Accuracy Booster: Performance Boosting using Feature Map Re-calibration

Pravendra Singh Pratik Mazumder Vinay P. Namboodiri
Department of Computer Science and Engineering, IIT Kanpur, India
{psingh, pratikm, vinaypn}@iitk.ac.in

Abstract

Convolution Neural Networks (CNN) have been extremely successful in solving intensive computer vision tasks. The convolutional filters used in CNNs have played a major role in this success, by extracting useful features from the inputs. Recently researchers have tried to boost the performance of CNNs by re-calibrating the feature maps produced by these filters, e.g., Squeeze-and-Excitation Networks (SENets). These approaches have achieved better performance by Exciting up the important channels or feature maps while diminishing the rest. However, in the process, architectural complexity has increased. We propose an architectural block that introduces much lower complexity than the existing methods of CNN performance boosting while performing significantly better than them. We carry out experiments on the CIFAR, ImageNet and MS-COCO datasets, and show that the proposed block can challenge the state-of-the-art results. Our method boosts the ResNet-50 architecture to perform comparably to the ResNet-152 architecture, which is a three times deeper network, on classification. We also show experimentally that our method is not limited to classification but also generalizes well to other tasks such as object detection.

1. Introduction

Convolutional neural networks (CNNs) have surpassed many traditional machine learning approaches in solving several computer vision tasks such as classification [9, 18], segmentation [2], detection [16, 12] and others. Various works [24, 25, 20, 23, 21, 19, 13, 26, 22] have been proposed for efficient deep learning. Researchers have recently been trying to improve CNN performance, by promoting channels (feature maps) that are more relevant [5]. Each channel or feature map is produced by a Convolutional filter, and each Convolutional layer can have multiple such filters. Therefore, the significance of a feature map points to the relevance of the Convolutional filter that produced it. It has been experimentally shown that increasing the contribution of relevant channels towards creating higher-level features improves the performance of CNNs [5]. Therefore, the recent works have focussed on *learning* the significance (re-calibration weights) of the feature maps. We will use the terms feature map and channel interchangeably to refer to the output produced by a Convolutional filter.

The [5] paper (SENet) captures this channel relevance and shows improvement over the base CNN models. The [29] (CBAM) paper improves upon the SENet idea by using a combination of channel importance and spatial importance to learn better features and improve the network performance further.

However, we find that these methods perform certain redundant transformations to find the re-calibration weights of feature maps, and in the process, they shoot up the architectural complexity of the base model. We propose an idea that performs better than these methods while requiring simpler and lighter modifications to the base networks as compared to these methods. Our method finds a single representative data-point for each channel and applies 1×1 Depth Wise Convolution operation with Batch Normalization (BN) and sigmoid activation to find the channel significance (re-calibration weights). Our lighter architecture, Accuracy Booster block (Figure 1), performs better than the existing methods. We also propose a heavier model (AB-Plus) that significantly beats all the other methods in performance while having a similar computation complexity (FLOPS).

In this work, we show that our design beats the state-of-the-art results in classification on ImageNet and CIFAR datasets and generalizes well for object detection on MS-COCO dataset. We analyze the effects of further reducing our model complexity in our detailed ablation studies.

Our major contributions are as follows:

- We propose a simpler and more efficient Accuracy Booster block (AB) that significantly boosts the performance of CNNs. We provide experimental results that vindicate our choice of architecture through extensive ablation studies.

- We show that our proposed block works well for various networks not only for classification but also for detection.

- We empirically show that our proposed block consistently performs better than SE blocks and other follow up architectures while introducing much lower complexity (extra FLOPs, Parameters, Runtime Memory) to the original model as compared the other blocks.

2. Related Works

Network Design Improvement: Improving the network architecture of CNNs has remained a hot topic from some time now [6, 33]. Improvements have targeted design changes that lead to better performance by the network on various tasks. With an increase in computation power and dataset size, researchers have looked towards even deeper network architectures for CNNs to improve their performance. Architectures like Inception models [27] and VGGNet [18] showed that increasing the depth of a network could significantly increase the quality of representations that it was capable of learning. While Deeper Architectures improved the performance of CNNs, they also introduced problems like vanishing gradients, longer training time and higher space requirements for training and deployment. ResNet [3] proposed skip-connections based on identity mapping, which reduced the optimization issues of deep networks. This allowed for using deeper and more complex networks. WideResNet [31] restricted the network depth and used wider layers to improve the performance, thereby modifying this idea. ResNeXt [30] proposed parallel aggregated transformations blocks and showed that increasing the number of such parallel blocks led to better performance. Our proposed approach improves network performance without significantly increasing the network depth or complexity.

Attention: Attention mechanisms in a network give higher importance to the most relevant components of the information flowing through the network [7, 10, 14, 1, 28].

Channel-wise attention provides the re-calibration weights for the channels (feature maps) generated by convolutional filters. The SE block [5] *Squeezes* the output channels, finds the re-calibration weights for each channel and then *Excites* the channels using these weights. But it also adds to the depth of the network, thereby increasing the complexity of the network and the network latency. CBAM [29] makes use of a combination of channel-wise attention and spatial attention to learn better representations for achieving the same goal. GE-Θ^+ model [4] uses the excitation module of the SE block as a black box and experiments with the squeeze module.

