RePr: Improved Training of Convolutional Filters

Aaditya Prakash
Brandeis University
aprakash@brandeis.edu

James Storer
Brandeis University
storer@brandeis.edu

Dinei Florencio, Cha Zhang
Microsoft Research
{dinei,chazhang}@microsoft.com

Abstract

A well-trained Convolutional Neural Network can easily be pruned without significant loss of performance. This is because of unnecessary overlap in the features captured by the network's filters. Innovations in network architecture such as skip/dense connections and Inception units have mitigated this problem to some extent, but these improvements come with increased computation and memory requirements at run-time. We attempt to address this problem from another angle - not by changing the network structure but by altering the training method. We show that by temporarily pruning and then restoring a subset of the model's filters, and repeating this process cyclically, overlap in the learned features is reduced, producing improved generalization. We show that the existing model-pruning criteria are not optimal for selecting filters to prune in this context and introduce inter-filter orthogonality as the ranking criteria to determine under-expressive filters. Our method is applicable both to vanilla convolutional networks and more complex modern architectures, and improves the performance across a variety of tasks, especially when applied to smaller networks.

1. Introduction

Convolutional Neural Networks have achieved state-of-the-art results in various computer vision tasks [1, 2]. Much of this success is due to innovations of a novel, task-specific network architectures [3, 4]. Despite variation in network design, the same core optimization techniques are used across tasks. These techniques consider each individual weight as its own entity and update them independently. Limited progress has been made towards developing a training process specifically designed for convolutional networks, in which *filters* are the fundamental unit of the network. A filter is not a single weight parameter but a stack of spatial kernels.

Because models are typically over-parameterized, a trained convolutional network will contain redundant filters [5, 6]. This is evident from the common practice of
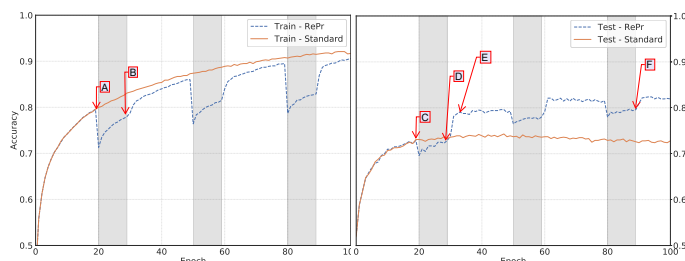

Figure 1: Performance of a three layer ConvNet with 32 filters each over 100 epochs using standard scheme and our method - RePr on CIFAR-10. The shaded regions denote periods when only part of the network is trained. Left: Training Accuracy, Right: Test Accuracy. Annotations [A-F] are discussed in Section 4.

pruning filters [7, 8, 6, 9, 10, 11], rather than individual parameters [12], to achieve model compression. Most of these pruning methods are able to drop a significant number of filters with only a marginal loss in the performance of the model. However, a model with fewer filters cannot be trained from scratch to achieve the performance of a large model that has been pruned to be roughly the same size [6, 11, 13]. Standard training procedures tend to learn models with extraneous and prunable filters, even for architectures without any excess capacity. This suggests that there is room for improvement in the training of Convolutional Neural Networks (ConvNets).

To this end, we propose a training scheme in which, after some number of iterations of standard training, we select a subset of the model's filters to be temporarily dropped. After additional training of the reduced network, we reintroduce the previously dropped filters, initialized with new weights, and continue standard training. We observe that following the reintroduction of the dropped filters, the model is able to achieve higher performance than was obtained before the drop. Repeated application of this process obtains models which outperform those obtained by standard training as seen in Figure 1 and discussed in Section 4. We observe this improvement across various tasks and over various types of convolutional networks. This training procedure is able to produce improved performance across a range of possible criteria for choosing which filters

to drop, and further gains can be achieved by careful selection of the ranking criterion. According to a recent hypothesis [14], the relative success of over-parameterized networks may largely be due to an abundance of initial sub-networks. Our method aims to preserve successful sub-networks while allowing the re-initialization of less useful filters.

In addition to our novel training strategy, the second major contribution of our work is an exploration of metrics to guide filter dropping. Our experiments demonstrate that standard techniques for permanent filter pruning are suboptimal in our setting, and we present an alternative metric which can be efficiently computed, and which gives a significant improvement in performance. We propose a metric based on the *inter*-filter orthogonality within convolutional layers and show that this metric outperforms state-of-the-art filter importance ranking methods used for network pruning in the context of our training strategy. We observe that even small, under-parameterized networks tend to learn redundant filters, which suggests that filter redundancy is not solely a result of over-parameterization, but is also due to ineffective training. Our goal is to reduce the redundancy of the filters and increase the expressive capacity of ConvNets and we achieve this by changing the training scheme rather than the model architecture.

2. Related Work

Training Scheme Many changes to the training paradigm have been proposed to reduce over-fitting and improve generalization. Dropout [15] is widely used in training deep nets. By stochastically dropping the neurons it prevents co-adaption of feature detectors. A similar effect can be achieved by dropping a subset of activations [16]. Wu *et al*. [15] extend the idea of stochastic dropping to convolutional neural networks by probabilistic pooling of convolution activations. Yet another form of stochastic training recommends randomly dropping entire layers [17], forcing the model to learn similar features across various layers which prevent extreme overfitting. In contrast, our technique encourages the model to use a linear combination of features instead of duplicating the same feature. Han *et al*. [18] propose Dense-Sparse-Dense (DSD), a similar training scheme, in which they apply weight regularization mid-training to encourage the development of sparse weights, and subsequently remove the regularization to restore dense weights. While DSD works at the level of individual parameters, our method is specifically designed to apply to convolutional filters.

