Computational Visual Media
https://doi.org/10.1007/s41095-020-0185-5 Vol. 6, No. 4, December 2020, 455–466
Research Article

Weight asynchronous update: Improving the diversity of filters in a 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 ability to learn a variety of features. A well-trained deep convolutional network can be compressed to 20%–40% of its original size by removing filters that make little contribution, as many overlapping features are generated by redundant filters. Model compression can reduce the number of unnecessary filters but does not take advantage of redundant filters since the 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 do not necessarily overcome the issue. To do so, we propose a new training strategy, weight asynchronous update, which helps to significantly increase the diversity of filters and enhance the representation ability of the network. The proposed method can be widely applied to different convolutional networks without changing the network topology. Our experiments show that the stochastic subset of filters updated in different iterations can significantly reduce filter overlap in convolutional networks. Extensive experiments show that our method yields noteworthy improvements in neural network performance.

Keywords deep convolutional network; model compression; convolutional filter

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. He, 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: 2020-04-06; accepted: 2020-06-16

1 Introduction

In the past few years, deep learning methods based on convolutional neural networks (CNNs) have obtained significant achievements in machine vision [1, 2], shape representation [3–5], speech recognition [6, 7], natural language processing [8–10], 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 [11, 12].

Deep convolutional networks possess strong learning capability owing to their rich sets of parameters. However, at times, the number of parameters can be excessive, which leads to overlapping and redundant features. It also causes overfitting to the training set and a lack of generalization to new data. Several modern networks, which have hundreds of layers (e.g., ResNet [13], DenseNet [11], and Inception [14]), employ an architectural approach to alleviate the above problems. One key idea is that residual connections in early layers and feature fusion can be considered to add noise in the feature space, which regularizes the network, and hence reduce the overlap of learned deep features.

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, instead of optimizing the capacity of networks in training, is to minimize the memory requirements and to accelerate the speed of inference without degrading performance. Exploring the best performance of the modern networks is still a challenge.

Our work aims to expand the capacity of a network by reducing overlap between its filters. Our method includes the following two major techniques, which are our key contributions:

- Weight asynchronous update (WAU). We perform backward propagation asynchronously to update a subset of convolutional filters to reduce overlap between them. As this does not change the original network architecture, it can be easily applied to neural networks to boost their performance on various visual tasks.
- Asynchronous–synchronous–asynchronous (ASA) training. Reducing the model capacity for every mini-batch would lead to missing relevant relationships between deep features and target outputs [15]. To address this issue, we first apply WAU to initialise the network and to provide it with orthogonality. Then, *sync* training is applied, which is beneficial to global learning, as it strengthens connections between filters and enhances the relationship between feature maps and output. Finally, *async* training is used again to disrupt the convergent evolution of the previous training phase and reduce overlap between filters.

The remainder of this paper is organized as follows: Section 2 briefly reviews related work on model compression, weight inactivation, and its extension to regularization. Section 3 explains the motivation behind weight asynchronous update. Sections 4 and 5 give details of our methods for weight asynchronous update and the ASA training flow. In Section 6, we evaluate our approach and compare it with other representative approaches. Finally, conclusions and future work are discussed in Section 7.

2 Related work

Recently, many works have been published on neural network optimization, and significant progress has been made on some longstanding problems. Methods can be grouped into three types according to the network optimization approach: (i) model compression, (ii) weight inactivation, or (iii) regularization. We will review each of these approaches in turn.

2.1 Model compression

In order to reduce computational and memory costs, pruning a well-trained model is currently the most widely used method of model compression [16]. This method finds an effective criterion to judge the importance of parameters and prunes redundant connections or filters. The smaller, pruned model is able to re-learn the knowledge from the original larger model without significant loss of performance. However, network pruning aims to reduce the redundancy of model, not to take advantage of increasingly deeper and wider networks. The reason is that network pruning reduces over-parameterization during inferencing, but does not help with training. We propose a training scheme that focuses on mitigating over-parameterization and 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 performance. Inspired by this characteristic, several training strategies have been explored to re-train the redundant filters with the aid of a ranking criterion. Dense–sparse–dense (DSD) [15] applies a hard threshold mask to kernel weights according to a Taylor expansion of the cost function [17]. It divides filters into two fixed groups via the hard mask $|w_i| < \lambda$ and prunes the less salient group in the second training phase, but it cannot break symmetry within these groups. Currently, RePr [18] has overtaken DSD. It prunes the top-N least orthogonal filters according to filter orthogonality ranking in the whole network in each epoch. However, RePr has the problem that lower dimensional filers tend to be pruned in practice, which leads to RePr performing well in shallow networks while its performance decreases in deep networks. Therefore, we argue that the degree of overlap is not well judged by using a fixed criterion. Our training scheme does not use an external criterion and generates kernel masks for re-training. It uses a simple and generic strategy with low computational cost.

