EDS Pooling Layer

Pravendra Singh*, Prem Raj, Vinay P. Namboodiri

Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, Kanpur, India

Abstract

Convolutional neural networks (CNNs) have been the source of recent breakthroughs in many vision tasks. Feature pooling layers are being widely used in CNNs to reduce the spatial dimensions of the feature maps of the hidden layers. This gives CNNs the property of spatial invariance and also results in speed-up and reduces over-fitting. However, this also causes significant information loss. All existing feature pooling layers follow a one-step procedure for spatial pooling, which affects the overall performance due to significant information loss. Not much work has been done to do efficient feature pooling operation in CNNs. To reduce the loss of information at this critical operation of the CNNs, we propose a new EDS layer (Expansion Downsampling learnable-Scaling) to replace the existing pooling mechanism. We propose a two-step procedure to minimize the information loss by increasing the number of channels in pooling operation. We also use feature scaling in the proposed EDS layer to highlight the most relevant channels/feature-maps. Our results show a significant improvement over the generally used pooling methods such as MaxPool, AvgPool, and StridePool (strided convolutions with stride > 1). We have done the experiments on image classification and object detection task. ResNet-50 with our proposed EDS layer has performed comparably to ResNet-152 with stride pooling on the ImageNet dataset.

Keywords: Feature pooling layer, Convolutional neural network, Deep learning, Object recognition

*Corresponding author.

Email addresses: psingh@iitk.ac.in (Pravendra Singh), praj@iitk.ac.in (Prem Raj), vinaypn@iitk.ac.in (Vinay P. Namboodiri)

1. Introduction and Related Works

Convolutional Neural Networks (CNNs) have attracted the attention of the researchers in recent years due to its outstanding performances in various computer vision tasks [1]. The CNNs consist of many different types of layers, such as convolutional layer, fully connected layer, and feature pooling layer, each having its own functionality. The feature pooling layer, when applied to the feature maps of a hidden layer in CNN, reduces the spatial dimension of the feature maps. This results in the benefits of a reduced number of parameters, computational speed up, and regularizes overfitting [2]. One more subtle benefit of feature pooling is that it increases the spatial invariance property of CNNs towards various image transformations [3].

There are different pooling methods proposed in the literature, such as max pooling, average pooling, mixed pooling [4], stride pooling (strided convolutions with stride > 1) [5], fractional pooling [6], and stochastic pooling [7]. In the feature pooling operation, the main aim is to downsample the input feature maps size with some factor. Average pooling averages out the features in the neighborhood, creating a blurring effect. One common thing in most of the pooling operations, such as max pooling, average pooling, and stochastic pooling, is that they are similar to nearest neighbor interpolation. Neighborhood interpolation might cause data discontinuities due to artifacts such as blurring [8].

Max pooling performs well but is known to cause over-fitting over the training data. To overcome this issue, other pooling methods such as mixed pooling [4] and stochastic pooling [7] are proposed in the literature. Mixed pooling at each run randomly selects either max pool or average pool during the training. The stochastic pool, on the other hand, randomly samples a feature in the pooling window depending upon the probability values of the features. These probabilities are calculated by normalizing the feature values in the window. However, other than these pooling methods, there are other methods proposed to overcome the over-fitting problem, such as dropout [9], and batch normalization [10]. Due to this, max pooling and average pooling are most commonly used in practice due to their simplicity, ease in implementation, and having comparable

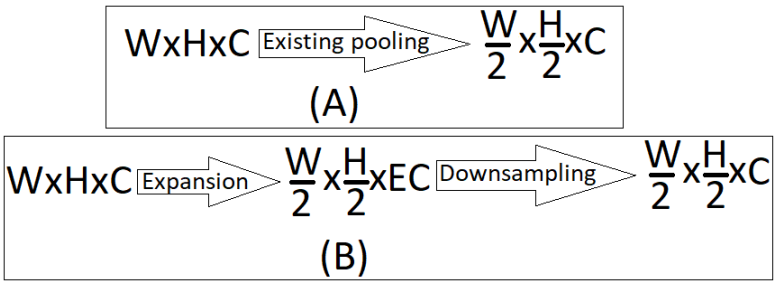

Figure 1: Figure (A) shows a one-step procedure for spatial pooling. Figure (B) shows the proposed two-step procedure for spatial pooling, where we have increased the number of channels (E times) in the pooling operation to preserve more information.

results with other pooling methods [11]. DFT pooling [12] uses 2D DFT over feature maps and then selects the magnitudes of the low frequencies as the new features. Various other feature pooling methods such as FBN [13], SORT [14], MPN-COV [15], and iSORT-COV [16] have been proposed in literature. The method iSORT-COV+EP [17] proposes an entropy-based feature weighting for semantics-aware feature pooling operation. In GFGP [18], pooling parameters are estimated by means of global statistics in the input feature map. Detail-preserving pooling (DPP) [19] proposes an adaptive pooling method that magnifies spatial changes and preserves important structural detail.

In most of the pooling operations, such as max pooling and stochastic pooling, a significant amount of features are simply discarded. In 2×2 max pooling with stride 2, 75% of the features are discarded, which affects the overall performance [20]. Thus, feature pooling is a very critical operation in the CNNs, where a significant amount of information might be lost, and data discontinuities might occur. We aim to design a more efficient way of feature pooling, which tries to reduce the information loss and do learning based feature scaling while maintaining the computational cost within budget.

All existing feature pooling operations follow a *one-step procedure* for spatial pooling (Figure 1 (A)). As shown in Figure 1 (A), a significant amount of information is simply discarded in the pooling operation (75% in the case mentioned in Figure 1 (A)), which affects the overall performance. We propose a *two-step procedure* to minimize the information loss in pooling operation, as shown in Figure 1 (B). In the first step, we

increase the number of channels by a factor E (termed as expansion ratio) simultaneously with a reduction in the spatial dimension, in order to preserve more information. In the second step, we down-sample the number of channels. We have shown experimentally that the proposed two-step procedure is much more efficient than the one-step procedure for feature pooling. This two-step procedure for feature pooling, that minimizes the information loss, is the major contribution of our work.

