Filter Grafting for Deep Neural Networks

Fanxu Meng[1,2*], Hao Cheng[2*†], Ke Li[2], Zhixin Xu[1], Rongrong Ji[3,4], Xing Sun[2], Guangming Lu [1†]
[1] School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China
[2] Tencent Youtu Lab, Shanghai, China
[3] Department of Artificial Intelligence, School of Informatics, Xiamen University, China
[4] Peng Cheng Laboratory, China
{louischeng, tristanli, winfredsun}@tencent.com, {18S151514,xuzhixin}@stu.hit.edu.cn, luguangm@hit.edu.cn, rrj@xmu.edu.cn

Abstract

*This paper proposes a new learning paradigm called **filter grafting**, which aims to improve the representation capability of Deep Neural Networks (DNNs). The motivation is that DNNs have unimportant (invalid) filters (e.g., l_1 norm close to 0). These filters limit the potential of DNNs since they are identified as having little effect on the network. While filter pruning removes these invalid filters for efficiency consideration, filter grafting re-activates them from an accuracy boosting perspective. The activation is processed by grafting external information (weights) into invalid filters. To better perform the grafting process, we develop an **entropy-based criterion** to measure the information of filters and an **adaptive weighting strategy** for balancing the grafted information among networks. After the grafting operation, the network has very few invalid filters compared with its untouched state, enpowering the model with more representation capacity. We also perform extensive experiments on the classification and recognition tasks to show the superiority of our method. For example, the grafted MobileNetV2 outperforms the non-grafted MobileNetV2 by about 7 percent on CIFAR-100 dataset. Code is available at https://github.com/fxmeng/filter-grafting.git.*

1. Introduction

Since Krizhevsky *et al*. [7] make a breakthrough in the 2012 ImageNet competition [17], researchers have got significant advancements in exploring various architectures for DNNs (Szegedy *et al*. [20]; He *et al*. [4]; Lu *et al*. [13, 12]; Zheng *et al*. [27]). DNNs gradually become very popular and powerful models in areas including computer vision [7, 11], speech recognition [2], and language processing [24]. However, recent studies show that DNNs have invalid (unimportant) filters [9]. These filters are identified as having a small effect on output accuracy. Removing certain filters could accelerate the inference of DNNs without hurting much performance. This discovery inspires many works studying how to decide which filters are unimportant [14] and how to effectively remove the filters with tolerable performance drop [19, 10].

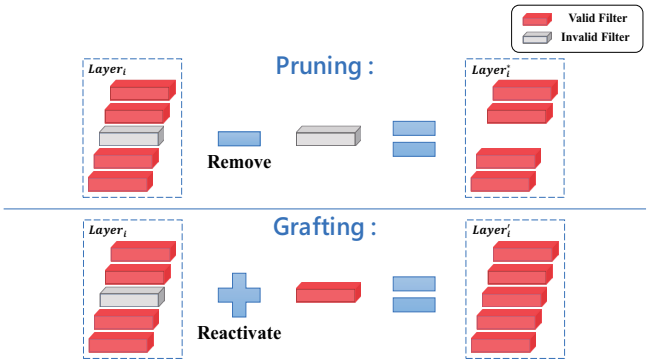

Figure 1. An illustration of the difference between filter pruning and filter grafting. For filter grafting, we graft external information into invalid filters without changing the model structure. (best viewed in color)

However, it is unclear that whether directly abandoning such filters and components is the best choice. What if, such traditional *invalid* filters are indeed useful in certain senses? The same story happens in the ensemble learning like boosting, where while a single weak classifier is poor, their combination and retraining might open a gate towards optimal performance. Besides, given multiple networks, it is unclear whether one network can learn from the others. In this paper, we investigate the possibility to re-activate the invalid filters in one network by bringing outside information. This is achieved by proposing a novel filter grafting scheme, as illustrated in Figure 1. Filter grafting differs from filter pruning in the sense that we re-activate filters by assigning

[1]In the author list, * denotes that authors contribute equally and are listed in alphabetical order; † denotes corresponding authors.

methods	without changing model structure ?	one stage ?	without supervision ?
filter pruning [9]	×	×	✓
distillation [6]	✓	×	×
deep mutual learning [25]	✓	✓	×
RePr [15]	✓	×	✓
filter grafting	✓	✓	✓

Table 1. The difference between filter grafting and other learning methods

new weights, which maintains the number of layers and filters within each layer as the same. The grafted network has a higher representation capability since more valid filters in the network are involved in processing information.

