Feature Recalibration in Deep Learning via Depthwise Squeeze and Refinement Operations

XINGPENG ZHANG<sup>©</sup>, (Member, IEEE), AND XIAOHONG ZHANG<sup>©</sup>
School of Big Data and Software Engineering, Chongqing University, Chongqing 400030, China
Corresponding author: Xiaohong Zhang (xhongz@cqu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772093, in part by the National Key Research and Development Program of China under Grant 2018YFB2101200, and in part by the Chongqing Major Theme Projects under Grant cstc2018jszx-cyztzxX0017 and Grant cstc2018jszx-cyztzxX0037.

ABSTRACT Feature recalibration is a very effective strategy of further improving performance in deep networks. The commonly used global pooling operation will lose the information of distinguishing features, which requires additional fully connected layers to adjust the relationship between feature maps. In this paper, we propose a novel architectural unit that aims to recalibrate feature maps based on the discriminative feature information contained in each feature map. To achieve this, we use two-layer depthwise convolution instead of global pooling to extract the distinguishing feature information of each feature map. Since convolution can discriminately learn the global information of each feature map, we discard the fully connected layer to ensure independence when adjusting the feature map. After getting the global information of each feature map, the nonlinear function is used directly to implement the refinement of the feature map to enhance the distinguishing feature map and suppress useless ones. Based on the design characteristics of the new unit, we codename it as the “Depthwise Squeeze and Refinement” (DSR) block. And it can be easily embedded into existing state-of-the-art deep structures to significantly improve the performance at a slight computational cost.

INDEX TERMS Feature recalibration, depthwise convolution, squeeze, global information.

I. INTRODUCTION

Deep convolution neural networks have dominated computer vision tasks due to their powerful representation capabilities. Continuous research shows that the performance of CNN is deeply affected by the depth and width of the network [1]–[5]. The depth of the network means stacking convolution layers to capture hierarchical patterns. The residual connection successfully makes the network achieving a highly impressive depth while avoiding accuracy degradation [4], [6]. And the inception structure proposed by GoogLeNet makes the width widely recognized [2], [7]. The network width is represented by multiple branches, thereby increasing the ability to capture spatial patterns. Another representation of the width is the number of channels. WRN (wide-residual-network) [8], [9] shows that the performance of the network will improve as the number of channels increases under the premise of fixed depth. Increasing the number of channels will result in multiplying parameters. Moreover, similar to the phenomenon that not all layers in very deep ResNet contribute to the network [5], [10], the contributions of different feature maps on the same layer are also widely divergent [11]–[13]. In response to this phenomenon, there are two research directions: model compression and feature recalibration. Model compression mainly refers to channel pruning [12], which scores each feature map by BN (Batch normalization) operation and prunes the feature map with a lower score. The feature recalibration is proposed in SENet (Squeeze-and-Excitation Networks) [14], which can enhance useful features and suppress useless features through squeeze and excitation operations. In CBAM (Convolutional Block Attention Module) [15], this is also called as attention mechanism, which further improves the SE block to enhance the recalibration of features.

Both the SE block and CBAM use global pooling operation to get the global information of each feature map. The global average and maximum pooling used by it are easy to implement and parameterless, but some detailed information is lost in the operation [16], [17]. The max-pooling operation loses detailed information, while the average pooling cannot highlight the contribution of significant information.
FIGURE 1. The Depthwise Squeeze and Refine block. The X is the input features, and \( \tilde{X} \) is the recalibrated features after DSR operation. And the color depth indicates the strength of the feature adjustment.

on each feature map [16], [17]. In fact, regardless of the average or maximum operation, the larger the pooling area, the more detail information is lost. This means that the global pooling operation cannot fully obtain the global information contained in the entire feature map. Fully connected in SENet learns the dependence between channels, which also reflects that it will disrupt the independence of channels.

The depthwise convolution has received much attention in lightweight networks due to its unique convolution characteristics. The Xception [18], MobileNet [19], MobileNet-v2 [20] and ShuffleNet-v2 [21], which apply depthwise convolution, has achieved very good performance with much fewer parameters. Unlike regular convolution operations, each kernel of depthwise convolution operates on only one feature map of the previous layer, which greatly reduces the parameters.

