Weight Excitation: Built-in Attention Mechanisms in Convolutional Neural Networks

Niamul Quader(✉), Md Mafijul Islam Bhuiyan, Juwei Lu, Peng Dai, and Wei Li

Huawei Noah's Ark Lab, Montreal, Canada
{niamul.quader1,juwei.lu,peng.dai,wei.li.crc}@huawei.com, mbhuiyan@ualberta.ca

Abstract. We propose novel approaches for simultaneously identifying important weights of a convolutional neural network (ConvNet) and providing more attention to the important weights during training. More formally, we identify two characteristics of a weight, its magnitude and its location, which can be linked with the importance of the weight. By targeting these characteristics of a weight during training, we develop two separate weight excitation (WE) mechanisms via weight reparameterization-based backpropagation modifications. We demonstrate significant improvements over popular baseline ConvNets on multiple computer vision applications using WE (e.g. 1.3% accuracy improvement over ResNet50 baseline on ImageNet image classification, etc.). These improvements come at no extra computational cost or ConvNet structural change during inference. Additionally, including WE methods in a convolution block is straightforward, requiring few lines of extra code. Lastly, WE mechanisms can provide complementary benefits when used with external attention mechanisms such as the popular Squeeze-and-Excitation attention block.

Keywords: Convolutional neural network · Convolution filter weights · Weight reparameterization · Attention mechanism

1 Introduction

Convolutional neural networks (ConvNets) are extremely powerful in analyzing visual imagery and have brought tremendous success in many computer vision tasks, e.g. image classification [9,29,40], video action and gesture recognition [14,34], etc. A ConvNet is made up of convolution blocks, each of which consists of a number of learnable parameters including convolution filter weights. The effectiveness of training these weights in convolution blocks can depend on the

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58577-8_6) contains supplementary material, which is available to authorized users.

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12375, pp. 87–103, 2020.
https://doi.org/10.1007/978-3-030-58577-8_6

ConvNet design, the training data available, choice of optimizer and many other factors [10,15,19,26,37].

As a ConvNet is trained, some of these weights become less important than others [18], and removing the less important weights in a ConvNet can often lead to compressed ConvNets without compromising on accuracy considerably [6–8,18,22,25]. Recently, Frankle & Carbin [4] hypothesized that an iteratively pruned network can be retrained from scratch to similar accuracy using its initial ConvNet parameters, suggesting that some weights may, in fact, start as being more important than others after random initialization of weights.

Assuming that some convolution kernel weights of a ConvNet are more important than others during the ConvNet training, we hypothesize that providing more attention to optimizing the more important weights during ConvNet training can improve ConvNet performances without the addition of computation costs or modifications in ConvNet structure during inference. To test this hypothesis, we implement two novel schemes that simultaneously identify the importance of weights and provide more attention to training the more important weights, and investigate their effects on ConvNet performances. The highlights of our contributions in this paper are:

1. We investigate two characteristics of weights (i.e., their magnitude and location in a ConvNet) that could provide indications on how important each of the weights is. Note that 'location' of a weight is defined by the layer the weight is in, and the input and output channel that the weight is connecting.
2. We propose two novel weight reparameterization mechanisms that target the identified characteristics of weights in (1) and modify the weights in a way that enables more attention to optimizing the more important weights via adjusting the backpropagated gradients. We broadly term such training mechanisms as Weight Excitation (WE)-based training.
3. We conduct several experiments on image classification (ImageNet [30], CIFAR100 [16]), semantic segmentation (PASCAL VOC [3], Cityscapes [2]), gesture recognition (Jester [33]) and action recognition (mini Kinetics [41]). The ConvNets studied for these tasks have varying types of convolutions (e.g. 2D convolution, 3D convolution, shift-based 3D convolution [20], etc.). Additionally, we experiment with ConvNets of different structures and model sizes, hyperparameter settings, normalization approaches, optimizers, and schedulers.

In all our experiments, WE-based training demonstrates considerable improvements over popular baseline ConvNets. The accuracy gains with using WE can be similar to that of adding the popular squeeze-and-excitation (SE) based attention to a baseline ConvNet [12]. Notably, in contrast to SE, WE-based training does not require any ConvNet structural changes or added costs at inference. Additionally, WE-based training complements prior attention mechanisms [12,27,35] in terms of improving accuracy. Finally, WE-based training provides considerably better acceleration and accuracy gains compared to the other weight reparameterization approaches [24,28,31].

2 Related Works

Identifying Important ConvNet Parameters: Identifying important parameters and then removing unimportant parameters is common in ConvNet pruning approaches [6–8,18,25]. Two common criteria for identifying important ConvNet parameters are - (1) higher minimal increase in training error when a parameter is removed corresponding to the parameter being of higher importance [8,18], and (2) higher magnitude parameter corresponding to higher importance (e.g. [22,25]). A less explored characteristic of a weight that can be linked to its importance is its location in a ConvNet. One evidence that the location of a weight could be important is in an ablation study in [12]. The ablation study shows that earlier layers in SE-added ConvNets tend to put more attention or importance on some activation map output channels than others regardless of the inputs to those layers [12], suggesting that the convolution filter weights involved in calculating those activation map output channels are more important than others. In this work, we use novel weight reparameterization approaches to provide more attention to training important weights identified using their magnitude and location characteristics.

Applying Attention on Weights: In prior works, attention mechanisms in ConvNets have been applied to activation maps [11,12,27,35]. Such activation map-based attention methods do not have fine-grained control on providing more attention to a particular weight in a convolution filter kernel - for example, in SE ConvNets [12], during a backpropagation through an excited activation map channel, attention is provided to all the weights that contributed to generating that activation map channel. In contrast, weight reparameterization-based WE methods can provide fine-grained weight-wise attention to the weights during backpropagation.

