BALANCED STRIPE-WISE PRUNING IN THE FILTER

Zheng Huo[1], *Chong Wang*[1,2*], *Weiwei Chen*[1], *Yuqi Li*[1], *Jun Wang*[2] *and Jiafei Wu*[3]
[1] Faculty of Electrical Engineering and Computer Science, Ningbo University, China
[2] School of Information and Control Engineering, China University of Mining and Technology, China
[3] SenseTime Research, China

Introduction

Stripe-Wise Pruning (SWP) combined the strength of aforementioned methods, which achieved more fine-grained pruning than traditional filter pruning and was still hardware friendly. By introducing a filter skeleton matrix to learn the importance of stripes in the filter, less significant stripes were pruned instead of the whole filter. However, SWP does not consider the local spatial distribution of the stripes and the correlation between different channels of the output feature maps, which results an imbalanced pruning result. In some cases, it may degrade the generalization ability or representation capacity of the model.

To address this issue, we propose a variant of SWP, namely balanced stripe-wise pruning (BSWP), to perform pruning in the filter. The key concept of BSWP shown in Fig. 1 is that stripes-wise pruning is performed under the constraint that the number of different types of stripes (intra-filter ones) are close, while the number of survived stripes in different filters (inter-filter ones) will also be balanced by a new pruning strategy. The main contribution of this work is twofold, (1) the idea of balanced pruning is firstly proposed for SWP; (2) the pruning threshold is dynamically adjusted according to the survival rate at each layer during the training process.

Dynamic Pruning Threshold

In the vanilla SWP , the pruning threshold δ is set to a fixed value for all convolutional layers. However, the value of δ is not directly connect to the pruning rate at each layer, which introduces another kind of layer-wise imbalance. Since it unable to control the exact pruning rate at specific layer, undesired poor performace may be triggered by a too low surviving rate in mid-layers.

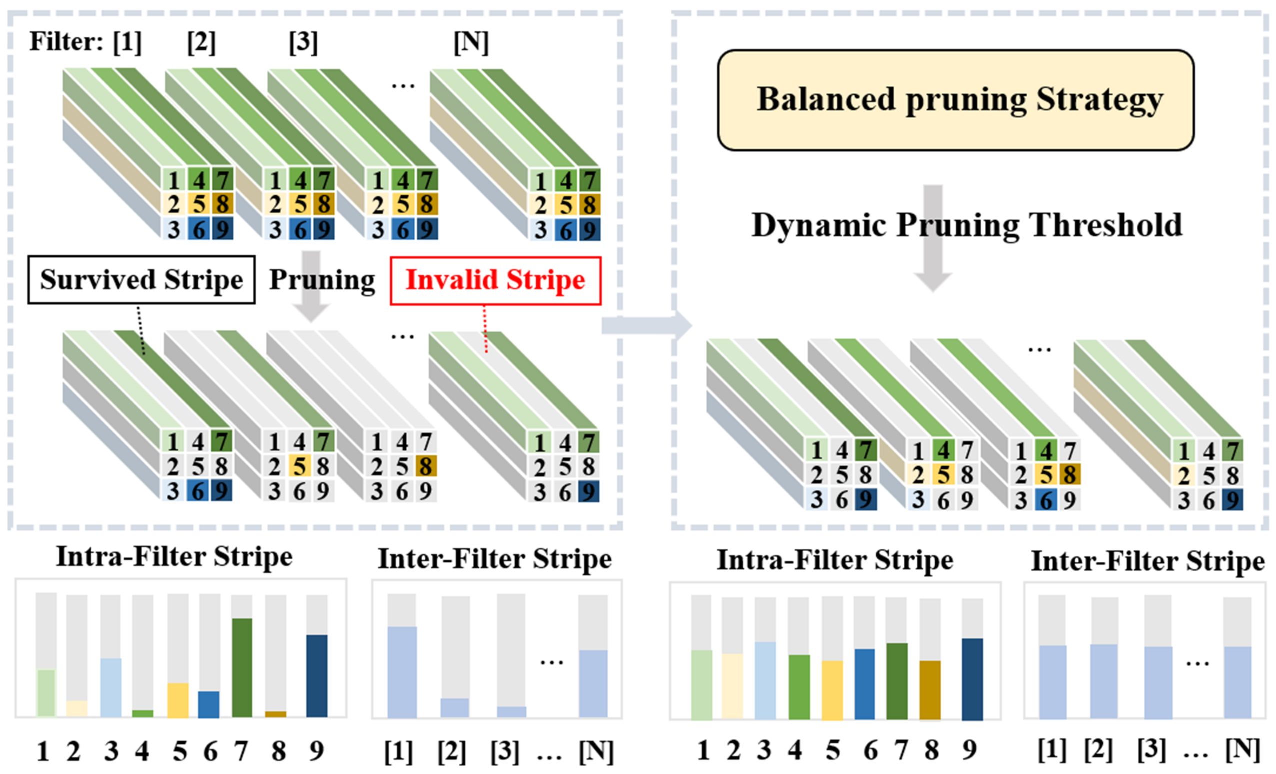
Therefore, the survival rates of every layer in the entire model are counted every 10 training epochs. Different pruning thresholds are assigned according to their current survival rates. In other words, a higher pruning threshold will be given to those layers with a higher survival rate, and vice versa. This dynamic pruning threshold δ^l at l-th layer is determined as,

