Channel Regeneration: Improving Channel Utilization for Compact DNNs

Ankit Sharma and Hassan Foroosh

Computational Imaging Lab
Department of Computer Science, University of Central Florida
Orlando, USA
ankit.sharma285@knights.ucf.edu, Hassan.Foroosh@ucf.edu

Abstract

Overparameterized deep neural networks have redundant neurons that do not contribute to the network's accuracy. In this paper, we introduce a novel *channel regeneration* technique that reinvigorates these redundant channels of efficient architectures by re-initializing its batch normalization scaling factor γ. This re-initialization of BN γ of these channels promotes regular weight updates during training. Furthermore, we show that channel regeneration encourages the channels to contribute equally to the learned representation and further boosts the generalization accuracy. We apply our technique at regular intervals of the training cycle to improve channel utilization. The solutions proposed in previous works either raise the total computational cost or increase the model complexity. Integrating the channel regeneration technique into the training methodology of efficient architectures requires minimal effort and comes at no additional cost in size or memory. Extensive experiments on several image classification benchmarks and on semantic segmentation task demonstrate the effectiveness of applying the channel regeneration technique to compact architectures.

1 Introduction

The recent success of large deep neural networks in fundamental computer vision tasks (Redmon and Farhadi 2017; Ren et al. 2017; Chen et al. 2018; He et al. 2016; Simonyan and Zisserman 2015) has accelerated the demand for deploying them commercially. However, overparameterized deep neural networks(DNN) have a large number of redundant neurons that do not contribute to the network output. Contemporary works have addressed this issue by pruning these ineffective neurons/channels from the network (Liu et al. 2017; Lin et al. 2020; He et al. 2018; Li et al. 2020). However, pruning results in an overall drop in accuracy. Another approach is to revitalize these irrelevant channels (Shao et al. 2020). Increasing channel utilization encourages more channels to contribute to boosting accuracy and improving generalization. In this work, we focus on the latter.

Previous works on improving the resource utilization of DNNs have proposed solutions that modify the conventional CNN building blocks such as convolutional layers and optimizers (Qiao et al. 2019) or add a self-organizing map in the training pipeline for increasing channel utilization (Baddar et al. 2021). Another line of work investigates the effect of the standard batch normalization(BN) layer on channels participation to the learned feature representations (Shao et al. 2020; Huang et al. 2019). However, these works either increase the total training computational cost (Shao et al. 2020; Huang et al. 2019; Baddar et al. 2021) or require modifying existing off-the-shelf network models limiting their usage (Shao et al. 2020; Huang et al. 2019; Qiao et al. 2019). We improve the channel utilization of compact DNNs at no additional computational cost during training or at inference.

A trained deep neural network suffers from the phenomenon of filter level sparsity wherein a large number of feature channels are inactive and always produce small val-

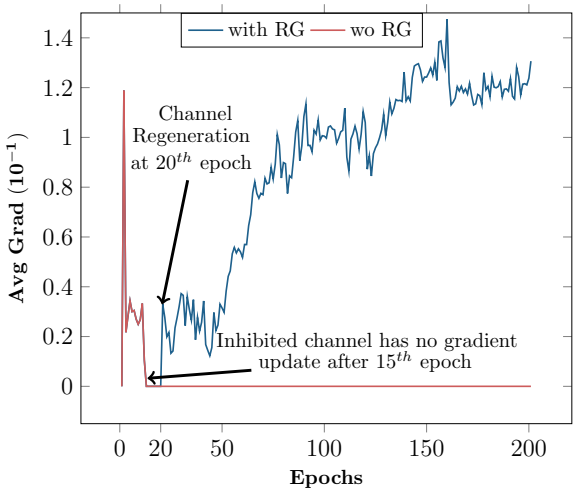

Figure 1: The graph plots the average gradient update of an inhibited channel with and without channel regeneration(RG). Under the standard training routine, the channel has no weight update after the 15^{th} epoch as the average gradient drops to zero. This limits the contribution of the corresponding filter to the network output. To address this problem of under utilization of network resources, channel regeneration restores inhibited (inactive) channels/filters and thereby ensuring a consistent gradient update for all the channels.

ues (Mehta, Kim, and Theobalt 2019). Specifically, a channel is considered inactive if the BN γ values of the channel is below $|10^{-2}|$. These *inhibited channels* (Shao et al. 2020) contribute little to the final performance of the network. The graph in Figure 1 shows the average gradient of an inhibited channel from the second layer of a trained MobileNetv2 network (Sandler et al. 2018) on CIFAR-100 during training. It can be observed that the average gradient value falls to zero quickly and remains zero for the remaining training epochs. Consequently, this inhibited channel undergoes no weight update and, therefore, has little or no impact on the final learned representation. The network would suffer from poor generalization performance if it relies only on few individual selective channels/filters (Morcos et al. 2018). This under-utilization of network resources is a cause for concern of pruned models and efficient architectures. This naturally raises the following question: *Is it possible to recover these inhibited channels and ensure better utilization of resources in edge environments without increasing the computational training cost?*