Inspired by this, we designed a new feature recalibration block, which consists of two depthwise convolution layers and a nonlinear refinement function. The role of the two depthwise convolution layers is to extract the global information of each feature map, corresponding to the squeeze operation in the SE block. The role of the nonlinear refinement function is to adjust each feature map according to the extracted global information to achieve selective emphasize informative features and suppress less useful ones.

The basic structure of our DSR block is shown in Fig 1. For any given feature maps \( X \), we can get the recalibrated features \( \tilde{X} \) via the DSR block. Two depthwise convolution layers use large-scale convolution kernels for the fast extraction of global information. The refine operation apply the most widely used sigmoid function to capture channel characteristics. Due to the nature of the convolution operation, the size of the convolution kernel in each stage is different, which will be discussed in section 3.

The core of our DSR block is the squeeze operation. Since the depthwise convolution is parameterized, it can obtain non-destructive global information through end-to-end learning of the network, so the refinement operation only needs to evaluate the learned result.

Both our DSR module and SE module [14] focus on the recalibration of features, but the basic ideas of the two modules are slightly different. The SE module [14] is more inclined to perform calibration based on the relationship between feature maps, while our module is based on the information contained in the feature maps. And the GENet (Gather-excite network) [22] mainly uses convolution and interpolation to achieve information aggregation for each pixel on the entire feature map, which is different from the SENet and our DSRNet.

The DSR block is simple to implement and can be combined directly with existing deep structures to further enhance the learning ability of convolution layers. Due to the depthwise convolution, the DSR is a lightweight calculation that even slightly reduces the complexity of the model compared to the SE block. We develop several DSR-Nets, namely as DSR-ResNet and DSR-ShuffleNet-v2. Then we do evaluations of the DSRNets on the ImageNet32 and CIFAR100 datasets. Compared with SENets, our DSRNets can achieve performance improvements.

II. RELATED WORK

A. GLOBAL POOLING

The global pooling layer proposed in NIN (Network in network) [23] is widely used before the deep learning classification layer to replace the fully connected layer [24]. Its most prominent effect is to integrate global information to make spatial information more robust [25]. Moreover, global pooling is more native to convolution architecture and can reduce the number of parameters [26]. Due to these characteristics, global pooling is used in the attention mechanism, which also face the problem of information loss.

B. ATTENTION MECHANISM

As has been introduced in SENet [14], attention can be seen as tool, which can effectively adjust resource allocation towards the most informative components. It has seen significant interest in recent years as a powerful addition...
to deep neural networks [27]. The attention mechanism is customarily achieved through a combination of gate functions [28], [29]. The introduction of the SE [14] and the CBAM [15] blocks effectively enhance the informative channels.

SE block [14], whose structure is shown in Fig 2, consists of squeeze and excitation. The squeeze operation is to squeeze the global spatial information into a channel descriptor via applying global average pooling. To fully capture channel-wise dependencies, a simple gating mechanism with a sigmoid activation is employed. To reduce parameters, a bottleneck with two fully-connected layers around the non-linearity is formed. CBAM [15] uses the global maximum pooling in addition to the global average pooling in the channel attention block.

### III. DEPTHWISE CONVOLUTION

The depthwise convolution is the ultimate form of group convolution (i.e. each channel is a group). Due to its convolution characteristics, it usually appears with pointwise convolution (i.e. the convolution with a size of $1 \times 1$), which is proposed by Xception [18]. Due to its superiority in parameters, it is favored by lightweight network design [19]–[21], [30], [31].

What’s more, depthwise convolution is also calculated on a single feature map, which is the same as global pooling. We propose to achieve this by introducing a depthwise convolution in feature recalibration.

### III. DEPTHWISE SQUEEZE AND REFINEMENT BLOCK

The Depthwise Squeeze and Refinement block is a computable end-to-end unit. For simplicity of exposition, let $X \in \mathbb{R}^{W \times H \times C}$ be the factorized input. We wish to get the output $\hat{X}$, which has been successfully recalibrated, enhancing the informative feature map. We hope that the feature map can be evaluated based on the global information of the feature map to achieve enhancement or suppression, regardless of the channel dependencies. A detailed diagram of an DSR building block is shown in Fig 2.