A key step in filter grafting is choosing proper information source (*i.e.*, where should we graft the information from). In this paper, we thoroughly study this question and claim that we should graft the information from outside (other networks) rather than inside (self-network). Generally, we could train several networks in parallel. During training at certain epochs, we graft a network's meaningful filters into another network's invalid filters. By performing grafting, each network could learn external information from other networks. The details can be found in Section 3.

There are three main contributions of this paper:

- We propose a new learning paradigm called **filter grafting** for DNNs. Grafting could re-activate the invalid filters to improve the potential of DNNs without changing the network structure.
- An **entropy based criterion** and an **adaptive weighting strategy** are developed to further improve the performance of filter grafting method.
- We perform extensive experiments on classification and recognition tasks and show grafting could substantially improve the performance of DNNs. For example, the grafted MobileNetV2 achieves 78.32% accuracies on CIFAR-100, which is about 7% higher than non-grafted MobileNetV2.

2. Related Work

Filter Pruning. Filter pruning aims to remove the invalid filters to accelerate the inference of the network. [9] first utilizes l_1 norm criterion to prune unimportant filters. Since then, more criterions came out to measure the importance of the filters. [**?**] utilizes spectral clustering to decide which filter needs to be removed. [19] proposes an inherently data-driven method that utilizes Principal Component Analysis (PCA) to specify the proportion of the energy that should be preserved. [21] applies subspace clustering to feature maps to eliminate the redundancy in convolutional filters. While instead of abandoning the invalid filters, filter grafting intends to activate them. It is worth noting that even though the motivation of filter grafting is opposite to pruning, grafting still involves choosing a proper criterion to decide which filters are unimportant. Thus different criterions from pruning are readily applied to grafting.

Distillation and Mutual Learning. Grafting may involve training multiple networks in parallel. Thus this process is similar to distillation [6] and mutual learning [25]. The difference between grafting and distillation is that distillation is a 'two-stage' process. First, we need to train a large model (teacher), then use the trained model to teach a small model (student). While grafting is a 'one-stage' process, we graft the weight during the training process. The difference between mutual learning and grafting is that mutual learning needs a mutual loss to supervise each network to learn and do not generalize well to multiple networks. While grafting does not need supervised loss and performs much better when we add more networks into the training process. Also, we graft the weight at each epoch instead of each iteration, thus greatly reduce communication costs among networks.

RePr. RePr [15] is similar to our work which considers improving network on the filter level. However, the motivation of RePr is that there exists unnecessary overlaps in the features captured by the networks filters. RePr first prunes overlapped filters to train the sub-network, then restores the pruned filters and re-trains the full network. In this sense, RePr is a multi-stage training algorithm. In contrast, the motivation of filter grafting is that the filter whose l_1 norm is smaller contributes less to the network output. Thus the filters that each method operates are different. Also grafting is a one-stage training algorithm which is more efficient. To better illustrate how grafting differs from the above learning types. We draw a table in Table 1. From Table 1, filter grafting is a one stage learning method, without changing network structure and does not need supervised loss.

3. Filter Grafting

This section arranges as follows: In Section 3.1, we study the source of information that we need during grafting process; In Section 3.2, we propose two criterions to calculate the information of filters; In Section 3.3, we discuss how to effectively use the information for grafting; In Section 3.4, we extend grafting method to multiple networks and propose our final entropy-based grafting algorithm.

3.1. Information Source for Grafting

In the remaining, we would call the original invalid filters as 'rootstocks' and call the meaningful filters or information to be grafted as 'scions', which is consistent with botany interpretation for grafting. Filter grafting aims to transfer information (weights) from scions to rootstocks, thus selecting useful information is essential for grafting. In this paper, we propose three ways to get scions.

3.1.1 Noise as Scions

A simple way is to graft gaussian noise $\mathcal{N}(0, \sigma_t)$ into invalid filters, since gaussian noise is commonly used for weight initialization of DNNs [8, 3]. Before grafting, the invalid filters have smaller l_1 norm and have little effects for the output. But after grafting, the invalid filters have larger l_1 norm and begin to make more effects to DNNs.

$$\sigma_t = a^t (0 < a < 1) \tag{1}$$

We also let σ_t decrease over time (see (1)), since too much noise may make the model harder to converge.

3.1.2 Internal Filters as Scions

Instead of adding random noise, we add the weights of other filters (l_1 norm is bigger) into the invalid filters (l_1 norm is smaller). Grafting is processed inside a single network. Specifically, for each layer, we sort the filters by l_1 norm and set a threshold γ. For filters whose l_1 norm are smaller than γ, we treat these filters as invalid ones. Then we graft the weights of the i-th largest filter into the i-th smallest filter. This procedure is illustrated in Figure 2.