3 Technical Approach

We begin by investigating the relationship between the importance of a weight and its magnitude/location properties (Fig. 1a, Fig. 2) (Sect. 3.1). Next, we implement a weight reparameterization approach that simultaneously identifies important weights via location characteristics and provides more attention to them (Fig. 1c, Fig. 3a) (Sect. 3.2). We call this method location-based WE (LWE). We then implement another WE method that targets magnitude property of a weight for identifying its importance (Fig. 1b, Fig. 3b) (Sect. 3.3). We call this second method magnitude-based WE (MWE).

3.1 Investigating the Importance of Weights

To investigate the importance of weights, we systematically decimate the effects of weights (i.e., by making them zeroes) that exhibit certain characteristics in a ConvNet and investigate the resulting decrease in ConvNet performance. This approach is similar to finding the importance of a ConvNet parameter in [8,18].

We study two separate characteristics of a weight in a ConvNet - the magnitude and the location of a weight. For these investigations, we use a ResNet50 [9] ConvNet pretrained on the ImageNet [30] since it is popularly and variously used in computer vision research (e.g. [12,21]).

Weight Magnitude and Importance: To study this relationship, we do the following: (1) sort weights in each convolution layer in ascending order of absolute values of weights, (2) make the bth percentile position ($b \in \{1,2,3,4, \ldots, 100\}$) weight of the sorted absolute weights of each layer zero and record the decrease in performance D (Fig. 1a). Figure 1a shows our findings on decrease in performance D when a weight at position b is zeroed. For higher b, D progressively increases (Fig. 1a), suggesting that importance increases with magnitude.

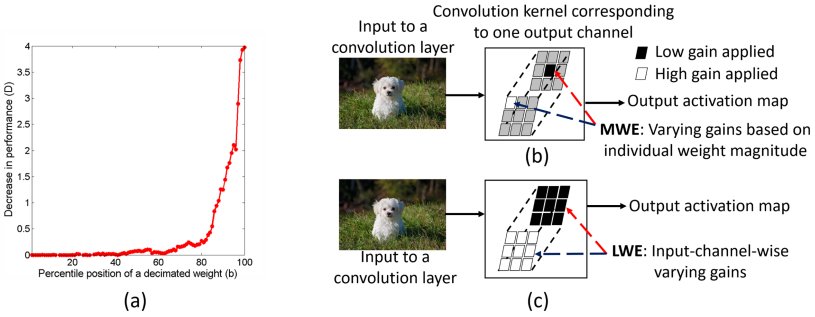

Fig. 1. (a) Higher magnitude weights correspond to larger D, and therefore are more important. To focus on optimizing more important weights, higher gain is provided to the more important weights via weight reparameterization. This reparameterization can be based on magnitude-importance relationship of each weight (b) (Sect. 3.3) or can be based on location-importance relationship of all weights along each of the input channels (c) (Sect. 3.2).

Weight Location and Importance: To study this relationship, we do the following: (1) select all the L convolutional blocks having 3×3 filters in the pretrained ResNet50 ($L = 16$ for ResNet50), (2) for each selected 3×3 convolution block (e.g. Fig. 3) (S_l, $l \in \{1, \ldots, L\}$), select N_1 output channels ($S_{l,O_j}, j \in \{1, \ldots, N_1\}$) equidistant from each other, (3) for each S_{l,O_j}, select N_2 input channels ($S_{l,O_j,I_i}, i \in \{1, \ldots, N_2\}$) equidistant from each other, and then (4) zero each of the weights in S_{l,O_j,I_i} and record the decrease in performance $D_{S_{l,O_j,I_i}}$. In this paper, we consider $N_1 = N_2 = 5$. This results in 5×5 D values for each layer of the pretrained ResNet50 (Fig. 2). Higher D values correspond to 3×3 filters that are more important to retaining the performances of the ResNet50. We notice larger fluctuations in D in the early layers, and the fluctuations in D tend to fade away in deeper layers, though some variations in D still remain at the deepest layer (Fig. 2). This suggests that the importance of weights in different locations of a ConvNet can be different and that the difference in importance is more obvious in the ConvNet's earlier layers.

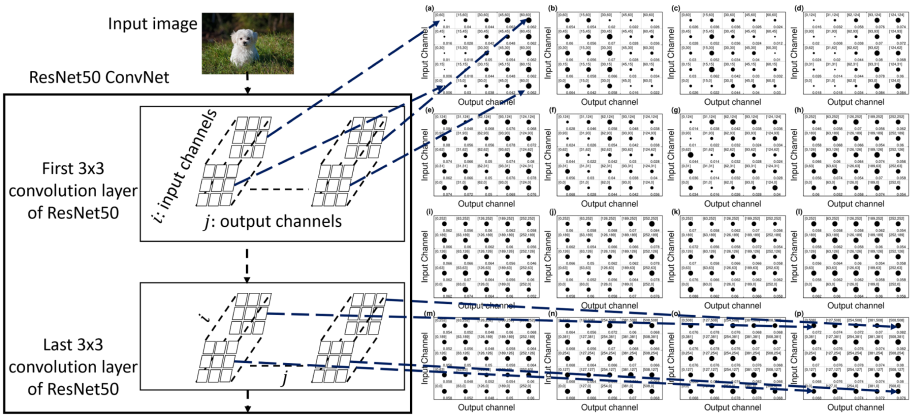

Fig. 2. Decrease in performance map (or importance map) of sampled 3×3 convolution kernels in various layers of the pretrained ResNet50. Here, (a) corresponds to the first convolution layer in the ConvNet that consists of 3×3 convolution kernels, and (b) to (p) correspond to progressively deeper layers containing 3×3 kernels. Early layers exhibit larger variation in 3×3 kernel importance than deeper layers.

