication that both kinds of redundancy – low-norm and duplicates – increase with over-parameterization of the network. [1]

3.2. Grouping duplicate filters

Considering the `fc1:500` MLP network as an example, the first layer weights ($W \in \mathbb{R}^{500\times3*32*32}$), can be viewed as 500 discriminative filters acting upon 3-channel 32×32 images, each filter being a $3 * 32 * 32$ dimensional vector. A similarity matrix S is then formed, where each entry s_{ij} measures the cosine similarity between filters i and j. By thresholding S at a particular value τ, we can induce a *similarity graph* G_τ over the filters in a layer. The *connected components* (CCs) of G_τ result in groups of near-duplicate filters at a particular threshold τ of their cosine similarity. For CNNs we consider the 4-D tensor of k convolutional kernels of spatial support $c \times c$ acting on d-dimensional inputs to be a set of k vectors, each of size dc^2. Some examples of such grouped filters in the first layers are shown for MLP and CNN in Figure 2(b). The effect of changing the similarity threshold τ is visualized for MLP filters in Figure 3.

3.3. Analysis of duplicates

Layer size and duplicate concentration. A method to quantify the degree of duplicate filters in a network is described here. Let η be the ratio of number of filter groups and the total number of filters at different thresholds τ in G_τ. The quantity $1 - \eta$, plotted as the y-axes of Figure 2(b-c) gives an estimate of the concentration of duplicates in a layer. For the MLP, the curves become higher as we in-

[1]We observe in passing that the distributions of filter norms, which are the lengths of high dimensional random variables, resemble a Rayleigh distribution with decreasing scale parameter for increasing layer size.

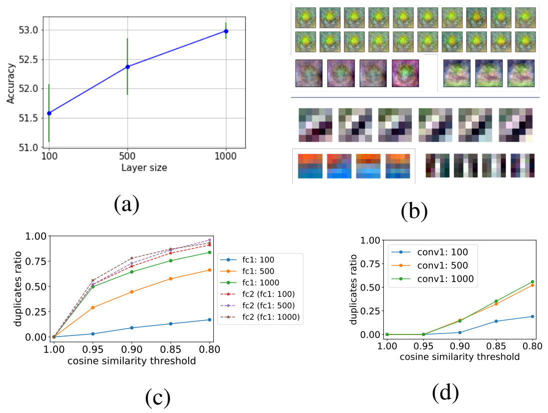


Figure 2. **(a)** Test accuracy of MLP on CIFAR-10 with increasing layer size over 100 runs; **(b)** Groups of similar filters in the first layer of a 500-unit MLP *(top)* and CNN *(bottom)*; **(c-d)** concentration of duplicates with increasing layer size for the MLP and CNN.

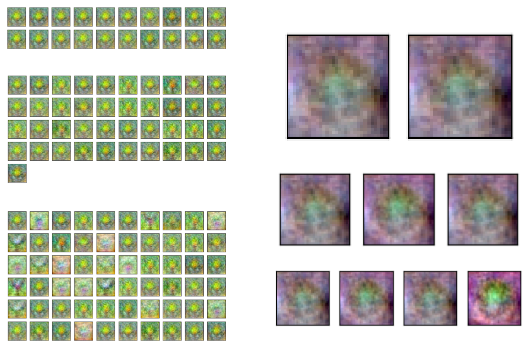


Figure 3. Visualizing MLP filter groups at various cosine similarity thresholds τ. *Rows from top to bottom:* $\tau = \{0.95, 0.9, 0.8\}$. At strict thresholds the filters are virtually indistinguishable from each other visually; in numerical terms this means that the filters have weight vectors pointing in almost the exact same direction. Relaxing the similarity threshold results in visually dissimilar filters being clustered together, as is particularly evident for the last row.

crease the number of filters in the first layer (Figure 2(b)). For the CNN (Figure 2(c)), near-duplicate filters is not very common in the first layer and do not occur at all in the later layers. We note from Figure 1 that for larger first layer size in MLPs, the *distribution of inter-filter cosine similarity* had heavier right-tails, indicating more filters with high similarity. This trend was not evident in CNNs. It is possible that having fewer parameters and being less prone to over-fitting, the *CNN architecture rarely develops completely identical filters, unlike the fully-connected MLP networks.*