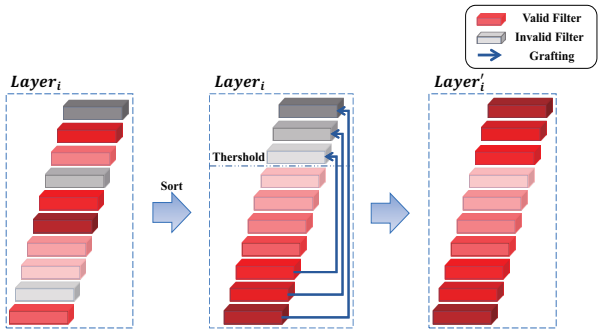

Figure 2. Grafting internal filters. We first sort the filters by l_1 norm, then graft the weights from filters with larger l_1 norm into filters with smaller l_1 norm. (best viewed in color)

Since the invalid filters have new weights with larger l_1 norm, they can be activated to have a bigger influence on the output. But this method does not bring new information to the network since the weights are grafted inside the self network. We further evaluate it via the language of information theory. To simplify the proving process, we deal with two filters in a certain layer of the network (See Theorem 1, proof can be found in the supplementary material). From Theorem 1, selecting internal filters as scions does not bring new information. The experiment in Section 4.1 is also consistent with our analysis.

Theorem 1 *Suppose there are two filters in a certain layer of the network, denoted as random variables X and Y. Z is another variable which satisfies $Z = X + Y$, then $H(X, Y) = H(X, Z) = H(Y, Z)$, where H denotes the entropy from information theory.*

3.1.3 External Filters as Scions

In response to the shortcomings of adding random noise and weights inside a single network, we select external filters from other networks as scions. Specifically, we could train two networks, denoted as M_1 and M_2, in parallel. During training at certain epochs, we graft the valid filters' weights of M_1 into the invalid filters of M_2. Compared to the grafting process in Section 3.1.2, we make two modifications:

- The grafting is processed at layer level instead of filter level, which means we graft the weights of all the filters in a certain layer in M_1 into the same layer in M_2 (also M_2 into M_1, inversely). Since two networks are initialized with different weights, the location of invalid filters are statistically different and only grafting information into part of filters in a layer may break layer consistency (see more analyses and experimental results in the supplementary material). By performing grafting, the invalid filters of two networks can learn mutual information from each other.

- When performing grafting, the inherent information and the extoic information are weighted. Specifically, We use $W_i^{M_2}$ denotes the weights of the i-th layer of M_2, $W_i^{M_2'}$ denotes the weights of the i-th layer of M_2 after grafting. Then:

$$W_i^{M_2'} = \alpha W_i^{M_2} + (1 - \alpha) W_i^{M_1} \quad (0 < \alpha < 1) \tag{2}$$

  Suppose $W_i^{M_2}$ is more informative than $W_i^{M_1}$, then α should be larger than 0.5.

The two networks grafting procedure is illustrated in Figure 3. From Equation (2) and Figure 3, there are two key points in grafting: 1) how to calculate the information of $W_i^{M_1}$ and $W_i^{M_2}$; 2) How to decide the weighting coefficient α. We thoroughly study these two problems in Section 3.2 and Section 3.3. Also, we hope to increase the diversity of two networks, thus two networks are initialized differently

and some hyper-parameters of two networks are also different from each other (e.g., learning rate, sampling order of data ...). It is worth noting that when performing grafting algorithm on two networks, the two networks have the same weights after grafting process from (2). But grafting is only performed at each epoch. For other iteration steps, since the two networks are learned with different hyper-parameters, their weights are different from each other. Also, this problem disappears when we add more networks ($N > 2$) in grafting algorithm. Multiple networks grafting can be found in Section 3.4.

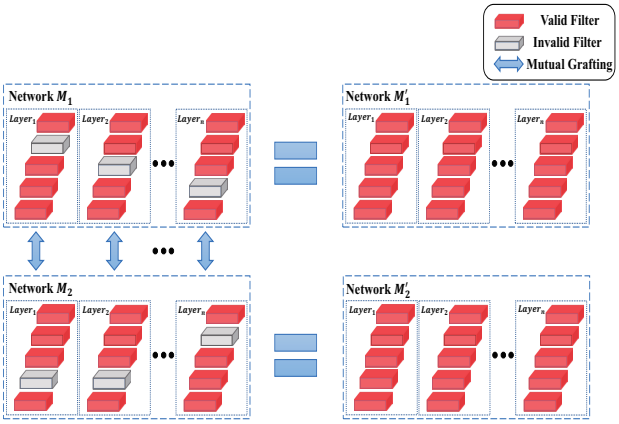

Figure 3. Grafting between two networks. Each network accepts information from the other network. (best viewed in color)

3.2. Criterions for Calculating Information of Filters and Layers

In this section, we study two criterions to calculate the information of filters or layers.

