Manuscript

Computational Visual Media
DOI 10.1007/s41095-xxx-xxxx-x Vol. x, No. x, month year, xx–xx

Article Type (Research)

Weight Asynchronous Update: Improves the Diversity of Filters in Deep Convolutional Network

Dejun Zhang[1], **Linchao He**[2], **Mengting Luo**[2], **Zhanya Xu**[1] (✉), **and Fazhi He**[3]

Abstract Deep convolutional networks have obtained remarkable achievements on various visual tasks due to their strong capabilities in learning abundant features. A well-trained deep convolutional networks can be compressed to 20%∼40% of the original size by trimming many under-expressive filters. This can be partly traced back to the fact that many overlapping features are generated by potentially redundant filters. Model compression is used to reduce the unnecessary filters but does not take advantage of redundant filters since training phase is not affected. Modern networks with residual/dense connections and inception blocks are considered to be able to mitigate the overlap in convolutional filters, but not necessarily overcoming the issue. To address these issues, we propose a new training strategy, called Weight Asynchronous Update (WAU), which significantly helps to increase the diversity of filters and enhance the representation ability of network. The proposed method can be widely applied to different convolutional networks without changing the network topology. Specifically, our experiments show that the stochastic subset of filters is updated in different iterations can significantly reduce the filter overlap of convolutional networks. Extensive experiments show that WAU yields noteworthy improvements in test error, e.g. 2.96% on CIFAR-100 and 1.2% AP@0.5 on COCO.

Keywords deep convolutional network, model compression, convolutional filter, image classification.

1 School of Geography and Information Engineering, China University of Geosciences, Wuhan 430074, China. E-mail: D. Zhang, zhangdejun@cug.edu.cn; Z. Xu, zhanyaxu@163.com

2 College of Information and Engineering, Sichuan Agricultural University, Yaan 625014, China. E-mail: L. Chao, fpsandnoob@hotmail.com; M. Luo, sookie0331@icloud.com.

3 School of Computer, Wuhan University, Wuhan 430072, China. E-mail: fzhe@whu.edu.cn.

Manuscript received: 2014-12-31; accepted: 2015-01-30.

1 Introduction

In the past few years, deep learning methods based on Convolutional Neural Networks (CNNs) have obtained significant achievements in machine vision [12, 33], shape representation [13, 14, 32], automatic speech recognition [3, 22], natural language processing [20, 34, 35], etc. In particular, many advanced deep convolutional networks have been proposed to handle visual tasks. For example, the success of Deep Residual Nets has inspired researchers to explore deeper, wider and more complex frameworks [8, 29].

Deep convolutional networks possess strong learning capability owing to their rich sets of parameters. However, this characteristic brings about the evident nuisance of over-parameterization, which further leads to overlapped/redundant features. It also causes overfitting to the training set and the lack of generalization to new data. Several modern networks, which have hundreds layers (*e.g.* ResNet [5], DenseNet [8], Inception [26]), employ their architectural advantages to alleviate the above problems. One main factor is that residual connections through early layers and feature fusion can be considered as noise addition in the feature space, with which the network is regularized and hence the overlap of learned deep features is reduced.

A trained network may be further compressed by pruning, quantization or binarization, which typically exploits the redundancy in the weights of the trained network. In general, the purpose of model compression is to minimize the memory cost, and to accelerate the speed of inference without losing performance, rather than optimizing the capacity of networks in training. Exploring the best performance of the modern networks is still a challenge.

To this end, this work aims to expand the capacity of the network by reducing the overlap of the filters. Our method includes the following two major techniques, which are also the key contributions of this work:

- Weight Asynchronous Update (WAU). We perform the backward propagation asynchronously to update a subset of convolutional filters to reduce the overlap of the filters. Since the proposed method does not change the original network architecture, it can be easily applied to neural network models to boost the performance of various visual tasks.
- Asynchronous-Synchronous-Asynchronous (ASA) training flow. Reducing the model capacity every mini-batch would lead to missing relevant relationship among deep features and target outputs [4]. To address this issue, we first apply WAU to "warm-up" the network and facilitate initial orthogonality. Then, *sync* training is applied, which is beneficial to global learning, as *sync* training strengthens the connection among filters and enhances the relationship between feature maps and output. Finally, *async* training is used again to break the convergent evolution of the previous training phase and reduce the overlap of filters.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related work on model compression, weight inactivate and its extension in regularization. Section 3 explains the motivation for designing WAU. Section 4 and Section 5 respectively elaborates on the proposed method and its enhanced method (*ASA training flow*) in detail. In section 6, we discuss the evaluation results and compare them with those of representative approaches. Finally, the conclusions and future work are discussed in section 7.

2 Related work

Recently, a large number of works have been published concerning network optimization. Over the past few years, with efforts of researchers in this area, significant progress has been made on some longstanding problems. These approaches can be grouped into three types according to network optimization: (1) model compression; (2) weight inactivate; (3) regularization. We review each of these approaches in turn.

2.1 Model compression

In order to reduce computational and memory costs, pruning the well-trained model is the most widely used method in current model compression [7]. This method finds an effective criterion to judge the importance of parameters and prunes the redundant connections or filters. The pruned smaller model is able to re-train the knowledge from the original larger model without significant loss of performance. However, network pruning aims to reduce the redundancy of model, but not take advantage of increasingly deeper and wider networks. The reason is that network pruning reduces over-parameterization in the inference phase, but does not provide a solution for the training phase. We propose a training scheme that also focuses on mitigating over-parameterization and especially increasing the capacity of deep convolutional networks.