Our proposed AB block uses a very simple transformation to get the re-calibration weights (RW) and still gets better performance than all the other existing methods derived from SE [5]. AB-Plus block achieves even better performance than the AB block but introduces a higher number of parameters with similar computational complexity.

All recent block architectures that have built upon the SE block architecture have increased the complexity of the base model further. Our Proposed AB block is the first to introduce lower complexity than the SE block and still perform consistently better than all other blocks.

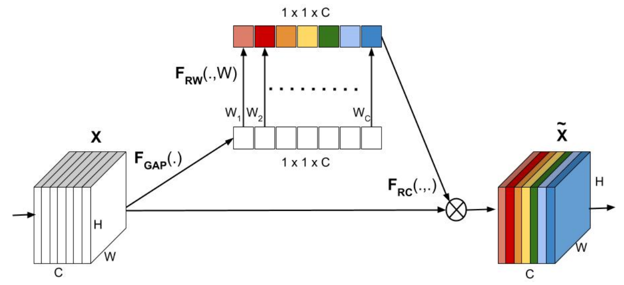

Figure 1. Figure shows the Accuracy Booster block with Depth Wise 1×1 Convolution (best viewed in color).

3. Accuracy Booster Blocks

We propose two types of AB block architectures: AB (Fig 1) and AB-Plus (Fig 2(C)).

In the AB block, first, we need a representative for each channel (feature map), on the basis of which we can judge their importance. We use the Global Average Pooling (F_{GAP}) operator for this purpose. Our input feature maps $X \in \mathbf{R}^{H \times W \times C}$ consist of C channels of height H and width W. Therefore, given an input X, we compute a channel-wise representative for it, $Y \in \mathbf{R}^C$, using global average pooling as follows:

$$y_k = F_{GAP}(x_k) = \frac{\sum_{i=1}^{H} \sum_{j=1}^{W} x_k(i,j)}{H \times W}$$

where each $y_k \in \mathbf{R}$ is a representative for the k^{th} channel and $Y = [y_1, y_2, .., y_k, .., y_C]$ is the channel-wise descriptor. $x_k \in \mathbf{R}^{H \times W}$ is the feature map for k^{th} channel of X where $X = [x_1, x_2, .., x_k, .., x_C]$. F_{GAP} is the the Global Average Pooling operator.

We use this simple technique (GAP) to get a descriptor for each channel because it adds no extra parameters. Other techniques for this purpose are discussed in the ablation studies.

Next, we need to use these representatives to find out the significance of each channel over the other (F_{RW}). We use C Depth Wise 1×1 Convolution operators $W = [W_1, W_2, ..., W_C]$, one for each of the channel representatives, where each $W_k \in \mathbf{R}$. This is followed by Batch Normalization (BN) and a sigmoid activation operator to get the re-calibration weight for each channel (Fig 1, Fig 2(D)). Formally, the re-calibration weights can be defined as,

$$P = F_{RW}(Y) = \sigma(BN(DWConv(Y)))$$

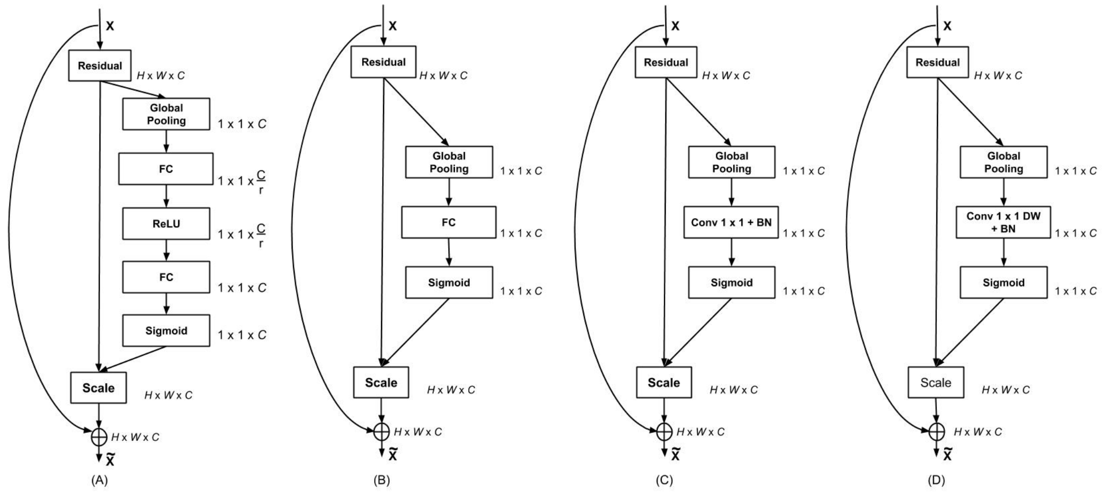

Figure 2. Figure shows the evolution of AB block from the SE block architecture. (A) SE block, (C) AB-Plus block, and (D) AB block. (B) and (C) are similar as fully connected layer performs the same operation as Conv 1×1 layer when operating on a 1×1 feature maps.