In this paper, we present a novel technique *channel regeneration* for compact models that regenerates inhibited channels by re-initializing the batch normalization scaling factor of all the inhibited channels in the network. This is shown in Fig 2. The key insight to our technique is that the BN γ of a channel plays an important role in determining the weight update δw of its corresponding filter during back propagation. We empirically show that the gradient update of a channel is zero or close to zero when the channel's BN γ is low. Re-initializing the BN γ of an inhibited channel increases the gradient and encourages weight update δw. Figure 1 shows that the average gradient of an inhibited channel rises after we apply our channel regeneration technique at 20^{th} epoch. Consequently, this makes the channel relevant and improves the overall network baseline capacity. Our technique offers two advantages over previous methods on improving channel utilization: (i) **First,** the total computational cost during training and at inference remains the same. Our technique is computationally efficient to execute and furthermore it does not increase the total network parameters, and (ii) **Second,** channel regeneration can be integrated with off-the-shelf models from the popular deep learning libraries. We apply channel regeneration at regular intervals while training commonly used compact neural networks to improve network accuracy and resource utilization.

Our contributions can be summarized as follows:

- We introduce a novel technique named *channel regeneration* that restores inhibited channels by re-initializing their batch normalization scaling factor.
- We conduct extensive experiments to show that channel regeneration increases channel utilization and boosts the generalization accuracy of the model.
- We demonstrate the effectiveness of our novel channel regeneration technique on classification and semantic segmentation tasks across several compact architectures and datasets.

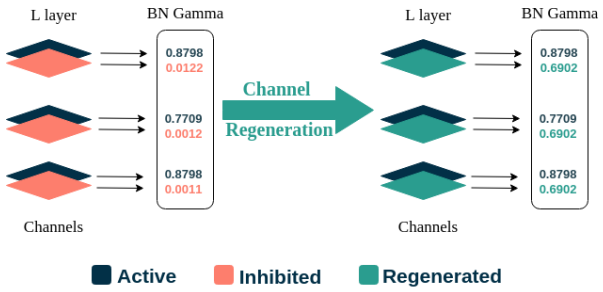

Figure 2: Channel Regeneration. Inhibited channels with low BN scaling factor γ are regenerated by re-initializing their γ to γ_{new} (regenerated channels seen in green). The value of γ_{new} is calculated at every regeneration epoch and is discussed in details in section 3.

Method	ΔParam	ΔTT	OTS
ITN (Huang et al. 2019)	No	Increase	No
CE (Shao et al. 2020)	Increase	Increase	No
NR (Qiao et al. 2019)	No	No	No
SRR (Baddar et al. 2021)	Increase	Increase	No
Ours	No	No	Yes

Table 1: Comparison of Channel Regeneration with other works. (TT: Training Time, OFS: Off-the-Shelf)

2 Related Works

Improving Model Capacity: The filter reinitialization method in Qiao et al. (2019) enforces L1 penalty on all filters to increase sparsity. Moreover, the weights/parameters of rejuvenated filters are randomly re-initialized. In addition, the method modifies the conventional CNN building blocks such as convolutional layers and optimizers for its application. In contrast, our proposed channel regeneration technique is straightforward since it only resets the BN scaling term of channels after regular intervals without reinitializing the filter weights. In Baddar et al. (2021), class activation maps are used to train self-organizing maps for re-organization purposes. Following this, they perform the channel rejuvenation step to revive neurons to increase model capacity. Compared to (Qiao et al. 2019; Baddar et al. 2021), our technique applies to any off-the-shelf CNN model at no additional training computational cost.

Normalizing Techniques: Shao et al. (2020) encourages channels to contribute equally by introducing an additional neural block after every BN layer in the network. This block performs batch decorrelation and instance reweighting for the batch of mini samples at every training iteration. DBN (Huang et al. 2018) applies ZCA whitening on the activations along with scaling to achieve better performance and generalization ability than the standard BN. However, the method performs eigenvalue decomposition, which is computationally expensive and increases the total training time. Iterative normalization (Huang et al. 2019) describes a methodology to perform ZCA whitening without computing

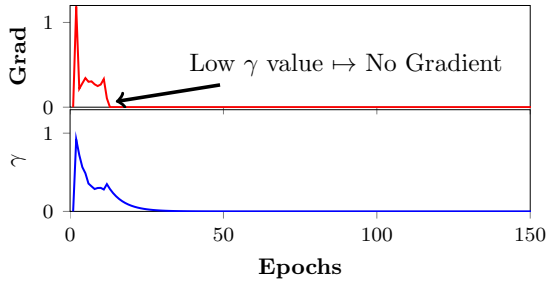

Figure 3: This graph shows the influence of BN γ on the average gradient update of an inhibited channel. Lower γ value results in no weight update of the corresponding filter. Thus, this network channel has no contribution to the network output, which results in poor channel/resource utilization.

the eigenvalue decomposition. All these normalizing methods described significantly increase the training time and do not apply to the off-the-shelf models available in popular libraries. Table 1 compares our proposed channel regeneration technique with previous works.