2.2 Weight inactivation

For model pruning, inactivating the least effective filters is beneficial for constructing efficient CNNs without sacrificing the performance. Inspired by this characteristic, several training strategies have been explored to re-train the redundant filters with the conduction of the ranking criterion. DSD [4] applies a hard threshold mask on kernel weight according to Taylor expansion of the cost function [18]. DSD intends to divide filters into two fixed groups via hard mask $|w_i| < \lambda$ and prunes the least salient group in the second training phase, but it cannot break the symmetry within the groups. Currently, RePr [21] prunes top-N least orthogonal filters according to the filter orthogonality ranking in the whole network in each epoch have overtaken DSD. However, there is a problem of RePr that lower dimensional filers tend to be pruned in practice. It leads to RePr performing well in shallow networks while the performance decreases in deep networks. Therefore, we argue that the degree of overlap is hardly conducted by a certain criterion. Our training scheme does not determine by an external criterion and generates the kernel masks to re-train. A simple and generic strategy requires less computation cost.

2.3 Regularization

In order to reduce the generalization error, some regularization methods are proposed in recent years. Dropout [25], an effective approach to overcome over-fitting, which ensembles several weak classifiers to gain a more robust strong classifier by dropping neuron randomly. Additionally, Batch Normalization (BN) is widely used in modern standard CNNs to reduce the *Internal Covariate Shift* problem. However, the theoretical and empirical evidence demonstrates that combining Dropout and BN usually causes unsatisfactory performance [15]. Recently, Shake-Shake regularization [28] has broken the records of CIFAR10, which disturbs the forward and backward propagation with the stochastic affine combination. But, Shake-Shake [30] slows down the convergence due to the strong interference, where it requires 1800 epochs to make ResNet-110 converge on CIFAR-10, and can only be used in specific 3-branch convolutional networks (e.g. ResNeXt [29]).

To overcome the above deficiencies, we design an effective WAU, which can also be regarded as a regularizer with disturbing learning to add noise into the network. Firstly, WAU is friendly to be used with BN and is available to any convolutional networks. Secondly, WAU does not cost more time to converge and even requires less number of parameters updating iterations. Experimental results demonstrate that the proposed WAU, with a simple but powerful weight update strategy, is superior to neuron ranking criterion and has good performance in the learning of deep networks.

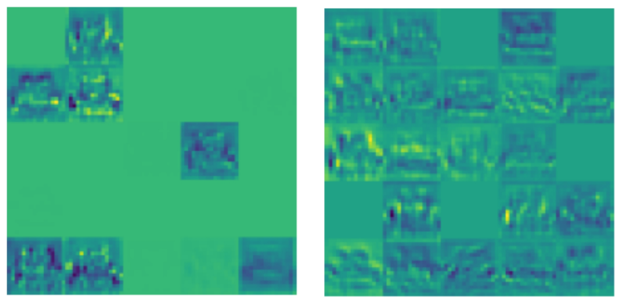
3 Motivation

In each standard backward propagation (BP) step, the parameters W of filters F are updated with learning rate η. For the example of a single layer perceptron, the updating equation of each parameter w_i of the hidden layer h is expressed as follows:

$$w_i = (1+\eta)\frac{\partial h}{\partial w_i}\frac{\partial \alpha}{\partial h}\frac{\partial l_m}{\partial \alpha}, \tag{1}$$

where α represents an activation unit. The partial derivative of w_i is determined by a mini-batch loss denoted as l_m, which originates from the same information entropy $-\sum_{j=1}^{n} y_j \log \hat{y_j}$. y_j and $\hat{y_j}$ represent the ground truth and prediction of sample x_j in n classes, respectively. The behavior of updating all parameters W at the same iteration is referred to as *synchronous learning*. However, updating all weights through the identical information entropy over thousands of iterations can result in poorly differentiated features within the same layer. In fact, this phenomenon widely exists in the modern deep neural networks, which we call *convergent evolution* in this paper.

The convergent evolution usually exists in filters from the same hidden layer. Because the most widely accepted understanding of adaptive filters in CNN is that the filters of bottom layers learn low-level visual features, while the filters of top layers learn high-level semantic information. It seems to show less correlation and discriminative semantic features in different layers. However, the previous work [16] shows that the ensemble of the residual blocks is also proved by lesion study of deleting individual blocks in [27], and the convergent evolution also appears in different convolution networks, so-called *convergent learning*. It shows that the *convergent learning* not only exits in the same layer, but also between different layers. This echoes the evolution theory in biology: the independent evolution of similar features in species of different lineages, due to the same type of environment and similar lifestyle [19]. In short, the low-discriminative filters result in the inefficiency of deep convolutional network.



(a) Synchronous (b) Asynchronous

Fig. 1 The comparison of features between *Sync* and *Async*. (a) Features learned by *sync* updating. (b) Features learned by *async* updating.