Relation to matrix rank. Having filters in a network with high cosine similarity means that the vectors are both pointing in the same direction, with a possible difference in scale. This would indicate that the rank of the weight matrix is small when it has many duplicate filters (the converse is not true – small rank does not necessarily indicate duplicates). Gunasekar et al. (2017) observe that stochastic gradient descent (SGD) is biased to finding the *minimum nuclear norm* solution. We conjecture that the presence of duplicates in over-parameterized networks could be an effect of finding a set of weight matrices with low nuclear norm, which implies low rank. Figure 4(d) plots a slightly modified matrix *trace complexity* (Srebro & Salakhutdinov, 2010) (for a matrix $W \in \mathbb{R}^{n\times m}$, $tc = \frac{||W||_{tr}}{\sqrt{nm}}$), which acts as an indicator of matrix rank, as we increase layer size in the MLP networks, averaged over 100 runs. This shows that with increasing layer size, there is a decrease in effective rank, normalized for matrix size.

The following discussion focuses on MLPs, since they exhibit duplicates more noticeably. We adapt the method of SV-CCA (Raghu et al., 2017) to compare the subspaces spanned by the filters as the layer is widened. The singular values of the filters are plotted in Figure 4 (a-c), which indicate a low effective rank of the weight matrices. The average correlation between the first layer filters of the networks after the CCA is very high across the different layer sizes, as shown in Figure 4 (e). This would indicate that the subspaces spanned by the weight vectors of the network as we increase the layer-size are quite similar. We observe qualitatively that filters across the models are roughly similar. The new filters being learned with the increased capacity are subtly different from the filters in a smaller network — they are tuned to fine-grained differences (e.g. a “brown" color detector may now be replaced by detectors for the shades “auburn" and “chestnut").

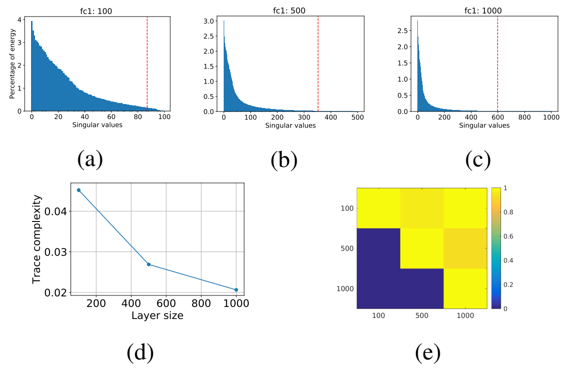


Figure 4. **(a-c)** Singular values with increasing `fc1` layer size in MLP. The vertical red line marks 99% of the singular values' energy. **(d)** Trace complexity of weight matrices of `fc:{100, 500, 1k}`, averaged over 100 runs each. **(e)** Modified SV-CCA similarity matrix (upper-triangle) between the MLP networks `fc1:100,500,1000`.

Dropout and neuron co-adaptation. Dropout (Hinton et al., 2012; Srivastava et al., 2014) was introduced to prevent overfitting in neural networks by reducing co-

Table 1. The fractions show $\frac{\#CC}{\#filters}$ in $G_{\tau=0.9}$ at different dropout rates on the MLP models. Smaller $\#CC$ means more duplicates.

Dropout:	*none*	0.2	0.5	0.8
`fc1:100`	91/100	95/100	95/100	97/100
`fc1:500`	277/500	272/500	290/500	285/500
`fc1:1k`	356/1k	365/1k	368/1k	384/1k

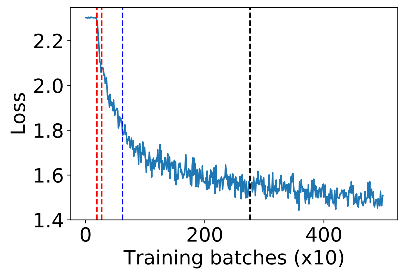
Table 2. Increasing weight decay in the MLP `fc:100`.