Where σ is the sigmoid activation operator, P is the channel-wise re-calibration weights, DWConv is the Depth Wise 1×1 Convolution operator, BN is Batch Normalization applied after performing the Depth Wise Convolution operations. The DWConv can also be seen as the channel-wise product of the re-calibration weights in W and the channel representatives in Y, i.e., $[W_1.y_1, W_2.y_2, ..., W_C.y_C]$.

Finally, the input $X = [x_1, x_2, .., x_k, .., x_C]$ is re-calibrated using the learned re-calibration weights $P = [p_1, p_2, .., p_k, .., p_C]$ ie. $\widetilde{X} = F_{RC}(P, X)$, such that,

$$\widetilde{x_k} = p_k.x_k$$

where, $p_k.x_k$ is a product of the scalar re-calibration weight p_k and the feature $x_k \in \mathbf{R}^{H\times W}$ of k^{th} the channel of the input. $\widetilde{X} = [\widetilde{x_1}, \widetilde{x_2}, .., \widetilde{x_k}, .., \widetilde{x_C}]$ is the re-calibrated output feature maps.

The AB-Plus block is almost the same as the AB block, but it uses C 1×1 Convolutional Operators instead of C Depth Wise 1×1 Convolutional Operators. The difference is that each of the C Depth Wise 1×1 Convolutiona1 layers learns 1 scalar weight W_k for mapping the k_{th} channel descriptor y_k to its re-calibration weight p_k,i.e. it can be thought of as a one to one connection between 2 sets of C nodes each. Whereas, each of the C Full 1×1 Convolutional layers learns a weight vector $\widetilde{W_k} \in \mathbf{R}^C$ for using the entire channel-wise descriptor Y to learn the re-calibration weight p_k of the k^{th} channel,i.e. it can be thought of as a fully connected layer between 2 sets of C nodes each. This means that the AB-Plus block has more parameters than the AB block.

The proposed block can be added after any convolutional layer. In the case of residual blocks, the proposed block should be added before the summation operator for the skip connection, as shown in Fig 2.

4. Relation to SE-Net

As mentioned earlier, the SE-Net [5] also learns channel (feature map) importance and uses them to promote the useful channels. As can be seen in Fig.2(A), the SE block *Squeezes* the channels produced by a Convolutional layer using Global Average Pooling, to get a channel-wise descriptor $Y = [y_1, y_2, .., y_k, .., y_C]$ where $y_k \in \mathbf{R}$ and $Y \in \mathbf{R}^C$. Next, a fully connected layer transforms Y of size C to another smaller descriptor of size $\frac{C}{r}$, where r is a hyperparameter. Then, another fully connected layer transforms the smaller descriptor back to the original C sized descriptor which is followed by a sigmoid activation operator to get the channel-wise re-calibration weights. These weights are used to *Excite* the channels.

A shallower design (Fig.2(B)) can have only one fully connected layer after the Global Average Pooling operator, which transforms the channel-wise descriptor Y to another descriptor of the same size. The sigmoid activation is then applied to it, to get the re-calibration weights.

This design is equivalent to using C 1×1 Convolutional operators, which in effect is the same as a fully connected layer of C nodes. This is our AB-Plus model (Fig.2(C)).

Our proposed design (Fig.2(D)) replaces the C Convolutional operator in the previous with C Depth Wise Convolutional operators. Depth Wise Convolutional operators create a one to one connection between the channel-wise descriptors, before and after the transformation, as can be seen in Fig.1. This means our design is lighter than the model in Fig.2(C) and significantly lighter than the SE-Block Fig.2(A).

To the best of our knowledge, this is the first work that reduces the complexity of the SE block while consistently

Table 1. Analysis of Extra Parameters, Extra FLOPS, and Extra Run Time Memory (RTM) introduced per block for B batch size.

Models	AB	SE	CBAM
Extra Params.	C	$\frac{2C^2}{r}$	$> \frac{2C^2}{r}$
Extra FLOPS	$C.B$	$\frac{2C^2.B}{r}$	$> \frac{2C^2.B}{r}$
RTM	$4C(1+B)$	$\frac{8C(C+B)}{r}$	$> \frac{8C(C+B)}{r}$

performing better than it.

Our Experiments and Ablation studies show that the compressing and expanding of the channel-wise descriptor by the SE block prevents the SE block from achieving the full potential of improvements that can be obtained by using channel-wise re-calibrations. Our approach improves the CNN performance further by overcoming this architectural drawback.

Further, complexity analysis of our proposed design and previous designs are given in the next section.

5. Analysis of Model Complexity

One of the major goals of Network Architecture improvement is to avoid increasing the network complexity significantly in the process of improving network performance. We compare the complexity introduced by our block to that done by the other recent designs. We compare the designs on the basis of the extra parameters, extra computation, and extra run time memory requirement introduced by the performance-boosting approaches.