### A. DEPTHWISE SQUEEZE OPERATION

The global depthwise convolution can be employed to directly extract the overall information of the feature map, which is also consistent with the global pooling operation. However, due to the large size of the feature map in the early layers, the global depthwise convolution will result in a relatively large parameter burden. So we use two depthwise convolution layers to get global information in early layers.

To quickly decrease the size, the kernel and stride size of the first depthwise convolution layer needs to be specifically set. The second depthwise convolution layer is the global convolution, that is, the size of the kernel is equal to the size of the feature map. In the later layers, the size of the feature map is customarily small, so we only use one global depthwise convolution. Table 1 shows the configuration of the depthwise convolution at each stage. And the “dwconv” denote the first depthwise convolution layer, the “global dwconv” is the second convolution.

![Table 1. Configuration of the depthwise convolution at each stage. And the “dwconv” denote the first depthwise convolution layer, the “global dwconv” is the second convolution.](image)

### B. NONLINEAR REFINEMENT OPERATION

We hope to fulfill our objective, we employ the sigmoid function as a nonlinear refinement function.

$$s = f_{ex}(Z)$$

where $f_{ex}$ refers to the sigmoid function.

To fully guarantee the extraction of complete global information, neither the depthwise convolution layer passes the activation function [21]. For the current feature map size, two convolutional layers are sufficient to fulfill our needs. When the size of the feature map is further increased, it may be considered to increase the number of layers to reduce the parameters.
The calibrated output $\tilde{X}$ is obtained by applying the scale $s$ containing the global information of the feature map to the input $X$.

$$\tilde{x}_c = f_s(x_c, s_c) = s_c \cdot x_c$$

where $\tilde{X} = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_C]$ and $f_s$ denotes the channel-wise multiplication between the feature map $x_c$ and the scalar $s_c$, $c = 1, 2, \ldots, C$.

**IV. DSRNets**

Since the function of the DSR block is to recalibrate the feature, it usually needs to appear with the convolution. In addition to the basic version, we also designed a DSR block with residuals as shown in Fig. 2 (d). We put the DSR block into the existing deep structure and develop a series of DSRNets that integrate with ResNet [4], [6] and ShuffleNet-v2 [21] respectively. Table 2 show the architecture of DSR-ResNet and DSR-ShuffleNet-v2.

We should mention that all the three deep models are designed for the ImageNet dataset, which regularly is trained with a size of $224 \times 224$. To let it satisfies the size of $32 \times 32$, we adopt the configuration proposed in ShuffleNet-v2 [21], which as $\{4, 8, 4\}$. And the DSR-ResNet applies configuration as $\{3, 4, 6, 3\}$, which is proposed in ResNet [4]. Because ResNet contains two many parameters, we reduce one stage to propose a new residual network whose parameters can be greatly reduced. To distinguish the two networks, they are named ResNet-ori and ResNet-new.

For ResNet, the DSR block is located after the convolution operation and before the residual connection. Due to the particularity of the ShuffleNet-v2 block, only half of the channels in the block learn new features during each training session, so the DSR block only recalibrates the features of this half one. The bottleneck and ShuffleNet-v2 with DSR block is shown in Fig. 3. It is also worth noting that ShuffleNet-v2 [21] is also generally considered as lightweight network due to its remarkably few parameters compared to the residual network.

In these DSRNets, each conventional convolution operation contains the following three steps: Batch normalization (BN) [32], [33], Rectified Linear Unit (ReLU) [1], [34] function, and convolution layer [2], which are widely used in great number of deep models [6]. The order of these three operations has been thought to let the network has better performance [6]. The depthwise convolution used in ShuffleNet-v2 and our DSR block contains only two operations: BN and depthwise convolution. The ReLU function is removed to avoid information loss [21].
TABLE 2. The configuration of the DSRNets. (Left) DSR-ResNet50-ori. (Middle) DSR-ResNet50-new. (Right) DSR-ShuffleNetv2-50. The basic structure of the block is given in parentheses, and the number of stacks in the current stage is shown outside the brackets. The configuration of the DSR block for each stage is subject to Table 1. And the “GAP” denotes the global average pooling operation.