Wt. decay	Train acc.(%)	Test acc.(%)
0.001	81	54
0.010	63	55
0.018	55	52

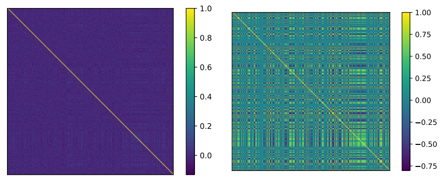
adaptations between neurons. Dropout randomly zeroes out a fraction of features at each training iteration — this amounts to randomly dropping the contributions of an entire row of a layer's weight matrix (or a whole filter in case of a conv-layer). The results in Table 1 show that even with dropout the filters can be grouped into clusters of duplicates – if there were no duplicates then the number of CCs in G_τ would be equal to the number of filters in that layer. Dropout decreases the number of duplicates by a small amount, but does not eliminate the issue. It is also to be noted that setting a very high dropout rate adversely affects the test-time performance of the network.

Initialization and convergence. A sample plot of inter-filter similarity over training iterations is shown in Figure 5. The learning rate is exponentially decayed over training epochs. Starting at random values, most of the weight vectors are nearly orthogonal to each other (Diaconis & Freedman, 1984) during the initial iterations, shown in Figure 5 (b). The first sharp drop in the loss, denoted by the *two red lines* in the loss plot Figure 5 (a), is accompanied by a significant change in similarity structure (b-c). Groups of filters attain similar values, evident in the block structure of Figure 5 (c). As training proceeds the filters gradually become distinct (Figure 5 (d)), and we get the similarity structure at the final stages ((Figure 5 (e)).

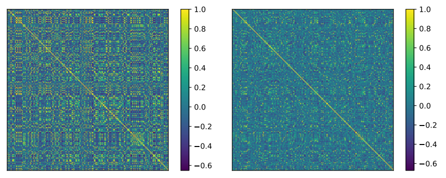
Effect of regularization. The results of varying weight decay are shown in Table 2 and the filters are visualized in Figure 6. The filters are seen to be quite noisy with low regularization and higher regularization decreases the performance slightly while resulting in smoother filters. Duplicate filters are visually evident in the rightmost plot (highest weight decay) in Figure 6 – in particular a prominent green patch is repeated quite frequently.



(a) loss



(b) 190 (c) 270



(d) 620 (e) 2760

Figure 5. **(a)** MLP `fc1:500` data loss over training iterations, batchsize of 100. **(b-e)** First layer filter similarity matrices (500 × 500) at different iterations (batches). Colors correspond to iterations at vertical lines in the loss plot. Best viewed in color and high zoom.

4. Network Reduction

Let us consider the two-layer MLP with weight matrices $W \in \mathbb{R}^{K_1 \times D}$ and $V \in \mathbb{R}^{K_2 \times K_1}$ for the first and second layer respectively, and input features $\mathbf{x} \in \mathbb{R}^D$. For ease of exposition, we discuss eliminating duplicate filters in the first layer, i.e. the rows of W, denoted as $\mathbf{w}_i$, with biases b_i and non-linearity $\phi(.)$.

4.1. Duplicate Neuron Reduction

Equivalence-preserving weight normalization. Although filters in a group have their weight vectors pointing in the same direction, their magnitudes are empirically found to be quite different (see Figure 7). By dividing each `fc1` layer filter (rows of W) by its L2-norm and then multiplying the corresponding connections to the `fc2` layer (columns of V) by the same scalar, the network's functional map remains unchanged, but we end up with unit-normalized first layer filters. This requires $\phi(a\mathbf{x}) = a\phi(\mathbf{x})$ for some scalar $a \in \mathbb{R}^+$, which holds true for the widely used ReLU non-linearity. The symmetry of neural network weights under the multiplicative group of positive real numbers has been

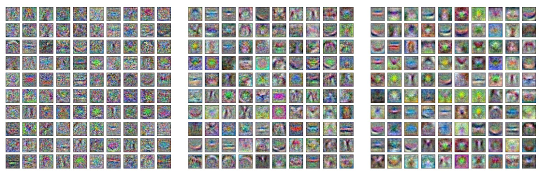


Figure 6. Effect of regularization on filters. First layer filters of the MLP `fc:100`, from left to right, with increasing weight decay: 0.001, 0.01, 0.018.

referred to in Neyshabur et al. (2015); Badrinarayanan et al. (2015).

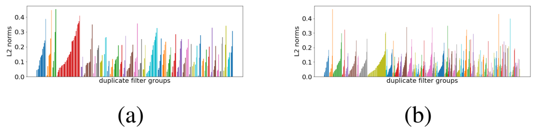


Figure 7. Plotting the L2-norms of the weight vectors of filters from **(a)** `fc1:500` and **(b)** `fc1:1k` MLPs. Same color denotes filters within the same group (colors are cycled). The filters are grouped using $\tau = 0.9$. Filters within the same group are observed to have widely varying weight vector magnitudes.