2.3 Regularization

In order to reduce generalization error, various regularization methods have recently been proposed. Dropout [19] is an effective approach to overcome over-fitting; it makes an ensemble of several weak classifiers by dropping neurons randomly to provide a more robust strong classifier. Additionally, batch

normalization (BN) is widely used in modern CNNs to reduce the *internal covariate shift* problem. However, theoretical and empirical evidence demonstrates that combining Dropout and BN usually causes unsatisfactory performance [20]. Recently, Shake-Shake regularization [21] has broken benchmarking records on CIFAR10: it disturbs forward and backward propagation using a stochastic affine combination. However, Shake-Shake [22] slows down convergence due to its strong interference. 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 [12]).

To overcome the above deficiencies, we have designed an effective approach for WAU, which can also be regarded as a regularizer with disturbance to add noise into the network. Firstly, WAU works well with BN and works for any convolutional network. Secondly, WAU does not increase convergence time, and can even require fewer parameter update iterations. Experimental results demonstrate that our proposed WAU method, with a simple but powerful weight update strategy, is superior to using a neuron ranking criterion and provides good performance in deep network learning.

3 Motivation

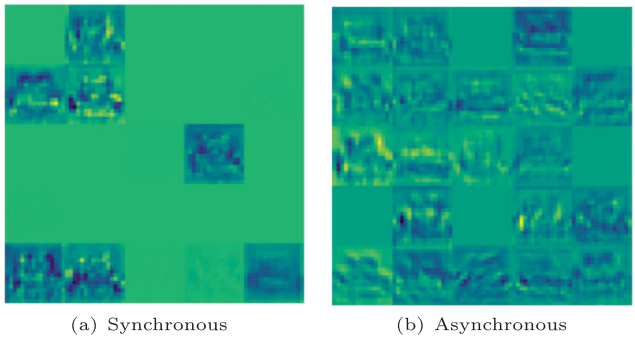
In each standard backpropagation (BP) step, the parameters W of filters F are updated with learning rate η. In the case of a single layer perceptron, each parameter w_i of the hidden layer h is updated as follows:

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

where α represents an activation unit. The partial derivative of w_i is determined by a mini-batch loss denoted by 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 for sample x_j in n classes, respectively. If all parameters in W are updated in the same iteration, this is referred to as *synchronous learning*. However, updating all weights using identical information entropy over thousands of iterations can result in poorly differentiated features within each layer. This phenomenon widely occurs in modern deep neural networks; we call it *convergent evolution* in this paper.

Convergent evolution usually happens in filters in the same hidden layer. The most widely accepted understanding of adaptive filters in a CNN is that filters in the lower layers learn low-level visual features, while the filters in the upper layers learn high-level semantic information. This would seem to imply low correlation and discriminative semantic features in different layers. However, previous work [23] has shown that resemblance of residual blocks is demonstrated by deleting individual blocks [24], and that convergent evolution also appears in a variety of convolution networks, being referred to as *convergent learning*. This work shows that *convergent learning* does not only exist within a layer, but also between different layers. This echoes evolutionary theory in biology: independent evolution of similar features occurs in species of different lineages, when placed in the same type of environment and having a similar lifestyle [25]. In short, poorly discriminated filters result in inefficient deep convolutional networks.

This motivates us to introduce our weight asynchronous update method, to prevent convergent evolution and network symmetry. WAU updates different filters in different iterations. This increases the diversity of filters within the same layer and between different layers. To demonstrate the effectiveness of WAU, we compare features generated by kernels with *sync* and *async* updating. As shown in Fig. 1(a), in the former case, the highly related filters lead to features with similar and weakly differentiated representations. It is clear that more varied and diverse features are generated by the WAU training strategy. As Fig. 1(b) shows, it mitigates overlap and improves the representation ability of a convolutional network. Here, L2 regularization is used to constrain filters close to zero. There are more plain green regions which do not contain useful information in Fig. 1(a) than in Fig. 1(b).



(a) Synchronous (b) Asynchronous

Fig. 1 Features learned by *sync* and *async* updating.