In existing feature pooling operations, feature selection is made across the spatial dimension only. We use learning based feature scaling to make feature selection across the channels (to highlight the most relevant feature-maps). Other work [21, 22] explores feature scaling only for non-pooling layers while we explore feature scaling especially only for pooling layers. For example, in the case of ResNet-56 on the CIFAR dataset, Squeeze-and-Excitation Networks (SE) [21] will perform feature re-calibration at 27 layers (one layer in each block) while we perform feature scaling for only 2 layers (pooling layers). Further, SE feature re-calibration can be augmented with the proposed approach for further accuracy improvement.

The following are our contributions:

- We propose a two-step procedure for feature pooling to minimize the information loss, which results in a significant performance improvement.
- We show that by increasing the number of channels in the pooling operation, we can preserve more pieces of information.
- We propose a learning based feature scaling to make feature selection across the channels in pooling operation to highlight the most relevant feature-maps.
- We show that by replacing the conventional pooling with a proposed EDS pooling method, the accuracy can be increased significantly. This is an interesting finding because pooling layers only constitute a small part of a deep CNN.
- We show that our proposed approach works well for various networks not only for classification but also for detection. The proposed EDS layer is demonstrated to improve the performances on various models (VGG, ResNet, WideResNet, MobileNet, and Faster R-CNN) and datasets (CIFAR, ImageNet, and MS-COCO).

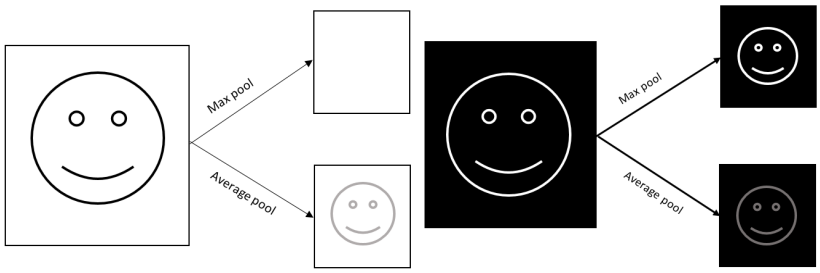

Figure 2: A toy example, Left: Max pool discards the important features, and Right: Max pool preserve the features.

2. Background

The objective of the feature pooling layer is to discard redundant information and preserve the more useful information. To decide which features are informative and which are irrelevant/redundant is very crucial for any feature pooling method. Max pool and average pool are known to perform well and are much used in practice due to their simplicity [11]. But which pooling method can perform better than others depends upon the input data and features [2].

This can be understood by a toy example given in Figure 2. In the case of Figure 2 (left), the relevant feature is black, and the background is white. Max pooling completely wipes out the features while downsampling the feature map and average pooling preserves the feature information, though it causes a blurring effect. On the other hand, in the case of Figure 2 (right), max pooling preserves the features in downsampling the feature map. The problem with these pooling methods is that they are fixed in nature and do not involve any learnable parameters to improve their performance depending upon the input dataset. Our pooling method involves learnable parameters. Hence it does not suffer from these defects. The pooling parameters are learned end-to-end during backpropagation, depending upon the input data.

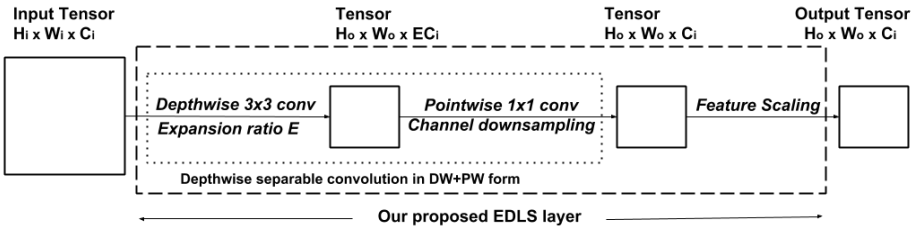

Figure 3: Block diagram of our proposed EDS layer.

3. Proposed Method

3.1. Implementation

Our proposed EDS layer has three important aspects. First, its parameters are not fixed but are end-to-end learnable during backpropagation, unlike other pooling methods such as max pooling and average pooling, which have fixed parameters. Secondly, we are using a two-step procedure for feature pooling to minimize information loss, which results in a significant performance improvement. Thirdly, we are doing channel-wise feature scaling, which highlights the important channels/feature-maps and suppresses the unimportant channels/feature-maps. To the best of our knowledge, no such prior work has been done in case of a feature pooling layer. Figure 3 depicts the block diagram of our proposed EDS layer.

Let's say we have an input tensor (feature maps) of size $H_i \times W_i \times C_i$, where H_i, W_i, C_i are the height, width, and the number of channels respectively. Then as the first part of our EDS layer, depthwise separable convolutions [23, 24] are applied in the DWC+PWC form (depthwise convolution followed by a pointwise convolution). Since depthwise separable convolutions are much cheaper (in terms of parameters and FLOPS) than standard convolutions, we have got a tradeoff to increase the number of depthwise convolutional filters by the factor *E*, which is termed as *expansion ratio*.