Based on this motivation, we introduce the Weight Asynchronous Update (WAU) to prevent the convergent evolution and network symmetry. WAU allows different filters to be updated in different iterations. This constraint increases the diversity of filters within the same layer and between the different layers. To demonstrate the effectiveness of WAU, we compare the generated features by kernels in *sync* updating and *async* updating. As shown in Fig. 1(a), the highly related filters lead features to gain similar and weakly differentiated representations. Apparently, the various and diverse features are generated by WAU training strategy. As shown in Fig. 1(b), it mitigates the overlap and improves the representation ability of convolutional networks. The L2 regularization is applied to constrain the filters that are close to zero. Thus, more plain green regions do not contain useful information in Fig. 1(a) than Fig. 1(b).

4 Weight asynchronous update

WAU aims to reduce the potential overlap by updating the dynamic subset $\hat{F}$ of convolutional filters F in each mini-batch. Each convolutional filter ($3D$ tensor) is considered as a single neural unit. To ensure the sparsity of a single layer, we sample filters on layer-level by fixing the *async rate* $r \in [0, 1]$. In other words, all the filters would be updated one time on average within $1/r$ cycles. The expectation of the number of $\hat{F}$ of layer l in iteration t is defined as follows:

$$E[\hat{F}_{l,t}] = |F_{l,t}|r. \tag{2}$$

We use the SGD algorithm as an example to explain how the filters update asynchronously in a mini-batch. The standard SGD optimizer is shown in the equations as below:

$$g_t = \nabla_{\theta_{t-1}} f(\theta_{t-1}), \tag{3}$$

$$\Delta\theta_t = -\eta * g_t, \tag{4}$$

$$\theta_t = \theta_{t-1} + \Delta\theta_t. \tag{5}$$

The SGD optimizer calculates the gradient g_t of the objective function with respect to the current parameter $f(\theta_{t-1})$. In Eq. 4, η denotes the learning rate and $\Delta\theta_t$ is the descent gradient at iteration t. θ_t can be obtained by updating θ_{t-1} with $\Delta\theta_t$.

For each mini-batch, we sample active filters $\hat{F_{l,t}}$ from $F_{l,t}$ to make every filter having the same probability r of weight update via stochastic sampling function $\mathcal{S}$. Updating the weight of the network asynchronously is implemented by applying a mask function ψ which depends on $\hat{F_{l,t}}$ as shown below:

$$\theta_t = \theta_{t-1} + \psi(\Delta\theta_t) \tag{6}$$

$$\psi(\Delta\theta_t) = \begin{cases} \Delta\theta_t & \text{if } \theta_t \in \hat{F_{l,t}} \\ 0 & \text{if } \theta_t \in F_{l,t} - \hat{F}_{l,t} \end{cases} \tag{7}$$

$$\hat{F}_{l,t} = \mathcal{S}(F_{l,t}, r) \tag{8}$$

Figure 2 illustrates the weight asynchronous update training strategy in t-th iteration. There is no impact on forward propagation (all convolutional filters F are active as normal networks). During the back-propagation, each layer has a dynamic subset $\hat{F}$ which does not update the parameters in t-th iteration. These convolutional filters are represented as transparent kernels by multiplying kernel mask ψ in shape $1 \times 1 \times c_l$.

The goal of WAU is to change the way of updating the convolution filters. Our sampling function $\mathcal{S}$ does not explicitly influence the work of the optimizer and BN, but only decides whether the weight is updated or not. Specially, for the adaptive optimizer [1, 10] which needs to save the previous variables, it also works in the normal way when using our training scheme.

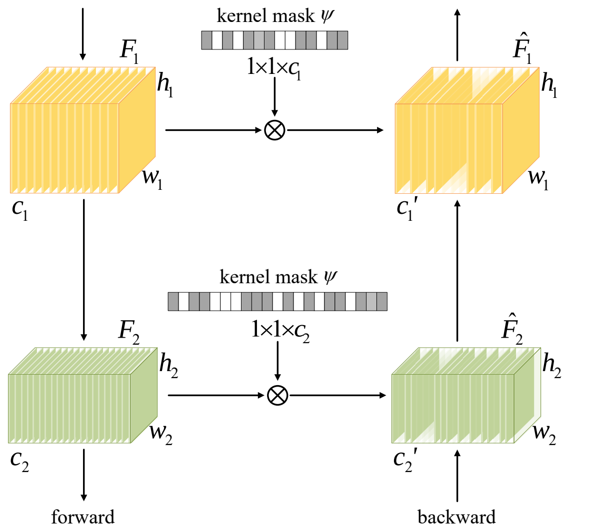


Fig. 2 Weight asynchronous update training strategy. c, w and h are represented as channel, width and height, respectively.

5 ASA training flow

The extended version of WAU is introduced in this section. Inspired by [4], we propose an ASA training flow that includes three processes: async weight update, sync weight update and re-async weight update.

- *Async.* The CNNs [5, 8, 12] use various weight initialization to avoid learning redundant features, but they are unable to handle the redundant features in the training phase. The first Async step not only learns the values of the weights, but also aims to expand the gap between the filters and warm up the network, which is equivalent to initializing the network by learning the real-world data.
- *Sync.* The hierarchical relationships of the features generated by kernels in different layers, which are formed in standard back-propagation. The weights of the network are updated synchronously to enhance the relationship of deep features and the outputs [4].
- *Re-Async.* The filters are updated asynchronously to ensure that the weights of the redundant filters can be differentiated in different directions. The hyper-parameters, such as *async* rate and weight decay r, are consistent with the first Async step. The re-Async step increases the diversity of the kernel, as well as the capacity of the network. Compared with sparse networks, it is possible to converge to better local minima from Sync step.