4 Weight asynchronous update

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

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

We use the stochastic gradient descent (SGD) algorithm as an example to explain how the filters are updated asynchronously in a mini-batch. The standard SGD optimizer is as in the equations 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 for iteration t. θ_t can be obtained by updating θ_{t-1} by $\Delta\theta_t$.

For each mini-batch, we sample active filters $\hat{F_{l,t}}$ from $F_{l,t}$ via stochastic sampling function $\mathcal{S}$; every filter has the same probability r of being updated. 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:

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

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

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

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

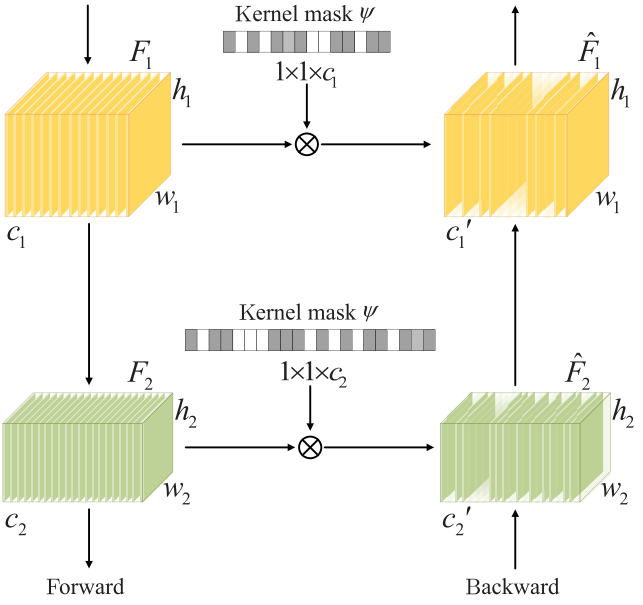


Fig. 2 Weight asynchronous update training strategy. c, w, and h are channel, width, and height, respectively.

The goal of WAU is to change how convolution filters are updated. 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, the adaptive optimizer [26, 27], which needs to save previous variables, also works in the normal way when using our training scheme.

5 ASA training flow

ASA, an extended version of WAU, is now introduced in this section. Inspired by Ref. [15], our ASA training flow includes three processes: async weight update, sync weight update, and re-async weight update.

- *Async.* CNNs [1, 11, 13] use various weight initialization schemes to avoid learning redundant features, but they are unable to handle redundant features during the training phase. The first async step not only learns values of the weights, but also aims to expand the distance between the filters and to warm up the network, which is equivalent to initializing the network by learning real-world data.
- *Sync.* Hierarchical relationships of the features are generated by kernels in different layers, which are formed by standard backpropagation. The weights of the network are updated synchronously to enhance the relationships between deep features and the outputs [15].

- *Re-Async*. The filters are updated asynchronously to ensure that the weights of the redundant filters can be differentiated in different directions. Hyper-parameters, such as async rate and weight decay r, are chosen consistently with the first async step. The re-async step increases the diversity of the kernel, as well as the capacity of the network. Compared to using sparse networks, it is possible to converge to better local minima from the sync step.

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 standard CNNs [1, 11–13, 29–31] with convolutional networks that add the WAU strategy; in addition, we apply them to some visual tasks. Secondly, we verify the effectiveness of ASA training flow. We experimentally demonstrate that, in agreement with our ideas, the WAU method effectively reduces filter overlap. 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, e.g., using the same strategy for decreasing learning rate and the same hyper-parameters. The corresponding code and model are available at `https://github.com/djzgroup/wau`.

6.1 WAU improves accuracy of CNNs

Unless otherwise stated, the hyper-parameters are set to the same values 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 significantly improved accuracy using various convolutional networks.

Table 1 shows accuracy for various networks with and without WAU on CIFAR-10 and CIFAR-100.

On CIFAR-10, the accuracy of AlexNet [1] is enhanced by 1.66% when using our WAU method compared to the baseline. For other well-known convolutional networks (e.g., ResNet [13], VGG [1], PreResNet [30], ResNext [12], Wide ResNet [31], and DenseNet [11]), WAU also improves their capacity and provides better accuracy than the baselines (giving at least 0.43% accuracy improvement). For DenseNet-40 and DenseNet-100, our WAU makes a bigger improvement in accuracy (1.63% and 1.10% respectively) compared to the sync training method. In our experiment, ResNext [12] uses two special convolutional structures, pointwise convolution and group convolution. It can be seen from Table 1 that ResNext is significantly improved by using WAU.