Neuron ranking Interest in finding the least salient neurons/weights has a long history. LeCun [19] and Hassibi*et al*. [20] show that using the Hessian, which contains second-order derivative, identifies the weak neurons and performs better than using the magnitude of the weights. Computing the Hessian is expensive and thus is not widely used. Han *et*

al. [12] show that the norm of weights is still effective ranking criteria and yields sparse models. The sparse models do not translate to faster inference, but as a neuron ranking criterion, they are effective. Hu *et al*. [21] explore Average Percentage of Zeros (APoZ) in the activations and use a data-driven threshold to determine the cut-off. Molchanov *et al*. [9] recommend the second term from the Taylor expansion of the loss function.We provide detail comparison and show results on using these metrics with our training scheme in Section 5.

3. Motivation for Orthogonal Features

A feature for a convolutional filter is defined as the point-wise sum of the activations from individual kernels of the filter. A feature is considered useful if it helps to improve the generalization of the model. A model that has poor generalization usually has features that, in aggregate, capture limited directions in activation space [22]. On the other hand, if a model's features are orthogonal to one another, they will each capture distinct directions in activation space, leading to improved generalization. For a trivially-sized ConvNet, we can compute the maximally expressive filters by analyzing the correlation of features across layers and clustering them into groups [23]. However, this scheme is computationally impractical for the deep ConvNets used in real-world applications. Alternatively, a computationally feasible option is the addition of a regularization term to the loss function used in standard SGD training which encourages the minimization of the covariance of the activations, but this produces only limited improvement in model performance [24, 5]. A similar method, in which the regularization term instead encourages the orthogonality of filter weights, has also produced marginal improvements [25, 26, 27, 28]. Shang *et al*. [29] discovered the low-level filters are duplicated with opposite phase. Forcing filters to be orthogonal will minimize this duplication without changing the activation function. In addition to improvements in performance and generalization, Saxe *et al*. [30] show that the orthogonality of weights also improves the stability of network convergence during training. The authors of [28, 31] further demonstrate the value of orthogonal weights to the efficient training of networks. Orthogonal initialization is common practice for Recurrent Neural Networks due to their increased sensitivity to initial conditions [32], but it has somewhat fallen out of favor for ConvNets. These factors shape our motivation for encouraging orthogonality of features in the ConvNet and form the basis of our ranking criteria. Because features are dependent on the input data, determining their orthogonality requires computing statistics across the entire training set, and is therefore prohibitive. We instead compute the orthogonality of filter weights as a surrogate. Our experiments show that encouraging weight orthogonality through a regu-
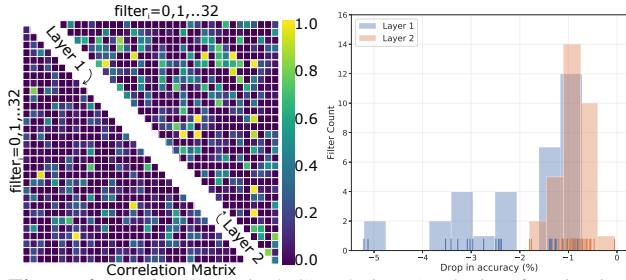
Figure 2: Left: Canonical Correlation Analysis of activations from two layers of a ConvNet trained on CIFAR-10. Right: Distribution of change in accuracy when the model is evaluated by dropping one filter at a time.

larization term is insufficient to promote the development of features which capture the full space of the input data manifold. Our method of dropping overlapping filters acts as an implicit regularization and leads to the better orthogonality of filters without hampering model convergence.

We use Canonical Correlation Analysis [33] (CCA) to study the overlap of features in a single layer. CCA finds the linear combinations of random variables that show maximum correlation with each other. It is a useful tool to determine if the learned features are overlapping in their representational capacity. Li *et al*. [34] apply correlation analysis to filter activations to show that most of the well-known ConvNet architectures learn similar representations. Raghu *et al*. [35] combine CCA with SVD to perform a correlation analysis of the singular values of activations from various layers. They show that increasing the depth of a model does not always lead to a corresponding increase of the model's dimensionality, due to several layers learning representations in correlated directions. We ask an even more elementary question - how correlated are the activations from various filters within a single layer? In an over-parameterized network like VGG-16, which has several convolutional layers with 512 filters each, it is no surprise that most of the filter activations are highly correlated. As a result, VGG-16 has been shown to be easily pruned - more than 50% of the filters can be dropped while maintaining the performance of the full network [9, 34]. Is this also true for significantly smaller convolutional networks, which under-fit the dataset?

We will consider a simple network with two convolutional layers of 32 filters each, and a softmax layer at the end. Training this model on CIFAR-10 for 100 epochs with an annealed learning rate results in test set accuracy of 58.2%, far below the 93.5% achieved by VGG-16. In the case of VGG-16, we might expect that correlation between filters is merely an artifact of the over-parameterization of the model - the dataset simply does not have a dimensionality high enough to require every feature to be orthogonal to every other. On the other hand, our small network has clearly failed to capture the full feature space of the training data, and thus any correlation between its filters is due to

inefficiencies in training, rather than over-parameterization.