3.2 Location-Based Weight Excitation

As we see in Sect. 3.1, importance of weight kernels can vary depending on where they are located (Fig. 2). Therefore, for weights W having dimensions $Out \times In \times h \times w$ in a convolutional layer that connects In input channels to Out output channels (e.g. Fig. 3a where $h = 3$ and $w = 3$), importance of each $h \times w$ weight kernels can vary. To provide separate attention to each of the $Out \times In$ weight kernels, an $Out \times In$-sized location-importance multiplier map ($m \in \mathbb{R} : m \in [0, 1]$) can be multiplied to each of the $h \times w$ weight kernels separately. This will result in larger backpropagated gradients through weight kernels that were multiplied with larger m values. To find the multiplier map m, a simple approach could be to instantiate $Out \times In$-sized learnable parameters that are dynamically trained during training; however, this results in drastically increased ConvNet parameters (e.g. around 60% increase in parameters for ResNet50 [9]). Instead, we propose to use a simple subnetwork that takes in $In \times h \times w$ weights and generates In sized importance map values. The same subnetwork is re-used for all the Out pathways in W to generate m. While this subnetwork could have many different structures, we chose the SE block [12] for its efficient design - a small artificial neural network comprising of only two fully connected layers and some activation functions. The subnetwork (Fig. 3a) is defined as:

$$m_j = A_2(FC_2(A_1(FC_1(Avg(W_j))))) \tag{1}$$

where W_j is the weights across the jth output channel [28], Avg is an average pooling operation that averages every $h \times w$ to one averaged value, A_1 and A_2 are two activation functions (instantiated as ReLU and Sigmoid, respectively), FC_1

and FC_2 are two fully connected layers. To extend this formulation to 1D and 3D convolution layer, Avg is modified to average over w for a 1D convolution and over $t \times h \times w$ in a 3D convolution. Finally, m_j is expanded by value replication to a $In \times h \times w$, and multiplied elementwise to W_j (Fig. 3a). Repeating this across all output channels result in LWE transformed weights W_{LWE} (Fig. 3a).

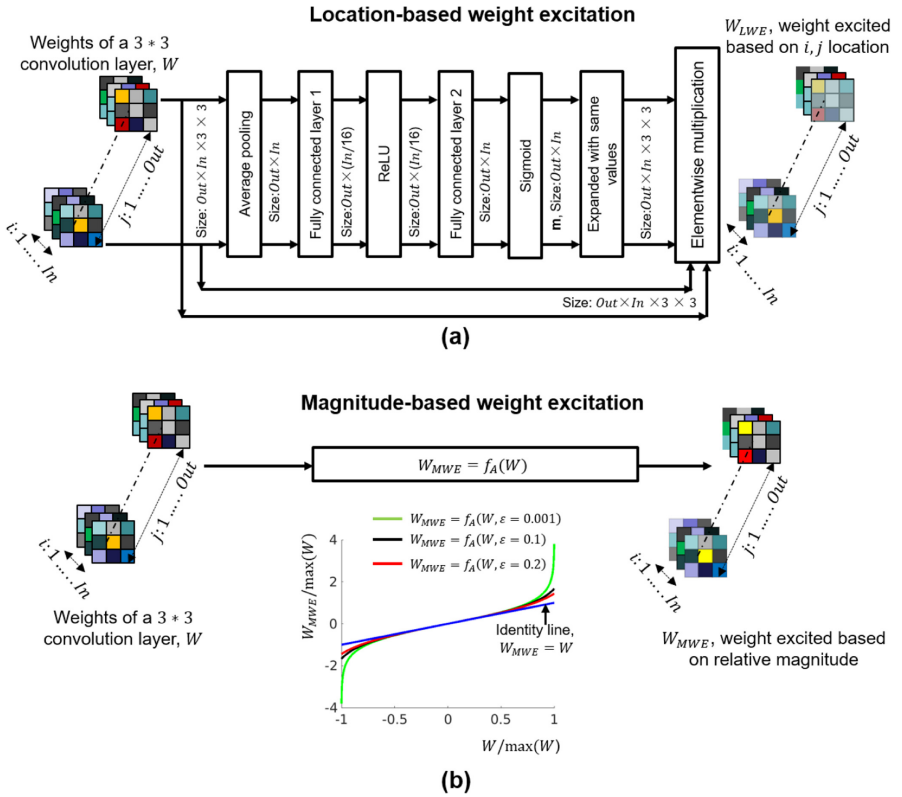

Fig. 3. (a) Outline of our LWE method that identifies location-wise importance map m, and then provides more attention to important locations via elementwise multiplication. (b) Outline of our MWE method that uses the f_A activation function to provide more attention to higher magnitude weights. This behavior is seen for all f_A under different ϵ_A values, with lower ϵ_A increasing this effect of attention. As ϵ_A value is increased, f_A behaves closer to an identity line (blue line). (Color figure online)

3.3 Magnitude-Based Weight Excitation

Our magnitude-based weight mechanism (MWE) is a novel activation function $f_A(\omega)$ that takes in a weight ω of W, and provides relatively larger gains to ω if it has a relatively larger magnitude among all weights of W. Additionally,

$f_A(\omega)$ is differentiable and avoids vanishing and exploding gradient problems. We define $f_A(\omega)$ which transforms ω to ω_{MWE} as:

$$\omega_{MWE} = f_A(W) = M_A \times 0.5 \times ln\frac{1+\omega/M_A}{1-\omega/M_A}, \tag{2}$$

where $M_A = (1+\epsilon_A) \times M$, M is the maximum magnitude of a weight in W and ϵ_A is a hyperparameter with a small value greater than 0 (e.g. $0 < \epsilon_A < 0.2$). The gradient for ω (i.e., ∇_ω) becomes:

$$\nabla_\omega = {M_A}^2/({M_A}^2 - \omega^2) \times \nabla_{\omega_{MWE}}. \tag{3}$$