| Layers | Output size | DSR-ResNet-ori | DSR-ResNet-new | DSR-ShuffleNet-v2 |
|--------|-------------|----------------|----------------|-------------------|
|        | 32×32       | conv, 3×3, 64  | conv, 1×1, 58  | conv, 1×1, 58, 24 |
| Stage1 | 32×32       | conv, 1×1, 64  | conv, 1×1, 58  | conv, 1×1, 58, 24 |
|        |             | conv, 3×3, 64  | conv, 3×3, 64  | conv, 3×3, 64, 24 |
|        |             | conv, 1×1, 256 | conv, 1×1, 256 | conv, 1×1, 256, 24 |
|        |             | DSR, 526      | DSR, 512       | DSR, 512          |
| Stage2 | 16×16       | conv, 1×1, 128 | conv, 1×1, 116 | conv, 1×1, 116, 24 |
|        |             | conv, 3×3, 128 | conv, 3×3, 116 | conv, 3×3, 116, 24 |
|        |             | conv, 1×1, 512 | conv, 1×1, 512 | conv, 1×1, 512, 24 |
|        |             | DSR, 512      | DSR, 512       | DSR, 512          |
| Stage3 | 8×8         | conv, 1×1, 256 | conv, 1×1, 232 | conv, 1×1, 232, 24 |
|        |             | conv, 3×3, 256 | conv, 3×3, 232 | conv, 3×3, 232, 24 |
|        |             | conv, 1×1, 1024| conv, 1×1, 1024| conv, 1×1, 1024, 24 |
|        |             | DSR, 1024     | DSR, 1024      | DSR, 1024          |
| Stage4 | 4×4         | conv, 1×1, 128 | conv, 1×1, 128 | conv, 1×1, 128, 24 |
|        |             | conv, 3×3, 128 | conv, 3×3, 128 | conv, 3×3, 128, 24 |
|        |             | conv, 1×1, 2048| conv, 1×1, 2048| conv, 1×1, 2048, 24 |
|        |             | DSR, 2048     | DSR, 2048      | DSR, 2048          |

V. EXPERIMENTAL EVALUATION

In this section, we conduct extensive experiments on the ImageNet32 and CIFAR100 datasets to illustrate the advantages of our DSR block. The main comparison test was conducted between DSRNets and SENets. ImageNet32 [35], [36] is a downsampled version of ImageNet, contains exactly the same number of classes and images information. Hence it is comprised of 1.28 million training images and 50K validation images from 1000 classes, but only occupies about 4 Gb of storage space. The emergence of this data set effectively solves the problem that the CIFAR dataset is too small to make the network sufficiently convincing, while the original ImageNet dataset is too large to facilitate. This data set can be downloaded from the official website of ImageNet. Because some information will be lost in the downsampling process, the classification accuracy on the Imagenet32 data set is worth studying.

A. TRAINING INFRASTRUCTURE

To train our network, we apply SGD (Stochastic Gradient Descent) and momentum as the optimizer. And all networks are both trained for up to 50 iterations. The initial learning rate is set to 0.01 and is divided by 10 when the iteration at the {20, 30, 40}. We set the batch size as 128. Besides, we also use a weight decay of $10^{-4}$ and a momentum [4], [5] of 0.9. In addition, the convolution layer and the fully connected layer in all networks use an L2 regularization [37] operation with a coefficient of 0.002 to prevent over-fitting.
TABLE 4. The accuracy of the DSRNets on CIFAR100 compared with SENets. The values in parentheses indicate the increased precision of DSRNet over SENet.

| models                  | Params  | FLOPs       | epoch time | top1-acc | top5-acc |
|-------------------------|---------|-------------|------------|----------|----------|
| ResNet50-ori            | 20.0M   | 1465.9M     | 68s        | 73.62    | 93.72    |
| + SE                    | 2.2M    | 1470.7M     | 72s        | 73.94    | 93.85    |
| + SE-residual           | 2.2M    | 1470.7M     | 75s        | 73.19    | 93.59    |
| + DSR (ours)            | 2.1M    | 1535.2M     | 87s        | 76.11(2.17) | 94.15(0.4) |
| + DSR-residual (ours)   | 2.1M    | 1535.2M     | 90s        | 73.53(1.01) | 93.99(0.4) |
| ResNet50-new            | 6.7M    | 1349.1M     | 63s        | 73.12    | 93.34    |
| + SE                    | 7.4M    | 1343.5M     | 68s        | 71.88    | 92.65    |
| + SE-residual           | 7.4M    | 1343.5M     | 71s        | 73.93    | 93.87    |
| + DSR (ours)            | 7.2M    | 1436.7M     | 77s        | 72.87(0.99) | 93.35(0.6) |
| + DSR-residual (ours)   | 7.2M    | 1436.7M     | 80s        | 74.34(0.41) | 94.06(0.19) |