Efficient Architectures: MobileNetv2 (Sandler et al. 2018) consists of inverted residual blocks as building blocks to design a light-weight network topology. These blocks are composed of inverted residuals and depthwise separable convolutions to improve the efficiency of the network. ShuffleNet (Zhang et al. 2018) consists of a shuffle unit which is made up of 1x1 group convolution with channel shuffle operation followed by 3x3 depthwise convolution and another 1x1 group convolution. Tan and Le (2019) introduces a compound coefficient method to uniformly scale across dimensions and leverage neural architecture search to construct a family of networks, *EfficientNets* which are computationally efficient and provide better performance than networks with a similar computational budget.

Channel Pruning: Channel pruning involves removing unimportant channels across the network based on saliency criteria. Liu et al. (2017) channel-level regularization is enforced on scaling factors from batch normalization layers. Peng et al. (2019) examines the dependency between channels by using a novel loss function based on Taylor expansion. In (Luo, Wu, and Lin 2017), filters are pruned based on statistical information from immediately subsequent layers. (He et al. 2020) introduces a learnable differentiable pruning criteria sampler to find the appropriate pruning criteria for each layer. Lin et al. (2020) prunes the filters with low-rank feature maps as these maps contain less information. Unlike pruning, our technique aims to rehabilitate these redundant network channels and improve the network accuracy.

3 Method

We begin by establishing the relationship between the gradient update of a channel/filter and its corresponding BN γ. Then, we introduce our channel regeneration technique and its effect on the gradient update of all channels in a network.

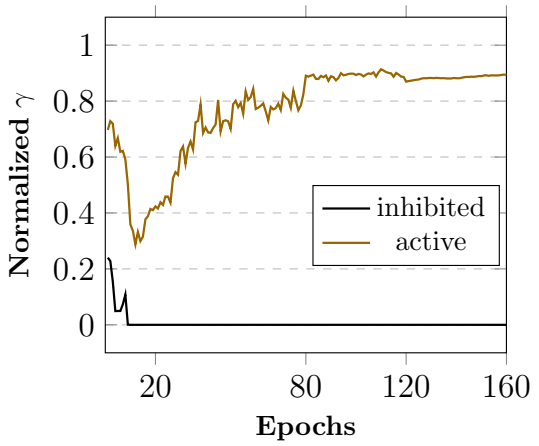

Figure 4: The graph plots the normalized gamma values of an inhibited and an active channel during training. The γ value of the inhibited channel falls drastically in the initial epochs and never recover even after several training epochs. In contrast, the γ value of the active channel has the γ value close to the median value during the early epochs and rises to a very high value in the later training epochs.

Network	BL	Median	Mean	0.2	Max
MobileNetv2	69.10	**71.90**	71.50	71.25	71.81
ShuffleNetv2	70.18	**72.90**	72.02	71.99	72.54

Table 2: Accuracy for different values of γ_{new} on CIFAR-100

3.1 Background

Consider the conventional *Conv-BN-ReLU* building block of a CNN model. During training, the weight update of the convulational filter F in the block is:

$$F_{update} = F + LR * \frac{\partial \mathcal{L}}{\partial F}; \ \frac{\partial \mathcal{L}}{\partial F} = \frac{\partial \mathcal{L}}{\partial O} . \frac{\partial O}{\partial F} \tag{1}$$

where $\mathcal{L}$ is the training loss, LR is the learning rate and O is the CNN filter output. The gradient to update the filter $\frac{\partial \mathcal{L}}{\partial F}$ is the product of the loss gradient from the BN layer of the building block $\frac{\partial \mathcal{L}}{\partial O}$ and the local gradient $\frac{\partial O}{\partial F}$. The BN layer transforms the CNN filter output O with its affine learnable parameters in the following manner (Ioffe and Szegedy 2015):

$$BN_{out} = \gamma \hat{O} + \beta; \ \hat{O} = \frac{O - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{2}$$

where γ and β are the BN scaling and shifting parameters; μ and σ are the mean and standard deviation values of input activations over the mini-batch B respectively. To establish the dependency of the filter update $\frac{\partial \mathcal{L}}{\partial F}$ on BN γ, we must show that the value of $\frac{\partial \mathcal{L}}{\partial O}$ is dependent on γ . From (Ioffe and Szegedy 2015), we know that:

$$\frac{\partial \mathcal{L}}{\partial O} = \frac{\partial \mathcal{L}}{\partial \hat{O}} \frac{1}{\sqrt{\sigma^2 + \epsilon}} + \frac{\partial \mathcal{L}}{\partial \sigma^2} \frac{2(O - \mu)}{B} + \frac{1}{B} \frac{\partial \mathcal{L}}{\partial \mu} \tag{3}$$