For $\epsilon_A > 0$, $\omega < M_A$ and the denominator of ${M_A}^2/({M_A}^2 - \omega^2)$ remains numerically stable. Minimum value of ${M_A}^2/({M_A}^2 - \omega^2)$ is 1 when ω has a zero magnitude. As magnitude of ω increases, ${M_A}^2/({M_A}^2 - \omega^2)$ progressively increases upto a maximum value defined by ϵ_A (e.g. maximum of 5.76 when $\epsilon_A = 0.1$) for ω that has the highest magnitude in W. Thus, more attention or gain is provided to backpropagated gradients corresponding to higher magnitude weights. For larger values of ϵ_A, f_A becomes closer to an identity line, whereas smaller values of ϵ_A increases the level of attention provided via MWE (Fig. 3b).

Note that, before feeding W as input for LWE or MWE, we can normalize W similar to normalizing an input before being fed to a neural network. We do this by standardizing W across each j^{th} output channel similar to [28]. Using normalized weight W_n instead of W as input results in additional small performance improvements on most ConvNets.

4 Experiments and Results

4.1 Experimental Setup

We test effectiveness of our proposed WE methods on image classification (ImageNet [30], CIFAR100 [16]), and semantic segmentation (PASCAL VOC [3]). Additionally, we test WE methods on a standard 3D convolution-based ConvNet [34] for action recognition on the Mini-Kinetics dataset [41], and on a temporal shift convolution-based 3D ConvNet [20] for gesture recognition on Jester dataset [33]. In all experiments, we modify all convolution blocks such that WE methods are added to the convolution blocks during training only, so no ConvNet structural change or additional cost is required at inference. We use normalized weights as inputs for WE methods, except when WE methods are used on depthwise convolutions. This exception is because ConvNet performances deteriorate when weights are normalized in depthwise convolutions [39].

4.2 ImageNet Image Classification

ImageNet is a large-scale standard dataset containing around 1.28 million training and 50K validation images [30]. For experiments on ImageNet, we used a

standard data augmentation and training recipe as described in [42]. We adopt the data augmentation used in [9], and adopt the standard single-crop evaluation for testing [12,35]. We used a batch size of 128 for ResNet152 [9] and a batch size of 256 for all other ConvNets, and optimized with stochastic gradient descent (SGD) with backpropagation [17] having momentum 0.9 and a decay of 10^{-4}. Each of the ConvNets were trained for 100 epochs. Learning rate was initialized at 0.1 and reduced by a factor of 10^{-1} at 30^{th}, 60^{th} and 90^{th} epochs. We used MWE with $\epsilon_A = 0.1$ in all convolution blocks. Since our LWE outperforms our MWE method, we primarily investigate efficacy of LWE on different ConvNets (Table 1).

Additionally, we conduct a series of ablation studies on the ImageNet image classification dataset [30]: (1) compare performance gains achieved with LWE against performance gains with activation map-based attention (or external attention) based approaches [12,27,35], (2) compare convergence speed with LWE-based training against convergence speeds with popular weight normalizing reparameterization approaches [24,28,31], (3) study effectiveness of LWE for different normalization approaches, optimizers and schedulers, (4) compare utility of LWE against that of MWE, (5) check validity of learned attention multiplier in LWE, and (6) study hyperparameter sensitivity.

WE Improves Accuracy on Baseline ConvNets: LWE-based training on popular baseline ConvNets of varying parameter sizes (e.g. MobileNetV2 [32], ResNet50 [9], ResNeXt50 (32×4d) [40], ResNet152-SE [12], Wide ResNet50-2 [43]) provides significant accuracy gains over baseline ConvNets (Table 1, Fig. 4a). MWE-based training also provides considerable accuracy gain on ResNet50 [9]; however, this accuracy gain is much lower than that with LWE. To provide perspective on the accuracy gains achieved with WE-based training, we compare them with the accuracy gains obtained using the popular SE blocks [12]. We find the two sets of accuracy gains to be similar (Table 1), except for MobileNetV2 [32] where SE [12] provides better accuracy gain. This exception is likely due to LWE's inability in providing excitations on depthwise convolution blocks that have a size of $Out \times In \times h \times w$ where $In = 1$. In such a case, we conjecture that any excitation provided by LWE's importance map m of sized $Out \times 1$ likely gets absorbed by the following batch normalization (BN) that also has a $Out \times 1$ learnable scaling paramter. This limitation of LWE in depthwise convolution is not shared by MWE - adding MWE on depthwise convolutions of MobileNetV2 provides improved performance gain (MobileNetV2-WE in Table 1).

Comparing LWE with Activation Map-Based Attention Mechanisms: LWE-based built-in weight attention mechanism brings large accuracy gains on baseline ConvNets similar to popular attention methods such as SE [12] (Table 1); however, in contrast to prior attention approaches [12,27,35] LWE does not burden the baseline ConvNets with any structural changes or any added computation costs at inference. Also, training costs with LWE methods are lower than prior attention approaches such as SE [12] since LWE is applied on weights whereas prior attention approaches are applied on considerably larger activation

maps (e.g. lower training memory with LWE, see Table 1). Finally, LWE-based attention is applied on each of the In input filter kernels separately for each output channel j (Fig. 3), which is complementary to prior attention methods that provide attention to only the output channels separately [12,27,35]. We conjecture that, due to this complementary attention mechanism, consistent accuracy gains (Table 1) and speed in convergence (Fig. 4b) are observed when LWE is used with other attention methods [12,35]. Overall, the utility of LWE is in its ability to provide considerable accuracy gains that are comparable to SE [12] with lower training costs than SE and without addition of inference costs, and in LWE's complementary nature to prior attention approaches.