The pseudo-code of our proposed approach is shown in Algorithm 1.

Algorithm 1 ASA training flow.

```
for all N epochs do
    F̂ = S(F, r);
    for all C mini-batches do
        Network back propagation with F̂;
        F̂ = S(F, r);
    end for
end for
for all N epochs do
    Reinitialize the optimizer state and the learning rate schedule;
    for all C mini-batches do
        Network back propagation with F;
    end for
end for
for all N epochs do
    Reinitialize the optimizer state and the learning rate schedule;
    F̂ = S(F, r);
    for all C mini-batches do
        Network back propagation with F̂;
        F̂ = S(F, r);
    end for
end for
```

6 Results

First, we compare the standard CNNs [5, 6, 8, 12, 24, 29, 31] with convolutional networks that added WAU strategy; in addition, we extend to some visual tasks. Secondly, we verify the effectiveness of ASA training flow. As our theoretical analysis shows, we have experimentally proved that the WAU method effectively reduces the overlap filters. Finally yet importantly, we demonstrate that the WAU method has a faster convergence speed and should be more friendly combined with BN for better performance. In order to prove the effectiveness of our approach, we follow the original training protocol to train the neural networks without fine-tuning, such as the same strategy of decreasing learning rate and hyper-parameters. The corresponding code and model are available at our community (https://github.com/djzgroup/wau).

6.1 Comparison of different convolutional networks

It is worth mentioning here that if there is no special mention, the hyper-parameters are set to be the same in all experiments (for example, the weight decay rate is set to 0.0001 and the default asynchronous rate is r=0.5). Compared to synchronous weight update, our WAU method achieves significant accuracy enhancement on different convolutional networks.

Table 1 shows the testing accuracy of WAU on CIFAR-10 and CIFAR-100. Specifically, on CIFAR-10, AlexNet [12] gets large improvement by using our *async* training method, where the accuracy is enhanced by 1.66% compared to the baseline. For other famous convolutional networks (e.g. ResNet [5], VGG [12], PreResNet [6], ResNext [29], Wide ResNet [31], DenseNet [8]), our WAU is also effective to improve their capacity and get better performance than the baselines (at least 0.43% accuracy improvement). For DenseNet-40 and DenseNet-100, our WAU strategy promote them achieve better results (1.63% and 1.10% improvement respectively) compared to sync training method. In our experiment, ResNext[29] uses two special convolutional structures, pointwise convolution and group convolution. It can be found from Table 1 that ResNext has significantly improved after using WAU.

Furthermore, for the more complex and challenging dataset CIFAR-100, which is a 100-class classification problem and that requires much more diversified filters, the performance of networks trained with WAU is still better even obtains larger improvement than CIFAR-10 compared with the baseline.

For AlexNet [12], the accuracy enhancement is 1.06% (2.72% vs 1.66%) more on CIFAR-100 than on CIFAR-10. For VGG-BN [24], the performance is boosted by 2.96% and 2.53% (2.96% vs 0.43%) compared to the baseline and CIFAR-10 respectively. Particularly, DenseNet-40 gains the biggest boost, which improves the relative accuracy is 4.13%.

Table 1 exhibits that our method WAU can improve the performance of most convolutional network frameworks. The WAU training flow is a generic strategy since it does not depend on the specific network framework. Simply, in contrast to the weight synchronous update, our WAU method can produce more diverse filters (Fig. 1(b)) to learn more discriminative data representation; therefore, better performance can be obtained.

In addition, the effectiveness of the proposed WAU strategy can also improve performance of other models in various tasks, e.g. object detection which is shown in Table 2 and Table 3.

As shown in Table 2, we experimented with Faster R-CNN using VGG-16 pre-trained on ImageNet. The model was trained on the COCO trainval35k dataset and evaluated on the minival set. AP represents the average results of different classes with 10 Intersection

Tab. 1 Overview of the performance improvement (Imp.) from our WAU. The proposed WAU shows significant gain over the baseline on both CIFAR-10 and CIFAR-100 [see [11], chap. 3].

Networks	Depth	Test Accuracy (%)			
		CIFAR-10		CIFAR-100	
		Baseline	WAU	Baseline	WAU
AlexNet [12]	-	77.26	78.92 (1.66↑)	45.17	47.89 (2.72↑)
ResNet-50 [5]	50	92.68	93.38 (0.70↑)	70.58	71.41 (0.83↑)
VGG-BN [24]	19	93.01	93.44 (0.43↑)	70.76	73.73 (2.96↑)
PreResNet [6]	110	93.58	94.12 (0.54↑)	72.53	73.23 (0.70↑)
ResNext [29]	29	95.56	96.11 (0.55↑)	80.54	82.75 (2.21↑)
Wide ResNet [31]	28	95.54	96.16 (0.62↑)	80.91	81.10 (0.19↑)
DenseNet [8]	40	89.91	91.54 (1.63↑)	63.28	67.41 (4.13↑)
DenseNet [8]	100	91.10	92.20 (1.10↑)	68.08	70.22 (2.14↑)

Tab. 2 Object detection results of Faster R-CNN [23] on the COCO minival set [17]. All models are trained on the trainval35k set with images of image scale 600 pixels.

Network	AP@.5	AP
VGG-16 w/o WAU	46.9	26.9
VGG-16 with WAU	48.1 (1.2↑)	27.4 (0.5↑)

Tab. 3 Object detection results using Faster R-CNN [23] tested on the Pascal VOC 2007 test set. Models are trained on the Pascal VOC 2007 trainval set.