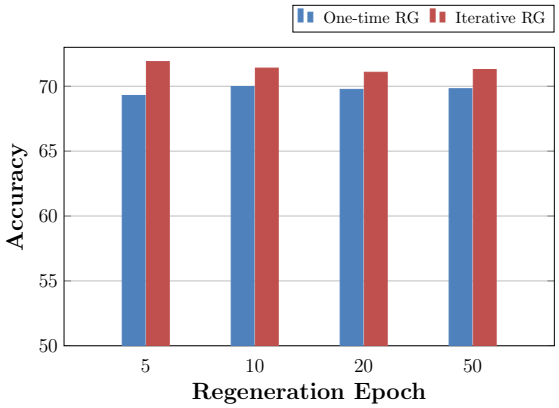

Figure 5: The graph reports the accuracy under different regeneration settings.

Also, the gradients $\frac{\partial \mathcal{L}}{\partial \sigma^2}$ and $\frac{\partial \mathcal{L}}{\partial \mu}$ can be found by using the $\frac{\partial \mathcal{L}}{\partial \hat{O}}$ in the following manner:

$$\frac{\partial \mathcal{L}}{\partial \sigma^2} = \sum \frac{\partial \mathcal{L}}{\partial \hat{O}} \cdot (O - \mu) \cdot \frac{-1}{2}(\sigma^2 + \epsilon)^{\frac{-3}{2}} \tag{4}$$

$$\frac{\partial \mathcal{L}}{\partial \mu} = (\sum \frac{\partial \mathcal{L}}{\partial \hat{O}} \frac{-1}{\sqrt{\sigma^2 + \epsilon}}) + \frac{\partial \mathcal{L}}{\partial \sigma^2} \cdot \frac{\sum -2(O - \mu)}{B} \tag{5}$$

Finally, the loss gradient $\frac{\partial \mathcal{L}}{\partial \hat{O}}$ is:

$$\frac{\partial \mathcal{L}}{\partial \hat{O}} = \frac{\partial \mathcal{L}}{\partial BN_{out}} \cdot \gamma \tag{6}$$

From equations [1-6], we notice that the value of γ has an important role to play in determining the gradient update of the filter F.

To empirically verify the influence of γ on $\frac{\partial \mathcal{L}}{\partial F}$, we train a MobileNetv2 model on CIFAR-100 and plot the average gradient update of an inhibited channel and its corresponding γ value for the entire training in Fig 3. The initial value of BN γ is 1.0. From the figure, we make an important observation that the average gradient weight update and BN γ value of an inhibited channel falls to zero only after a few training epochs. A lower value of γ impedes the learning of the channel/filter, which results in no contribution to the final learned representation of the network.

3.2 Channel Regeneration

Channel regeneration works by re-initializing the BN scale factor value γ of inhibited channels to a new value γ_{new}. In section 3.1, we establish that the a lower value of a channel's BN γ value leads to no gradient weight update of its corresponding filter. Increasing the BN γ value of an inhibited channel boosts its importance to the network, and further improving the network baseline capacity.

New γ value of inhibited channels. We plot the optimization trajectory of normalized BN scaling factor values of an inhibited channel and an active channel from the second layer of a trained MobileNetv2 model in Figure 4. As expected, the γ value of the inhibited channel drops to zero by 10^{th} epoch. The active channel has the γ value close to the mean γ value around the 30^{th} epoch. However, the active channel's γ rises to a very high-value mid-way during training. We hypothesize that a channel with a γ value close to the mean or higher would receive regular gradient updates. We evaluate this hypothesis by training a network with the channel regeneration technique. We report results on different values of γ_{new} in Table 2. Training with our technique achieves better accuracy than the baseline for all values of γ_{new}. This result concurs with our rationale that increasing the γ value of inhibited channels above a small threshold would result in higher network accuracy. The models achieve the best accuracy when the value of γ_{new} is greater than equal to the median. For all our experiments, we re-initialize the γ of inhibited channels to γ_{med} of the network.

Algorithm 1: Channel Regeneration

Input: randomly initialized network, Training epochs $\mathcal{T}_E$, regeneration epochs E_{RG}, total number of channels $\mathcal{C}$

Train the model
for $t \leftarrow 1$ to $\mathcal{T}_E$ **do**
  $\mathcal{L} = \mathcal{L}_{\mathcal{CE}}(output, label)$ ▷ Cross entropy loss
  $\gamma_{all} = \{\}$
  if $t = E_{RG}$ **then** ▷ Regeneration Epoch
    for $k \leftarrow 1$ to $\mathcal{C}$ **do**
      $\gamma_{all}.append(\gamma_k)$
    end for
    γ_{med} = Median(Sort(γ_{all})) ▷ Threshold γ
    for $k \leftarrow 1$ to $\mathcal{C}$ **do**
      if $\gamma_k \leq \gamma_{med}$ **then**
        $\gamma_k = \gamma_{med}$ ▷ Channel Regeneration
      end if
    end for
  end if
end for