Why is the *expansion ratio* required? Let's try to understand the intuition behind using the *expansion ratio (E)*. In CNN, a significant amount of information might be lost due to decrements in the spatial dimension of the feature maps (i.e., pooling). This loss of information can be minimized by increasing the number of channels. The

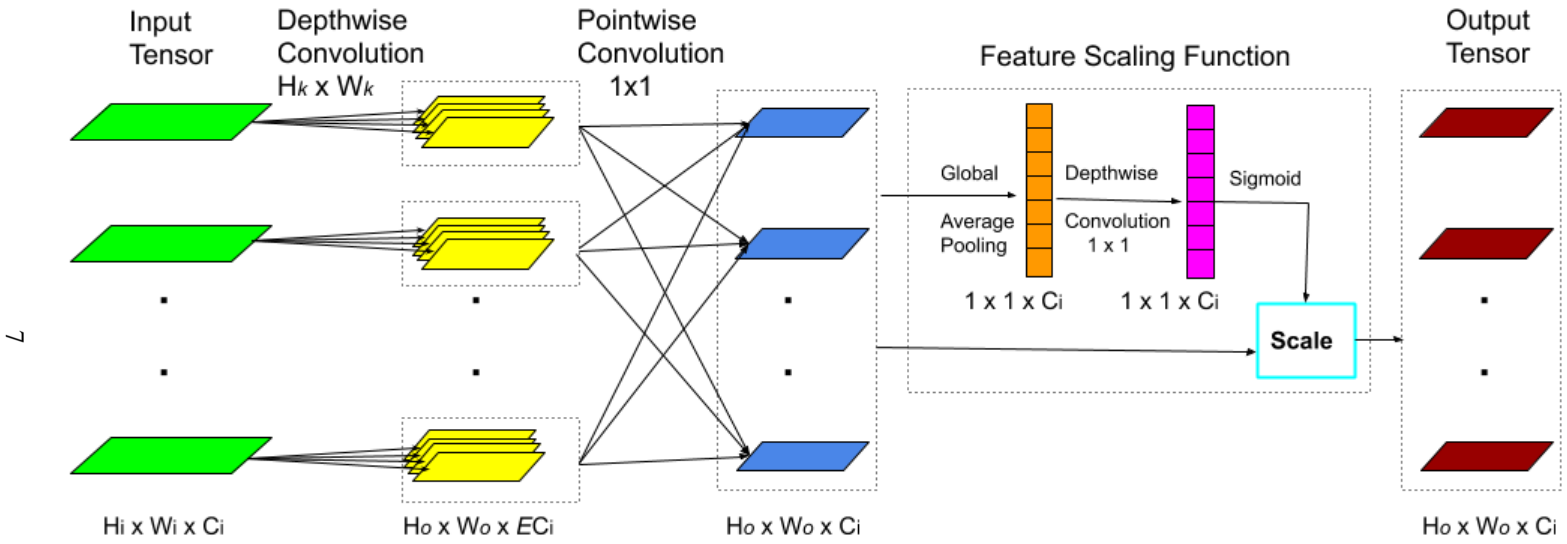

Figure 4: Conceptual details of our proposed EDS layer (best viewed in color).

increment in the number of channels helps to capture more features. However, this fact is never used within the pooling operation. Therefore, we have increased the number of channels by the factor *E* (termed as *expansion ratio*) at the place where the spatial dimension is reduced. This can be visualized in Figure 4, where the number of channels is increased from C_i to EC_i when depthwise convolutions is applied to the input tensor. However, the spatial dimension is decreased from $H_i \times W_i$ to $H_o \times W_o$.

In our experiments, we have used different values of expansion ratio ($E \in \{1,2,4,8\}$). As shown in Figure 4, corresponding to each channel in the input tensor (feature maps), *E* number of depthwise convolutional filters are applied which produces *E* different maps (channels) of size $H_o \times W_o$. Thus corresponding to an input tensor of size $H_i \times W_i \times C_i$, we have an output of size $H_o \times W_o \times EC_i$. After that, pointwise convolutions [23] are applied which are similar to standard convolution operations with 1×1 filter size. With an appropriate number of such 1×1 filters (C_i), the pointwise convolution operation produces an output tensor of size $H_o \times W_o \times C_i$ (channel downsampling).

The second part of our EDS layer is feature scaling. In the pooling operation, the aim is to discard unnecessary features and to keep only important features. Based on this principle, our scaling function scales up important channels/feature-maps and scales down the unimportant channels/feature-maps. This scaling is done on the feature maps, which is end-to-end trainable.

Why is *feature scaling* required? The intuition behind doing channel-wise feature scaling is that we also want a sense of feature selection across the channels. In traditional feature pooling operations, feature selection and pooling are done across the spatial dimension only. In our case, first spatial pooling is done by depthwise separable convolutions (stride> 1) with *expansion ratio E* followed by channel downsampling. The second part of our EDS layer (feature scaling) does a kind of feature selection across the channels.

In this feature scaling operation, as shown in the Figure 4, first global average pooling (GAP) is applied on the tensor of size $H_o \times W_o \times C_i$ which produces a vector of size $1 \times 1 \times C_i$. This vector is then convolved by depthwise 1×1 convolution followed by batch normalization layer. The resulting output is then operated by the sigmoid

activation operator to make the values in between 0 and 1. Based on these C_i values (in between 0 and 1), each channel is scaled (up/down) by simple multiplication. The final output of size $H_o \times W_o \times C_i$ is obtained as shown in the Figure 4. Therefore, we perform a learning based feature scaling to make feature selection across the channels in pooling operation to highlight the most relevant feature-maps. In section 5, we present an ablation on feature scaling to show it's significance.

3.2. Complexity Analysis

Pooling layers only constitute a small part of a deep CNN. Therefore, the contribution of pooling layers parameters will be small in the total model parameters. Similarly, the contribution of pooling layers FLOPS will be small in the total model FLOPS. However, we are providing the number of FLOPS and parameters for the proposed EDS layer.

3.2.1. FLOPS (Computational Complexity) calculation for the proposed EDS layer

Floating point operations per second (FLOPS) is a measure to estimate the computational complexity. The number of FLOPS for our EDS layer will be the summation of the number of FLOPS in depthwise convolution, pointwise convolution, and feature scaling. Let us say the size of input tensor is $H_i \times W_i \times C_i$, and after pooling, the output tensor size is $H_o \times W_o \times C_i$, where the height and width are reduced, but the number of channels remains the same.