$$\delta^l = \delta_0 \times \rho(r),$$

where δ_0 denotes initial pruning threshold, r is the survival rate at l-th layer, and $\rho(r)$ is an adjustment function.

Balanced Pruning Strategy

Fig.1. The illustration of the balanced stripe-wise pruning. The number of survived stripes will be balanced.



If the stripe $\bar{W}^l_{n,i,j}$ at location (*i*, *j*) is pruned, it means the information at (*i*, *j*) within the $K \times K$ local window is not used in producing the output feature maps $X^{l+1}_{N,H,W}$. If the stripes at certain locations are rarely survived, some useful patterns may be missing in $X^{l+1}_{N,H,W}$. It may even degrade the generalization ability of the model. A new constraint is imposed to keep the numbers $S^l = \{S^l_1, S^l_2, \ldots, S^l_{K\times K}\}$ of different intra-filter stripes to be balanced. Specifically, a new loss function L_s is defined based on the variation,

$$L_s = \frac{1}{K \times K} \sum_{l=1}^{L} \sum_{m=1}^{K\times K} \left(S^l_m - \bar{S}^l\right)^2$$

Similarly, the stripe imbalance between N filters at l-th layer may also cause performance degradation. Thus, the second balance strategy is proposed in this work to avoid such extreme case for inter-filter stripes. The number of survived stripes in each filter can be denoted as $f^l = \{f^l_1, f^l_2, \ldots, f^l_N\}$, while the second constraint is defined by the inter-filter loss L_f,

$$L_s = \frac{1}{K \times K} \sum_{l=1}^{L} \sum_{m=1}^{K\times K} \left(S^l_m - \bar{S}^l\right)^2$$

It is worth noticing that it is hard to compute the gradient of the thresholding function. Thus the following formula is used to simulate it,