Given a trained model, we can evaluate the contribution of each filter to the model's performance by removing *(zeroing out)* that filter and measuring the drop in accuracy on the test set. We will call this metric of filter importance the *"greedy Oracle"*. We perform this evaluation independently for every filter in the model, and plot the distribution of the resulting drops in accuracy in Figure 2 (right). Most of the second layer filters contribute less than 1% in accuracy and with first layer filters, there is a long tail. Some filters are important and contribute over 4% of accuracy but most filters are around 1%. This implies that even a tiny and under-performing network could be filter pruned without significant performance loss. The model has not efficiently allocated filters to capture wider representations of necessary features. Figure 2 (left) shows the correlations from linear combinations of the filter activations (CCA) at both the layers. It is evident that in both the layers there is a significant correlation among filter activations with several of them close to a near perfect correlation of 1 (*bright yellow spots* ▇). The second layer (upper right diagonal) has lot more overlap of features the first layer (lower right). For a random orthogonal matrix any value above 0.3 (*lighter than dark blue* ▇) is an anomaly. The activations are even more correlated if the linear combinations are extended to kernel functions [36] or singular values [35]. Regardless, it suffices to say that standard training for convolutional filters does not maximize the representational potential of the network.

4. Our Training Scheme : RePr

We modify the training process by cyclically removing redundant filters, retraining the network, re-initializing the removed filters, and repeating. We consider each filter (3D *tensor*) as a single unit, and represent it as a long vector - (f). Let $\mathbf{M}$ denote a model with $\mathcal{F}$ filters spread across $\mathbf{L}$ layers. Let $\widehat{\mathcal{F}}$ denote a subset of $\mathcal{F}$ filters, such that $\mathbf{M}_{\mathcal{F}}$ denotes a complete network whereas, $\mathbf{M}_{\mathcal{F}-\widehat{\mathcal{F}}}$ denotes a sub-network without that $\widehat{\mathcal{F}}$ filters. Our training scheme alternates between training the complete network ($\mathbf{M}_{\mathcal{F}}$) and the sub-network ($\mathbf{M}_{\mathcal{F}-\widehat{\mathcal{F}}}$). This introduces two hyper-parameters. First is the number of iterations to train each of the networks before switching over; let this be S_1 for the full network and S_2 for the sub-network. These have to be non-trivial values so that each of the networks learns to improve upon the results of the previous network. The second hyper-parameter is the total number of times to repeat this alternating scheme; let it be N. This value has minimal impact beyond certain range and does not require tuning.

The most important part of our algorithm is the metric used to rank the filters. Let $\mathcal{R}$ be the metric which asso-

ciates some numeric value to a filter. This could be a norm of the weights or its gradients or our metric - *inter*-filter orthogonality in a layer. Here we present our algorithm agnostic to the choice of metric. Most sensible choices for filter importance results in an improvement over standard training when applied to our training scheme *(see Ablation Study 6)*.

Our training scheme operates on a macro-level and is not a weight update rule. Thus, is not a substitute for SGD or other adaptive methods like Adam [37] and RmsProp [38]. Our scheme works with any of the available optimizers and shows improvement across the board. However, if using an optimizer that has parameters specific learning rates *(like Adam)*, it is important to re-initialize the learning rates corresponding to the weights that are part of the pruned filters $(\widehat{\mathcal{F}})$. Corresponding Batch Normalization [39] parameters $(\gamma \& \beta)$ must also be re-initialized. For this reason, comparisons of our training scheme with standard training are done with a common optimizer.

We reinitialize the filters $(\widehat{\mathcal{F}})$ to be orthogonal to its value before being dropped and the current value of non-pruned filters $(\mathcal{F} - \widehat{\mathcal{F}})$. We use the QR decomposition on the weights of the filters from the same layer to find the null-space and use that to find an orthogonal initialization point.

Our algorithm is training interposed with **Re**-initializing and **Pr**uning - **RePr** *(pronounced: reaper)*. We summarize our training scheme in Algorithm 1.

Algorithm 1: RePr Training Scheme

---
1   **for** N *iterations* **do**
2   **for** S_1 *iterations* **do**
3 | Train the full network: $\mathbf{M}_\mathcal{F}$
4   **end**
5   Compute the metric : $\mathcal{R}(f) \ \forall f \in \mathcal{F}$
6   Let $\widehat{\mathcal{F}}$ be bottom $p_\%$ of $\mathcal{F}$ using $\mathcal{R}(f)$
7   **for** S_2 *iterations* **do**
8 | Train the sub-network : $\mathbf{M}_{\mathcal{F} - \widehat{\mathcal{F}}}$
9   **end**
10   Reinitialize the filters $(\widehat{\mathcal{F}})$ s.t. $\widehat{\mathcal{F}} \perp \mathcal{F}$
11   *(and their training specific parameters*
12   *from BatchNorm and Adam, if applicable)*
13 **end**

We use a shallow model to analyze the dynamics of our training scheme and its impact on the train/test accuracy. A shallow model will make it feasible to compute the greedy Oracle ranking for each of the filters. This will allow us to understand the impact of training scheme alone without confounding the results due to the impact of ranking criteria. We provide results on larger and deeper convolutional networks in Section Results 8.