3.2.1 L_1 norm

In previous sections, we use l_1 norm to measure the information of filters. Denote $W_{i,j} \in \mathbb{R}^{N_i \times K \times K}$ as the weight of the j-th filter in the i-th convolutional layer, where N_i is the number of filters in i-th layer. Its l_1 norm can be presented by:

$$\|W_{i,j}\|_1 = \sum_{n=1}^{N_i} \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} |W_{i,j}(n, k_1, k_2)| \tag{3}$$

The l_1 norm criterion is commonly used in many research [9, 23, 22]. But recent studies show smaller-norm-less-important criterion is not always true. One special case is that 0-1 regularly arranged filters are better than all 1 filters. [5] also points out that there are some pre-requisites to utilize this smaller-norm-less-important criterion. Otherwise, pruning may hurt valid filters.

3.2.2 Entropy

While l_1 norm criterion only concentrates on the absolute value of filter's weight, we pay more attention to the variation of the weight. A problem of l_1 norm criterion is that l_1 norm neglects the variation of the weight. Suppose a filter's weight $W_{i,j} \in \mathbb{R}^{N_i \times K \times K}$ satisfies $W_{i,j}(n, k_1, k_2) = a$ for each $n \in \{1, \ldots, N_i\}$ and $k_1, k_2 \in \{1, \ldots, K\}$, Each single value in $W_{i,j}$ will be the same. Thus when using $W_{i,j}$ to operate convolution on the input, each part of the input is contributed equally to the output even though a is big. Thus the filter can not discriminate which part of the input is more important. Based on the above analyses, we choose to measure the variation of the weight. We suppose each value of $W_{i,j}$ is sampled from a distribution of a random variable X and use the entropy to measure the distribution. Suppose the distribution satisfies $P(X = a) = 1$, then each single value in $W_{i,j}$ is the same and the entropy is 0. While calculating the entropy of continuous distribution is hard, we follow the strategy from [18, 1]. We first convert continuous distribution to discrete distribution. Specifically, we divide the range of values into m different bins and calculate the probability of each bin. Finally, the entropy of the variable can be calculated as follows:

$$H(W_{i,j}) = -\sum_{k=1}^{B} p_k \log p_k \tag{4}$$

Where B is the number of bins and p_k is the probability of bin k. A smaller score of $H(W_{i,j})$ means the filter has less variation (information).

Suppose layer i has C filters, then the total information of the layer i is:

$$H(W_i) = \sum_{j=1}^{C} H_{i,j} \tag{5}$$

But one problem of (5) is that it neglects the correlations among the filters since (5) calculates each filter's information independently. To keep layer consistency, we directly calculate the entropy of the whole layer's weight $W_i \in \mathbb{R}^{N_i \times N_{i+1} \times K \times K}$ as follows:

$$H(W_i) = -\sum_{k=1}^{B} p_k \log p_k \tag{6}$$

Different from (4), the values to be binned in (6) are from the weight of the whole layer instead of a single filter. In the supplementary material, we prove layer consistency is essential for grafting algorithm.

3.3. Adaptive Weighting in Grafting

In this part, we propose an adaptive weighting strategy for weighting two models' weight from (2). Denote $W_i^{M_1}$

Algorithm 1 Entropy-based Multiple Networks Grafting

Input:

Number of networks K, M_k denotes the k-th network; Number of layers L; Training iterations $\mathcal{N} = \{1, \ldots, N_{max}\}$; Number of iterations for each epoch N_T; Training dataset $\mathcal{D}$; Initial weights for each layer of each network $\{\mathbf{W}_l^{M_k} : k = 1, \ldots, K; l = 1, \ldots, L\}$; Different hyper-parameters for each network $\{\lambda_k : k = 1, \ldots K\}$.

Iteration:

for $n = 1$ to N_{max}
  for $\mathbf{k} \in \{1, \ldots K\}, \mathbf{l} \in \{1, \ldots L\}$ **parallel do**
    Update model parameters $\mathbf{W}_l^{M_k}$ based on $\mathcal{D}$ with λ_k //Update model weights at each iteration.
    if $n \ mod \ N_T = 0$
      Get the weighting coefficient α from (7) //Graft model weights at each epoch.
      $\mathbf{W}_l^{M_k} = \alpha \mathbf{W}_l^{M_k} + (1 - \alpha)\mathbf{W}_l^{M_{k-1}}$
    end if
  end for
end for

and $H(W_i^{M_1})$ as the weight and information of layer i in network M_1, respectively. The calculation of $H(W_i^{M_1})$ can be referred to (6). We enumerate two conditions that need to be met for calculating the coefficient α.