Output: Trained network

When to apply channel regeneration. To ensure better channel utilization, we must address the following question: how often should we employ channel regeneration? To answer this question, we train a MobileNetv2 model on the CIFAR-100 dataset and report the results in Figure 5. We investigate two different settings: (i) apply channel regeneration only one time at different epochs = $\{5, 10, 20, 50\}$ (ii) apply it at regular intervals of $\{5, 10, 20, 50\}$ epochs. From the figure, we notice that applying our technique at regular intervals achieves significantly higher accuracy. This result suggests that the restoration of a few inhibited channels may be momentary when we employ channel regeneration only once. The regular use of our technique aids in the proper restoration of inhibited channels and consequently improves the final performance. We define a hyper-parameter *regeneration epoch* E_{RG}- the interval of epochs at which we regenerate channels in an iterative manner during training. From the Figure 5, we note that our technique is robust to the E_{RG} value as we increase the accuracy by over 2% over the baseline for different values of E_{RG}.

	CIFAR-100						CIFAR-10					
		RG Epoch (E_{RG})						RG Epoch (E_{RG})				
Model	w/o	5	10	15	20	25	w/o	5	10	15	20	25
MobileNetv2	69.10	**71.90**	71.40	71.65	71.08	71.47	90.74	92.24	92.42	92.18	**92.56**	92.42
ShuffleNetv2	70.18	72.90	**73.07**	72.32	72.41	72.91	91.88	**93.20**	92.92	92.89	92.67	92.75
ResNet-18	76.20	**77.18**	76.99	76.90	76.96	76.41	94.90	**95.14**	95.14	95.04	94.98	95.06
ResNet-34	77.31	**78.14**	77.97	77.84	77.97	77.84	95.28	95.37	95.24	95.53	95.60	**95.69**

Table 3: Performance of compact models on CIFAR-10/100 when trained with and without Channel Regeneration

Algorithm. We describe the Channel Regeneration in Algorithm 1. During regeneration epochs E_{RG}, we first identify inhibited channels in every layer with scaling factor values lesser than γ_{med}. Then, we regenerate these channels by re-initializing their scaling factor values to γ_{med}.

4 Experiments and Results

4.1 Datasets

CIFAR-10/100: The CIFAR datasets consist of natural color images with 32 x 32 dimensions. The datasets contain $50,000$ images for training and $10,000$ images for validation. With regards to the classification problem, we follow the training routine described in (Devries and Taylor 2017), which is initial LR = 0.1 divided by 5 at 60th, 120th, 160th epochs, train for 200 epochs with batch size 128 and weight decay 5e-4, Nesterov momentum of 0.9.

Tiny-ImageNet: Tiny-ImageNet is a subset of the full ImageNet dataset and essentially a downsampled version of the dataset. Tiny-ImageNet contains images from 200 classes and each image is of spatial resolution 64 x 64. Each class has 500 for training and 50 images for validation. We follow the same training routine for classification. We apply random horizontal flipping, rotation and normalize images during training for data augmentation. The batch size is 32 for all architectures reported on this dataset. We train for 90 epochs and decrease the learning rate by 0.1 at $[30, 60]$ epochs.

ImageNet: The ImageNet dataset contains 1.2 million training images and 50,000 validation images of 1000 classes. We train a MobileNetv2 model for 150 epochs with an initial learning rate = 0.05 and, we follow a cosine scheduling policy. We set the batch size to 256, and weight decay is $4e{-}5$. We report the single-center-crop validation error of the final model.

4.2 Image Classification

Results on Efficient Architectures In this section, we demonstrate the effectiveness of *channel regeneration* technique on efficient architectures such as MobileNetv2 (Sandler et al. 2018), ShuffleNetv2 (Ma et al. 2018), Mnas-Net (Tan et al. 2019), ResNets-18/34 (He et al. 2016), RegNet (Radosavovic et al. 2020), Efficient-Net (Tan and Le 2019), ResNext-50 (Xie et al. 2017). We adopt a "plug and play" approach in our experimental setup, where in we apply channel regeneration at regular intervals during training on these efficient architectures.

		RG Epoch (E_{RG})			
Model	w/o	5	10	15	20
MobileNetv2	41.87	**46.40**	45.92	45.31	45.53
ShuffleNetv2	44.50	47.62	46.50	**47.94**	47.65
Mnas-net	39.06	**45.40**	44.82	43.57	43.13
MobileNet3-small	33.34	40.10	39.10	**40.22**	37.97
EfficientNet-B0	42.10	45.46	**46.04**	46.16	44.48
RegNet	50.08	51.49	**51.82**	51.29	51.49

Table 4: Results on Tiny-ImageNet Dataset

		RG Epoch		
Model	w/o	5	10	20
MobileNetv2	71.70	72.02	**72.08**	72.04
ShuffleNetv2	67.89	67.98	**68.11**	68.01
ResNet-50	75.58	75.86	**75.61**	75.60
ResNeXt-50	76.53	**76.91**	76.72	76.73