Network	Baseline	Ours(WAU)
Faster R-CNN w/o WD	70.10	70.74 (0.64↑)
Faster R-CNN with WD	69.80	70.80 (1.00↑)

over Union (IoU) thresholds of 0.50:0.05:0.95, and AP@.5 is result with the IoU thresholds of 0.5. Remarkably, our method gain 1.2% AP@.5 and 0.5% AP improvement compared to the VGG-16 baseline. This model has many fewer parameters (by a factor of 11×) 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.

The object detection networks are trained on the train split and tested on the test split of the Pascal VOC dataset. We use AP@.5 for characterizing the performance that is the standard Pascal VOC metric. Table 3 shows that our WAU strategy also boosts the performance of the baseline on Pascal VOC dataset in object detection task. However, we notice that training Faster R-CNN in WAU training strategy presents different performances without and with Weight Decay (WD), which respectively improve by 0.64% mAP and 1% mAP. The weight decay will be briefly analyzed in Section 6.6.

To sum up, the results of experiments demonstrate the simple WAU method is suitable for modern frameworks and various tasks. The improvement of the performance is partly traceable to the contribution of diverse filters which generated by WAU, which is discussed in next subsection.

6.2 The diversity of kernels with WAU training

Prakash et al. [21] and Li et al. [16] employed correlation analysis to estimate the similarity between filters. In this paper, the diversity of kernels are also evaluated by correlation matrix of filters according to the Pearson Correlation Coefficient of Canonical Correlation Analysis (CCA):

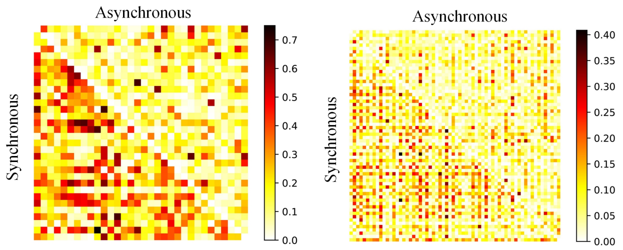
$$\begin{aligned} &P_{i,j} = E[(F_i - \mu_i)(F_j - \mu_j)]/\sigma_i\sigma_j, \\ &\text{where, } \mu_i = E(F_i),\ \sigma_i = \sqrt{E[(F_i-\mu_i)^2]} \end{aligned} \tag{9}$$

where μ and σ respectively denote as mean value and standard deviation, and E represents the mean of filters.

To demonstrate the diversity of the kernels increased by WAU, we visualize the correlation matrix of filters, which are random sampled within a layer or a residual block of ResNet-110.

Figure 3(a) illustrates the CCA of the filter activations at the same layer in the trained ResNet-110. The darker color patches represents higher correlation. It shows that almost half filters reveal a large correlation to others in the lower triangle, and many filters are in high correlation. There is no surprise that *convergent evolution* of kernels usually occur within the same layer, since the phenomenon is demonstrated in section 3.

In Fig. 3(b) illustrates the CCA visualization of filter activation between two basic blocks inside a Residual Block of ResNet-110, which reveal the *convergent*



(a) Within Layer (b) Inter Layer

Fig. 3 Comparison between *Sync* and *Async*. For (a) and (b), the upper triangle and lower triangle respectively represent the results of *Sync* and *Async* weight updating training methods. (a) The correlation of 32 filters within a single layer. (b) The correlation of 64 filters between two layers inside a Residual Block.

evolution also exists at the layer level.

From the perspective the lower triangle of Fig. 3(a) and Fig. 3(b), there is no strong evidence show that widening and deepening the model can result in increasing the diversity of kernels. That is because of the *convergent evolution* in the model.

Therefore, preventing the *convergent evolution* and increasing the diversity of kernels are beneficial for expanding the capacity of original model. The upper triangle of two correlation matrix in Fig. 3 demonstrates that the *async* flow training strategy significantly reduces the filters correlation within layers and between layers. Apparently, the trained kernels learn representations from different directions by using WAU, and the mitigation of over-parameterization increases performance.

6.3 Analysis of the ASA training flow

More studies were carried out on the ASA training flow. We conduct experiments with ResNet-32 trained on the CIFAR-10 training set and evaluate on the testing set. We set the Async rate used by the ASA strategy is "0.5, 1.0, 0.5". A Sync-Async-Sync (SAS) training flow is introduced as our baseline. We set the training epochs to $N = 164$, thus the total training epochs are 492. We re-initialize the optimizer state and the learning rate schedule when we change the weight update method.

As shown in Table 4, both training flow strategies improve the test accuracy due to asynchronous weight update. The ASA training flow gets better result at the 1st training phase compared to SAS training flow. In the 2nd phase, ASA training flow method obtains more significant improvement than SAS training flow approach (0.54% vs 0.16%). Both methods show similar performance enhancement at the 3rd phase (0.11% v.s. 0.12%). After the final phase, the test accuracy gain of our ASA training reaches to 0.65% compared to the 1st phase, and it surpasses SAS training flow strategy by 0.64% (93.40% v.s. 92.76%).

Table 5 shows that compared to other related training strategies [2, 4, 21], WAU is more effective to boost the performance of deep networks and easier to implement. Strict ranking criterion requires long training phase for meaningful neuron ranking, and in most training time the filters are still in a state of *sync* updating. In contrast, our stochastic weight inactivation enables the updatable filters $\hat{F}$ to continue to change in different iterations.