ShuffleNetv2-116

| models                  | Params  | FLOPs       | epoch time | top1-acc | top5-acc |
|-------------------------|---------|-------------|------------|----------|----------|
| + SE                    | 1.31M   | 145.3M      | 48s        | 69.89    | 90.21    |
| + SE-residual           | 1.47M   | 145.5M      | 53s        | 68.94    | 91.87    |
| + DSR (ours)            | 1.44M   | 166.6M      | 52s        | 70.11    | 92.24    |
| + DSR-residual (ours)   | 1.44M   | 166.6M      | 54s        | 73.67(3.83) | 93.51(1.64) |

ShuffleNetv2-176

| models                  | Params  | FLOPs       | epoch time | top1-acc | top5-acc |
|-------------------------|---------|-------------|------------|----------|----------|
| + SE                    | 2.49M   | 298.5M      | 65s        | 72.56    | 92.97    |
| + SE-residual           | 2.86M   | 298.9M      | 71s        | 71.90    | 92.40    |
| + DSR (ours)            | 2.68M   | 331.1M      | 67s        | 73.12    | 93.05    |
| + DSR-residual (ours)   | 2.86M   | 331.1M      | 68s        | 74.47(1.35) | 93.89(0.84) |

All networks were implemented applying the tensorflow.keras framework and trained on the Tesla V100 GPU with 32G memory. The code of the DSRNets is also available at https://github.com/HedwigZhang/DSR.

B. ImageNet32 CLASSIFICATION

On the ImageNet32 dataset, the DSR and DSR-residual block is also added to the ResNet and ShuffleNet-v2, whose configuration is shown in the Table 2. Table 3 shows the accuracy of the network after adding DSR and DSR-residual blocks respectively.

The results given in Table 3 illustrate the significant performance improvement of introducing the DSR and DSR-residual block into models compared with introducing the SE blocks. For the ResNet50 with four stages, the DSR block can both increased about 0.7 percentage point in top-1 and top-5 accuracy than the SE block. As for the experiments of ResNet50 with three stages, the DSR block outperforms the SE block by 3.87% in top-1 accuracy and 2.69% in top-5 accuracy. Though the residual connection added the SE block can let the SE-ResNet has similar performance with the ResNet by costing only consumes one-third of the parameter amount.

The training process of ResNet50 shown in Fig. 4 illustrates that the DSRNets has higher stability than SENets in the all 50 epochs. It is worth noting that the DSR block makes ShuffleNet-v2 lower than the SE block in the first 30 pieces of training, and only achieves the anti-over time when the learning rate is reduced last time. When the ShuffleNet-V2 has not added SE or DSR block, it also has similar training curves as SENet. This is because although each DSR block is a channel enhancement block, it also contains convolution operations, which will inevitably have a tremendous influence on the network. Then the performance of the network continues to increase as the learning rate decreases. ShuffleNet does not have this phenomenon on the CIFAR100 dataset compared to Fig. 5 (c) and (d). This is because there is a particularly generous amount of data in the ImageNet32 dataset.

C. CIFAR100 CLASSIFICATION

On the CIFAR100 dataset, the configuration of DSRNets in Table 2 is still satisfied, just change the last classification layer to 100. The performance of the DSRNets on the test set is shown in Table 4, which shows that DSR blocks consistently improve performance with even fewer parameters than the SE block.

As shown in Table 3, both our DSR and DSR-residual block has effectively improved the performance than SE and SE-residual block when introducing into both architectures. Except for the unexplained improvement in ResNet with three stages, it has increased by at least two percentage points in ResNet50-ori, Shufflenet-v2-116, and Shufflenet-v2-176.
FIGURE 4. The training process of DSRNets on ImageNet32 dataset compared with SENets. (a): ResNet50 with four stages; (b): ResNet50 with three stages; (c): ShuffleNet-v2 with 116 channels at the first stages.