5.1. Extra Parameters

As mentioned in Table 1, the AB block introduces only C extra parameters which are the parameters in the C Depth Wise 1×1 Convolutional operators. The SE block adds $\frac{2C^2}{r}$ since it uses 2 fully connected layers of size $\frac{C}{r}$ and C respectively. The CBAM block uses the SE block for the channel attention along with another subnetwork for spatial attention. Therefore, the number amount of extra parameters it introduces is greater than that of SE. So we can see that the AB block introduces only extra parameters of the order of $O(C)$, which very less compared to the others, and still performs better. The AB-Plus block adds C^2 extra parameters, which makes it heavier than the others, but it performs significantly better than all the other designs.

5.2. Extra FLOPS

FLoating point OPerations per Second (FLOPS) for a model can be used to describe its computational complexity. FLOPS can be used to represent the total number of computations. The FLOPS are calculated using the process described in [20].

As mentioned in Table 1, the AB block introduces only $C.B$ extra FLOPS. The SE block adds $\frac{2C^2.B}{r}$ extra FLOPS. Since the CBAM block uses the SE block for the channel attention along with another subnetwork for spatial attention, therefore it introduces extra FLOPS greater than that of SE. So the AB block requires lesser extra FLOPS than both of them while the AB-Plus model adds $C^2.B$ extra FLOPS.

5.3. Extra Run Time Memory Size Requirements

Run Time Memory (RTM) denotes the memory space required to store the feature maps and the model parameters. The extra Run Time Memory is calculated using the process described in [20].

As mentioned in Table 1, the AB block introduces only $C \times 4 + 1 \times 1 \times C \times 4 \times B = 4C(1+B)$ extra RTM. The SE block adds $\frac{2C^2}{r} \times 4 + \frac{2C}{r} \times 4 \times B = \frac{8C(C+B)}{r}$ extra RTM. Since the CBAM block uses the SE block for the channel attention along with another subnetwork for spatial attention, therefore it introduces extra RTM greater than that of SE. So the AB block has the lowest extra RTM requirement.

6. Experiments

This section explores the experimental results of incorporating the AB block into various CNN architectures for various tasks and datasets.

6.1. Image Classification

In this section, we explore how the AB block improves the performance of networks like ResNet-50 and others on the image classification task.

Experiments are conducted on the ImageNet large scale dataset [17], CIFAR-10 and CIFAR-100 datasets [8]. The ImageNet dataset has around 1.28 million training images and 50000 validation images from 1000 different classes. The training is done on the training set, and the Top-1 and Top-5 errors on the validation set are reported.

For the ImageNet dataset, we verify the performance of ResNet-50 [3], WideResNet-18 (widen=2) [31] with and without the AB block. For the CIFAR-10 and CIFAR-100 datasets, we verify the performance of ResNet-56 [3], ResNet-164 [3], WideResNet-22 (widen=10) [31] with and without the AB block. For ImageNet experiments, we use the same settings and setup as mentioned in CBAM [29]. The results on the CIFAR-10/100 datasets for all the architectures have been reproduced in the PyTorch [15].

We also compare our AB and AB-Plus results with the networks modified with SE blocks and CBAM blocks. The additional tricks used by SE Blocks, such as repeated training with lower learning rate when the loss plateaus and additional augmentation techniques such as pixel jittering, image rotation, are not used in our experiments in order to maintain standard conditions.

As can be seen in Table 2, AB blocks significantly improve the network performance over the baseline ResNet-50 and also exceed the improvement produced by the SE

Table 2. Single-crop error rate (%) on the ImageNet validation set and complexity comparisons for ResNet-50.

Models	Top-1	Top-5	Params	FLOPS
ResNet-50 (Baseline) [29, 3]	24.56	7.50	25.56M	3.858G
ResNet-152 [5, 3]	22.42	6.34	60.19M	11.30G
SE [29, 5]	23.14	6.70	28.09M	3.860G
CBAM [29]	22.66	6.31	28.09M	3.864G
AB (Ours)	**22.4**	**6.2**	25.57M	3.858G
AB-Plus (Ours)	**22.1**	**6.1**	45.67M	3.878G

Table 3. Inference Time per batch needed for base ResNet-50 network and base modified by SE and AB blocks on ImageNet dataset.

Models	Batch Size	Time in sec	FLOPS
ResNet-50 (Baseline)	256	0.152	3.858G
SE	256	0.187	3.860G
AB (Ours)	256	0.163	3.858G
AB-Plus (Ours)	256	0.192	3.878G

Table 4. Single-crop error (Top-1 and Top-5 error rate) (%) on the ImageNet validation set and complexity comparisons for WideResNet-18 with widen=2 (WRN).

Models	Top-1	Top-5	Params	FLOPS
WRN-18-2 (Baseline) [29, 31]	25.63	8.20	45.62M	6.696G
SE [29, 5]	24.93	7.65	45.97M	6.696G
CBAM [29]	24.84	7.63	45.97M	6.697G
AB (Ours)	**24.7**	**7.6**	45.62M	6.696G
AB-Plus (Ours)	**24.6**	**7.5**	48.40M	6.698G

Table 5. Classification error (%) on the CIFAR-10 for ResNet-56, ResNet-164, WideResNet-22-10.