Consider a n layer vanilla ConvNet, without a skip or dense connections, with X filter each, as shown below:

$$\text{Img} \longmapsto \left[\text{CONV}(X) \rightarrow \text{RELU} \right]^n \longmapsto \text{FC} \longmapsto \text{Softmax}$$

We will represent this architecture as $C^n(X)$. Thus, a $C^3(32)$ has 96 filters, and when trained with SGD with a learning rate of 0.01, achieves test accuracy of 73%. Figure 1 shows training plots for accuracy on the training set (left) and test set (right). In this example, we use a RePr training scheme with $S_1 = 20, S_2 = 10, N = 3, p_\% = 30$ and the ranking criteria $\mathcal{R}$ as a greedy Oracle. We exclude a separate validation set of 5K images from the training set to compute the Oracle ranking. In the training plot, annotation [A] shows the point at which the filters are first pruned. Annotation [C] marks the test accuracy of the model at this point. The drop in test accuracy at [C] is lower than that of training accuracy at [A], which is not a surprise as most models overfit the training set. However, the test accuracy at [D] is the same as [C] but at this point, the model only has 70% of the filters. This is not a surprising result, as research on filter pruning shows that at lower rates of pruning most if not all of the performance can be recovered [9].

What is surprising is that test accuracy at [E], which is only a couple of epochs after re-introducing the pruned filters, is significantly higher than point [C]. Both point [C] and point [E] are same capacity networks, and higher accuracy at [E] is not due to the model convergence. In the standard training (orange line) the test accuracy does not change during this period. Models that first grow the network and then prune [40, 41], unfortunately, stopped shy of another phase of growth, which yields improved performance. In their defense, this technique defeats the purpose of obtaining a smaller network by pruning. However, if we continue RePr training for another two iterations, we see that the point [F], which is still at 70% of the original filters yields accuracy which is comparable to the point [E] (100% of the model size.

Another observation we can make from the plots is that training accuracy of RePr model is lower, which signifies some form of regularization on the model. This is evident in the Figure 4 (Right), which shows RePr with a large number of iterations ($N = 28$). While the marginal benefit of higher test accuracy diminishes quickly, the generalization gap between train and test accuracy is reduced significantly.

5. Our Metric : *inter*-filter orthogonality

The goals of searching for a metric to rank least important filters are twofold - (1) computing the greedy Oracle is not computationally feasible for large networks, and (2) the greedy Oracle may not be the best criteria. If a filter which captures a unique direction, thus not replaceable by a linear combination of other filters, has a lower contribution to

accuracy, the Oracle will drop that filter. On a subsequent re-initialization and training, we may not get back the same set of directions.

The directions captured by the activation pattern expresses the capacity of a deep network [42]. Making orthogonal features will maximize the directions captured and thus expressiveness of the network. In a densely connected layer, orthogonal weights lead to orthogonal features, even in the presence of ReLU [32]. However, it is not clear how to compute the orthogonality of a convolutional layer.

A convolutional layer is composed of parameters grouped into spatial kernels and sparsely share the incoming activations. Should all the parameters in a single convolutional layer be considered while accounting for orthogonality? The theory that promotes initializing weights to be orthogonal is based on densely connected layers (FC-layers) and popular deep learning libraries follow this guide[1] by considering convolutional layer as one giant vector disregarding the sparse connectivity. A recent attempt to study orthogonality of convolutional filters is described in [31] but their motivation is the convergence of very deep networks (10K layers) and not orthogonality of the features. Our empirical study suggests a strong preference for requiring orthogonality of individual filters in a layer (inter-filter & intra-layer) rather than individual kernels.

A filter of kernel size $k \times k$ is commonly a 3D tensor of shape $k \times k \times c$, where c is the number of channels in the incoming activations. Flatten this tensor to a 1D vector of size $k * k * c$, and denote it by f. Let J_ℓ denote the number of filters in the layer ℓ, where $\ell \in \mathbf{L}$, and $\mathbf{L}$ is the number of layers in the ConvNet. Let $\mathbf{W}_\ell$ be a matrix, such that the individual rows are the flattened filters (f) of the layer ℓ.

Let $\hat{\mathbf{W}}_\ell = \mathbf{W}_\ell / ||\mathbf{W}_\ell||$ denote the normalized weights. Then, the measure of Orthogonality for filter f in a layer ℓ (denoted by O_ℓ^f) is computed as shown in the equations below.

$$\mathbf{P}_\ell = |\hat{\mathbf{W}}_\ell \times \hat{\mathbf{W}}_\ell^T - I| \qquad (1)$$

$$O_\ell^f = \frac{\sum \mathbf{P}_\ell[f]}{J_\ell} \qquad (2)$$

$\mathbf{P}_\ell$ is a matrix of size $J_\ell \times J_\ell$ and $\mathbf{P}[i]$ denotes i^{th} row of $\mathbf{P}$. Off-diagonal elements of a row of $\mathbf{P}$ for a filter f denote angle (direction overlap) with all the other filters in the same layer with f. The sum of a row is minimum when other filters are orthogonal to this given filter. We rank the filters least important (thus subject to pruning) if this value is largest among all the filters in the network. While we compute the metric for a filter over a single layer, the ranking is computed over all the filters in the network. We do not enforce per layer rank because that would require learning a hyper-parameter $p_\%$ for every layer and some layers are