FIGURE 5. Training curves of DSRNets on CIFAR100. (a): The ResNet50 with four stages; (b): The ResNet50 with three stages; (c): The ShuffleNet-v2 with $1 \times$, i.e. the channel of the first stage is 116; (d): The ShuffleNet-v2 with $1.5 \times$, i.e. the channel of the first stage is 176.

Also in addition to the ResNet50 with three stages, the DSR and SE block without residual can achieve better performance than the blocks with residual. In ResNet50 with three stages, our DSR block with residual achieves the best result and 0.41% higher than SE block with residual. And the DSR is also 0.99% higher than the SE block. For the ResNet50 with...
four stages, the DSR block can improve 2.17% higher than the SE block.

The DSR block improves the lightweight network more than the traditional deep network. For ShuffleNet-v2 with 116 channels, the DSR block outperforms 3.83% than the SE block. When the channels increased to 176, the DSR block is about 4 percentage points higher than the SE block. When the residual connection is also added to the DSR block, the performance can also increase by about 2 percentage points. Compared with deep models, lightweight networks have more room for improvement due to its special convolution characteristics.

The training results, which are shown in Fig. 5, also indicate the learning process concerning DSRNets and SENets. In the four deep networks, the DSR and DSR-residual have faster convergence speed.

1) THE EFFECT OF REFINEMENT FUNCTION
Since we have illustrated that the refinement function should satisfy two criteria for better performance. We also do the experiment by comparing it with the softmax, which is another widely used in gate function, and ReLU, which violates our guidelines. The result on ResNet and ShuffleNetv2 are shown in Table 5.

Because the ReLU function does not meet the criteria we proposed before, the loss and accuracy of both networks are far worse than sigmoid. Although softmax satisfies the criteria, its performance is also significantly lower than sigmoid function because of its functional characteristics which may cause some channels to be excessively depressed. And the tanh function violates our first criterion, which also means that all feature maps have contributions to the current network, and negative values should not be used to suppress the contribution ability of feature maps.

2) THE DISCUSS OF THE TWO DEPTHWISE CONVOLUTION LAYERS
To avoid the parameter increase caused by the size of the previous layers, we use two convolution layers to obtain global information. We also show the performance of a network where all blocks use global convolution, as shown in Table 6.

When the ResNet50 with DSR block, which applies global depthwise convolution, has increased the 1M parameters than the network using the SE block, solely an insignificant increase in performance. For lightweight networks, the global depthwise convolution adds significantly more parameters than the SE block. Although the performance has been greatly improved, the DSR block that uses two depthwise convolution layers can achieve more distinguished network performance improvements with less parameter cost. When the number of depthwise convolutional layers is further increased, although the parameters are slightly reduced, the performance improvement is not obvious.

3) THE DISCUSSION OF TWO FULLY CONNECTED LAYERS
To fully capture channel-wise dependencies, the excitation layer in SEnet is implemented through two fully connected layers. Since we have extracted the global information of each feature map without loss, we expect to adjust the feature map effectively through the refinement operation. The performance impact of the two FC layers on our DSR-ShuffleNet-v2 is shown in Table 7 and Fig. 6. Limited by the number of...
channels in the ShuffleNet-v2 network, the scaling ratio for the two fully connected layers is set as 4.

Table 7 shows that when the fully connected layer is added to ShuffleNet-v2, the parameters of the network will increase significantly, but the performance will decrease. And Fig. 6 also indicates that the use of the fully connected layer will slightly reduce the convergence speed of the network.

The experiment on the fully connected layer also further illustrates that each feature map can contain sufficient information through network learning. The existence of the fully connected layer instead disrupts the independence between feature maps, thereby reducing the effect of feature recalibration.

4) THE COMPARISONS ON RUNNING TIME
The DSR block adds more FLOPs because of the application of depthwise convolution, which consumes more time than SENets when training the network. As can be seen from Tables 3 and 4, The increase in parameters and time on ResNet is particularly significant. And the DSR module consumes almost the same time as the SE module on ShuffleNet-v2. The main reason for this phenomenon is that ShuffleNet-v2 is a lightweight network with a small number of channels.

5) THE COMPARISONS WITH CBAM ATTENTION METHOD
CBAM is an excellent attention module after SENet, so we have also added a comparison test with CBAM. The result is shown in Table 8.