Models	Original	SE	AB	AB-Plus
ResNet-56	6.5	5.6	**5.34**	**5.26**
ResNet-164	5.5	4.4	**4.0**	**3.9**
WideResNet-22-10	4.4	4.0	**3.7**	**3.7**

and CBAM blocks. The ResNet-50 network with AB blocks also performs comparably to the ResNet-152 network which is 3 times deeper, uses more than twice the number of parameters and requires almost 3 times the FLOPS required by the ResNet-50 network with AB blocks. We can also see that the SE and CBAM blocks add 2.5M more parameters to base ResNet-50 network, whereas our AB blocks just add 0.01M extra parameters. AB-Plus blocks further improve the performance, but since it introduces more parameters, there is a trade-off. Though the AB-Plus blocks introduce a lot of parameters, the FLOPS do not increase by much because the extra parameters belong only to the 1×1 Convolutions applied on $1 \times 1 \times C$ sized feature maps.

From Table 4, we can also see that AB blocks significantly improve the network performance over the baseline WideResNet-18 (widen=2) and those modified by the SE and CBAM blocks. AB-Plus blocks further reduce the error rates for WideResNet.

From the Table 5, we can see that the networks modified with AB blocks show significant improvement in performance over the baseline ResNet-56 and ResNet-56 modified with the SE blocks for classification over the CIFAR-10

Table 6. Classification error (%) on the CIFAR-100 for ResNet-56, ResNet-164, WideResNet-22 (widen=10).

Models	Original	SE	AB	AB-Plus
ResNet-56	28.6	27.3	**26.9**	**26.3**
ResNet-164	24.3	21.8	**21.5**	**21.3**
WideResNet-22-10	20.6	19.3	**19.0**	**18.8**

Table 7. Performance of Mobile optimised networks: Single-crop error rates (%) on the ImageNet validation set and complexity comparisons for ShuffleNet.

Models	Top-1	Top-5	Params	FLOPS
ShuffleNet (Baseline) [5, 32]	32.6	12.5	1.80M	140.0M
SE [5, 32]	31.0	11.1	2.40M	142.0M
AB (Ours)	**30.5**	**11.0**	1.82M	140.5M
AB-Plus (Ours)	**30.3**	**10.9**	6.70M	145.3M

dataset. The AB modified ResNet-56 even performs similarly to the deeper ResNet-164 base model. AB-Plus blocks further reduce the error rate.

From the Table 6, we can see that the networks modified with AB blocks show significant improvement in performance over the baseline ResNet-56 dataset and ResNet-56 modified with the SE blocks for classification over the CIFAR-100 dataset. AB-Plus blocks further reduce the error rates. If we remove Batch Normalization from the AB block, we observe 0.3%, 0.2%, and 0.2% reduction in the accuracy reported in Table 6 for ResNet-56, ResNet-164, and WideResNet-22-10 respectively.

6.1.1 Inference Time

From Table 3, we can see that our modification results in a significant reduction in the inference time of the SE modified ResNet-50 on ImageNet dataset. The AB model reduces the extra inference time needed by the SE block by one-third because the SE block introduces more latency by using 2 layers in the calibration process while the AB block uses 1 layer. Therefore, the AB block has a lower inference time than the SE block, although the FLOPS for the models are similar. All these experiments were run on a single Titan X GPU, with batch size 256. Since CBAM [29] and GE-Θ^{+} [4] both use the SE block as a black box and adds further layers/computation on top of it, they are bound to have more inference time that the SE block and the AB block.

6.1.2 Mobile-optimized networks

We verify the performance of the AB blocks on mobile-optimized networks such as ShuffleNet [32]. We use the same training settings, as mentioned in the SE-Net paper [5]. Apart from comparing our results with the baseline ShuffleNet, we also compare our results with the ShuffleNet with SE blocks. As can be seen in Table 7, AB and AB-Plus blocks significantly improve the network performance over the baseline network and also exceed the improvement produced by the SE blocks.

Table 8. Single-crop error rate (%) on the ImageNet validation set and complexity comparisons for ResNeXt-101 (32×4d).

Models	Top-1	Top-5	Params	FLOPS
ResNeXt-101 (Baseline)	21.2	5.6	44.18M	7.99G
SE	20.7	5.01	48.96M	8.00G
GE-θ^+	20.5	4.8	57.92M	8.02G
GE-θ^+ with AB (Ours)	**20.2**	**4.6**	**53.23**M	**8.00**G

Table 9. Performance on Object Detection: Object detection mAP (%) on the MS COCO validation set using Faster R-CNN.

Base Model	AP@IoU=0.5	AP@IoU=0.5:0.95
ResNet-50 [5, 16]	45.2	25.1
SE ResNet-50 [5]	46.8	26.4
AB ResNet-50 (Ours)	**47.2**	**26.7**

6.1.3 Improving GE-Θ^+ using AB block

The top-performing model (GE-θ^+) of [4] uses the excitation phase of SE (2 fully connected layers with reduction ratio $r = 16$) as a black box to perform excitation. Therefore, if we replace the SE Excitation module used in GE-θ^+ by our proposed Excitation module (1×1 DW Conv + BN), then the performance improvement is guaranteed since our proposed excitation module performs significantly better than the SE excitation module.