The more complex and challenging dataset CIFAR-100 is a 100-class classification problem which requires much more diverse filters, and WAU makes a better improvement to the baseline than for CIFAR-10.

For AlexNet [1], the accuracy enhancement is further improved by 1.06% (2.72% vs. 1.66%) for CIFAR-100 than for CIFAR-10. For VGG-BN [29], 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: the accuracy is improved by 4.13%.

Table 1 demonstrates that our WAU method can improve the performance of most convolutional network frameworks. The WAU training flow is a generic strategy which does not depend on a specific network framework. In contrast with weight synchronous update, our WAU method can produce

Table 1 WAU shows significant performance improvement over the baseline on both CIFAR-10 and CIFAR-100 (see Ref. [28], chap. 3)

Network	Depth	Test accuracy (%)			
		CIFAR-10		CIFAR-100	
		Baseline	WAU	Baseline	WAU
AlexNet [1]	—	77.26	78.92 (1.66↑)	45.17	47.89 (2.72↑)
ResNet-50 [13]	50	92.68	93.38 (0.70↑)	70.58	71.41 (0.83↑)
VGG-BN [29]	19	93.01	93.44 (0.43↑)	70.76	73.73 (2.96↑)
PreResNet [30]	110	93.58	94.12 (0.54↑)	72.53	73.23 (0.70↑)
ResNext [12]	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 [11]	40	89.91	91.54 (1.63↑)	63.28	67.41 (4.13↑)
DenseNet [11]	100	91.10	92.20 (1.10↑)	68.08	70.22 (2.14↑)

more diverse filters (Fig. 1(b)) to learn a more discriminative data representation, thereby giving better performance.

The effectiveness of the proposed WAU strategy can also improve performance of models for various other tasks, e.g., for object detection: see Tables 2 and 3.

As Table 2 shows, 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@.5 is the result with an intersection over union (IoU) threshold of 0.5, while AP represents the average result for different classes with 10 IoU thresholds of from 0.50 to 0.95 in steps 0f 0.05. Remarkably, our method provides an accuracy gain of 1.2% for AP@.5 and 0.5% for general AP compared to the VGG-16 baseline. This model has many fewer parameters (by a factor of 11) than plain ConvNet, leading to significantly higher error rates, but we chose to equalize inferencing time rather than parameter count, due to the importance of inferencing time in many practical applications.

In another test, object detection networks were trained and tested on the Pascal VOC dataset using the usual splits. We use AP@.5 to characterize performance, as it is the standard Pascal VOC metric. Table 3 shows that our WAU strategy boosts the performance of the baseline on the Pascal VOC dataset for the object detection task. However, we notice that using WAU training for Faster R-CNN leads to different performance changes without and with weight decay (WD), with respectively improvements of 0.64% mAP and 1% mAP. Weight decay will be briefly analyzed later in Section 6.6.

To sum up, these experiments demonstrate that the simple WAU method is suitable for various modern frameworks and tasks. The performance improvement is partly traceable to the contribution of filter diversity generated by WAU, which we will discuss in the next section.

Table 2 Object detection accuracy (%) for Faster R-CNN [32] on the COCO minival set [33]. All models were trained on the trainval35k set with images of size 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↑)

Table 3 Object detection accuracy (%) using Faster R-CNN [32] on the Pascal VOC 2007 test set. Models were 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↑)

6.2 Diversity of kernels with WAU training

Prakash et al. [18] and Li et al. [23] employed correlation analysis to estimate the similarity of filters. In this paper, the diversity of kernels is also evaluated by computing a correlation matrix of filters according to the Pearson correlation coefficient of canonical correlation analysis (CCA):

$$P_{i,j} = E[(F_i - \mu_i)(F_j - \mu_j)]/\sigma_i\sigma_j \tag{9}$$

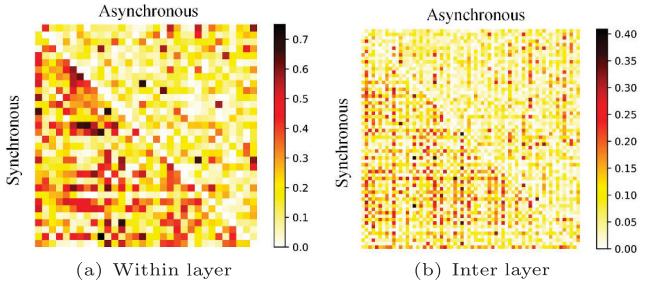
where

$$\mu_i = E(F_i), \qquad \sigma_i = \sqrt{E[(F_i - \mu_i)^2]}$$

μ and σ respectively denoting mean value and standard deviation, and E represents the mean filter value.