- The coefficient α from (2) should be equal to 0.5 if $H(W_i^{M_2}) = H(W_i^{M_1})$ and larger than 0.5 if $H(W_i^{M_2}) > H(W_i^{M_1})$.
- Each network should contain part of self information even though $H(W_i^{M_2}) \gg H(W_i^{M_1})$ or $H(W_i^{M_2}) \ll H(W_i^{M_1})$.

In response to the above requirements, the following adaptive coefficient is designed:

$$\alpha = A * (arctan(c * (H(W_i^{M_2}) - H(W_i^{M_1})))) + 0.5 \quad (7)$$

where A and c from (7) are the fixed hyper-parameters. α is the coefficient of (2). We further depict a picture in Figure 4. We can see this function well satisfies the above conditions.

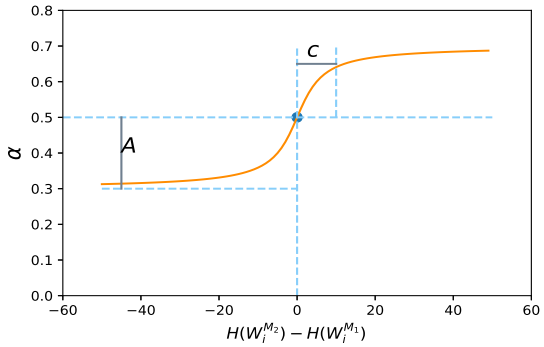

Figure 4. The adaptive coefficient in grafting process.

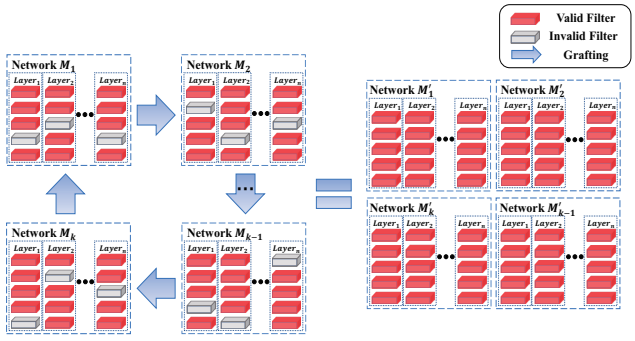

Figure 5. Grafting with multiple networks. The network M_k accepts information from M_{k-1}. (best viewed in color)

3.4. Extending Grafting to Multiple Networks

Grafting method can be easily extended to a multi-networks case, as illustrated in Figure 5. At each epoch during training, each network M_k accepts the information from M_{k-1}. After certain training epochs, each network contains information from all the other networks. The weighting coefficient is also calculated adaptively. From Section 4.5, we find that by using grafting to train multiple networks, each network achieves much performance gain. We propose our entropy-based grafting in Algorithm 1. It is worth noting that grafting is performed on multiple networks in parallel, which means when we use $\mathbf{W}_l^{M_{k-1}}$ to update $\mathbf{W}_l^{M_k}$, $\mathbf{W}_l^{M_{k-1}}$ has not been updated by grafting yet.

4. Experiment

This section arranges as follows: In Section 4.1, we examine how different information sources affect grafting method; In Section 4.2, we prove entropy-based grafting is better than l_1 norm-based grafting; In Section 4.3, we an-

alyze the training diversity when performing grafting; In Section 4.4, we compare grafting with other learning methods; In Section 4.5, we show by using multiple-networks, grafting could greatly improve the performance of the network; In Section 4.6 and Section 4.7, we examine grafting on close-set classification and open-set recognition tasks; In Section 4.8, we further analyze the effectiveness of grafting algorithm. All the experiments are reproducible. The code is available upon requirement and will be released online.

4.1. Selecting Useful Information Source

We propose three ways to get scions in Section 3 and experimentally examine the three ways on CIFAR-10 and CIFAR-100 datasets in Table 2. Vanilla DNN training without grafting is taken as the baseline. All the methods use MobileNetV2 as the base model. For a fair comparison, the same hyper-parameters are deployed for each method: mini-batch size (256), optimizer (SGD), initial learning rate (0.1), momentum (0.9), weight decay (0.0005), number of epochs (200), learning rate decay (0.1 at every 60 epochs). 'External' here involves training two networks in parallel. In practice, we find the performance of each network in the 'external' method is very close to each other. Thus in the remaining, we always record the first network's performance.