For our proposed EDS layer, a depthwise convolutional filters ($H_k \times W_k$ kernel size) with expansion ratio E are applied, therefore the total number of FLOPS ($FLOPS_T$) can be calculated as follow:

$$FLOPS_T = FLOPS_{DW} + FLOPS_{PW} + FLOPS_R \tag{1}$$

$FLOPS_{DW}$ is the number of FLOPS in depthwise convolutions, which can be calculated as follow:

$$FLOPS_{DW} = H_k W_k H_o W_o C_i E \tag{2}$$

$FLOPS_{PW}$ is the number of FLOPS in pointwise convolutions, which can be calculated as follow:

$$FLOPS_{PW} = C_i E H_o W_o C_i \tag{3}$$

$FLOPS_R$ is the number of FLOPS in the feature scaling component, which can be calculated as follow:

$$FLOPS_R = C_i \tag{4}$$

Note that, the computational complexity of the sigmoid and scaling module is negligible with respect to the other modules and $FLOPS_R << (FLOPS_{DW} + FLOPS_{PW})$. Therefore, the total number of FLOPS ($FLOPS_T$) can be approximated as follow:

$$FLOPS_T \approx FLOPS_{DW} + FLOPS_{PW} \tag{5}$$

Using Equations 5, $FLOPS_T$ can be calculated as follow:

$$FLOPS_T = (H_k W_k E + C_i E) H_o W_o C_i \tag{6}$$

3.2.2. FLOPS comparison with stride pooling

The number of FLOPS for the stride pooling (strided convolutions with stride > 1) would be $H_k W_k C_i H_o W_o C_i$. The ratio of number of FLOPS in our EDS layer and the number of FLOPS in stride pooling would be as follow :

$$\frac{FLOPS_{Ours}}{FLOPS_{stride}} = \frac{(H_k W_k E + C_i E)}{H_k W_k C_i} \tag{7}$$

In the practical implementation, the depthwise convolutional filters with 3×3 kernel size is chosen ($H_k = W_k = 3$).

For depthwise convolutional filters (3×3 kernel size) with expansion ratio E = 4, equation 7 would be:

$$\frac{FLOPS_{Ours}}{FLOPS_{stride}} = \frac{4}{9} + \frac{36}{9Ci} \approx \frac{4}{9} \tag{8}$$

The ratio is approximated to $\frac{4}{9}$ as $\frac{36}{9Ci} << \frac{4}{9}$, when number of channels $C_i \in \{128, 512, 1024, 2048\}$.

Similarly, for depthwise convolutional filters (3×3 kernel size) with expansion ratio E = 8, equation 7 would be:

$$\frac{FLOPS_{Ours}}{FLOPS_{stride}} = \frac{8}{9} + \frac{72}{9Ci} \approx \frac{8}{9} \tag{9}$$

It can be concluded that the number of FLOPS in our method is not greater than the number of FLOPS in stride pooling. Particularly it is much lesser for $E = 4$. But the performance of our EDS layer is superior to stride pooling, which is discussed in the results section. Also, In section 5.1 it is shown that expansion ratio $E = 4$ and $E = 8$ performed comparably but EDS layer with $E = 4$ has approximately 55% lesser FLOPS than stride pooling.

3.2.3. Parameters comparison with stride pooling

The number of parameters for the stride pooling (strided convolutions with stride $>$ 1) layer would be $H_k W_k C_i C_i$. The total number of parameters in our EDS layer would be equal to $H_k W_k C_i E$ (depthwise convolutions) + $C_i E C_i$ (pointwise convolutions) + C_i (feature scaling component). The ratio of the number of parameters in our EDS layer and the number of parameters in stride pooling would be as follows :

$$\frac{\#PARAM_{Ours}}{\#PARAM_{stride}} = \frac{H_k W_k E + C_i E + 1}{H_k W_k C_i} \tag{10}$$

For depthwise convolutional filters (3×3 kernel size) with expansion ratio E = 4, equation 10 would be:

$$\frac{\#PARAM_{Ours}}{\#PARAM_{stride}} = \frac{37 + 4C_i}{9C_i} \tag{11}$$

The values of the above ratio are equal to 0.5, 0.45 when the number of channels (C_i) is 64, 512 respectively. Therefore, in the setting mentioned above, the number of parameters in the EDS layer would be at least 50% less than stride pooling.

4. Experiments

The proposed approach is tested with various CNN architectures by replacing their pooling methods with our proposed EDS layer. Network architectures such as VGG-

16 [25], ResNet-56, ResNet-50 [26], WideResNet-28-10, Wide-ResNet-18-2 [27], MobileNet [28], and Faster R-CNN [29] are used. Datasets such as CIFAR [30], ImageNet [31], and MS-COCO [32] are used for the experiments. Different network architectures use one or a combination of different pooling methods such as max pooling, average pooling, or stride pooling. In EDS, expansion ratio $E \in \{1,2,4,8\}$ and depthwise convolutional filters of 3×3 kernel size with stride 2 are taken in all experiments.

4.1. Image Classification on CIFAR-10

CIFAR-10 dataset [30] designed for image classification problem which consists of 60,000 colour images with 10 classes. We have used 50,000 images for the training and 10,000 images for the testing. We have done extensive number of experiments with CIFAR-10 dataset using various network architectures such as VGG-16, ResNet-56, MobileNet, and WideResNet-28-10 [25, 26, 28, 27].

For optimization, Stochastic Gradient Descent (SGD) [33] is used with momentum 0.9 and 128 minibatch size. Initially, the learning rate is set to 0.1 and is decreased by a factor of 5 every 50 epochs. The models are trained from scratch for 250 epochs. We set weight decay to 0.0005 for model parameters. The standard data augmentation strategy of random crop and horizontal flip is used.

VGG-16 uses five 2×2 max pooling layers in its architectures. We replaced all five max pool layers with our proposed EDS and other pooling methods. As shown in Table 1, EDS (E = 4) outperforms all other pooling methods by a significant margin.