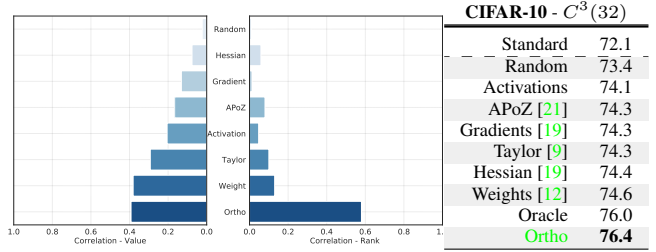[1]tensorflow:ops/init_ops.py#L543 & pytorch:nn/init.py#L350

Figure 3: Left: Pearson correlation coefficient of various metric values with the accuracy values from greedy Oracle. Center: Pearson correlation coefficient of filter ranks using various metric with rank from greedy Oracle Right: Test accuracy on CIFAR-10 using standard training and RePr training with various metrics
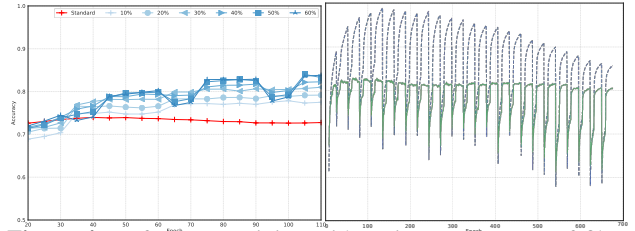
CIFAR-10 - $C^3(32)$	
Standard	72.1
Random	73.4
Activations	74.1
APoZ [21]	74.3
Gradients [19]	74.3
Taylor [9]	74.3
Hessian [19]	74.4
Weights [12]	74.6
Oracle	76.0
Ortho	**76.4**

Figure 4: Left: RePr training with various percentage of filters pruned. Shows average test accuracy over 5 epochs starting from epoch 20 for better visibility. Right:Marginal returns of multiple iterations of RePR - Training and Test accuracy on CIFAR-10

more sensitive than others. Our method prunes more filters from deeper layers compared to the earlier layers. This is in accordance with the distribution of contribution of each filter in a given network (Figure 2 right).

Computation of our metric does not require expensive calculations of the inverse of Hessian [19] or the second order derivatives [20] and is feasible for any sized networks. The most expensive calculations are L matrix products of size $J_\ell \times J_\ell$, but GPUs are designed for fast matrix-multiplications. Still, our method is more expensive than computing norm of the weights or the activations or the Average Percentage of Zeros (APoZ).

Given the choice of Orthogonality of filters, an obvious question would be to ask if adding a soft penalty to the loss function improve this training? A few researchers [25, 26, 27] have reported marginal improvements due to added regularization in the ConvNets used for task-specific models. We experimented by adding $\lambda * \sum_\ell \mathbf{P}_\ell$ to the loss function, but we did not see any improvement. Soft regularization penalizes all the filters and changes the loss surface to encourage random orthogonality in the weights without improving expressiveness.

6. Ablation study

Comparison of pruning criteria We measure the correlation of our metric with the Oracle to answer the question - how good a substitute is our metric for the filter importance ranking. Pearson correlation of our metric, henceforth referred to as Ortho, with the Oracle is 0.38. This is not

a strong correlation, however, when we compare this with other known metrics, it is the closest. Molchanov *et al.* [9] report Spearman correlation of their criteria (Taylor) with greedy Oracle at 0.73. We observed similar numbers for Taylor ranking during the early epochs but the correlation diminished significantly as the models converged. This is due to low gradient value from filters that have converged. The Taylor metric is a product of the activation and the gradient. High gradients correlate with important filters during early phases of learning but when models converge low gradient do not necessarily mean less salient weights. It could be that the filter has already converged to a useful feature that is not contributing to the overall error of the model or is stuck at a saddle point. With the norm of activations, the relationship is reversed. Thus by multiplying the terms together hope is to achieve a balance. But our experiments show that in a fully converged model, low gradients dominate high activations. Therefore, the Taylor term will have lower values as the models converge and will no longer be correlated with the inefficient filters. While the correlation of the values denotes how well the metric is the substitute for predicting the accuracy, it is more important to measure the correlation of the rank of the filters. Correlation of the values and the rank may not be the same, and the correlation with the rank is the more meaningful measurement to determine the weaker filters. Ortho has a correlation of 0.58 against the Oracle when measured over the rank of the filters. Other metrics show very poor correlation using the rank. Figure 3 (Left and Center) shows the correlation plot for various metrics with the Oracle. The table on the right of Figure 3 presents the test accuracy on CIFAR-10 of various ranking metrics. From the table, it is evident that Orthogonality ranking leads to a significant boost of accuracy compared to standard training and other ranking criteria.

Percentage of filters pruned One of the key factors in our training scheme is the percentage of the filters to prune at each pruning phase ($p_\%$). It behaves like the Dropout parameter, and impacts the training time and generalization ability of the model *(see Figure: 4)*. In general the higher the pruned percentage, the better the performance. However, beyond 30%, the performances are not significant. Up to 50%, the model seems to recover from the dropping of filters. Beyond that, the training is not stable, and sometimes the model fails to converge.

Number of RePr iterations Our experiments suggest that each repeat of the RePr process has diminishing returns, and therefore should be limited to a single-digit number (see Figure 4 (Right)). Similar to Dense-Sparse-Dense [18] and Born-Again-Networks [43], we observe that for most networks, two to three iterations is sufficient to achieve the maximum benefit.