Table 5: Results on ImageNet

Results on CIFAR-10/100: Here, we train different instances of a specific architecture, such as MobileNetv2, with and without applying channel regeneration. The value of E_{RG} determines when we re-initialize channels with γ_{med} during training. We report the results of different architectures in Table 3. We observe that the application of the channel regeneration technique offers a significant improvement in the accuracy of all networks at different values of E_{RG}. The technique boosts the accuracy by over 2% at different E_{RG} values for MobileNetv2 and ShuffleNetv2 networks. The result demonstrates that channel regeneration promotes resource utilization by reinvigorating less influential channels and, thereby increasing model capacity. Moreover, we show that our technique applies to diverse architectures by achieving higher accuracy on ResNet-18/34 models than the performance obtained without channel regeneration.

Results on Tiny-ImageNet: We report the results on the Tiny-ImageNet dataset in Table 4. The accuracy of models trained with the channel regeneration technique is significantly higher than the baseline. This result emphasizes the effective utilization of channels on several architectures by

Dataset	Model	RG Epoch (E_{RG}) w/o	5	10
CIFAR-100	MobileNetv2	68.9	**71.81**	71.33
	ShuffleNetv2	69.71	72.33	**72.88**
Tiny-ImageNet	MobileNetv2	41.53	45.56	**45.71**
	ShuffleNetv2	45.05	**47.80**	47.13
	EfficientNet	43.20	**46.61**	46.31
	ResNext-50	56.99	**58.01**	57.92
ImageNet	MobileNetv2	71.00	71.38	**71.41**

Table 6: Results on Quantized Networks

our method when compared with the vanilla training protocol. The technique boosts the accuracy by at least 3% for different E_{RG} values on MobileNetv2, ShuffleNetv2 and EfficientNet-B0. We outperform the baseline by over 5% on Mnas-net and MobileNet3-small.

Results on ImageNet: We report the performance of MobileNetv2, ShuffleNetv2, ResNet-50 and ResNeXt-50 on ImageNet dataset in Table 5. As shown in the table, training a model with the channel regeneration technique results in better performance. In particular, network trained with $E_{RG} = 5$ records the best accuracy on ResNeXt model when compared model trained without the technique. For all the other architectures, we achieve the best performance when the regeneration epoch is set to 10 epochs. This improvement comes with no additional increase in model complexity or FLOP computation, which further establishes the efficacy of the technique.

Results on Quantized DNNs We assess the effectiveness of the *channel regeneration* technique on quantized neural networks. We adopt the mixed-precision approach for training these networks as this methodology is well-developed and accessible on existing frameworks. We evaluate three different architectures and report the results in Table 6. The results show that training with channel regeneration raise the accuracy of all quantized networks reported on three different datasets. On CIFAR-100, we boost the accuracy by 2% and 3% on MobileNetv2 and ShuffleNetv2 respectively. On some architectures, quantization has increased the accuracy of models when compared with non-quantized instances of architectures in Table 4 on TinyImage-Net. Channel regeneration improves the final accuracy by 4%, 2% and 3% on MobileNetv2, ShuffleNetv2 and EfficientNet respectively. On ImageNet, we increase the accuracy by over 0.4% with respect to the baseline.

Results on Pruned Models Here, we evaluate the channel regeneration technique on pruned models. We use the pruning methods described in NS (Liu et al. 2017), FPGM (He et al. 2019) and EB (You et al. 2020) to prune an original model to attain a pruned model. We integrate the channel regeneration technique on a pruned network during the finetuning step of the pruning pipeline and report the results in Table 7. We also provide the accuracy achieved without

Dataset	Model	Method	RG Epoch (E_{RG}) BL	3	5
CIFAR-100	Res-56	FPGM	69.00	**69.42**	69.33
ImageNet	Res-18	NS	66.28	**66.57**	66.53
		EB	67.15	67.26	**67.39**
		FPGM	66.11	**66.38**	66.35

Table 7: Results on Pruned Models

Backbone	Baseline	E_{RG}=1	E_{RG}=2
MobileNetv2	69.69	70.64 (**0.95**↑)	70.82 (**1.13**↑)
ResNet-18	56.06	58.12 (**2.06**↑)	56.89 (**0.83**↑)
ResNet-50	65.15	67.23 (**2.08**↑)	65.93 (**0.78**↑)

Table 8: Semantic segmentation results on ADE20K dataset

the integration of our technique in these methods (noted as BL in Table 7). On CIFAR-100, we observe that channel regeneration improves the accuracy attained with the FPGM method by 0.42. Similarly, our technique improves upon the accuracy attained by FPGM, NS, and EB on the ImageNet dataset.