In Table 8, GE-θ^+ with AB is the model which uses our proposed Excitation module (1×1 DW Conv + BN) in place of SE Excitation module (two FC with $r = 16$) in GE-θ^+. From Table 8 we can see that GE-θ^+ with AB significantly reduces the number of parameters (by around 4.7 Million) when compared to GE-θ^+ and also performs better than it.

6.2. Object Detection

We explore how the AB block improves object detection performance. We use the MS-COCO dataset [11]. It consists of around 80,000 training and 40,000 validation images. The ResNet-50 model used in the Faster R-CNN network [16] is modified with AB blocks to explore how the AB blocks generalize well to object detection tasks.

Table 9 shows the validation set performance of the object detector using the base ResNet-50 and the modified ResNet-50 with the AB and SE blocks. The AB modified ResNet-50 shows improvement over the base and SE model.

We can conclude from these experiments that AB blocks induce better improvements in the network performance across a number of architectures, datasets and task than other existing methods.

7. Ablation Study

We perform ablation experiments on the AB block architecture to explore the significance of the design choices that we have made. Performance of AB modified ResNet-50/ResNet-56 is computed for Classification task on ImageNet/CIFAR-100 dataset. The Standard data augmentation strategy of random crop and random horizontal flip is used for carrying out the ablation experiments. All the architectures have been extensively trained with the same settings for fair comparisons.

Table 10. Classification accuracy % on CIFAR-100 dataset for the ResNet-56 network with modifications as given in Fig 2 (A,C,D).

Models	Accuracy(%)	Depth
ResNet-56 (baseline)	71.4	0
SE (2 FC layers)	72.7	2
AB (1×1 DWConv+BN)	**73.1**	1
AB-Plus (1×1 Conv+BN)	**73.7**	1

7.1. Network Depth

As shown in Fig 2, our AB block (Fig.2(D)) reduces the number of transformations used in the SE block. The SE block (Fig.2(A)) uses 2 fully connected (FC) layers. A shallower design (Fig.2(B)) can have only one FC layer after the Global Average Pooling, which transforms the channel-wise descriptor Y to another descriptor of the same size. This is equivalent to using C 1×1 Convolutional operators which is in effect the same as a fully connected layer of C nodes (Fig.2(C)). This is the architecture of our AB-Plus block. Our AB block (Fig.2(D)) replaces the C Convolutional operators in the previous design with C Depth Wise Convolutional operators.

We checked the performance improvement induced by the designs in Fig 2(A,C,D) on the ResNet-56 network on Classification accuracy.

Table 10 shows that the AB design (Fig 2(D)) beats the SE design (Fig 2(A), 2 FC layers). This shows that our choice of architecture (having fewer parameters and lower computational complexity than SE) does not hurt the CNN performance but improves it further. The AB-Plus design (Fig 2(C)) beats all the other designs but has more parameters (with similar computational complexity) than all the others.

7.2. Average Pooling

We further experimented with the AB block design by removing the 1×1 Depth Wise Convolutional layer. The first design (E) used only Global Average Pooling (GAP) to get the channel-wise descriptors followed by a sigmoid operation to get the re-calibration weights. The second design (F) modified the design (E) by using 1D Batch Normalization after the GAP operation. Since the GAP operation gives equal importance to all the points in each channel while finding the average, we checked if a weighted average with learnable weights can improve the performance further. The third design (G), uses C Depth Wise Convolutional operators of the same spatial size as the channels or feature maps ($W \times H$) produced by the Convolutional filters of the base model. Each of the C Depth Wise Convolutional operators is for one of the C channels. So it learns a weight for each point in the input feature map. The fourth

Table 11. Classification accuracy % on CIFAR-100 dataset for the AB ResNet-56 network with modifications on the averaging process after removing the 1×1 DWConv layer in AB.

Models	Accuracy(%)
ResNet-56 (baseline)	71.4
AB (1×1 DWConv+BN)	73.1
AB-Plus (1×1 Conv+BN)	73.7
(E) Only GAP	72.0
(F) Only GAP + BN	71.9
(G) Global DW Wt. Avg	72.8
(H) Global Wt. Avg	**73.8**

Table 12. Classification accuracy % on CIFAR-100 dataset for the ResNet-56 network with modifications in AB block to use calibration for each point in the Feature Map (fine-grained calibration).

Models	Accuracy(%)
(I) 3×3 Conv	71.36
(J) 3×3 DW Conv	70.6
(K) 7×7 Conv	72.3
(L) 7×7 DW Conv	72.3

design (H), uses C Convolutional operators of the same size as the set of feature maps ($W \times H \times C$) given as input to the block.

From Table 11, we can see that using "Only GAP" (E design) improves the network performance over the baseline but still lags behind the performance of the AB blocks. Adding Batch Normalization to GAP (F design) is of no help too. The Depth Wise Global Weighted Average method (G design) improves the performance but still falls short of our AB design. The Global Weighted Average design (H) exceeds even the AB-Plus block results. But it is not practical to use this design since it adds $W \times H \times C \times C$ parameters for every block and will drastically shoot up the computational complexity (in the order of $O(WHC^2)$ per block.