	CIFAR-10	CIFAR-100
baseline	92.42	71.44
noise	92.51	72.34
internal	92.68	72.38
external	**92.94**	**72.90**

Table 2. Comparison of different scion sources.

From Table 2, the performance of 'internal scions' is similar to 'noise', since we prove in Theorem 1 that choosing internal filters as scions does not bring new information to the network. While choosing external filters as scions achieves the best result among the three methods. In the remaining, all the grafting experiments choose external filters as scions.

4.2. Comparison of L_1 norm & Entropy Criterions

We propose two criterions to measure the inherent information of filters in Section 3.2. In this part, we quantitatively evaluate the l_1 norm-based grafting and the entropy-based grafting on CIFAR-10 and CIFAR-100 dataset. The results are listed in Table 3. Two networks are used for grafting, with an identical model structure and training hyper-parameters. From Table 3, we can find that, entropy-based grafting beats l_1 norm-based grafting on every model and dataset setting.

model	method	CIFAR-10	CIFAR-100
ResNet32	baseline	92.83	69.82
	l_1 norm	93.24	70.69
	entropy	**93.33**	**71.16**
ResNet56	baseline	93.50	71.55
	l_1 norm	94.09	72.73
	entropy	**94.28**	**73.09**
ResNet110	baseline	93.81	73.21
	l_1 norm	94.37	73.65
	entropy	**94.60**	**74.70**
MobileNetV2	baseline	92.42	71.44
	l_1 norm	92.94	72.90
	entropy	**93.53**	**73.26**

Table 3. Comparison of grafting by l_1 norm & entropy.

4.3. Evaluation of Training Diversity in Grafting

We find that the performance raises when we increase the training diversity of two networks. Since grafting is about transferring weights between models, the network can learn better if the external information (weights) has more variations. To achieve this, we could diversify the hyper-parameters setting (sampling order and learning rate in our case) to see how these factors affect grafting performance. The results are listed in Table 4. Cosine annealing LR schedule with different initial learning rate is set for each model in different LR case (This ensures that at each step, the learning rate for each model is different). We find that the weight variations brought by sampling order and learning rate enrich the grafting information and thus encourage the models to learn better. In the remaining, when performing grafting, all the networks use different hyper-parameters in terms of data loader and learning rate.

different order	different LR	CIFAR10	CIFAR100
×	×	93.05	71.91
✓	×	93.53	73.26
✓	✓	**94.20**	**74.15**

Table 4. Hyper-parameters verification for grafting. The backbone is MobileNetV2.

4.4. Comparing Grafting with Other Methods

We thoroughly study the difference between grafting and other learning methods in Table 1. In this part, we experimentally compare grafting with other methods on CIFAR-10 and CIFAR-100 datasets in Table 5.

For a fair comparison, 'distillation', 'mutual learning' and 'filter grafting' all involve training two networks. The difference between distillation and grafting is that distillation is a two-stage training procedure. When performing distillation, we first train one network until convergence,

Dataset	method	ResNet32	ResNet56	ResNet110	MobileNetV2	WRN28-10
CIFAR-10	baseline	92.83	93.50	93.81	92.42	95.75
	distillation [6]	93.11	92.05	92.34	92.37	95.70
	mutual learning [25]	92.80	–	–	–	95.66
	RePr [15]	93.90	–	94.60	–	–
	filter grafting	**93.94**	**94.73**	**94.96**	**94.20**	**96.40**
CIFAR-100	baseline	69.82	71.55	73.21	71.44	80.65
	distillation [6]	70.96	72.03	73.32	73.37	81.03
	mutual learning [25]	70.19	–	–	–	80.28
	RePr [15]	69.90	–	73.60	–	–
	filter grafting	**71.28**	**72.83**	**75.27**	**74.15**	**81.62**

Table 5. Comparion of filter grafting with other learning methods. '–' denotes the result is not reported in the corresponding paper.

then we use the network, as a teacher, to distill knowledge into the student network. For a fair comparison with grafting, the network structrue for teacher and student is the same which is consistent with the setting in [25]. While for grating, training is completed in one stage without the retraining process. The difference between mutual learning and grafting is that mutual learning trains two networks with another strong supervised loss and communication costs are heavy between networks. One should carefully choose the coefficient for mutual supervised loss and main loss when using the mutual learning method. While for grafting, transferring weights does not need supervision. We graft the weights by utilizing entropy to adaptively calculate the weighting coefficient which is more efficient. The results from Table 5 show that filter grafting achieves the best results among all the learning methods.