Optimizer and S1/S2 Figure 5 (left) shows variance in improvement when using different optimizers. Our model
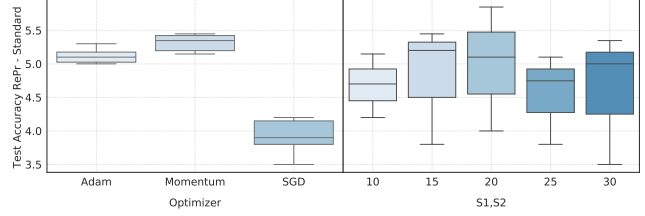

Figure 5: Left: Impact of using various optimizers on RePr training scheme. Right: Results from using different S1/S2 values. For clarity, these experiments only shows results with $S1 = S2$
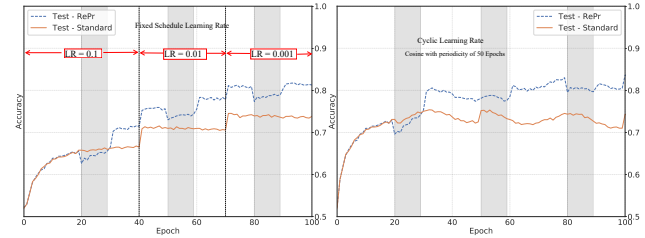

Figure 6: Test accuracy of a three layer ConvNet with 32 filters each over 100 epochs using standard scheme and our method - RePr on CIFAR-10. The shaded regions denote periods when only part of the network is trained for RePr. Left: Fixed Learning Rate schedule of 0.1, 0.01 and 0.001.Right: Cyclic Learning Rate with periodicity of 50 Epochs, and amplitude of 0.005 and starting LR of 0.001.

works well with most well-known optimizers. Adam and Momentum perform better than SGD due to their added stability in training. We experimented with various values of $S1$ and $S2$, and there is not much difference if either of them is large enough for the model to converge temporarily.

Learning Rate Schedules SGD with a fixed learning rate does not typically produce optimal model performance. Instead, gradually annealing the learning rate over the course of training is known to produce models with higher test accuracy. State-of-the-art results on ResNet, DenseNet, Inception were all reported with a predetermined learning rate schedule. However, the selection of the exact learning rate schedule is itself a hyperparameter, one which needs to be specifically tuned for each model. Cyclical learning rates [44] can provide stronger performance without exhaustive tuning of a precise learning rate schedule. Figure 6 shows the comparison of our training technique when applied in conjunction with fixed schedule learning rate scheme and cyclical learning rate. Our training scheme is not impacted by using these schemes, and improvements over standard training is still apparent.

7. Orthogonality and Distillation

Our method, RePr and Knowledge Distillation (KD) are both techniques to improve performance of compact mod-
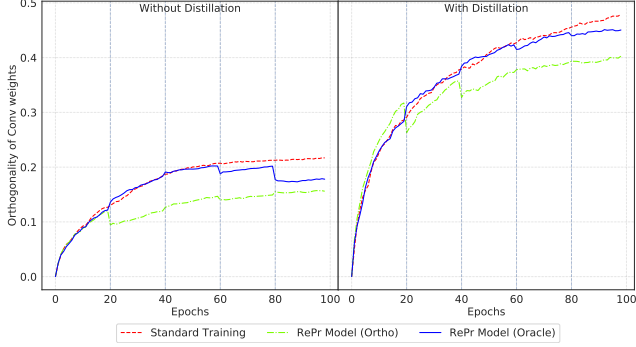
Figure 7: Comparison of orthogonality of filters (Ortho-sum - eq 2) in standard training and RePr training with and without Knowledge Distillation. **Lower value** signifies less overlapping filters. Dashed vertical lines denotes filter dropping.

$C^3(32)$	Std	KD	RePr	KD+RePr
CIFAR-10	72.1	74.8	76.4	**83.1**
CIFAR-100	47.2	56.5	58.2	**64.1**

Table 1: Comparison of Knowledge Distillation with RePr.

els. RePr reduces the overlap of filter representations and KD distills the information from a larger network. We present a brief comparison of the techniques and show that they can be combined to achieve even better performance.

RePr repetitively drops the filters with most overlap in the directions of the weights using the *inter*-filter orthogonality, as shown in the equation 2. Therefore, we expect this value to gradually reduce over time during training. Figure 7 (left) shows the sum of this value over the entire network with three training schemes. We show RePr with two different filter ranking criteria - Ortho and Oracle. It is not surprising that RePr training scheme with Ortho ranking has lowest Ortho sum but it is surprising that RePr training with Oracle ranking also reduces the filter overlap, compared to the standard training. Once the model starts to converge, the least important filters based on Oracle ranking are the ones with the most overlap. And dropping these filters leads to better test accuracy *(table on the right of Figure 3)*. Does this improvement come from the same source as the that due to Knowledge Distillation? Knowledge Distillation (KD) is a well-proven methodology to train compact models. Using soft logits from the teacher and the ground truth signal the model converges to better optima compared to standard training. If we apply KD to the same three experiments (see Figure 7, right), we see that all the models have significantly larger Ortho sum. Even the RePr (Ortho) model struggles to lower the sum as the model is strongly guided to converge to a specific solution. This suggests that this improvement due to KD is not due to reducing filter overlap. Therefore, a model which uses both the techniques should benefit by even better generalization. Indeed, that is the case as the combined model has significantly better performance than
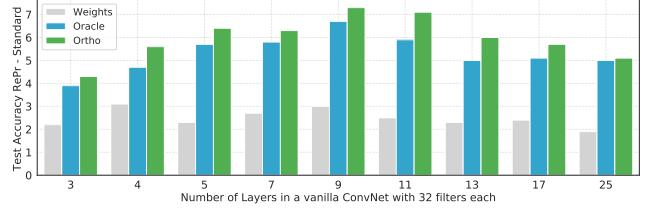