4.3 Results on Semantic Segmentation

In this section, we assess the generalization of our channel regeneration technique on the semantic segmentation task. Similar to our previous approach to efficient architectures, we adopt a "plug and play" methodology in our experimental setup, where we apply channel regeneration at regular intervals during training. We follow the training and evaluation protocols described in (Zhou et al. 2016). We evaluate three different backbone architectures on the ADE20K dataset and report the results in Table 8. Each backbone network is pretrained on ImageNet. We train all the models for 20 epochs on the semantic segmentation task. The results show that our proposed technique can consistently improve performance. We report our results for for different values of E_{RG}.

4.4 Comparing Channel Regeneration with normalizing techniques

We compare our work with CE (Shao et al. 2020) and ITN (Huang et al. 2019) and report the results in Table 1. For our experiment, the baseline method implements the standard BN layer in the network. The methods CE and ITN are not applicable for off-the-shelf popular DNN configurations available on existing frameworks as these methods need to be reimplemented. We run the experiment for MobileNetv2 architecture on CIFAR-100 on our own and compute the accuracy. We report a higher accuracy with no increase in the total training time(TT).

5 Analysis

Channel regeneration increases channel utilization. We conduct experiments to demonstrate the efficacy of our technique by ablating channels and report on the network's

Method	BL	Acc	Parameters	TT
ITN	68.29	68.72	1x	5x
CE	68.29	68.86	1.2x	3x
Ours	68.29	**71.90**	1x	1x

Table 9: Comparison with normalizing techniques on MobileNetv2 on CIFAR-100 dataset [TT: Total Training Time]

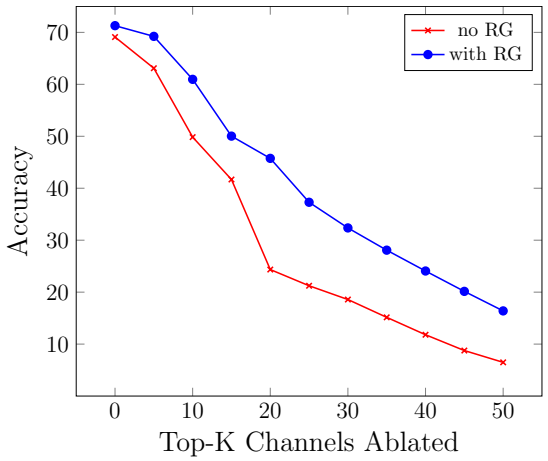

Figure 6: The graph plots the generalization accuracy against the number of top-k channels ablated with and without channel regeneration. Channel regeneration reports gradual drop in generalization accuracy and thus encouraging channels to contribute more equally to the network output.

generalization accuracy. The *generalization accuracy* is the classification accuracy reported on a held-out validation dataset (Zhou et al. 2015). Here, we consider a middle convolutional layer of size 96 of a trained MobileNetv2 model. First, we sort the channels from the convolutional layer according to their BN γ values. Then, we ablate the top-K channels by setting the weight and biases to 0, thus these ablated channels will not contribute to the prediction of any input image. We pass the images into the network and then compute the generalization accuracy. Figure 6 plots the generalization accuracy against the number of top-k channels ablated with and without channel regeneration. Under ideal conditions, the drop in generalization accuracy should be gradual as we increase the number of top-k channels ablated. Without channel regeneration, we notice that the accuracy drop is significant when we ablate any channels from the top-20 channels. In contrast, the channel regeneration achieves a gradual fall in accuracy. This indicates that our technique encourages all channels to contribute equally and be less reliant on a few selective channels.

Channel Regeneration improves average gradient statistics. Figure 7 plots the average gradient update of channels from the first layer of the MobileNetv2 network. Without channel regeneration, we note that the gradient update for several channels is zero for a large duration of training after the initial epochs. From the left graph, we observe that a few

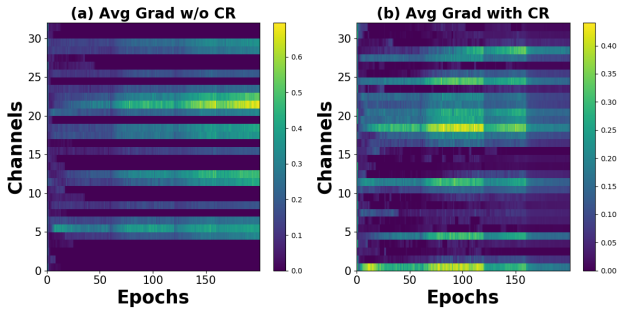

Figure 7: We show the average gradient statistics of all the channels from the first layer of the Mobilenetv2 network during training without and with channel regeneration (CR). In the left graph, we observe several inhibited channels after a few training epochs as their average gradient value drop to zero. From the plot on the right, we note that more channels receive regular gradient updates during training.

selective channels are chosen early to receive regular gradient updates and continue to do so throughout the training scheme. However, 50% of the channels receive no update, which limits their contribution to the final learned representation. The right graph plots the average gradient update when we apply channel regeneration at $E_{RG} = 5$ epoch. We note that more channels get a regular update during training, indicating that more channels would contribute to the network output. The percentage of inhibited channels is only 10% of the layer channels. Channel regeneration enforces the network to reduce the reliance on a few channels.