7.3. Calibration Level

In the AB block, we learn a single calibration weight for each channel. We experiment with the concept of learning calibration weights for each point in the feature map. The first design (I) uses C 3×3 Convolutional operators with padding=1 to get another same sized point-wise descriptor for the set of feature maps ($\mathbf{R}^{W \times H \times C}$). This is then passed through a sigmoid operator, which gives separate re-calibration weights for each point in the set of channels (feature maps). The re-calibrated output is obtained by performing element-wise multiplication (Hadamard product) of each point in the original set of channels and in the re-calibration matrix. The second design (J) modifies the design (I) to use C 3×3 Depth Wise Convolutional operators instead. The third (K) and fourth (L) design are similar to the design (I) and (J), except that they use 7×7 sized kernels for convolution with padding=3 so that a same sized (same size as the input set of channels $\mathbf{R}^{W \times H \times C}$) re-calibration matrix can be obtained.

Table 13. Classification accuracy % on ImageNet dataset for the ResNet-50 network with modifications in AB block to use calibration for each point in the Feature Map (fine-grained calibration).

Models	Accuracy(%)
ResNet-50 (baseline)	75.44
AB (1×1 DWConv+BN)	77.6
AB-Plus (1×1 Conv+BN)	77.9
3×3 DW Conv + BN	76.6
9×9 DW Conv + BN	77.2

Table 14. Classification accuracy % on CIFAR-100 dataset for the ResNet-56 network with modifications in AB block to use the 2 types of calibration fine-grained and channel-wise.

Models	Accuracy(%)
ResNet-56 (baseline)	71.4
SE	72.7
AB (1×1 DWConv+BN)	73.1
AB-Plus (1×1 Conv+BN)	73.7
(M) 7×7 DW Conv + GAP	72.8
(N) 7×7 DW Conv + GAP + 1×1 DW Conv	73.0
(O) 7×7 DW Conv + GAP + 1×1 Conv	73.5

From Table 12, we can see that the designs with 3×3 filters perform worse than the baseline. The 7×7 filter designs perform better but do not come close to the performance of AB blocks.

We also perform fine-grained calibration (for each point) on the large scale dataset, Imagenet using the ResNet-50 architecture. We use 3×3 and 9×9 Depth Wise Convolution with suitable padding to get re-calibration weights for each point in the set of feature maps. The results in Table 13 show that both the designs are unable to beat the performance of the AB blocks. Therefore, fine-grained calibration is unable to beat channel-wise calibration.

7.4. Combining two types of Calibration Levels

We also experiment with using the two types of calibration (channel-wise and fine-grained) simultaneously. The first design (M) combines the design (L) (7×7 DW Conv) with a GAP operator followed by a sigmoid to get a final channel-wise relevance. The second design (N) combines the design (L) (7×7 DW Conv) with the AB block (GAP + 1×1 DWConv + BN) design. It consists of C 7×7 Depth Wise convolutional operators with padding=3 to get another same sized point-wise descriptor for the set of feature maps ($\mathbf{R}^{W \times H \times C}$). This is followed by a GAP operator, C 1×1 Depth Wise Convolutional operators, Batch Normalization and the sigmoid operator as present in the AB block. The third design (O) is same as the design (N) but uses the AB-Plus block (GAP + 1×1 Conv + BN) in place of the AB block design, which uses 1×1 Convolutional operators in place of Depth Wise Convolution.

From Table 14, we can see that the design (N), which uses the AB block, performs worse than the standalone AB block. The design O, which uses the AB-Plus block, performs worse than the standalone AB-Plus block. Therefore,

Table 15. Classification Accuracy (%) on the CIFAR-100 test set and complexity comparisons for ResNet-164.

Models	Acc(%)	Params	FLOPS
ResNet-164 (Baseline)	75.70	1.734M	246.58M
AB G= #Channel (DW conv)	**78.50**	**1.758**M	**246.59**M
AB G (number of groups) = 16	78.53	1.847M	246.68M
AB G (number of groups) = 8	78.57	1.944M	246.78M
AB G (number of groups) = 4	78.62	2.137M	246.97M
AB G (number of groups) = 2	78.66	2.524M	247.36M
AB-Plus G=1 (Standard conv)	**78.70**	3.298M	248.14M

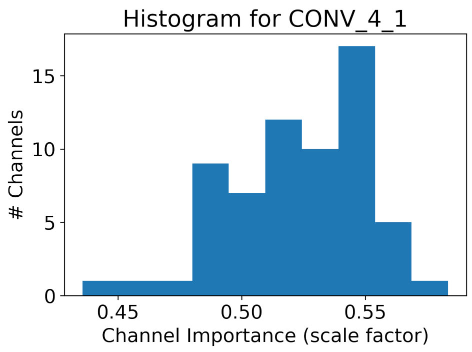

Figure 3. Figure shows the histogram of average channel-wise re-calibration weights (scale factor) for the Conv4_1 layer of AB ResNet-56 trained on CIFAR-10

adding the fine-grained re-calibration to the channel-wise re-calibration results in better performance than the base network and the SE modified network but fails to reach the performance of the AB blocks. Also, since this approach combines two types of calibration, it has more parameters and computations than the AB blocks.