Redundant filter aggregation. Following unit-normalization of the filters, we go through the groups of near-identical filters at a layer, say `fc1`, and replace all filters in a group with the within-group mean filter. The implementation of this procedure is simplified by the connected component (CC) step for clustering similar filters. We collapse all the filters within a connected component and appropriately modify their upstream connections, resulting in a network that has #filters equal to #CCs at the pruned layer. The details of the procedure are as follows:

Consider an `fc2` layer feature: $\mathbf{y} = \mathbf{v}_1\phi(\mathbf{w}_1\mathbf{x} + b_1) + \mathbf{v}_2\phi(\mathbf{w}_2\mathbf{x} + b_2) + \mathbf{v}_3\phi(\mathbf{w}_3\mathbf{x} + b_3) + \ldots$.

If $\mathbf{w}_1 = \mathbf{w}_2$ and $b_1 = b_2$, then $\mathbf{y} = (\mathbf{v}_1 + \mathbf{v}_2)\phi(\mathbf{w}_1\mathbf{x} + b_1) + \mathbf{v}_3\phi(\mathbf{w}_3\mathbf{x} + b_3) + \ldots$

Therefore, after collapsing the duplicates in the `fc1` layer weights W, their corresponding connections in the `fc2` layer weight matrix V should be aggregated by *summation*. This results in an unaltered output when the collapsed filters are identical. In practice, we also have to approximate the bias terms by their mean[2].

4.2. Network Reduction Experiments

For each network, while varying the first layer size, a fixed set of thresholds is applied on the filter similarity matrix.

[2]Applying weight decay on the bias terms during network training can make the bias values negligible, so empirically we do not lose much performance due to this approximation.

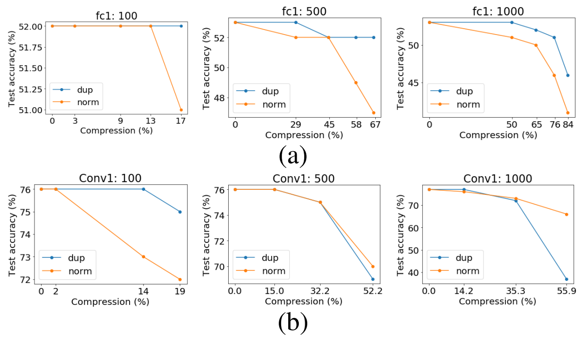


Figure 8. Plotting *compression vs. performance* as filters are aggregated at fixed similarity thresholds $\tau = \{1, 0.95, 0.9, 0.85, 0.8\}$. **(a)** accuracy of the MLP; **(b)** accuracy of the CNN. The x-axis in each plot shows the reduction (%) in number of filters.

A smaller threshold results in more filters being marked as identical. These duplicates are then pruned following the reduction procedure in Sec. 4.1. The duplication-removal procedure (***"dup"***) is compared with a simple norm-based pruning strategy (***"norm"***) that sorts a layer's filters based on their L1-norm and keeps the largest K filters (Li et al., 2017). The K for the baseline is chosen to match the number of filters obtained using our method. To be consistent with the author's method for norm-based pruning, we use the L1-norm in this case instead of the L2-norm and note that Li et al. (2017) mention that both the norms give equivalent results. Both the methods of eliminating redundant filters that we compare here are data-independent — we do not require access to a suitably large dataset to get statistics of activations or any other such heuristic. This appears to be a reasonable choice for the norm-based baseline, since Li et al. (2017) report the filter norm-based pruning strategy yields comparable-to-better empirical performance than activation-based approaches.

The results are summarized in Figure 8. In the case of the **MLP**, the first fully-connected layer exhibits a large number of duplicates as we keep increasing its size. As a result of this, our method is able to reduce a majority of the filters without causing too much change in the input-output map of the network, and consequently lesser drop in test accuracy (Figure 8(a)), than the baseline method of eliminating filters based on low L1-norm. The convolutional layers in the **CNN** have a fewer number of near-duplicate filters compared to the MLP. Setting conservative thresholds on the similarity ends up with almost no duplicates and more aggressive thresholds results in merging filters that are sufficiently dissimilar so as to impact the performance of the network (Figure 8(b)).