6 Conclusion

In this paper, we focus on revitalizing inhibited channels of compact architectures. To this end, we propose the *channel regeneration* technique that restores inhibited channels by re-initializing its batch normalization scaling factor. Channel regeneration increases model accuracy without increasing the total parameters or altering the structure of standard layers. Furthermore, we show that our novel channel regeneration technique increases channel contribution by reducing the reliance of a network on a few selective channels. Experimental results show that our technique improves the classification accuracy of various lightweight networks, quantized networks, and pruned models. Moreover, our channel regeneration exhibits good generalization ability in semantic segmentation tasks. Channel regeneration is easy to implement and can work with existing deep neural networks and hardware accelerators.

References

Baddar, W. J.; Han, S.; Rhee, S.; and Han, J. 2021. Self-Reorganizing and Rejuvenating CNNs for Increasing Model Capacity Utilization. *CoRR*.

Chen, L.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L. 2018. DeepLab: Semantic Image Segmenta-

tion with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.*

Devries, T.; and Taylor, G. W. 2017. Improved Regularization of Convolutional Neural Networks with Cutout. *CoRR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

He, Y.; Ding, Y.; Liu, P.; Zhu, L.; Zhang, H.; and Yang, Y. 2020. Learning Filter Pruning Criteria for Deep Convolutional Neural Networks Acceleration. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020,*.

He, Y.; Kang, G.; Dong, X.; Fu, Y.; and Yang, Y. 2018. Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*.

He, Y.; Liu, P.; Wang, Z.; Hu, Z.; and Yang, Y. 2019. Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

Huang, L.; Yang, D.; Lang, B.; and Deng, J. 2018. Decorrelated Batch Normalization. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*.

Huang, L.; Zhou, Y.; Zhu, F.; Liu, L.; and Shao, L. 2019. Iterative Normalization: Beyond Standardization Towards Efficient Whitening. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, JMLR Workshop and Conference Proceedings.

Li, Y.; Gu, S.; Mayer, C.; Gool, L. V.; and Timofte, R. 2020. Group Sparsity: The Hinge Between Filter Pruning and Decomposition for Network Compression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*.

Lin, M.; Ji, R.; Wang, Y.; Zhang, Y.; Zhang, B.; Tian, Y.; and Shao, L. 2020. HRank: Filter Pruning Using High-Rank Feature Map. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning Efficient Convolutional Networks through Network Slimming. In *IEEE International Conference on Computer Vision, ICCV*.

Luo, J.; Wu, J.; and Lin, W. 2017. ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression. In *IEEE International Conference on Computer Vision, ICCV 2017*.

Ma, N.; Zhang, X.; Zheng, H.; and Sun, J. 2018. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In *ECCV European Conference on Computer Vision*.

Mehta, D.; Kim, K. I.; and Theobalt, C. 2019. On Implicit Filter Level Sparsity in Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*.

Morcos, A. S.; Barrett, D. G. T.; Rabinowitz, N. C.; and Botvinick, M. M. 2018. On the importance of single directions for generalization. In *6th International Conference on Learning Representations, ICLR*.

Peng, H.; Wu, J.; Chen, S.; and Huang, J. 2019. Collaborative Channel Pruning for Deep Networks. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*.

Qiao, S.; Lin, Z.; Zhang, J.; and Yuille, A. L. 2019. Neural Rejuvenation: Improving Deep Network Training by Enhancing Computational Resource Utilization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

Radosavovic, I.; Kosaraju, R. P.; Girshick, R. B.; He, K.; and Dollár, P. 2020. Designing Network Design Spaces. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Redmon, J.; and Farhadi, A. 2017. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE Computer Society.

Ren, S.; He, K.; Girshick, R. B.; and Sun, J. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.*

Sandler, M.; Howard, A. G.; Zhu, M.; Zhmoginov, A.; and Chen, L. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

Shao, W.; Tang, S.; Pan, X.; Tan, P.; Wang, X.; and Luo, P. 2020. Channel Equilibrium Networks for Learning Deep Representation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015,*.

Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; and Le, Q. V. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Tan, M.; and Le, Q. V. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, Proceedings of Machine Learning Research.

Xie, S.; Girshick, R. B.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated Residual Transformations for Deep Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*.

You, H.; Li, C.; Xu, P.; Fu, Y.; Wang, Y.; Chen, X.; Baraniuk, R. G.; Wang, Z.; and Lin, Y. 2020. Drawing Early-Bird Tickets: Toward More Efficient Training of Deep Networks. In

8th International Conference on Learning Representations, ICLR 2020,.

Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2018. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

Zhou, B.; Khosla, A.; Lapedriza, À.; Oliva, A.; and Torralba, A. 2015. Object Detectors Emerge in Deep Scene CNNs. In *3rd International Conference on Learning Representations, ICLR*.

Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; and Torralba, A. 2016. Semantic understanding of scenes through the ade20k dataset. *arXiv preprint arXiv:1608.05442*.