4.5. Grafting with Multiple Networks

The power of filter grafting is that we could greatly increase the performance by involving more networks in grafting algorithm. We examine the effect of multi-networks grafting in Table 6.

method	CIFAR-10	CIFAR-100
baseline	92.42	71.44
2 models grafting	94.20	74.15
3 models grafting	94.55	76.21
4 models grafting	95.23	77.08
6 models grafting	**95.33**	**78.32**
8 models grafting	95.20	77.76
6 models ensemble	94.09	76.75

Table 6. Grafting with multiple networks (MobileNetV2).

As we raise the number of networks, the performance gets better. For example, the performance with 6 models grafting could outperform the baseline by about 7 percent which is a big improvement. The reason is that MobileNetV2 is based on depth separable convolutions, thus the filters may learn insufficient knowledges. Filter grafing could help filters learn complementary knowledges from other networks, which greatly improves the network's potential. Also it is worth noting that the result of 6 models grafting is even better than 6 models ensembles. But unlike ensemble, grafting only maintains one network for testing. However, the performance stagnates when we add the number of models to 8 in grafting algorithm. We assume the cause might be that the network receives too much information from outside which may affect its self-information for learning. How to well explain this phenomenon is an interesting future work.

4.6. Grafting on ImageNet

To test the performance of grafting on a larger dataset, we also validate grafting on ImageNet, an image classification dataset with over 14 million images. We compare grafting with the baseline on ResNet18 and ResNet34 models. The baseline hyper-parameters' setting is consistent with official PyTorch setting for ImageNet[1]: minibatch size (256), initial learning rate (0.1), learning rate decay (0.1 at every 30 epochs), momentum (0.9), weight decay (0.0001), number of epochs (90) and optimizer (SGD). To increase the training diversity, we use different learning rates and data loaders for two networks when performing grafting. The other hyper-parameters' setting is consistent with the baseline. The results inTable 7 shows that grafting can also handle larger datasets.

4.7. Grafting on ReID Task

Grafting is a general training method for convolutional neural networks. Thus grafting can not only apply to the classification task but also other computer vision tasks. In this part, we evaluate the grafting on Person re-identification (ReID) task, an open set retrieval problem in distributed multi-camera surveillance, aiming to match people appearing in different non-overlapping camera views.

[1]https://github.com/pytorch/examples/tree/master/imagenet

model	method	top-1	top-5
ResNet18	baseline	69.15	88.87
	grafting	**71.19**	**90.01**
ResNet34	baseline	72.60	90.91
	grafting	**74.58**	**92.05**
ResNet50	baseline	75.92	92.81
	grafting	**76.76**	**93.34**

Table 7. Grafting on ImageNet Dataset

We conduct experiments on two person ReID datasets: Market1501 [26] and DukeMTMC-ReID (Duke) [16, 28]. The baseline hyper-parameters' setting is consistent with [29]: mini-batch size (32), pretrained (True), initial learning rate (0.1), learning rate decay (0.1 at every 20 epochs), number of epochs (60). Besides data loaders and learning rate, the other hyper-parameters' setting is consistent with the baseline. Table 8 shows that for each model and each dataset, grafting performs better than the baseline. Besides, as mentioned before, increasing the number of networks in grafting can further improve the performance.

model	method	Market1501		Duke	
		mAP	rank1	mAP	rank1
ResNet50	baseline	67.6	86.7	56.2	76.2
	2 models	70.6	87.8	60.8	79.8
	4 models	**73.33**	**89.2**	**62.1**	**79.8**
MobileNetV2	baseline	56.8	81.3	47.6	71.7
	2 models	63.7	85.2	53.4	76.1
	4 models	**64.5**	**85.8**	**54.3**	**76.3**

Table 8. Grafting on ReID Task

4.8. Effectiveness of Grafting

In this part, we further analyze the effectiveness of the grafting method. To prove grafting does improve the potential of the network, we calculate the number of invalid filters and information gain after the training process. We select MobileNetV2, which is trained on CIFAR-10 with grafting algorithm, for this experiment. The same network structure without grafting is chosen as the baseline. Experimental results are reported in Figure 6 and Figure 7.