6.4 Discussion of convergence speed

There is no doubt that meaningful filters are able to modify the model performance. As shown in Fig. 1(a) and Fig. 1(b), we collect learned features from a group of filters in ResNet-110 [5] via different training methods. Most channels learn meaningless deep features by using standard BP. In contrast, after taking advantage of *async* learning, the filters learn more distinguished information and exploit extra deep network potentiality.

In order to reduce the influence of hyperparameters, we followed the training strategy of most convolutional neural networks. Figure 4 shows the behaviors of different convolutional networks. In all experiments, we used a strategy of decreasing learning rate. So it would be a huge increase around $6k$, $8k$, $15k$ iteration because of the learning rate decay.

Our method converges much faster than the *sync* method before the first learning rate is reduced. A large number of error information transferred to the filters, which significantly accelerates the convergence speed than the *sync* training strategy. This is the key reason why our model achieves high accuracy in the early stage, with four times less iterations.

6.5 What's the difference with dropout

Our WAU has some similarities to the well-known Dropout. The asynchronously updating scheme can be intuitively regarded as Dropout only on back propagation. However, there are two major differences between WAU and Dropout:

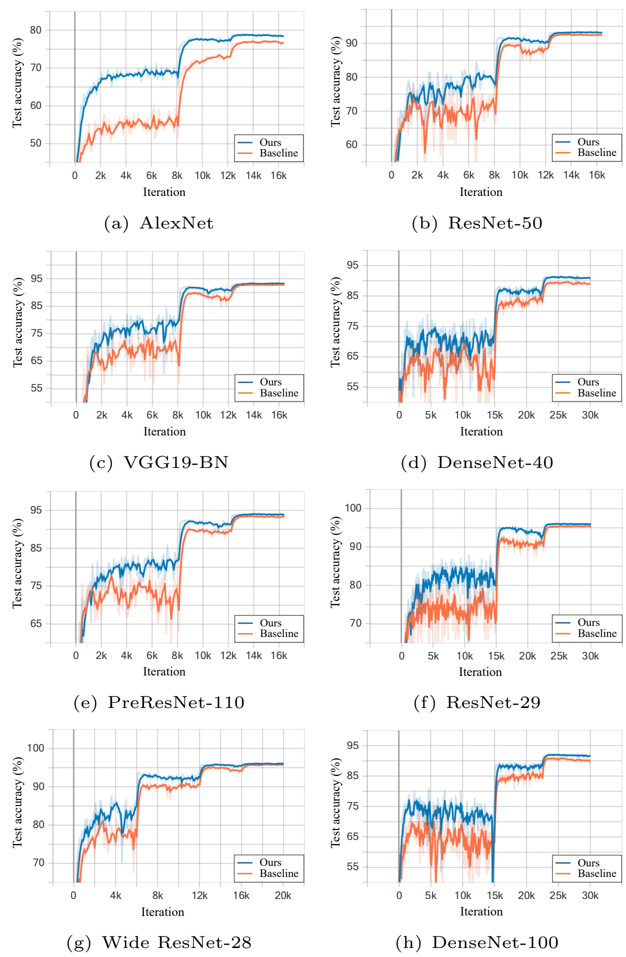
- *The approach of weight inactivation.* The goal of Dropout is main to prevent overfitting. It employs Bernoulli random variable r to multiply every single element-wise with the outputs h of hidden layer. Each r takes the value 1 with the hyper-parameter probability p and has probability

Tab. 4 Overview of different training flow. Both training flow strategies have obtained improvement, and the ASA training flow performs better.

	1st phase	2nd phase	3rd phase
ASA	92.75	93.29 (0.54↑)	93.40 (0.65↑)
SAS	92.49	92.65 (0.16↑)	92.76 (0.27↑)

Tab. 5 Comparison of test error on CIFAR-10.

Baseline	Various Training Schemes			WAU	
Original [5]	DSD [4]	BAN [2]	RePr [21]	Asynchronous	ASA
8.7	7.8	8.2	7.7	7.49 (1.21↓)	7.06 (1.64↓)



(a) AlexNet (b) ResNet-50 (c) VGG19-BN (d) DenseNet-40 (e) PreResNet-110 (f) ResNet-29 (g) Wide ResNet-28 (h) DenseNet-100

Fig. 4 Comparing the testing accuracy and convergence speed of our WAU method with the baseline on various convolutional networks on CIFAR-10.

$1-p$ of being 0, which is a vector independent of each other. By contrary, WAU method is to balance sparsity of inactivation, which prevents the nodes are inactivated in extreme cases. The weight inactive mask is randomly formed by a hyper-parameter *async* rate that fixes the sparsity of each layer and each iteration.

- *Cooperate with batch normalization.* Dropout and BN play significant roles in deep network regularization. However, these two powerful techniques do not produce double power in CNNs when used together (and may actually cause higher generalization error). Previous work [15] revealed that this is due to the disharmony between normalization of BN and Dropout. BN accumulates the statistics variance during training phase and maintain it in inference. Dropout transfer the variance from training to inference.

We further found that there is still a conflict between Dropout and affine transform $y_l = \gamma_l \hat{x}_l + \beta_l$ of BN, where $\hat{x}_l$ is denoted as normalized input x_l. Affine transform avoids mapping the inputs to saturated region by the activation function after normalization [9]. However, Dropout break the harmony on scale and shift part.