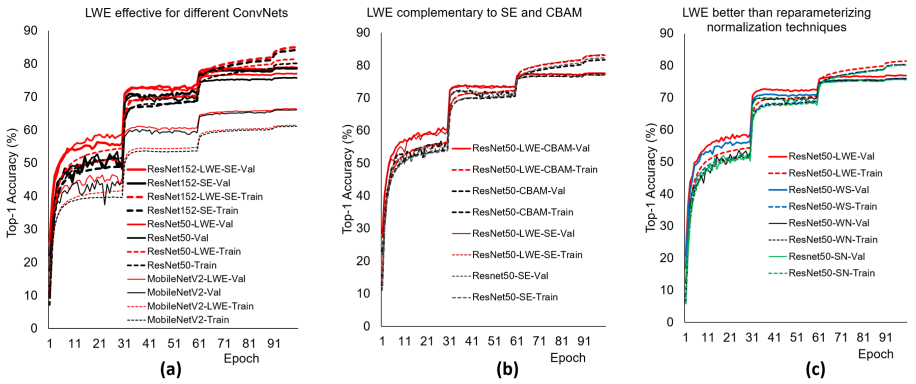

Fig. 4. (a) When LWE-based training (red lines) is applied on baseline ConvNets, consistent gains in performance are sustained throughout the training process. (b) Using LWE with other baseline activation map-based attention methods provide further performance gains. (c) Compared with weight normalizing reparameterization approaches that can speed up training convergence, LWE demonstrates faster convergence and better accuracy gains. (Color figure online)

Comparing LWE with Weight Normalizing Approaches: Weight normalizing reparameterization approaches have shown promise in speeding ConvNet training and in providing accuracy gains to baseline ConvNets [24,28,31]. We compare accuracy gains and speed of convergence of LWE-based training with three weight normalizing reparameterization approaches [24,28,31] for the ResNet50 [9] ConvNet. We find that LWE-based training outperforms all weight normalizing reparameterizing approaches by a considerable margin in terms of accuracy gain (Table 2) and speed of convergence (Fig. 4c). Taking training time into consideration, since LWE-based training adds around 7% training time compared to baseline ResNet50, we conservatively reduce total training epochs of LWE from 100 to 90 (with learning rate decays at 27th, 54th and 81st epochs), and still find improvements over other approaches (Table 2).

Applicability to Group Normalization, AdamW and Cosine Learning Rate: As a preliminary study to identify whether LWE retains its beneficial

Table 1. Validation accuracy of different baseline ConvNet architectures without and with WE-based training on the ImageNet image classification task. Memory requirements for training and inference are shown for batch size of 128. LWE consistently provides considerable accuracy gains that are comparable to accuracy gains achieved when SE blocks [12] are used with baseline ConvNet. In contrast to SE [12], WE methods do not add inference costs or ConvNet structural changes. Also, training with WE in baseline ConvNets require relatively small extra training memory compared to the extra memory requirement of SE (extra memory with WE around 1/3 to 2/3 of the extra memory with SE). LWE also complements different attention methods [12,27,35] in improving accuracy.

Method	Train Param. (M)	Train Mem. (GB)	Inference Param. (M)	Inference Mem. (GB)	GFLOPs	Top-1 Acc. (%)	Top-5 Acc. (%)
Small sized models							
MobileNetV2 [32]	3.5	9.5	3.5	2.9	0.317	66.2	87.1
MobileNetV2-**LWE**	4.1	9.6	3.5	2.9	0.317	66.5	87.3
MobileNetV2-**WE**	4.1	**9.6**	**3.5**	**2.9**	**0.317**	**67.0**	**87.5**
MobileNetV2-SE	3.6	9.8	3.6	3.0	0.320	67.3	87.8
MobileNetV2-**WE**-SE	4.2	9.9	**3.6**	**3.0**	**0.320**	**68.1**	**88.2**
ResNet18	11.7	4.9	11.7	2.2	1.80	70.6	89.5
ResNet18-**LWE**	11.9	**5.1**	**11.7**	**2.2**	**1.80**	**71.0**	**90.0**
ResNet18-SE	11.8	5.4	11.8	2.2	1.81	71.0	90.1
ResNet18-**LWE**-SE	11.9	5.6	**11.8**	**2.3**	**1.81**	**71.7**	**90.5**
Medium sized models							
ResNet50	25.6	14.0	25.6	2.6	3.86	75.8	92.8
ResNet50-BAM	25.9	15.8	25.9	2.8	3.94	76.0	92.8
ResNet50-**LWE**	28.1	**14.8**	**25.6**	**2.6**	**3.86**	**77.1**	**93.5**
ResNet50-SE	28.1	16.7	28.1	2.9	3.87	77.1	93.5
ResNet50-CBAM	28.1	19.3	28.1	3.4	3.87	77.2	93.7
ResNet50-**LWE**-SE	30.6	17.6	28.1	**2.9**	3.87	77.5	93.8
ResNet50-**LWE**-BAM	28.4	16.7	25.9	2.8	3.94	77.4	93.7
ResNet50-**LWE**-CBAM	30.6	20.2	**28.1**	3.4	**3.87**	**77.7**	**93.9**
ResNeXt50	25.0	16.8	25.0	2.4	4.24	77.2	93.4
ResNeXt50-**LWE**	27.9	**17.2**	**25.0**	**2.4**	**4.24**	**77.7**	**93.8**
ResNeXt50-SE	27.5	19.5	27.5	2.9	4.25	77.8	93.9
ResNeXt50-**LWE**-SE	30.4	20.0	**27.5**	**2.9**	**4.25**	**78.1**	**94.1**
Large sized models							
WideResNet50-2	68.9	18.3	68.9	5.2	11.46	77.3	93.5
WideResNet50-2-**LWE**	72.3	**20.0**	**68.9**	**5.2**	**11.46**	**78.3**	**94.2**
WideResNet50-2-SE	71.4	21.2	71.4	5.29	11.48	78.3	94.2
WideResNet50-2-**LWE**-SE	74.9	22.8	**71.4**	**5.29**	**11.48**	**78.6**	**94.3**
ResNet152	60.2	25.8	60.2	3.27	11.30	77.9	93.8
ResNet152-**LWE**	67.3	**29.7**	**60.2**	**3.27**	**11.30**	**78.6**	**94.3**
ResNet152-SE	66.8	31.4	66.8	3.46	11.32	78.7	94.3
ResNet152-**LWE**-SE	73.9	35.4	**66.8**	**3.46**	**11.32**	**79.0**	**94.6**