ResNet is the first network architecture that uses skip connections to train deeper networks with layers of more than a hundred. Otherwise, deeper networks are hard to train due to the vanishing gradient problem. ResNet-56 has two stride pool layers with stride value two, where pooling happens (32×32 to 16×16 and 16×16 to 8×8). We replace both of these pooling layers with our proposed EDS and other pooling methods while keeping all other parameters same. Similarly, we replace stride pooling used in Wide-ResNet-28(depth)-10(widen) with our EDS and other pooling methods. As shown in Table 2 and 3, we achieve a significant accuracy gain by just using EDS (E = 4) pooling at two layers.

MobileNet is a computationally efficient CNN architecture design for mobile and

Table 1: Comparison of different models and pooling methods on the CIFAR-10 dataset for VGG-16.

Model	Pooling Method	Accuracy
VGG-16 (Baseline)	Max	93.6
VGG-16 (Baseline)	Stride Pool	93.0
VGG-16 (Baseline)	Average	93.8
VGG-16 (Baseline)	Mixed	93.7
VGG-16 (Baseline)	Stochastic	93.4
VGG-16 (Ours)	EDS (E = 1)	93.9
VGG-16 (Ours)	EDS (E = 2)	94.1
VGG-16 (Ours)	**EDS (E = 4)**	**94.4**
VGG-16 (Ours)	EDS (E = 8)	94.2

Table 2: Comparison of different models and pooling methods on the CIFAR-10 dataset for ResNet-56.

Model	Pooling Method	Accuracy
ResNet-56 (Baseline)	Max	93.2
ResNet-56 (Baseline)	Stride Pool	93.5
ResNet-56 (Baseline)	Average	93.4
ResNet-56 (Baseline)	Mixed	93.6
ResNet-56 (Baseline)	Stochastic	93.8
ResNet-56 (Ours)	EDS (E = 1)	94.2
ResNet-56 (Ours)	EDS (E = 2)	94.4
ResNet-56 (Ours)	**EDS (E = 4)**	**94.6**
ResNet-56 (Ours)	EDS (E = 8)	94.6

Table 3: Comparison of different models and pooling methods on the CIFAR-10 dataset for WideResNet-28-10.

Model	Pooling Method	Accuracy
WideResNet-28-10 (Baseline)	Max	95.4
WideResNet-28-10 (Baseline)	Stride Pool	95.6
WideResNet-28-10 (Baseline)	Average	95.5
WideResNet-28-10 (Baseline)	Mixed	95.7
WideResNet-28-10 (Baseline)	Stochastic	95.8
WideResNet-28-10 (Ours)	EDS (E = 1)	96.0
WideResNet-28-10 (Ours)	EDS (E = 2)	96.2
WideResNet-28-10 (Ours)	**EDS (E = 4)**	**96.3**
WideResNet-28-10 (Ours)	EDS (E = 8)	96.2

embedded systems. MobileNet uses stride pooling layers (stride = 2) with 3×3 kernel size. We replaced each of these stride pooling with our proposed EDS and other pooling methods while keeping all other parameters same. EDS (E = 4) outperforms all other pooling methods by a significant margin, as shown in Table 4.

4.2. Image Classification on ImageNet

ImageNet [31] is a large scale dataset mainly popularized by Large Scale Visual Recognition Challenge (ILSVRC) containing 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 errors on the validation set are reported. We experimented with our proposed EDS layers on the ImageNet dataset using ResNet-50 and Wide-ResNet-18-2 network architectures.

For these experiments, we perform standard data augmentation methods of random cropping to a size of 224×224 and random horizontal flipping. For optimization, Stochastic Gradient Descent (SGD) is used with momentum 0.9 and a mini-batch size

Table 4: Comparison of different models and pooling methods on the CIFAR-10 dataset for MobileNet.

Model	Pooling Method	Accuracy
MobileNet (Baseline)	Max	91.0
MobileNet (Baseline)	Stride Pool	91.1
MobileNet (Baseline)	Average	91.0
MobileNet (Baseline)	Mixed	91.1
MobileNet (Baseline)	Stochastic	91.2
MobileNet (Ours)	EDS (E = 1)	91.1
MobileNet (Ours)	EDS (E = 2)	91.6
MobileNet (Ours)	**EDS (E = 4)**	**92.3**
MobileNet (Ours)	EDS (E = 8)	92.3

of 256. Initially, the learning rate is set to 0.1 and is decreased by a factor of 10 every 30 epochs. The models are trained from scratch for 100 epochs.

We replace stride pooling (stride = 2) used in ResNet-50 and WideResNet-18(depth)-2(widen) [26, 27] with our propose EDS layers (3×3 kernel size), while keeping other settings same as [26, 27]. As shown in Table 5, ResNet-50 with our EDS layers now has accuracy comparable to ResNet-152 [21, 26] (with stride pooling). Results are shown in the Table 5 for ResNet-50 and Table 6 for WideResNet-18-2.

Our pooling approach outperforms various state-of-the-art approaches on ResNet-50 model over ImageNet dataset as shown in Table 5.

4.3. Object Detection on MS-COCO

The above set of experiments shows improvement in accuracy due to our proposed EDS layer in classification tasks over various popular datasets on various CNN architectures. In this section, we show that our proposed EDS layer not only perform well on image classification task but also for other tasks such as object detection.

Table 5: Experiments on ImageNet dataset for ResNet-50 (https://github.com/KaimingHe/deep-residual-networks) network architecture (Top-1 classification accuracy on the validation set is reported). We are using the same settings and hyperparameters as mention in [17] for a fair comparison. Bold values indicate the best results obtained by our pooling method in the comparison.