We test block B_1, which consists of a sequence of layers i.e. **Conv-Dropout-BN-ReLU**, with 0.5 dropping probability. As the green curve shows in Fig. 5, combining BN and Dropout slows down the convergence speed and drops the network performance to 55%. When using B_2 i.e. **Conv-Dropout-Normalize-ReLU**, which removes the affine transform of BN, it mitigates the conflict with Dropout and boosts the accuracy by 20%.

The red carve in Fig. 5 shows that the cooperative effort between WAU and BN achieves a competitive 93.6% accuracy. Replacing B_1 by B_2 has a subtle effect on the network representation performance. WAU maintains the balance of internal covariate

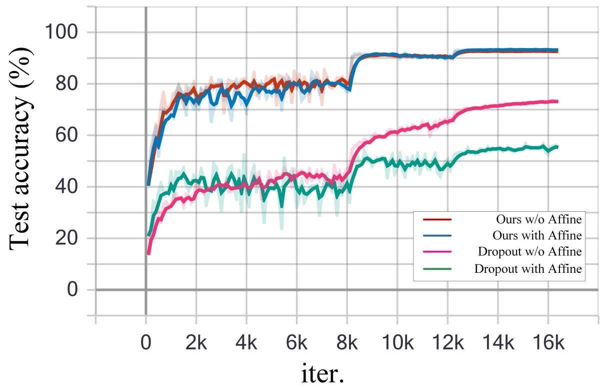


Fig. 5 Ablation of Affine Transform in BN. The performance curve of four different models on the task of image recognition. They are respectively represent WAU with/without affine transform of BN, and Dropout with/without affine transform of BN.

shift (ICS) [9] when going from training to inference. Because WAU is quite different from Dropout, it does not prune or inactivate the neuron in forward propagation and thus has no impact on the network architecture.

6.6 Hyper-parameters

6.6.1 Asynchronous rate

In this section, we focus on the effect of the hyper-parameter r on the deep neural models. We take $r \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, and WAU will collapse to the baseline when $r = 1.0$. The performance curves of AlexNet [12] on CIFAR-10 are plotted in Fig. 6 and the corresponding test results are reported in Table 6. With the extreme small *async* rate $r \in \{0.1, 0.2, 0.3\}$ in which the updating rate is very low, the performance of our method is worse than the baseline (light green). With other higher *async* rate ($r \in [0.4, 0.9]$), the testing accuracy of us is much better than the baseline, because of our models have faster convergence speed in early epoch. For all the extensive experiments that are conducted in this paper with different CNNs, our models get significant accuracy gain by taking default *async* rate $r = 0.5$ without careful tuning.

6.6.2 Weight decay

The proposed WAU strategy independently optimizes the dynamic filter subset $\hat{F}$, which allows to explore larger weight space. Therefore, it has more probability to escape saddle points and reach local minimum points. To reduce the exploration of optimizer as the learning rate decays for building more stable model, we increase the punishment of weight and set the weight decay rate to 0.001.

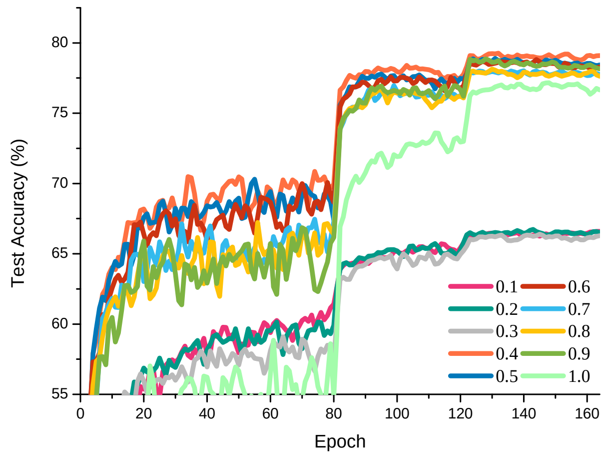


Fig. 6 The Influence of Hyperparameter r. Appropriate *async* rate all result in improved performance, and the parameters are not strictly sensitive.

Tab. 6 Results of classification by using WAU with different *async* rate.

Rate	Accuracy
0.1	66.59 (-10.67↓)
0.2	66.74 (-10.52↓)
0.3	66.40 (-10.86↓)
0.4	79.29 (+2.03↑)
0.5	78.92 (+1.66↑)
0.6	78.12 (+0.86↑)
0.7	78.82 (+1.56↑)
0.8	78.15 (+0.89↑)
0.9	78.95 (+1.69↑)
1.0 (baseline)	77.26

7 Conclusions

We proposed a novel training strategy WAU, which is able to reduce the overlap of convolutional filters and produce filters that are more diverse by updating weight asynchronously. In addition, we present a new training flow method termed *Async-Sync-Async* training flow to enhance the relationship among filters by inserting a *sync* updating in training phase, which further reduces the generation error. Experiments on various convolutional networks and different visual tasks demonstrate that the explored WAU method provides an effective solution to obtain faster convergence speed and improve the performance of convolutional models. In particular, visualizations show that our WAU method changes the behavior of convolutional filters and obtains better data representation. Remarkably, compared to the baseline, the network that added WAU can achieve 2.96% accuracy improvement in CIFAR-100 and 1.2% AP@.5 in COCO object detection task.