As can be seen in Table 8, our DSR module has achieved a very significant performance improvement effect on ResNet and ShuffleNet-v2. And it has achieved substantial performance improvements on lightweight networks. This further illustrates that using depthwise convolution to obtain global information of the feature map is more effective than the pooling method.

D. DISCUSSION
Our DSR block also contains two operations, the core of the block is different from the SE block. The core of the SE block is the excitation operation which is implemented by two fully connected layers to capture the channel-wise dependencies. Since our block is designed to adjust the feature map by the information contained in the feature map, the acquisition of global information for each feature map is the key to the block. Because of the convolution nature, the application of depthwise convolution to implement the squeeze operation can effectively avoid the loss of global information.

The refinement operation scores each feature map based on the extracted global information, thereby implementing feature recalibration. This idea is different from the SE block, which is why the fully connected layer is not required in the DSR block.

The lossless global information has enough influence on the feature map, and then learning the relationship between the feature maps will adversely affect the performance. The experiment also proves that increasing the total connection layer will not only affect the independence of the feature map but also reduce the performance and increase the parameters.

VI. CONCLUSION
In this paper, we proposed the DSR block, a new type of attention unit designed to do feature recalibration by refining the global information of each feature map obtained using depthwise convolution. The depthwise convolution can effectively extract the global information of each feature map, which also avoids the global pooled information loss problem. And using two layers can effectively reduce the parameters. And sigmoid as the refinement function can effectively evaluate each feature map to achieve feature adjustment. Our block also abandoned the fully connected layer to ensure the independence of each feature map. Extensive experiments on several deep models demonstrate the effectiveness of our DSR block, which can have better performance than SENets.

REFERENCES
[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., Lake Tahoe, NV, USA, Dec. 2012, pp. 1097–1105.
[2] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. Int. Conf. Learn. Represent., San Diego, CA, USA, May 2015.
[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Boston, MA, USA, Jun. 2015, pp. 1–9.
[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
[5] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA, Jul. 2017, pp. 4700–4708.
[6] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in Proc. Eur. Conf. Comput. Vis., Amsterdam, The Netherlands, Mar. 2016, pp. 630–645.
[7] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in Proc. AAAI Conf. Artif. Intell., San Francisco, CA, USA, Feb. 2017, pp. 4278–4284.
[8] S. Zagoruyko and N. Komodakis, “Wide residual networks,” in Proc. Brit. Mach. Vis. Conf., York, U.K., Sep. 2016, pp. 87:1–87:12.
[9] Y. Lee, H. Kim, E. Park, X. Cui, and H. Kim, “Wide-residual-inception networks for real-time object detection,” in Proc. IEEE Intell. Vehicles Symp. (IV), Redondo Beach, CA, USA, Jun. 2017, pp. 758–764.
[10] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, “Deep networks with stochastic depth,” in Proc. Eur. Conf. Comput. Vis., Amsterdam, The Netherlands, Oct. 2016, pp. 646–661.
[11] Y. He, X. Zhang, and J. Sun, “Channel pruning for accelerating very deep neural networks,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Venice, Italy, Oct. 2017, pp. 1389–1397.

[12] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, “Learning efficient convolutional networks through network slimming,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Honolulu, HI, USA, Oct. 2017, pp. 2736–2744.

[13] X. Li, W. Wang, X. Hu, and J. Yang, “Selective kernel networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Long Beach, CA, USA, Jun. 2019, pp. 510–519.

[14] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in Proc. Conf. Comput. Vis. Pattern Recognit., Salt Lake City, UT, USA, Jun. 2018, pp. 7132–7141.

[15] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, “Cham: Convolutional block attention module,” in Proc. Eur. Conf. Comput. Vis., Munich, Germany, Sep. 2018, pp. 3–19.

[16] M. Sun, Z. Song, X. Jiang, J. Pan, and Y. Pang, “Learning pooling for convolutional neural network,” Neurocomputing, vol. 224, pp. 96–104, Feb. 2017.

[17] N. Akhtar and U. Raghavendran, “Interpretation of intelligence in CNN-pooling processes: A methodological survey,” Neural Comput. Appl., vol. 32, no. 3, pp. 879–898, Feb. 2020.

[18] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA, Jul. 2017, pp. 1800–1807.