To demonstrate that kernel diversity is increased by WAU, we visualize the filter correlation matrix, random sampled within a layer or a residual block of ResNet-110.

Figure 3(a) illustrates the CCA of filter activations



(a) Within layer (b) Inter layer

Fig. 3 Filter correlation using *sync* and *async*. (a) Correlation of 32 filters within a single layer. (b) Correlation of 64 filters between two layers inside a residual block. Upper and lower triangles respectively represent the results of *sync* and *async* weight updating training methods.

in the same layer in a trained ResNet-110. Darker patches represent higher correlation. The lower triangle (sync) shows that almost half of the filters are strongly correlated to others. It is no surprise that convergent evolution of kernels often occurs within the same layer, as discussed in Section 3. Figure 3(b) illustrates the CCA of filter activations between two basic blocks inside a residual block of ResNet-110, the lower triangle revealing that convergent evolution also exists at the layer level. These lower triangles show no strong evidence that widening and deepening the model can result in increasing kernel diversity. Convergent evolution is a problem. Preventing it and increasing the diversity of kernels can help to expand the capacity of the original model. The two upper triangles in Fig. 3 demonstrate that the *async* flow training strategy significantly reduces filter correlation both within and between layers. It is apparent that the trained kernels learn representations in different directions by using WAU, and the mitigation of over-parameterization increases performance.

6.3 Analysis of the ASA training flow

Further studies were carried out on the ASA training flow. We conducted experiments with ResNet-32 trained on the CIFAR-10 training set and evaluated on its testing set. We set the async rate used by the ASA strategy to 0.5, 1.0, 0.5. A sync–async–sync (SAS) training flow was considered as an alternative. We set the training epochs to $N = 164$, with 492 training epochs in total. We re-initialized the optimizer state and the learning rate schedule when changing the weight update method.

Table 4 shows that both training flow strategies improved accuracy due to asynchronous weight update. The ASA training flow achieves a better result in the 1st training phase than the SAS training flow. In the 2nd phase, ASA training gains a greater improvement than SAS training (0.54% vs. 0.16%). Both methods show similar performance enhancement in the 3rd phase (0.11% vs. 0.12%). After the final phase, the accuracy gain of ASA training reaches 0.65% compared to the 1st phase, and it exceeds the SAS training flow gain by 0.64% (93.40% vs. 92.76%).

Table 4 Different training flows. Both strategies lead to accuracy (%) improvement for ResNet-32 trained on CIFAR-10, but ASA is 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↑)

Table 5 shows that compared with other related training strategies [15, 18, 34], WAU can more effectively boost the performance of deep networks and is easier to implement. A strict ranking criterion requires a long training phase for meaningful neuron ranking, and during most of the training time, the filters are still in a state of *sync* updating. In contrast, our stochastic weight inactivation approach enables the updatable filters $\hat{F}$ to continue to change in different iterations.