Model	Pooling Method	Accuracy
ResNet-50 (CVPR'16)	Stride Pool [26]	75.3
ResNet-50 (ECCV'18)	DFT-Pooling [12]	75.9
ResNet-50 (ICCV'17)	FBN [13]	76.0
ResNet-50 (ICCV'17)	SORT [14]	76.2
ResNet-50 (CVPR'18)	DPP [19]	76.6
ResNet-50 (ICCV'17)	MPN-COV [15]	77.3
ResNet-50 (ICCV'19)	GFGP with DPP [18]	77.4
ResNet-50 (ICCV'19)	GFGP with Gate [18]	77.8
ResNet-50 (CVPR'18)	iSORT-COV [16]	77.9
ResNet-50 (ICCV'19)	iSORT-COV+EP [17]	78.0
ResNet-50 (Ours)	**EDS (E = 4)**	**78.2**

MS-COCO dataset [32] is designed for object detection and segmentation task. It consists of around 80,000 training and 40,000 validation images. We use the MS-COCO object detection dataset with a Faster R-CNN object detector. In our experiment, Faster R-CNN uses ResNet-50 (with stride pooling layers) as a base architecture. We replace each of its stride pooling layers with our proposed EDS layer while keeping all other parameters and settings same as [21, 29, 26]. EDS (E = 4) outperforms baseline by a significant margin as shown in Table 7.

Table 6: Experiments on ImageNet dataset for WideResNet-18-2 [27] network architectures (Top-1 classification accuracy on the validation set are reported).

Model	Pooling Method	Accuracy
WideResNet-18-2 (Baseline)	Stride Pool	74.4
WideResNet-18-2 (Ours)	**EDS (E = 4)**	**75.3**

Table 7: Experiments on MS-COCO dataset for Faster R-CNN object detector with ResNet-50 as a base network [21, 29, 26].

Pooling Method	AP@IoU=0.5	AP@IoU=0.5:0.95
Stride Pool (Baseline)	45.2	25.1
EDS E = 4 (Ours)	**47.3**	**26.8**

5. Ablation Studies

We validate the proposed approach using extensive ablation studies. We perform ablation experiments to examine the effect of the expansion ratio and feature scaling.

5.1. Effect of Expansion Ratio

Our pooling method uses depthwise separable convolution operations, which is computationally much cheaper (in terms of parameters and FLOPS) than standard convolution operations. This gives us the flexibility to increase the number of channels/feature-maps in the intermediate step of pooling operation in the EDS layer (See Figure 1,4).

Each output channel/feature-map is a unique representation of the input tensor. Thus increasing the number of channels of an intermediate step in the spatial pooling operation gives a chance to preserve the more number of features at the place where a large number of spatial features are simply discarded.

The next question which comes in mind is: What should be the value of the hyper-parameter expansion ratio? We have experimented with expansion ratio values of 1,2,4 and 8. In section 3.2.1 through FLOPS calculation, we have demonstrated that our

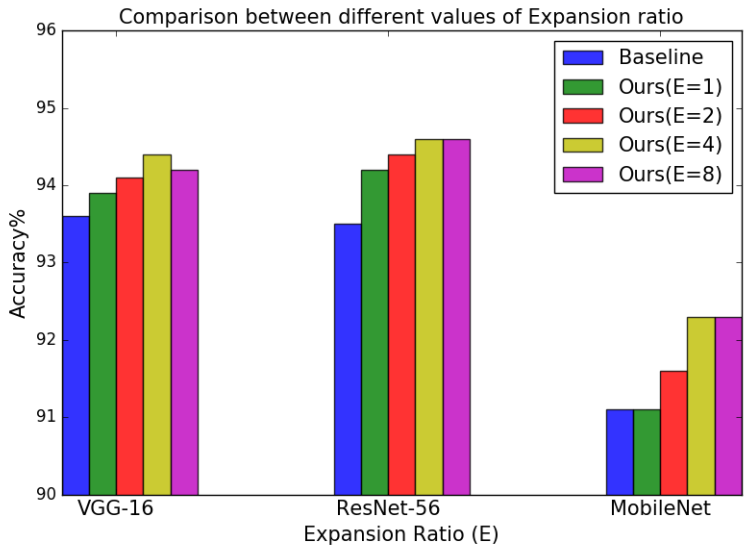

Figure 5: Effect of expansion ratio (E) on the performance of our EDS layers (best viewed in color).

pooling method with expansion ratio 4 is computationally much cheaper than stride pooling. Plots in Figure 5 summarizes the effect of different values of expansion ratio on the overall accuracy. The experiments use the CIFAR-10 dataset with three different network architectures VGG-16, ResNet-56, and MobileNet. In the case of VGG-16, EDS pooling with expansion ratio 4 has the best performance of accuracy 94.4 as compared to 94.2 with expansion ratio 8. The diminishes in performance in case of expansion ratio 8 is due to overfitting. In the case of ResNet-56 and MobileNet, experiments show that expansion ratios 4 and 8 perform equally good. From this, it can be observed that expansion ratio 4 is the right choice, and increasing the value of expansion ratio beyond four is not only computationally expansive but also doesn't improve the performance of CNN.

5.2. Ablation on Feature Scaling

Feature scaling has been used in our EDS layer to highlight the important channels/feature-maps. In feature scaling, all the channels are scaled by a value in the range between 0

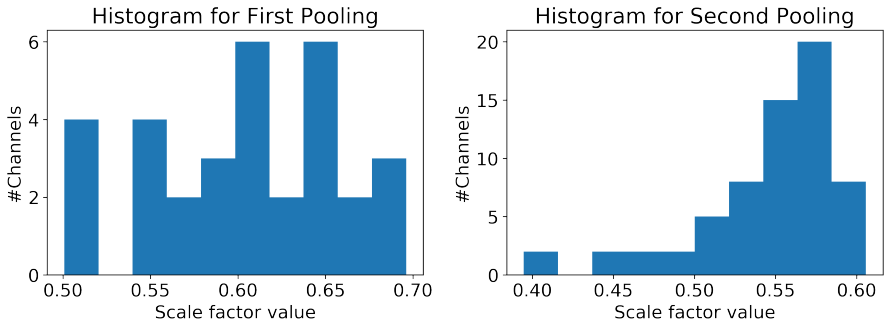

Figure 6: ResNet-56 on CIFAR-10 dataset: Histogram plots of the average scale factor value for different channels/feature-maps obtained on the whole test set (E = 4).

to 1. In this ablation, ResNet-56 architecture with the CIFAR-10 dataset is used. Our proposed EDS layer is applied at two places to downsample from 32×32 to 16×16 and 16×16 to 8×8 in ResNet-56. Figure 6 shows the histogram plots of average scale factor value obtained by different channels at the two different EDS layers (pooling layers) with E = 4 on the whole test set. It can be observed from Figure 6 that the feature scaling operation in our EDS layer is actually doing the scaling across channels.