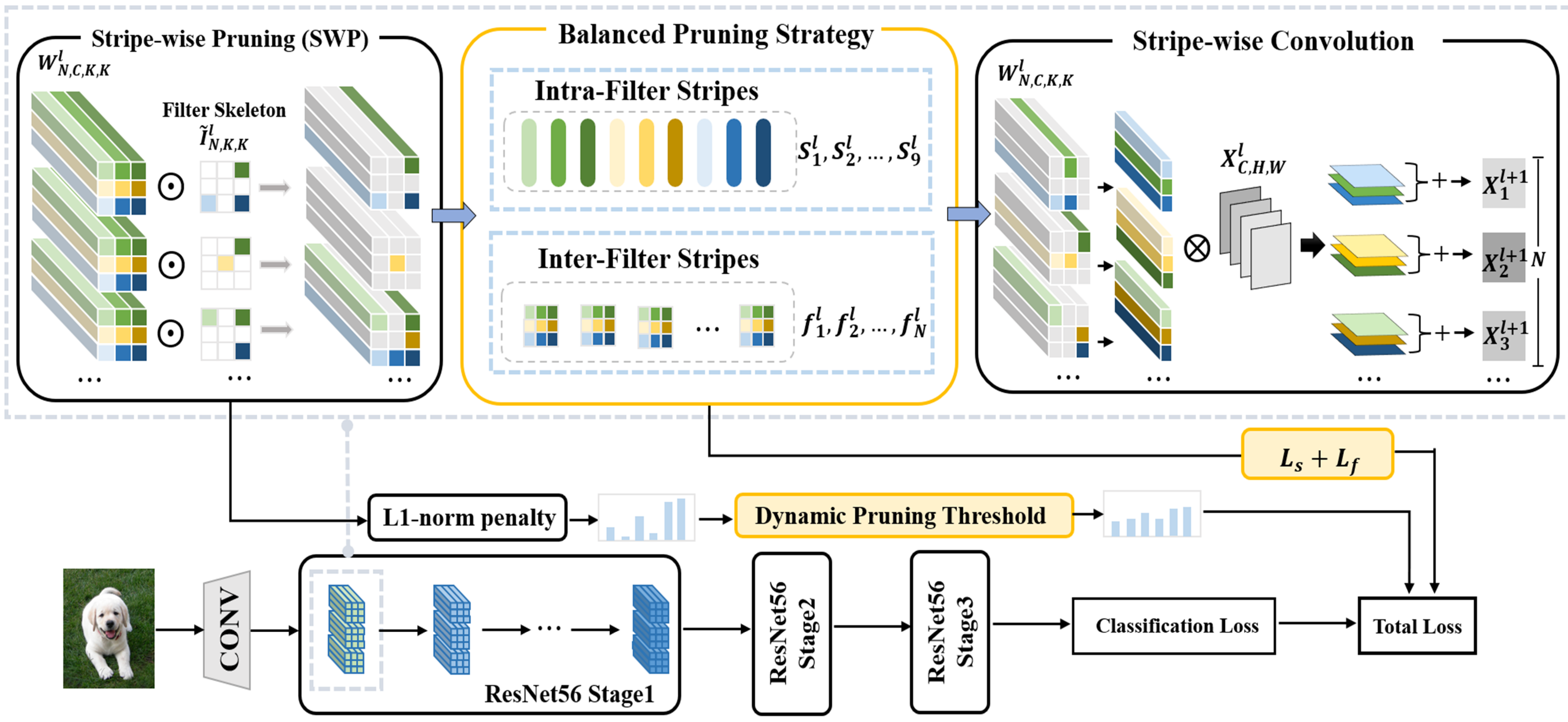
$$\sigma\left(I^l_{n,i,j}\right) = \frac{1}{e^{-q\times\left(I^l_{n,i,j}-\delta\right)} + 1}$$

In summary, the final loss function can be written as,

$$L = L_{cla} + \lambda_1 L_{lasso} + \lambda_2\left(\mu L_s + (1-\mu) L_f\right)$$

Network Architecture

Fig2. The framework of the proposed balanced strip-wise pruning strategy in ResNet56. The main contributions of this work are marked in orange blocks, while the original stripe-wise pruning is denoted in black blocks.



The main components of BSWP will be discussed in this section. As shown in Fig. 2, a new balanced pruning strategy is proposed to improve the vanilla SWP . Not only the number of survived inter-filter stripes but also the number of intra-filter ones are constrained to be close. Furthermore, the survival rate at each layer is also dynamically adjusted to achieve a layer-wise balance.

Experimental Results

Table 1: Pruning results of ResNet56 on CIFAR-10.

ResNet56			
Metrics	Param (%) ↓	FLOPS (%) ↓	Acc.
Baseline [17]	0	0	**93.10**
L1 [12]	13.7	27.6	**93.12**
CP [10]	-	50	92.10
NISP [19]	42.6	43.6	93.07
DCP [20]	70.3	47.1	**93.11**
IR [21]	-	67.7	92.70
GBN [22]	66.7	70.3	93.07
HRnk [23]	68.1	74.1	90.72
SWP [16]	75.6	77.7	92.98
BSWP	**75.8**	**81.1**	**93.10**

Table 2: Pruning results of VGG16 on CIFAR-10.

VGG16			
Metrics	Param (%) ↓	FLOPS (%) ↓	Acc.
Baseline [18]	0	0	93.26
ThiNet [24]	63.95	64.02	90.76
L1 [12]	64	34.3	93.40
SSS [13]	73.8	41.6	93.02
SFP [25]	63.95	63.91	92.08
GAL [26]	77.6	39.6	92.03
Hinge [27]	80.05	39.07	93.59
HRank [23]	82.9	53.5	93.43
SWP [16]	92.66	71.16	93.65
BSWP	**93.83**	**76.34**	**93.84**

Comparing with other recent state-of-arts pruning works, the results of the proposed BSWP with ResNet56 on CIFAR-10 is shown in Table 1. All the works except SWP are channel-wise or filter-wise pruning methods. It can be seen that our BSWP reduces the number of parameters by 75.8% and the number of Flops by 81.1% without losing network accuracy. As shown in Table 2, when VGG16 is used as the baseline model, the proposed model reduces the number of parameters by 93.83% and the FLOPS by 76.34%. It is worth noting that the network accuracy (93.84%) is even higher than the baseline.