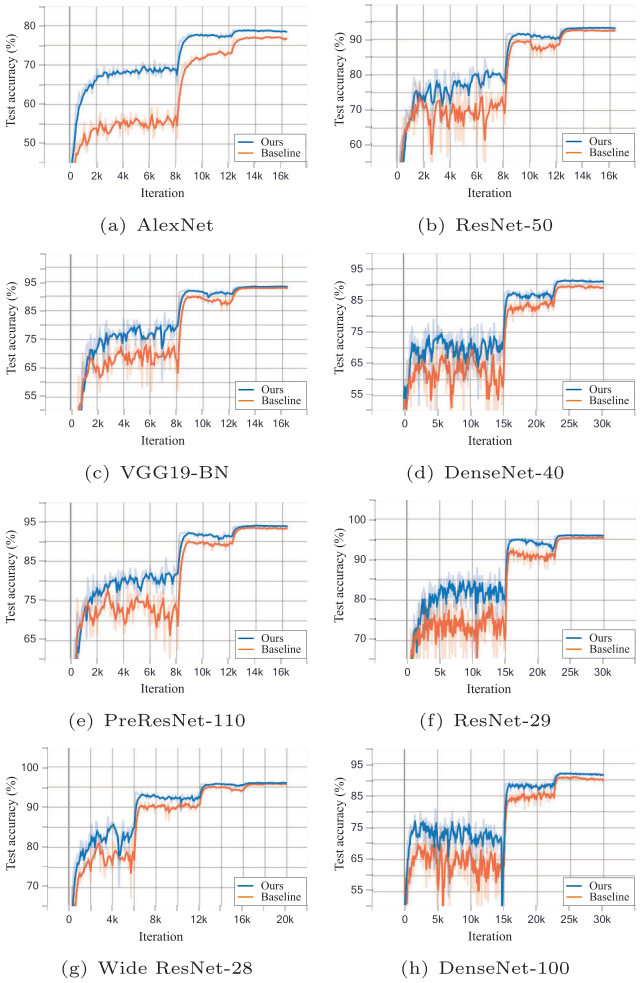
6.4 Speed of convergence

There is no doubt that meaningful filters can modify the model performance. Figures 1(a) and 1(b) collect learned features from a group of filters in ResNet-110 [13] 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 distinctive information and exploit extra deep network potentiality.

In order to reduce the influence of hyperparameters, we followed the training strategy used by most convolutional neural networks. Figure 4 shows the behavior of different convolutional networks. In all experiments, we used a strategy of decreasing learning rate, leading to huge increases at around 6k, 8k, and 15k iterations. Our method converges much faster than the *sync* method before the first learning rate reduction. A large amount of error information is transferred to the filters, which significantly improves the convergence speed over the *sync* training strategy. This is the key reason why our model achieves high accuracy in the early stages, with four times fewer iterations.

Table 5 Comparison of the test error (%) of WAU on CIFAR-10 with other related training strategies

Baseline	Various training schemes			WAU	
Original [13]	DSD [15]	BAN [34]	RePr [18]	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 Test accuracy and convergence speed of our WAU method and a baseline, for various convolutional networks, on CIFAR-10.

6.5 Dropout

Our WAU approach has some similarities with the well-known Dropout. Our asynchronous updating scheme can be intuitively thought of as Dropout only on back propagation. However, there are two major differences between WAU and Dropout:

- **Approach to weight inactivation.** The main goal of Dropout is to prevent overfitting. It employs a Bernoulli random variable r to multiply every single element with the outputs h of the hidden layer. Each r takes the value 1 with the hyper-parameter probability p and has probability $1-p$ of being 0, which is a vector of independent values. In contrast, WAU balances sparsity of inactivation, which prevents nodes from being inactivated in extreme cases. The inactive weight mask is randomly formed by a hyper-parameter *async rate* that fixes the sparsity of each layer and each iteration.
- **Cooperation with batch normalization.** Dropout and BN play significant roles in deep network regularization. However, these two powerful techniques do not lead to the sum of their individual improvements in CNNs when used together (and may actually cause higher generalization error). Previous work [20] revealed that this is due to incompatibility between the normalization of BN and Dropout. BN accumulates variance during the training phase and maintains it during inference. Dropout transfers variance from training to inference.

We have found that there is a further conflict between Dropout and the affine transform $y_l = \gamma_l \hat{x}_l + \beta_l$ of BN, where $\hat{x}_l$ is the normalized input x_l. Affine transform avoids mapping the inputs to a saturated region by the activation function after normalization [35]. However, Dropout breaks the agreement of scaling and shifting parts.

We test block B_1, which consists of a sequence of layers, i.e., **Conv–Dropout–BN–ReLU**, with 0.5 drop probability. As the green curve shows in Fig. 5, combining BN and Dropout slows convergence 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 curve in Fig. 5 shows that cooperation between WAU and BN achieves a competitive 93.6% accuracy. Replacing B_1 by B_2 has a subtle effect on network representation performance. WAU maintains the balance of internal covariate shift (ICS) [35] when going from training to inferencing. Because WAU is quite different from Dropout, it does not prune or inactivate the neuron in forward propagation and

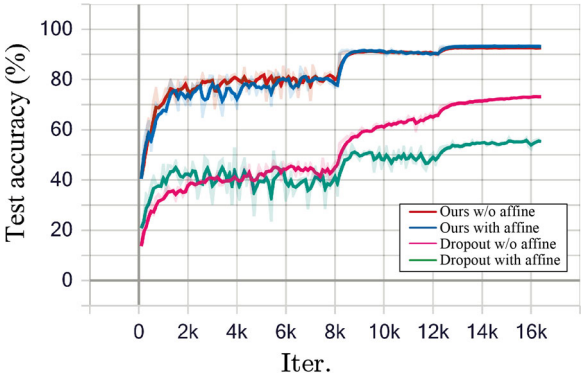


Fig. 5 Ablation of affine transform in BN. Performance on an image recognition task is shown for four different models: WAU and Dropout, each with and without BN affine transform.