7.5. Group Number

We perform experiments on the number of groups (from 1 to number of channels) in the Convolution operation in the AB block. As can be seen in Table 15 there is no significant performance improvement by reducing the group number.

7.6. Relevance of Calibration

Fig 3 shows a histogram over the channel (feature map) re-calibration weights produced by the AB block on the Conv4_1 layer, i.e., the first layer of the last block, of AB ResNet-56 after being trained on CIFAR-10. The re-calibration weights were averaged over 10,000 test images in the CIFAR-10 dataset. We can see that the network gives different re-calibration weights to different channels.

The calibration weights produced by the AB block, re-calibrates the channels (feature maps) of the output produced the Convolutional filters. We perform two types of experiments to find out how these weights affect the network performance. First, we progressively zero out x% of the highest channel re-calibration weights at each AB block. Second, we progressively zero out x% of the lowest channel re-calibration weights at each AB block. x is varied from 5% to 25%.

Table 16. Classification accuracy % on CIFAR-10 dataset for the ResNet-56 network with AB block by zeroing out the high value re-calibration weights and by zeroing out the low value re-calibration weights.

% of Channels	Acc(%) High wts.	Acc(%) Low wts.
0 (baseline AB)	94.66	94.66
5	76.13	91.32
10	53.52	84.50
15	40.01	74.26
20	29.24	63.11
25	18.22	52.40

Table 16 shows that, if we zero out the top 5% high value channel re-calibration weights, the classification accuracy crashes to 76.13% from 94.66%. This catastrophic drop continues as we increase the percentage of the high-value channel re-calibration weights that are to be zeroed, falling to 18.22% after only 25% high-value channel re-calibration weights have been zeroed. On the other hand, if we zero out the top 5% low-value channel re-calibration weights, the classification accuracy drops by only 3% and by the time we zero out 25% low-value channel re-calibration weights, the classification accuracy is still at 52.4%, which is much higher than the other case. This signifies that those channels which had higher re-calibration weights had highly relevant features in them and zeroing them caused a drastic drop in the classification accuracy. Whereas, those channels which had lower re-calibration weights had not so relevant features and zeroing them could not affect the classification accuracy in such a drastic manner as the most relevant channels were still functioning. Therefore, we can conclude that the channels for which the AB block gives high re-calibration weights contain highly relevant features.

8. Conclusion

In this paper, we proposed the Accuracy Booster block, a performance boosting block for CNNs that uses channel-wise (feature map) re-calibration. Our analysis showed how the AB block is lighter than other recent approaches. Through our several experiments, we show that the AB block performs consistently better than other designs of higher complexity. In our ablation study, we justified our architectural choices while experimenting on various architectures. Since the SE blocks were introduced, the general trend has been to further increase the complexity of such blocks to improve the performance of the base model. However, through extensive ablation studies, we show that too much increase in the complexity of such blocks may not always increase the model performance. We also show that our architecture generalizes to detection as well. Therefore, the Accuracy Booster block is a useful tool to be utilized in Neural Networks for boosting their performance.

References

[1] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua. Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6298–6306. IEEE, 2017.

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

[3] 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.

[4] J. Hu, L. Shen, S. Albanie, G. Sun, and A. Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 9401–9411, 2018.

[5] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 7, 2017.

[6] 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.

[7] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[8] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[9] 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.

[10] H. Larochelle and G. E. Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In *Advances in neural information processing systems*, pages 1243–1251, 2010.

[11] 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.

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[13] P. Mazumder, P. Singh, and V. Namboodiri. Cpwc: Contextual point wise convolution for object recognition. *arXiv preprint arXiv:1910.09643*, 2019.

[14] V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[15] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[16] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

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

[19] P. Singh, V. S. R. Kadi, and V. P. Namboodiri. Falf convnets: Fatuous auxiliary loss based filter-pruning for efficient deep cnns. *Image and Vision Computing*, page 103857, 2019.

[20] P. Singh, V. S. R. Kadi, N. Verma, and V. P. Namboodiri. Stability based filter pruning for accelerating deep cnns. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1166–1174. IEEE, 2019.

[21] P. Singh, R. Manikandan, N. Matiyali, and V. Namboodiri. Multi-layer pruning framework for compressing single shot multibox detector. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1318–1327. IEEE, 2019.

[22] P. Singh, M. Varshney, and V. P. Namboodiri. Cooperative initialization based deep neural network training. *arXiv preprint arXiv:2001.01240*, 2020.

[23] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri. Leveraging filter correlations for deep model compression. *arXiv preprint arXiv:1811.10559*, 2018.

[24] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri. Hetconv: Beyond homogeneous convolution kernels for deep cnns. *International Journal of Computer Vision*, pages 1–21, 2019.

[25] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri. Hetconv: Heterogeneous kernel-based convolutions for deep cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4835–4844, 2019.

[26] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri. Play and prune: Adaptive filter pruning for deep model compression. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[27] 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.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[29] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. Cbam: Convolutional block attention module. In *Proc. of European Conf. on Computer Vision (ECCV)*, 2018.

[30] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017.

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

[32] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices, 2017.

[33] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.