The Weight Asynchronous Update improves the performance of various deep convolutional networks shown in the results of experiments. In the future, we intend to extend this work to generic network frameworks like Multi-layer Perceptron, Recurrent Neural Network, and make it available to Natural Language Processing and Speak Processing tasks. Additionally, another important future direction is to design an effective and general criterion to accurately describe the similarity between filters of different dimensions. The criterion is used to evaluate the redundancy of the kernels. We can re-train the redundancy parameters in the network according to the criterion to improve accuracy.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 61702350.

References

[1] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[2] T. Furlanello, Z. Lipton, M. Tschannen, L. Itti, and A. Anandkumar. Born-again neural networks. In *International Conference on Machine Learning*, pages 1602–1611, 2018.

[3] H. Geoffrey, D. Li, Y. Dong, E. D. George, and A.-r. Mohamed. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[4] S. Han, J. Pool, S. Narang, H. Mao, E. Gong, S. Tang, E. Elsen, P. Vajda, M. Paluri, J. Tran, et al. Dsd: Dense-sparse-dense training for deep neural networks. *arXiv preprint arXiv:1607.04381*, 2016.

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

[6] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[7] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[8] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

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

[11] A. Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[13] J. Li, B. M. Chen, and G. Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018.

[14] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, and L. Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.

[15] X. Li, S. Chen, X. Hu, and J. Yang. Understanding the disharmony between dropout and batch normalization by variance shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2682–2690, 2019.

[16] Y. Li, J. Yosinski, J. Clune, H. Lipson, and J. E. Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *FE@ NIPS*, pages 196–212, 2015.

[17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[18] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

[19] J. D. Norton. Science and certainty. *Synthese*, pages 3–22, 1994.

[20] Y. Pan, F. He, and H. Yu. A novel enhanced collaborative autoencoder with knowledge distillation for top-n recommender systems. *Neurocomputing*, 332:137–148, 2019.

[21] A. Prakash, J. Storer, D. Florencio, and C. Zhang. Repr: Improved training of convolutional filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10666–10675, 2019.

[22] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[23] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2016.

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

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

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[27] A. Veit, M. J. Wilber, and S. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems*, pages 550–558, 2016.

[28] X. Xavier. Shake-shake regularization of 3-branch residual networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*, 2017.

[29] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[30] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise. Shakedrop regularization for deep residual learning. *IEEE Access*, 7:186126–186136, 2019.

[31] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[32] D. Zhang, F. He, Z. Tu, L. Zou, and Y. Chen. Pointwise geometric and semantic learning network on 3d point clouds. *Integrated Computer-Aided Engineering*, 27(1):57–75, 2020.

[33] D. Zhang, L. He, Z. Tu, F. Han, S. Zhang, and B. Yang. Learning motion representation for real-time spatio-temporal action localization. *Pattern Recognition*, page 107312, 2020.

[34] D. Zhang, M. Hong, L. Zou, F. Han, F. He, Z. Tu, and Y. Ren. Attention pooling-based bidirectional gated recurrent units model for sentimental classification. *International Journal of Computational Intelligence Systems*, 12(2):723–732, 2019.

[35] D. Zhang, M. Luo, and F. He. Reconstructed similarity for faster gans-based word translation to mitigate hubness. *Neurocomputing*, 362:83–93, 2019.

Dejun Zhang received the Ph.D. degree from the department of computer school, Wuhan University, China, in 2015. He is currently a lecturer with the faculty of Information Engineering, China University of Geosciences, China. Since 2015, he has been serving as a senior member of the China Society for Industrial and Applied Mathematics (CSIAM) and a committee member of the geometric design & computing of CSIAM. He is a member of the China Computer Federation(CCF). He was a technical program for the 5th Asian Conference on Pattern Recognition (ACPR 2019). His research areas include computer vision, computer graphics, image and video processing, deep learning. He has published more than 20 refereed articles in journals and conference proceedings.

Linchao He is currently a senior student in the College of Information and Engineering, Sichuan Agricultural University (SICAU) in Yaan, China. He is a member of the China Computer Federation (CCF). His research interests include image classification, object detection, action recognition and deep learning.

Mengting Luo is currently a senior student in the College of Information and Engineering, Sichuan Agricultural University (SICAU) in Yaan, China. She is a member of the China Computer Federation (CCF). Her research interests include image classification, object detection and action recognition.

Zhanya Xu received the PhD degree from the China University of Geosciences in 2010. He is currently a lecturer with the School of Grography and Engineering, China University of Geosciences, Wuhan, China. He is a member of the China Computer Federation(CCF). His research areas include spatial information services, big Data Processing and intelligent computing, and he has published more than 20 papers in journals and conferences.

Fazhi He received Ph.D. degree from Wuhan University of Technology. He was post-doctor researcher in The State Key Laboratory of CAD & CG at Zhejiang University, a visiting researcher in Korea Advanced Institute of Science & Technology and a visiting faculty member in the University of North Carolina at Chapel Hill. Now he is a professor in School of Computer, Wuhan University. He has been serving as a senior member of the China Society for Industrial and Applied Mathematics (CSIAM) and a committee member of the geometric design & computing of CSIAM. Currently, he is a member of the editorial board for the Journal of Computer-Aided Design & Computer Graphics. His research interests are computer graphics, computer-aided design and computer supported cooperative work.