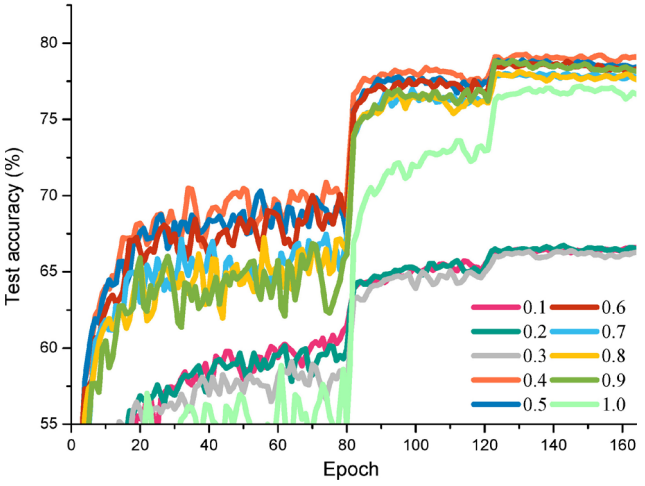


Fig. 6 Influence of hyper-parameter r. High enough *async* rate always results in improved performance, and the method is not sensitive to the hyperparameter.

thus has no impact on 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 deep neural models. We take $r \in \{0.1, 0.2, \ldots, 1.0\}$; WAU degenerates to the baseline when $r = 1.0$. Performance curves for AlexNet [1] on CIFAR-10 are plotted in Fig. 6 and the corresponding test results are reported in Table 6. With low *async* rate, $r \in \{0.1, 0.2, 0.3\}$, in which the update rate is very low, the performance of our method is worse than the baseline (light green). With higher *async* rate ($r \in [0.4, 0.9]$), testing accuracy is much better than the baseline, because our models have faster convergence in early epochs. For all experiments conducted in this paper with different CNNs, our models get significant accuracy gain by using the 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 exploration of a larger weight space. Therefore, it has more chance to escape saddle points and reach local minima. To reduce the exploration of the optimizer as the learning rate decays, to build a more stable model, we increase the weight penalty and set the weight decay rate to 0.001.

Table 6 Comparison of the classification accuracy (%) of 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 have proposed a novel training strategy, weight asynchronous updating, which is able to produce more diverse convolutional filters with reduced overlap. In addition, we present a new *Async–Sync–Async* training flow to enhance the relationships between filters by including *sync* updating during training, which further reduces the generation error. Experiments on various convolutional networks and different visual tasks demonstrate that the WAU method provides faster convergence and improves the performance of convolutional models. In particular, visualizations show that our WAU method changes the behavior of convolutional filters and obtains a better data representation. Remarkably, compared to the baseline, a network that added WAU achieves 2.96% accuracy improvement for CIFAR-100 and 1.2% improvement for AP@.5 for the COCO object detection task.

Weight asynchronous update improves the performance of various deep convolutional networks as shown by the results of our experiments. In the future, we intend to extend this work to generic

network frameworks like multi-layer perceptrons, recurrent neural networks, and make it available for natural language processing and speech processing tasks. Another important future direction is to design an effective and general criterion to accurately describe similarity between filters of different dimensions, to evaluate kernel redundancy. We can re-train the redundant 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 No. 61702350.

References

[1] Krizhevsky, A.; Sutskever, I.; Hinton, G. E. ImageNet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, Vol. 1, 1097–1105, 2012.

[2] Zhang, D. J.; He, L. C.; Tu, Z. G.; Zhang, S. F.; Han, F.; Yang, B. X. Learning motion representation for real-time spatio-temporal action localization. *Pattern Recognition* Vol. 103, 107312, 2020.

[3] Li, J.; Xu, K.; Chaudhuri, S.; Yumer, E.; Zhang, H.; Guibas, L. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics* Vol. 36, No. 4, 1–14, 2017.

[4] Li, J.; Chen, B. M.; Lee, G. H. SO-Net: Self-organizing network for point cloud analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 9397–9406, 2018.

[5] Zhang, D. J.; He, F. Z.; Tu, Z. G.; Zou, L.; Chen, Y. L. Pointwise geometric and semantic learning network on 3D point clouds. *Integrated Computer-Aided Engineering* Vol. 27, No. 1, 57–75, 2019.

[6] Rabiner, L. R. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* Vol. 77, No. 2, 257–286, 1989.