properties when different normalizations, optimizers and schedulers are used, we study the following on ResNet50 [9] baseline - (a) effectiveness of LWE when used with BN [13] and group normalization (GN) [38] (Fig. 5), and (b) effectiveness of LWE when using AdamW optimzer [23] with initial learning rate

of 10^{-3} and weight decay of 0.1, and a cosine learning rate with initial restart period set as 5 epochs and a restart period multiplier of 1.2 [23], and run for four cosine annealing cycles (Fig. 5). We find that LWE-based training provides similar accuracy gains on ResNet50 [9] ConvNets with both BN [13] and GN [38] (Table 4) (Fig. 5). We also find that ResNet50-LWE continually shows better training and validation accuracy compared to baseline ResNet50 and ResNet50-SE at every epoch for the AdamW and cosine learning rate optimizer/scheduler setting (accuracy at last epoch: baseline ResNet50 70.9% vs. ResNet50-SE 71.2% vs. ResNet50-LWE 72.0%) (Fig. 5). Notably, LWE performs considerably better than SE [12] in this optimizer/scheduler setting. These results provide preliminary evidence that LWE can be effective in various normalization, optimizer and learning rate settings.

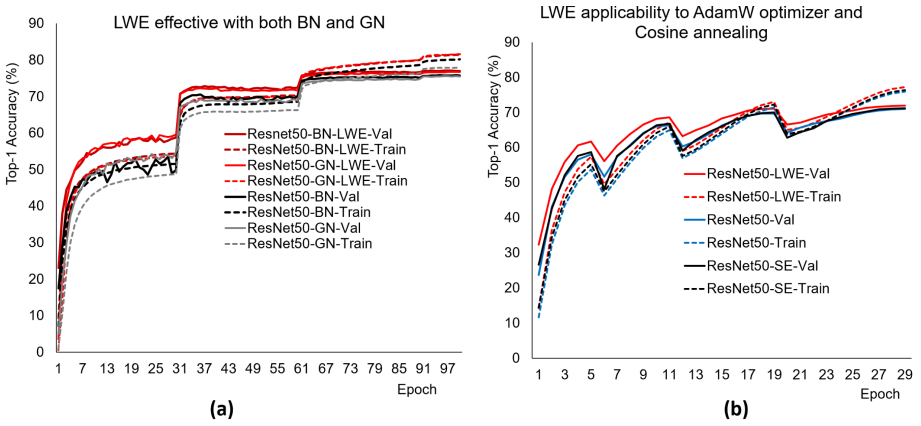

Fig. 5. (a) LWE-based training improves on baseline ResNet50 with both GN [38] and BN [13] normalizations. (b) LWE-based training on ResNet50 consistently performs better than baseline ResNet50 or ResNet50-SE for AdamW optimizer and Cosine annealing scheduler.

Utility of LWE and MWE Methods: While MWE provides a substantial improvement over baseline ResNet50, LWE causes an even larger improvement (Table 3). This improved utility of LWE is perhaps because it is based on a data-dependant learning approach and also because it provides a sense of relative structural importance (e.g. grouping weights in a 3×3 filter together, and comparing relative importance with other groups). Combining LWE and MWE together did not result in any noticeable accuracy gain (77.07% with LWE vs. 77.09% with LWE+MWE).

While our LWE method outperforms our MWE method, MWE has some advantages over LWE. First, as explained earlier, MWE is effective on depthwise convolutions whereas LWE is not (Table 1). Also, since LWE has an effect of suppressing some input channels, it is detrimental for shift-based convolutions [20,36] - this is because suppressing an input channel can result in consistently

Table 2. WE-based weight reparameterization provides considerable accuracy gains compared to weight reparameterization approaches

Method	Top-1 Acc.(%)	Top-5 Acc.(%)	△Top-1
ResNet50	75.8	92.8	+0.0
+WS [28]	76.1	92.9	+0.3
+WN [31]	76.0	92.9	+0.2
+SN [24]	75.7	92.8	-0.1
+MWE	76.5	93.1	+0.7
+LWE,90Epoch	**76.9**	**93.4**	**+1.1**
+LWE	**77.1**	**93.5**	**+1.3**

Table 3. Effectiveness of each WE processes in improving accuracy from ResNet50 baseline (△Top-1).

Method	Top-1 Acc.(%)	Top-5 Acc.(%)	△Top-1
ResNet50	75.83	92.8	+0.00
+MWE	76.52	93.1	+0.69
+LWE	77.07	93.5	+1.24
+LWE+MWE	77.09	93.5	+1.26

Table 4. LWE provides considerable accuracy gains with both batch normalization- [13] and group normali zation-based [38] ResNet50.