Figure 8: Accuracy improvement using RePr over standard training on Vanilla ConvNets across many layered networks [$C^n(32)$]

ResNet-20 on CIFAR-10					
Baseline		Various Training Schemes			
Original	**Our**	**DSD**	**BAN**	**RePr**	**RePr**
[1]	**Impl**	**[18]**	**[43]**	**Weights**	**Ortho**
8.7	8.4	7.8	8.2	7.7	6.9

Table 2: Comparison of test error from using various techniques.

either of the individual models, as shown in Table 1.

8. Results

We present the performance of our training scheme, RePr, with our ranking criteria, *inter*-filter orthogonality, Ortho, on different ConvNets [45, 1, 46, 47, 48]. For all the results provided RePr parameters are: $S_1 = 20$, $S_2 = 10$, $p_\% = 30$, and with three iterations, $N = 3$.

We compare our training scheme with other similar schemes like BAN and DSD in table 2. All three schemes were trained for three iterations *i.e.* N=3. All models were trained for 150 epochs with similar learning rate schedule and initialization. DSD and RePr (Weights) perform roughly the same function - sparsifying the model guided by magnitude, with the difference that DSD acts on individual weights, while RePr (Weights) acts on entire filters. Thus, we observe similar performance between these techniques. RePr (Ortho) outperforms the other techniques and is significantly cheaper to train compared to BAN, which requires N full training cycles.

Compared to modern architectures, vanilla ConvNets show significantly more inefficiency in the allocation of their feature representations. Thus, we find larger improvements from our method when applied to vanilla ConvNets, as compared to modern architectures. Table 3 shows test errors on CIFAR 10 & 100. Vanilla CNNs with 32 filters each have high error compared to DenseNet or ResNet but their inference time is significantly faster. RePr training improves the relative accuracy of vanilla CNNs by 8% on CIFAR-10 and 25% on CIFAR-100. The performance of baseline DenseNet and ResNet models is still better than vanilla CNNs trained with RePr, but these models incur more than twice the inference cost. For comparison, we also consider a reduced DenseNet model with only 5 layers, which has similar inference time to the 3-layer vanilla ConvNet. This model has many fewer parameters (by a factor

Layers	Params (×000)	Inf. Time (relative)	CIFAR-10		CIFAR-100	
			Std	RePr	Std	RePr
Vanilla CNN [32 filters / layer]						
3	20	1.0	27.9	23.6	52.8	41.8
8	66	1.7	26.8	19.5	50.9	36.8
13	113	2.5	26.6	20.6	51.0	37.9
18	159	3.3	28.2	22.5	51.9	39.5
DenseNet [k=12]						
5	1.7	0.9	39.4	36.2	43.5	40.9
40	1016	8.0	6.8	6.2	26.4	25.2
100	6968	43.9	5.3	5.6	22.2	22.1
ResNet						
20	269	1.7	8.4	6.9	32.6	31.1
32	464	2.2	7.4	6.1	31.4	30.1
110	1727	7.1	6.3	5.4	27.5	26.4
182	2894	11.7	5.6	5.1	26.0	25.3

Table 3: Comparison of test error on Cifar-10 & Cifar-100 of various ConvNets using Standard training vs RePr Training. Inf. Time shows the inference times for a single pass. All time measurements are relative to Vanilla CNN with three layers. Parameter count does not include the last fully-connected layer.

of $11\times$) than the vanilla ConvNet, leading to significantly higher error rates, but we choose to equalize inference time rather than parameter count, due to the importance of inference time in many practical applications. Figure 8 shows more results on vanilla CNNs with varying depth. Vanilla CNNs start to overfit the data, as most filters converge to similar representation. Our training scheme forces them to be different which reduces the overfitting (Figure 4 - right). This is evident in the larger test error of 18-layer vanilla CNN with CIFAR-10 compared to 3-layer CNN. With RePr training, 18 layer model shows lower test error.

RePr is also able to improve the performance of ResNet and shallow DenseNet. This improvement is larger on CIFAR-100, which is a 100 class classification and thus is a harder task and requires more specialized filters. Similarly, our training scheme shows bigger relative improvement on ImageNet, a 1000 way classification problem. Table 4 presents top-1 test error on ImageNet [49] of various ConvNets trained using standard training and with RePr. RePr was applied three times (N=3), and the table shows errors after each round. We have attempted to replicate the results of the known models as closely as possible with suggested hyper-parameters and are within $\pm 1\%$ of the reported results. More details of the training and hyper-parameters are provided in the supplementary material. Each subsequent RePr leads to improved performance with significantly diminishing returns. Improvement is more distinct in architectures which do not have skip connections, like Inception v1 and VGG and have lower baseline performance.