[7] Hinton, G.; Deng, L.; Yu, D.; Dahl, G.; Mohamed, A. R.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen. P.; Sainath, T. et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* Vol. 29, No. 6, 82–97, 2012.

[8] Zhang, D. J.; Hong, M. B.; Zou, L.; Han, F.; He, F. Z.; Tu, Z. G.; Ren, Y. F. Attention pooling-based bidirectional gated recurrent units model for sentimental classification. *International Journal of Computational Intelligence Systems* Vol. 12, No. 2, 723–732, 2019.

[9] Zhang, D. J.; Luo, M. T.; He, F. Z. Reconstructed similarity for faster GANs-based word translation to mitigate hubness. *Neurocomputing* Vol. 362, 83–93, 2019.

[10] Pan, Y. T.; He, F. Z.; Yu, H. P. A novel enhanced collaborative autoencoder with knowledge distillation for top-N recommender systems. *Neurocomputing* Vol. 332, 137–148, 2019.

[11] Huang, G.; Liu, Z.; Maaten, L. V. D.; Weinberger, K. Q. Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4700–4708, 2017.

[12] Xie, S.; Girshick, R.; Doll, P.; Tu, Z.; He, K. Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1492–1500, 2017.

[13] He, K. M.; Zhang, X. Y.; Ren, S. Q.; Sun, J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778, 2016.

[14] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich. A. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1–9, 2015.

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

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

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

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

[19] Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* Vol. 15, No. 56, 1929–1958, 2014.

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

[21] Gastaldi, X. Shake-shake regularization of 3-branch residual networks. In: Proceedings of the 5th International Conference on Learning Representations, 2017.

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

[23] Li, Y. X.; Yosinski, J.; Clune, J.; Lipson, H.; Hopcroft, J. Convergent Learning: Do different neural networks learn the same representations? *arXiv preprint* arXiv:1511.07543, 2015.

[24] Latham, P. E. Associative memory in realistic neuronal networks. In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, 237–244, 2001.

[25] Norton, J. D. Science and certainty. *Synthese* Vol. 99, No. 1, 3–22, 1994.

[26] Duchi, J. C.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* Vol. 12, 2121–2159, 2011.

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

[28] Krizhevsky, A. Learning multiple layers of features from tiny images. Master Thesis. University of Tront, 2009.

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

[30] He, K. M.; Zhang, X. Y.; Ren, S. Q.; Sun, J. Identity mappings in deep residual networks. In: *Computer Vision – ECCV 2016. Lecture Notes in Computer Science, Vol. 9908*. Leibe, B.; Matas, J.; Sebe, N.; Welling, M. Eds. Springer Cham, 630–645, 2016.

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

[32] Ren, S. Q.; He, K. M.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 39, No. 6, 1137–1149, 2017.

[33] Lin, T. Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C. L. Microsoft COCO: Common objects in context. In: *Computer Vision – ECCV 2014. Lecture Notes in Computer Science, Vol. 8693*. Fleet, D.; Pajdla, T.; Schiele, B.; Tuytelaars, T. Eds. Springer Cham, 740–755, 2014.

[34] Furlanello, T.; Lipton, Z.; Tschannen, M.; Itti, L.; Anandkumar, A. Born-again neural networks. In: Proceedings of the 35th International Conference on Machine Learning, 1602–1611, 2018.

[35] Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the International Conference on Machine Learning, 448–456, 2015.

Dejun Zhang received his Ph.D. degree from the Department of Computer Science, Wuhan University, China, in 2015. He is currently an associate professor of the School of Geography and Information Engineering, China University of Geosciences. Since 2015, he has served as a senior member of the China Society for Industrial and Applied Mathematics (CSIAM) and is a member of the geometric design & computing committee of CSIAM. Since 2020, he has been serving as a China Computer Federation (CCF) Senior Member. He was a technical program chair for the 5th Asian Conference on Pattern Recognition (ACPR 2019). His research areas include computer vision, computer graphics, image and video processing, and 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 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 CCF. Her research interests include image classification, object detection, and action recognition.

Zhanya Xu received his Ph.D. degree from China University of Geosciences in 2010. He is currently a lecturer in the School of Geography and Engineering, China University of Geosciences. He is a member of the CCF. His research areas include spatial information services, big data processing, and intelligent computing. He has published more than 20 papers in journals and conferences.

Fazhi He received his Ph.D. degree from Wuhan University of Technology. He was a postdoctoral 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 the School of Computing, Wuhan University. He has served as a senior member of CSIAM and a member of the geometric design & computing committee 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.