Method	Top-1 Acc.(%)	Top-5 Acc.(%)	△Top-1
ResNet50-GN	75.5	92.7	+0.0
ResNet50-GN-WN	75.9	92.9	+0.4
ResNet50-GN-WS	76.0	92.9	+0.5
ResNet50-GN-LWE	**76.8**	**93.3**	**+1.3**
ResNet50-BN	75.8	92.8	+0.0
ResNet50-BN-LWE	**77.1**	**93.5**	**+1.3**

Table 5. Effectiveness of MWE in improving accuracy over baseline ResNet50 for different ϵ_A.

Method	Top-1 Acc.(%)	Top-5 Acc.(%)	△Top-1
ResNet50	75.83	92.8	+0.0
+MWE, $\epsilon_A = 10^{-3}$	76.38	93.0	+0.55
+MWE, $\epsilon_A = 0.1$	76.52	93.1	+0.69
+MWE, $\epsilon_A = 0.2$	76.43	93.0	+0.50

ignoring some parts of the input signal. MWE remains effective for shift-based convolutions (Sect. 4.6).

Validity of Attention Multiplier of LWE: To verify that LWE's learned attention multiplier m provides excitation to the more important channel locations, we study decrease in performance after channels with the lowest LWE multiplier are decimated (D_L) in each layer and decrease in performance after channels with the highest LWE multiplier are decimated (D_H). As expected, we find $D_H > D_L$ ($D_H - D_L = 8.15$), meaning that the channels with high importance multiplier are more important and loss of such channels are more detrimental to the network. Interestingly, we find $D_L = -0.03$ meaning that removing the channels with the lowest LWE multiplier in each layer contributes to small improvement in accuracy, though the difference in accuracy is small and could turn out to be insignificant after repeated experiments. Investigating whether some of the channels in a ConvNet are adversarial or detrimental to the ConvNet's performance is interesting future work.

Sensitivity to Hyperparameter: We have used $\epsilon_A = 0.1$ in MWE. A large ϵ_A makes f_A close to an identity line thereby negating any effect that f_A may have, while ϵ_A close to 0 can amplify the maximum value in W_L towards infinity thereby causing exploding gradients. We investigate three values of ϵ_A: 10^{-3}, 0.1 and 0.2 (Table 5), and find consistent improvements over baseline in all settings, suggesting that use of ϵ_A within a reasonable range (e.g. between 10^{-3} and 0.2) may provide consistent performance gains. We also used reduction ratio of 16

in LWE (Fig. 3a). Similar to [12], reducing this ratio to 8 improved ImageNet accuracy slightly by 0.05%.

4.3 CIFAR-100 Image Classification

Continuing on our ablation studies on image classification task, we investigate LWE method applied on some other popular ConvNet architectures, VGG19 [29] and ResNeXt-29-16x64d [40]), on the standard CIFAR-100 dataset [16]. We used a batch size of 256, and optimized with SGD with backpropagation [17] having momentum 0.9 and a decay of 5×10^{-4}. Learning rate was initialized at 0.1 and reduced with a factor of 10^{-1} at 81 and 122 epochs, and training was completed at 164 epochs. We see consistent improvements when LWE is used with base ConvNets. LWE improves top-1 accuracy of VGG19 from 73.2% to 73.8%, and yields a larger top-1 accuracy improvement on ResNeXt-29-16x64d (80.5% with ResNeXt-29-16x64d vs. 81.5% with ResNeXt-29-16x64d-LWE) (Table 6).

Table 6. Efficacy of WE on different tasks and convolution operations. Superscript * denotes average result from 5 repeated experiments.

Baseline ConvNet	Convolution type	Performance metric	Baseline performance	WE type	△ Performance with WE added
CIFAR100 image classification					
VGG19	2D	Accuracy	73.2%	LWE	+0.6
ResNeXt-29-16x64d	2D	Accuracy	80.5%	LWE	+1.0
PASCAL VOC semantic segmentation					
DeepLabv3 (ResNet50)	2D	IOU	74.4%*	MWE	+0.8
DeepLabv3 (ResNet50)	2D	IOU	74.4%*	LWE	+1.3
CityScapes VOC semantic segmentation					
DeepLabv3 (ResNet50)	2D	IOU	76.0%	LWE	+0.7
Mini-Kinetics action recognition					
R3D-ResNet18	3D	Accuracy	43.7%	LWE	+0.6
Gesture recognition on Jester dataset					
TSM-ResNet50	Shift-based 3D	Accuracy	96.3%	MWE	+0.3

4.4 PASCAL VOC and CityScapes Semantic Segmentation

To explore applicability of WE methods for tasks other than image classification, we investigate semantic segmentation on the PASCAL VOC 2012 [3] and CityScapes [2] datasets. For PASCAL VOC, we use the training recipe in [28]

and use DeepLabv3 with ResNet50 as backbone [1] for its competitive performance, and find that MWE improves mean Intersection over Union (IOU) from 74.4% to 75.2% (+0.8% improvement) and LWE improves IOU to 75.7% (+1.3% improvement) (Table 6). For CityScapes, we also use DeepLabv3 with ResNet50 as backbone [1] and use the training recipe in [5].

4.5 Mini-Kinetics Action Recognition

To show the general applicability of WE mechanisms to improve training of 3D convolution kernels, we use LWE in the 3D convolution blocks of the R3D-ResNet18 ConvNet [34] and experiment on the Mini-Kinetics action recognition dataset [41]. We train R3D-ResNet18 on 16 densely sampled frames per video for 50 epochs using an initial learning rate of 0.02 which later gets reduced twice with a factor of 10^{-1} after 20^{th} and 40^{th} epochs. The weights of the network were randomly initialized, and trained using SGD [17] with momentum of 0.9 and a weight decay of 5×10^{-4}. A batch size of 64 was used. Evaluating on standard single center-crop [20], we find non-trivial accuracy improvement when LWE is used in R3D-ResNet18 (top-1 accuracy: R3D-ResNet18 at 43.7% vs. R3D-ResNet18-LWE at 44.3%, Table 6), suggesting LWE can be effective for ConvNets that use 3D convolution, without adding any computation cost at inference.