5. Conclusion

We have shown that duplication of filters occurs more in MLPs than CNNs, and this appears to be an outcome of over-parameterization in the fully-connected MLP model. Increasing the number of filters at a layer results in more duplicates for MLP and is less marked for CNNs. A method to reduce a network based on near-duplicate filters is introduced and is shown to work well for fully-connected nets in the regime of moderate compression. For CNNs, a norm-based pruning strategy works as well or better, underlining their differences with MLPs in how parameter redundancy is manifested.

Future directions to explore include the connections between the two heuristics for parameter redundancy – low-norm and duplication, and how this affects network compression strategies in large architectures beyond the controlled small-scale experiments performed in our current work.

Acknowledgement

This research is based in part upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) under contract number 2014-14071600010. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon.

ARC would like to thank Archan Ray for initial experiments on duplicate filters; Hang Su, Matheus Gadelha and Tsung-Yu Lin for very helpful discussions.

References

Amari, Shun-Ichi, Park, Hyeyoung, and Ozeki, Tomoko. Singularities affect dynamics of learning in neuromanifolds. *Neural computation*, 18(5):1007–1065, 2006.

Badrinarayanan, Vijay, Mishra, Bamdev, and Cipolla, Roberto. Symmetry-invariant optimization in deep networks. *arXiv preprint arXiv:1511.01754*, 2015.

Chen, An Mei and Hecht-Nielsen, Robert. On the geometry of feedforward neural network weight spaces. In *Artificial Neural Networks, 1991., Second International Conference on*, pp. 1–4. IET, 1991.

Cogswell, Michael, Ahmed, Faruk, Girshick, Ross, Zitnick, Larry, and Batra, Dhruv. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*, 2015.

Diaconis, Persi and Freedman, David. Asymptotics of graphical projection pursuit. *The annals of statistics*, pp. 793–815, 1984.

Fukumizu, Kenji. A regularity condition of the information matrix of a multilayer perceptron network. *Neural networks*, 9(5):871–879, 1996.

Gunasekar, Suriya, Woodworth, Blake, Bhojanapalli, Srinadh, Neyshabur, Behnam, and Srebro, Nathan. Implicit regularization in matrix factorization. *arXiv preprint arXiv:1705.09280*, 2017.

Hinton, Geoffrey E, Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Li, Hao, Kadav, Asim, Durdanovic, Igor, Samet, Hanan, and Graf, Hans Peter. Pruning filters for efficient convnets. *ICLR*, 2017.

Li, Yixuan, Yosinski, Jason, Clune, Jeff, Lipson, Hod, and Hopcroft, John. Convergent learning: Do different neural networks learn the same representations? In *Feature Extraction: Modern Questions and Challenges*, pp. 196–212, 2015.

Mariet, Zelda and Sra, Suvrit. Diversity networks. *ICLR*, 2016.

Molchanov, Pavlo, Tyree, Stephen, Karras, Tero, Aila, Timo, and Kautz, Jan. Pruning convolutional neural networks for resource efficient inference. 2016.

Neyshabur, Behnam, Salakhutdinov, Ruslan R, and Srebro, Nati. Path-sgd: Path-normalized optimization in deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 2422–2430, 2015.

Orhan, Emin and Pitkow, Xaq. Skip connections eliminate singularities. *arXiv preprint arXiv:1701.09175*, 2017.

Raghu, Maithra, Gilmer, Justin, Yosinski, Jason, and Sohl-Dickstein, Jascha. Svcca: Singular vector canonical correlation analysis for deep understanding and improvement. 2017.

Rodríguez, Pau, Gonzàlez, Jordi, Cucurull, Guillem, Gonfaus, Josep M, and Roca, Xavier. Regularizing cnns with locally constrained decorrelations. *arXiv preprint arXiv:1611.01967*, 2016.

Srebro, Nathan and Salakhutdinov, Ruslan R. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. In *Advances in Neural Information Processing Systems*, pp. 2056–2064, 2010.

Srinivas, Suraj and Babu, R Venkatesh. Data-free parameter pruning for deep neural networks. 2015.