Next, we show the efficacy of the feature scaling component in our EDS layer by two means. First, by a direct experiment over the CIFAR-10 dataset using ResNet-56 architecture. In one experiment, feature scaling is being used within the EDS layer, and in another experiment, feature scaling is absent in the EDS layer. The results of these two different runs are shown in Table 8. The accuracy results indicate the efficacy of the feature scaling component in our EDS layer. The improvement in accuracy due to scaling is significant as our EDS layer (pooling operation) is applied at only two places within the 56 layer network architecture. Whereas, the computational overhead due to feature scaling operation is insignificant, as can be seen by equation 4 in section 3.2.1.

Now we show the efficacy of the feature scaling component by an evaluation study. In the feature scaling operation, all the channels/feature-maps are scaled by a value in the range between 0 to 1. We drop some percentage of the channels/feature-maps after being scaled by the feature scaling component. Channels are dropped on different runs during our experiments from 5% to 40% in increments of 5% in each run.

Table 8: ResNet-56 on CIFAR-10: Experimental results of our proposed EDS layer with and without feature scaling.

Feature Scaling	Pooling Method	Accuracy
Absent (Baseline)	Stride Pool	93.5
Absent	EDS (E = 1)	93.4
Absent	EDS (E = 2)	93.8
Absent	EDS (E = 4)	94.1
Absent	EDS (E = 8)	94.1
Present (Ours)	EDS (E = 1)	94.2
Present (Ours)	EDS (E = 2)	94.4
Present (Ours)	**EDS (E = 4)**	**94.6**
Present (Ours)	EDS (E = 8)	94.6

The dropping of channels/feature-maps is done in two ways. In one setting, the most significant channels (high scale factor values) are dropped, and in another setting, the least significant channels (low scale factor values) are dropped. Figure 7 depicts the results of our evaluation study to show the efficacy of the learnable scaling function in our proposed EDS layer. It is observed from Figure 7 that performance is dropped lesser when least significant channels (low scale factor value) are dropped, and also the other way performance is dropped more when most significant channels (high scale factor value) are dropped. This shows that our proposed scaling function in the EDS layer can learn the importance of the channels/feature-maps. Feature scaling operation highlights the important channels/feature-maps by scaling them with high value and suppresses the less important channels/feature-maps by scaling them with low value. Hence feature scaling helps in increasing the overall performance of the CNNs by making feature selection across the channels.

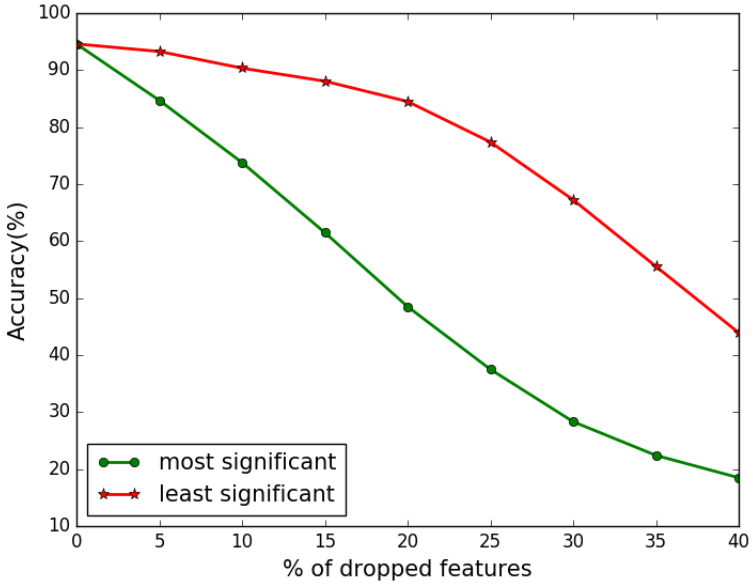

Figure 7: Results of an evaluation study for the feature scaling component use in our proposed EDS layer with E = 4 (best viewed in color).

5.3. Separable Convolution vs. Expansion Ratio

The proposed EDS pooling layer uses separable convolution, i.e., depthwise convolution followed by a pointwise convolution. Therefore one may think that the improvements are coming because of depthwise and pointwise convolutions. Therefore, to alleviate this concern, we also conduct experiments for different values of expansion ratio without using feature scaling, as shown in Table 8.

As shown in Table 8, EDS (E = 1) without feature scaling has accuracy similar to baseline. In both cases (with and without feature scaling), accuracy increase with the increase in E value. Therefore expansion ratio also plays a major role in the proposed EDS pooling layer.

5.4. Comparison with Heavier Models

It will be very logical to think that the additional parameters introduced by our modifications are the sole reason for the performance improvement. Pooling layers

Table 9: The table shows the comparison of different ResNet models on the CIFAR-10 dataset (**Acc**: Accuracy).

Models	Acc(%)	Params	FLOPS
ResNet-56 (Baseline)	93.5	0.856M	0.126G
ResNet-56 (E = 1)	94.2	0.883M	0.128G
ResNet-56 (E = 2)	94.4	0.888M	0.129G
ResNet-56 (E = 4)	**94.6**	**0.901M**	**0.130G**
ResNet-56 (E = 8)	94.6	0.926M	0.133G
ResNet-164 (Baseline)	94.4	1.712M	0.247G
ResNet-56_1.5x (Baseline)	94.6	1.922M	0.283G

only constitute a small portion of a deep CNN, and we use depthwise separable convolutions in the proposed EDS layer. Therefore, this will not increase the model parameters significantly, as shown in Table 9. However, in order to alleviate this concern, we also compare our results with different versions of networks having increased depth or width. This is done in order to compare our approach to heavier networks that have higher parameters/computation than ours.