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” 2017, arXiv:1704.04861. [Online]. Available: http://arxiv.org/abs/1704.04861

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, ‘MobileNetV2: Inverted residuals and linear bottlenecks,’ in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Lake City, UT, USA, Jun. 2018, pp. 4510–4520.

[21] N. Ma, X. Zhang, H. T. Zheng, and J. Sun, “Shufflenet v2: Practical guidelines for efficient CNN architecture design,” in Proc. Eur. Conf. Comput. Vis., Munich, Germany, Mar. 2018, pp. 116–131.

[22] J. Hu, L. Shen, S. Albanie, G. Sun, and A. Vedaldi, “Gather-excite: Exploiting feature context in convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., Montreal, QC, Canada, Dec. 2018, pp. 9401–9411.

[23] M. Lin, Q. Chen, and S. Yan, “Network in network,” in Proc. Int. Conf. Learn. Represent., Banff, AB, Canada, Apr. 2014.

[24] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, and S. Belongie, “Kernel pooling for convolutional neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA, Jul. 2017, pp. 2921–2930.

[25] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[26] Z. Gao, J. Xie, Q. Wang, and P. Li, “Global second-order pooling convolutional networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Long Beach, CA, USA, Jun. 2019, pp. 3024–3033.

[27] V. Mnih, N. Heess, and A. Graves, “Recurrent models of visual attention,” in Proc. Adv. Neural Inf. Process. Syst., Montreal, QC, Canada, Dec. 2014, pp. 2204–2212.

[28] M. F. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber, “Deep networks with internal selective attention through feedback connections,” in Proc. Adv. Neural Inf. Process. Syst., Montreal, QC, Canada, Dec. 2014, pp. 3545–3553.

[29] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Training very deep networks,” in Proc. Adv. Neural Inf. Process. Syst., Montreal, QC, Canada, Dec. 2015, pp. 2377–2385.

[30] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, “Searching for MobileNetV3,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Seoul, South Korea, Oct. 2019, pp. 1314–1324.

[31] S. Ma, W. Liu, W. Cai, Z. Zhang, and G. Liu, “Lightweight deep residual CNN for fault diagnosis of rotating machinery based on depthwise separable convolutions,” IEEE Access, vol. 7, pp. 57023–57036, Apr. 2019.

[32] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in Proc. Int. Conf. Mach. Learn., Lille, France, Feb. 2015, pp. 448–456.

[33] Q. Yuan and N. Xiao, “Scaling-based weight normalization for deep neural networks,” IEEE Access, vol. 7, pp. 7286–7295, Jan. 2019.

[34] Y. Ying, J. Su, P. Shan, L. Miao, X. Wang, and S. Peng, “Rectified exponential units for convolutional neural networks,” IEEE Access, vol. 7, pp. 101633–101640, Jul. 2019.

[35] A. Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” in Proc. Int. Conf. Mach. Learn., New York City, NY, USA, Jun. 2016, pp. 1747–1756.

[36] P. Chrabaszcz, I. Loshchilov, and F. Hutter, “A downsampling variant of ImageNet as an alternative to the CIFAR datasets,” 2017, arXiv:1707.08819. [Online]. Available: http://arxiv.org/abs/1707.08819

[37] T. van Laarhoven, “L2 regularization versus batch and weight normalization,” 2017, arXiv:1706.05350. [Online]. Available: http://arxiv.org/abs/1706.05350

------

XINGPENG ZHANG (Member, IEEE) received the B.S. degree from Jinan University, Shandong, China, in 2012, and the M.S. degree in applied mathematics from Chongqing University, Chongqing, China, in 2015, where he is currently pursuing the Ph.D. degree in software engineering. His current research interests include deep learning, image processing, and dynamics of fractional order chaotic systems.

XIAOHONG ZHANG received the M.S. degree in applied mathematics and the Ph.D. degree in computer software and theory from Chongqing University, China, in 2006. He is currently a Professor and the Vice Dean of the School of Big Data and Software Engineering, Chongqing University. His current research interests include data mining of software engineering, topic modeling, image semantic analysis, and video analysis. He has authored over 50 scientific articles and some of them are published in some authoritative journals and conferences, such as the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON IMAGE PROCESSING, PR, PRL, JSS, and IST.