Our model improves upon other computer vision tasks that use similar ConvNets. We present a small sample of

	ImageNet				
Model	Standard Training	RePr Training			Relative Change
		N=1	N=2	N=3	
ResNet-18	30.41	28.68	27.87	27.31	-11.35
ResNet-34	27.50	26.49	26.06	25.80	-6.59
ResNet-50	23.67	22.79	22.51	22.37	-5.81
ResNet-101	22.40	21.70	21.51	21.40	-4.67
ResNet-152	21.51	20.99	20.79	20.71	-3.86
VGG-16	31.30	27.76	26.45	25.50	-22.75
Inception v1	31.11	29.41	28.47	28.01	-11.07
Inception v2	27.60	27.15	26.95	26.80	-2.99

Table 4: Comparison of test error (Top-1) on ImageNet with different models at various stages of RePr. N=1, N=2, N=3 are results after each round of RePr.

results from visual question answering and object detection tasks. Both these tasks involve using ConvNets to extract features, and RePr improves their baseline results. For object detection on COCO [50], using Feature Pyramid Network [51] and ResNet-50, mAP improves from 38.2 to 42.3. For visual question answering on VQAv1, using VQA-LSTM-CNN model [52], accuracy on Open-Ended questions increases from 60.3% to 64.6%.

9. Conclusion

We have introduced RePr, a training paradigm which cyclically drops and relearns some percentage of the least expressive filters. After dropping these filters, the pruned sub-model is able to recapture the lost features using the remaining parameters, allowing a more robust and efficient allocation of model capacity once the filters are reintroduced. We show that a reduced model needs training before re-introducing the filters, and careful selection of this training duration leads to substantial gains. We also demonstrate that this process can be repeated with diminishing returns.

Motivated by prior research which highlights inefficiencies in the feature representations learned by convolutional neural networks, we further introduce a novel *inter*-filter orthogonality metric for ranking filter importance for the purpose of RePr training, and demonstrate that this metric outperforms established ranking metrics. Our training method is able to significantly improve performance in under-parameterized networks by ensuring the efficient use of limited capacity, and the performance gains are complementary to knowledge distillation. Even in the case of complex, over-parameterized network architectures, our method is able to improve performance across a variety of tasks.

10. Acknowledgement

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 7

[2] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 1

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. 1

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1

[5] Michael Cogswell, Faruk Ahmed, Ross B. Girshick, C. Lawrence Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. *ICLR*, abs/1511.06068, 2016. 1, 2

[6] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *ICLR*, abs/1608.08710, 2017. 1

[7] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. 1

[8] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *JETC*, 2017. 1

[9] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *ICLR*, abs/1611.06440, 2017. 1, 2, 3, 4, 5, 6

[10] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. 1

[11] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. 1

[12] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *ICLR*, abs/1510.00149, 2016. 1, 2, 5

[13] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *NIPS Workshop on Machine Learning of Phones and other Consumer Devices*, abs/1710.01878, 2017. 1

[14] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018. 2

[15] Haibing Wu and Xiaodong Gu. Towards dropout training for convolutional neural networks. *Neural networks : the official journal of the International Neural Network Society*, 71, 2015. 2

[16] Li Wan, Matthew D. Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013. 2

[17] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016. 2

[18] Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, Bryan Catanzaro, and William J. Dally. Dsd: Dense-sparse-dense training for deep neural networks. 2016. 2, 6, 7

[19] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *NIPS*, 1989. 2, 5

[20] Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *NIPS*, 1992. 2, 5

[21] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *CoRR*, abs/1607.03250, 2016. 2, 5

[22] Ari S. Morcos, David G. T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. *CoRR*, abs/1803.06959, 2017. 2

[23] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. In *ICML*, 2014. 2

[24] Pau Rodríguez, Jordi Gonzàlez, Guillem Cucurull, Josep M. Gonfaus, and F. Xavier Roca. Regularizing cnns with locally constrained decorrelations. *ICLR*, abs/1611.01967, 2017. 2

[25] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *ICLR*, abs/1609.07093, 2017. 2, 5

[26] Ben Poole, Jascha Sohl-Dickstein, and Surya Ganguli. Analyzing noise in autoencoders and deep networks. *NIPS Workshop on Deep Learning*, abs/1406.1831, 2013. 2, 5

[27] Pengtao Xie, Barnabás Póczos, and Eric P. Xing. Near-orthogonality regularization in kernel methods. In *UAI*, 2017. 2, 5

[28] Di Xie, Jiang Xiong, and Shiliang Pu. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[29] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *ICML*, 2016. 2

[30] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *ICLR*, abs/1312.6120, 2014. 2

[31] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel S. Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *ICML*, 2018. 2, 5

[32] Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Christopher Joseph Pal. On orthogonality and learning recurrent networks with long term dependencies. In *ICML*, 2017. 2, 5

[33] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4), 1936. 3

[34] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John E. Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *ICLR*, 2016. 3

[35] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *NIPS*, 2017. 3

[36] David R. Hardoon, Sándor Szedmák, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 2004. 3

[37] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, abs/1412.6980, 2015. 4

[38] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012. 4

[39] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 4

[40] Xiaoliang Dai, Hongxu Yin, and Niraj K. Jha. Nest: A neural network synthesis tool based on a grow-and-prune paradigm. *CoRR*, abs/1711.02017, 2017. 4

[41] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *NIPS*, 2015. 4

[42] Maithra Raghu, Ben Poole, Jon M. Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *ICML*, 2017. 5

[43] Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *ICML*, 2018. 6, 7

[44] Leslie N. Smith. Cyclical learning rates for training neural networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017. 6

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 7

[46] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 7

[47] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 7

[48] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. *CVPR*, 2017. 7

[49] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015. 8

[50] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 8

[51] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 8

[52] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015. 8