ResNet-56_1.5x is the wider version of ResNet-56, where we increase the number of filters to 1.5 times from each layer. ResNet-164 is the deeper version of ResNet. As shown in Table 9, the performance of ResNet-56 (E = 4) is similar to ResNet-56_1.5x and ResNet-164 while having almost half parameters and FLOPS than ResNet-56_1.5x and ResNet-164.

Therefore from Table 9, we can conclude that our approach improves the performance of deep CNNs without significantly increasing the number of parameters. Our approach shows similar/better performance than a heavier version of models with considerably higher parameters and computation.

Table 10: The table shows the results for VGG-16 on CIFAR-10 dataset in different setups (**Acc**: Accuracy, and **TD**: Training Data).

Model	Pooling Method	TD	(Train-Test) Acc(%)
VGG-16	Max	100%	6.1
VGG-16	EDS (E = 4)	100%	**5.2**
VGG-16	Max	20%	15.6
VGG-16	EDS (E = 4)	20%	**14.1**

5.5. Effect of EDS pooling on Over-fitting

We perform experiments for VGG-16 on the CIFAR-10 dataset to test the effect of EDS pooling on overfitting. We have conducted these experiments in two different settings. In the first setting, the experiment is carried out over a full dataset (100% of the training set). In the second setting, the experiment is carried out over 20% of the training set.

We use the difference between the training and test accuracy as a measure of over-fitting. Higher the difference, higher is the over-fitting. From Table 10, it is clear that the proposed EDS layer is less prone to overfitting problem than the max pooling layer since the gap between training and test accuracy is always lower.

6. Conclusion

In this work, we propose an EDS layer to substitute feature pooling layer in deep CNNs. Added functionality in our pooling, such as expansion ratio and feature scaling for feature selection across the channels, helps to improve the performance while not adding a significant computational overhead as compared to stride pooling. ResNet-50 with our proposed EDS layer has performed comparably to ResNet-152 with stride pooling on the ImageNet dataset. We show that our proposed approach works well for various networks not only for classification but also for detection. The proposed EDS

layer is demonstrated to improve the performance on various models such as VGG, ResNet, WideResNet, MobileNet, and Faster R-CNN.

References

[1] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, `http://www.deeplearningbook.org`.

[2] Y.-L. Boureau, J. Ponce, Y. LeCun, A theoretical analysis of feature pooling in visual recognition, in: Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 111–118.

[3] D. Scherer, A. Müller, S. Behnke, Evaluation of pooling operations in convolutional architectures for object recognition, in: Artificial Neural Networks–ICANN 2010, Springer, 2010, pp. 92–101.

[4] D. Yu, H. Wang, P. Chen, Z. Wei, Mixed pooling for convolutional neural networks, in: International Conference on Rough Sets and Knowledge Technology, Springer, 2014, pp. 364–375.

[5] J. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: The all convolutional net, in: The International Conference on Learning Representations (ICLR workshop track), 2015.

[6] B. Graham, Fractional max-pooling, in: arXiv preprint arXiv:1412.6071, 2014.

[7] M. D. Zeiler, R. Fergus, Stochastic pooling for regularization of deep convolutional neural networks, in: arXiv preprint arXiv:1301.3557, 2013.

[8] J. A. Parker, R. V. Kenyon, D. E. Troxel, Comparison of interpolating methods for image resampling, IEEE Transactions on medical imaging 2 (1) (1983) 31–39.

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

[10] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, 2015, pp. 448–456.

[11] M. A. Nielsen, Neural networks and deep learning, Vol. 25, Determination press USA, 2015.

[12] J. Ryu, M.-H. Yang, J. Lim, Dft-based transformation invariant pooling layer for visual classification, in: European Conference on Computer Vision, 2018.

[13] Y. Li, N. Wang, J. Liu, X. Hou, Factorized bilinear models for image recognition, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2079–2087.

[14] Y. Wang, L. Xie, C. Liu, S. Qiao, Y. Zhang, W. Zhang, Q. Tian, A. Yuille, Sort: Second-order response transform for visual recognition, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1359–1368.

[15] P. Li, J. Xie, Q. Wang, W. Zuo, Is second-order information helpful for large-scale visual recognition?, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2070–2078.

[16] P. Li, J. Xie, Q. Wang, Z. Gao, Towards faster training of global covariance pooling networks by iterative matrix square root normalization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 947–955.

[17] W. Wan, J. Chen, T. Li, Y. Huang, J. Tian, C. Yu, Y. Xue, Information entropy based feature pooling for convolutional neural networks, in: The IEEE International Conference on Computer Vision (ICCV), 2019.

[18] T. Kobayashi, Global feature guided local pooling, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 3365–3374.

[19] F. Saeedan, N. Weber, M. Goesele, S. Roth, Detail-preserving pooling in deep networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9108–9116.

[20] V. V. Romanuke, Appropriate number of standard 2× 2 max pooling layers and their allocation in convolutional neural networks for diverse and heterogeneous datasets, Information Technology and Management Science 20 (1) (2017) 12–19.

[21] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7132–7141.

[22] P. Singh, P. Mazumder, V. P. Namboodiri, Accuracy booster: Performance boosting using feature map re-calibration, in: arXiv preprint arXiv:1903.04407, 2019.

[23] J. Guo, Y. Li, W. Lin, Y. Chen, J. Li, Network decoupling: From regular to depthwise separable convolutions, in: The British Machine Vision Conference (BMVC), 2018.

[24] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258.

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

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

[27] S. Zagoruyko, N. Komodakis, Wide residual networks, in: British Machine Vision Conference 2016, British Machine Vision Association, 2016.

[28] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, in: arXiv preprint arXiv:1704.04861, 2017.

[29] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in neural information processing systems (NIPS), 2015, pp. 91–99.

[30] A. Krizhevsky, Learning multiple layers of features from tiny images, Tech. rep., Citeseer (2009).

[31] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

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

[33] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: International conference on machine learning, 2013, pp. 1139–1147.