4.6 Gesture Recognition on Jester Dataset

To show applicability in other recognition tasks on video, we experiment with the Jester dataset for gesture recognition [33]. Here, we choose the TSM [20] ConvNet as baseline for two reasons: First, TSM [20] uses an efficient shift-based operation [36] that can be used to reduce complexity of a 3D convolution, and we want to evaluate effectiveness of using WE in such shift-based convolutions; second, TSM outperforms most other ConvNets in gesture recognition on the Jester dataset [20].

TSM's temporal shift-based convolution operation assigns different temporal information to different input channels. To train TSM, we use the same training recipe described in Sect. 4.5 except we use a strided sampling [20]. Implementing MWE in shift-based convolutions of TSM-ResNet50 [20] yields improved single center-crop accuracy (96.3% for TSM only vs. 96.6% with TSM+MWE), with no additional inference cost.

5 Conclusion

In this paper, we proposed novel weight reparameterization mechanisms that simultaneously identify the relative importance of weights in a convolution block in a ConvNet, and provide more attention to training the more important weights. Training a baseline ConvNet using such WE mechanisms can provide strong performance gains, without adding any additional computational costs or

ConvNet structural change at inference. We demonstrated this potency of our WE mechanisms in diverse settings (e.g. varying computer vision tasks, different baseline ConvNets with and without activation map-based attention methods, multiple normalization methods, etc.). Finally, WE mechanisms can easily be implemented in a convolution block with minimal effort, requiring few lines of extra code. Overall, the diversity on applicability, the complementarity with previous attention mechanisms and the simplicity in implementation make our WE mechanisms valuable in variously and popularly used ConvNets in computer vision research.

References

1. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking Atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
2. Cordts, M., et al.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3213–3223 (2016)
3. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: a retrospective. Int. J. Comput. Vision **111**(1), 98–136 (2015)
4. Frankle, J., Carbin, M.: The lottery ticket hypothesis: finding sparse, trainable neural networks. In: International Conference on Learning Representations (2019)
5. Gongfan, F.: pytorch-classification (2020). https://github.com/VainF/DeepLabV3Plus-Pytorch
6. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
7. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems, pp. 1135–1143 (2015)
8. Hassibi, B., Stork, D.G.: Second order derivatives for network pruning: Optimal brain surgeon. In: Advances in Neural Information Processing Systems, pp. 164–171 (1993)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
10. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 558–567 (2019)
11. Hu, J., Shen, L., Albanie, S., Sun, G., Vedaldi, A.: Gather-excite: exploiting feature context in convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 9401–9411 (2018)
12. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141 (2018)
13. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
14. Kay, W., et al.: The kinetics human action video dataset. arXiv preprint arXiv:1705.06950 (2017)

15. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
16. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Technical report, Citeseer (2009)
17. LeCun, Y., et al.: Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989)
18. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Advances in Neural Information Processing Systems, pp. 598–605 (1990)
19. Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the loss landscape of neural nets. In: Advances in Neural Information Processing Systems, pp. 6389–6399 (2018)
20. Lin, J., Gan, C., Han, S.: Temporal shift module for efficient video understanding. arXiv preprint arXiv:1811.08383 (2018)
21. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988 (2017)
22. Liu, J., Xu, Z., Shi, R., Cheung, R.C.C., So, H.K.: Dynamic sparse training: find efficient sparse network from scratch with trainable masked layers. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=SJlbGJrtDB
23. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization (2018)
24. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957 (2018)
25. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440 (2016)
26. Nesterov, Y.: A method of solving a convex programming problem with convergence rate. In: Soviet Math. Dokl. vol. 27
27. Park, J., Woo, S., Lee, J.Y., Kweon, I.S.: Bam: Bottleneck attention module. arXiv preprint arXiv:1807.06514 (2018)
28. Qiao, S., Wang, H., Liu, C., Shen, W., Yuille, A.: Weight standardization. arXiv preprint arXiv:1903.10520 (2019)
29. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
30. Russakovsky, O., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vision **115**(3), 211–252 (2015)
31. Salimans, T., Kingma, D.P.: Weight normalization: a simple reparameterization to accelerate training of deep neural networks. In: Advances in Neural Information Processing Systems, pp. 901–909 (2016)
32. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv 2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
33. The 20bn-jester dataset v1: The 20BN-jester Dataset V1. Accessed 16 July 2020
34. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 6450–6459 (2018)
35. Woo, S., Park, J., Lee, J.-Y., Kweon, I.S.: CBAM: convolutional block attention module. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11211, pp. 3–19. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_1

36. Wu, B., et al.: Shift: a zero flop, zero parameter alternative to spatial convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9127–9135 (2018)
37. Wu, L., Zhu, Z., et al.: Towards understanding generalization of deep learning: perspective of loss landscapes. arXiv preprint arXiv:1706.10239 (2017)
38. Wu, Y., He, K.: Group normalization. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11217, pp. 3–19. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01261-8_1
39. Xiang, L., Shuo, C., Yan, X., Jian, Y.: Understanding the disharmony between weight normalization family and weight decay: *epsilon*− shifted l_2 regularizer. arXiv preprint arXiv:1911.05920 (2019)
40. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1492–1500 (2017)
41. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning: speed-accuracy trade-offs in video classification. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11219, pp. 318–335. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01267-0_19
42. Yang, W.: pytorch-classification (2017). https://github.com/bearpaw/pytorch-classification
43. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)