Srivastava, Nitish, Hinton, Geoffrey E, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

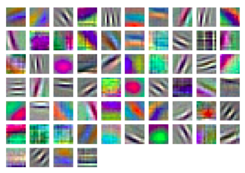
Sussmann, Héctor J. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural networks*, 5(4):589–593, 1992.

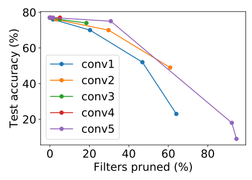
Wei, Haikun, Zhang, Jun, Cousseau, Florent, Ozeki, Tomoko, and Amari, Shun-ichi. Dynamics of learning near singularities in layered networks. *Dynamics*, 20(3), 2008.

Zeiler, Matthew D and Fergus, Rob. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Appendix

A. AlexNet on CIFAR-10


(a)



(b)

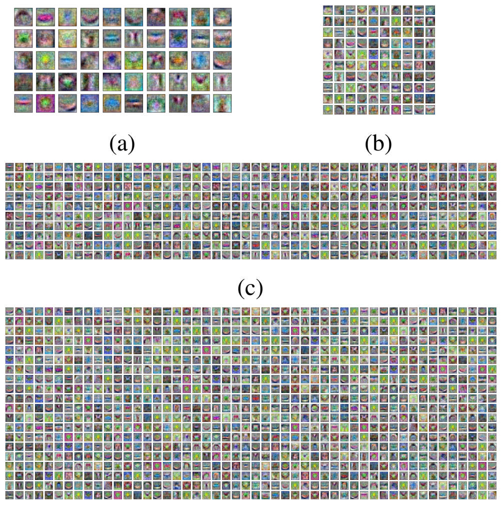
Figure 9. **(a)** First layer filters of AlexNet trained on CIFAR-10; **(b)** Plotting percentiles with layer-size. The different points on the curves correspond to different levels of filter pruning and corresponding accuracy at the following filter-similarity thresholds: $\tau = \{0.25, 0.3, 0.45, 0.6, 0.8, 0.9\}$.

Preliminary results with AlexNet (Krizhevsky et al., 2012) on CIFAR-10, pruning each layer individually, are shown in Figure 9(b). The layer sizes are: 64 (conv1), 192 (conv2), 384 (conv3), 256 (conv4) and 256 (conv5). Setting the thresholds $\tau = \{0.25, 0.3, 0.45, 0.6, 0.8, 0.9\}$ results in different numbers of filters being pruned.

`Conv1` is understandably sensitive to pruning – it has too few filters to have significant redundancy (Figure 9(a) shows this qualitatively). `Conv2` shows moderate drop in accuracy with a large percentage of filters being pruned. `Conv3` and `conv4` do not show much redundancy – even setting very low thresholds like 0.3 does not result in substantial reduction of filters. `Conv5` is again very sensitive to pruning.

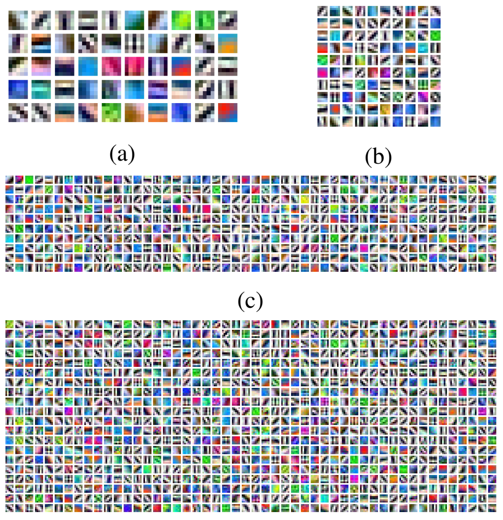
These results indicate that the AlexNet filters are sufficiently disparate that merging them requires (1) very relaxed thresholds of similarity, and (2) results in non-trivial drops in accuracy.

B. Filter visualizations



(d)

Figure 10. First layer filters of the MLP, `fc1`: **(a)**50, **(b)** 100, **(c)** 500, **(d)** 1000. A relatively high weight decay value (0.18) was chosen such that the filters look smooth. The increase in number of near-identical filters is quite apparent as layer size is increased.



(d)

Figure 11. First layer conv filters from the CNN, `conv1`: **(a)**50, **(b)** 100, **(c)** 500, **(d)** 1000.