From Figure 6, under the threshold of 1e-3, there are about 50% filters are invalid or unimportant for the base network, whereas the grafted network only has a small part of filters counted as 'invalid', which shows grafting does help network reduce invalid filters. From Figure 7, the model trained by grafting contains more information than the baseline. Also, the network can gain more information by training multiple networks for grafting method. Thus from the above analysis, we confirm that grafting could improve the potential of neural networks. More analyses can be found in the supplementary material, including the evaluation of

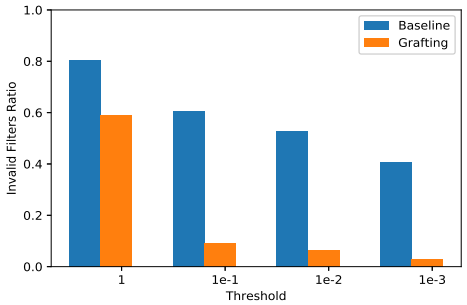

Figure 6. Ratio of filters whose l_1 norm under some threshold.

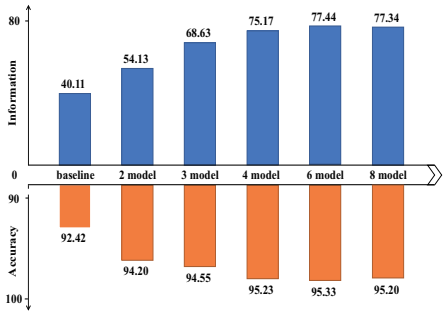

Figure 7. Entropy and accuracy of the baseline network and grafted network. The network's information is defined as the sum of all the layers' entropy in a **single** network. The x axis denotes the number of networks parallelly trained in grafting algorithm.

invalid filters' locations, necessity of keeping layer consistency and efficiency of adaptive weighting strategy.

5. Conclusion and Discussion

In this work, a new learning paradigm called '**filter grafting**' is proposed. We argue that there are two key points for effectively applying filter grafting algorithm: 1) How to choose proper criterion to calculate the inherent information of filters in DNNs. 2) How to balance the coefficients of information among networks. To deal with these two problems, we propose **entropy-based criterion** and **adaptive weighting strategy** to increase the network's performance. But this is not the only solution. Other criterions or methods could be developed to improve the grafting algorithm further. Heuristically, there are some future directions to be considered: 1) How to improve the network's performance with larger number of networks in grafting algorithm; 2) How to apply grafting on multiple networks with different network structures.

Acknowledgements This work is supported in part by the NSFC (No. 61906162), in part by the Shenzhen Fundamental Research Fund under Grants JCYJ20180306172023949, in part by the Medical Biometrics Perception and Analysis Engineering Laboratory, Shenzhen, China.

References

[1] Hao Cheng, Dongze Lian, Shenghua Gao, and Yanlin Geng. Utilizing information bottleneck to evaluate the capability of deep neural networks for image classification. *Entropy*, 21(5):456, 2019.

[2] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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

[8] Siddharth Krishna Kumar. On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863*, 2017.

[9] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[10] Shaohui Lin, Rongrong Ji, Chao Chen, Dacheng Tao, and Jiebo Luo. Holistic cnn compression via low-rank decomposition with knowledge transfer. *IEEE transactions on pattern analysis and machine intelligence*, 41(12):2889–2905, 2018.

[11] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.

[12] Y. Lu, G. Lu, R. Lin, J. Li, and D. Zhang. Sparse repeated group convolutional neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[13] Yao Lu, Guangming Lu, Bob Zhang, Yuanrong Xu, and Jinxing Li. Super sparse convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4440–4447, 2019.

[14] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019.

[15] Aaditya Prakash, James Storer, Dinei Florencio, and Cha Zhang. Repr: Improved training of convolutional filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10666–10675, 2019.

[16] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*, 2016.

[17] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[18] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[19] Xavier Suau, Luca Zappella, Vinay Palakkode, and Nicholas Apostoloff. Principal filter analysis for guided network compression. *arXiv preprint arXiv:1807.10585*, 2018.

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

[21] Dong Wang, Lei Zhou, Xueni Zhang, Xiao Bai, and Jun Zhou. Exploring linear relationship in feature map subspace for convnets compression. *arXiv preprint arXiv:1803.05729*, 2018.

[22] He Yang, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yang Yi. Soft filter pruning for accelerating deep convolutional neural networks. In *Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18*, 2018.

[23] Jianbo Ye, Lu Xin, Lin Zhe, and James Z. Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. 2018.

[24] Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

[25] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4320–4328, 2018.

[26] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. Scalable person re-identification: A benchmark. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1116–1124, Dec 2015.

[27] Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. Multinomial distribution learning for effective neural architecture search. In *ICCV*, 2019.

[28] Zhedong Zheng, Liang Zheng, and Yi Yang. Unlabeled samples generated by gan improve the person re-identification baseline in vitro. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 3754–3762, 2017.

[29] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Omni-scale feature learning for person re-identification. *arXiv preprint arXiv:1905.00953*, 2019.