Variational Convolutional Neural Network Pruning

Chenglong Zhao[1*] Bingbing Ni[1*†] Jian Zhang[1*] Qiwei Zhao[1] Wenjun Zhang[1] Qi Tian[2]
[1]Shanghai Jiao Tong University
[2]Huawei Noah's Ark Lab
{cl-zhao,nibingbing,stevenash0822,wwqqzzhi,zhangwenjun}@sjtu.edu.cn
tian.qi1@huawei.com

Abstract

We propose a variational Bayesian scheme for pruning convolutional neural networks in channel level. This idea is motivated by the fact that deterministic value based pruning methods are inherently improper and unstable. In a nutshell, variational technique is introduced to estimate distribution of a newly proposed parameter, called channel saliency, based on this, redundant channels can be removed from model via a simple criterion. The advantages are two-fold: 1) Our method conducts channel pruning without desire of re-training stage, thus improving the computation efficiency. 2) Our method is implemented as a stand-alone module, called variational pruning layer, which can be straightforwardly inserted into off-the-shelf deep learning packages, without any special network design. Extensive experimental results well demonstrate the effectiveness of our method: For CIFAR-10, we perform channel removal on different CNN models up to 74% reduction, which results in significant size reduction and computation saving. For ImageNet, about 40% channels of ResNet-50 are removed without compromising accuracy.

1. Introduction

Deep convolutional neural networks have achieved a significant success in computer vision community, such as object recognition [13, 25, 37, 44], semantic segmentation [2, 28], object detection [9, 26] and video analysis [45, 46]. Due to the huge storage and computation costs, deep models are difficult to implement on resource-constrained platforms, such as, mobile and wearable device.

To solve this problem, various methods have been proposed to improve the efficiency of CNN models or to compress the models into more compact representations. These approaches include tensor factorization [35, 48], network quantization [4, 40], and weight pruning [11, 32]. Though theoretically plausible, these methods usually require specific software or hardware implementation, which inevitably introduce extra overhead, *i.e.*, not practically applicable. Some other methods [15, 30] obtain compact models with hand-crafted manner, which need manual intervention and suffer from compromise performance.

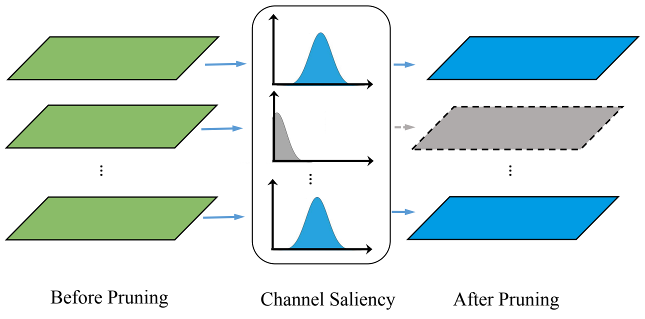


Figure 1. **Variational CNN Pruning**: We prune redundant channel based on distribution of channel saliency γ. Dotted line denotes pruned channel and solid line denotes preserved one. Best view in color.

An alternative method is channel pruning [14, 27, 29, 43], which removes redundant channels in models. Channel pruning operations are fully supported by off-the-shelf deep learning library, thus they are very flexible in practical implementation. Moreover, channel pruning reduces memory footprint dramatically (*i.e.*, feature maps). Recent methods [14, 29] remove one layer by minimizing the reconstruction error between the consecutive layers. However, these methods adopt the greedy algorithm for channel selection, which is time-consuming, as the computation complexity is linearly proportional to the layers of model. Sparsity based methods [23, 43] impose L_p norm on a group of weights, for channel shrinkage. However, despite their favorable performance, approaches for channel pruning have inherent drawbacks. 1) **Complexity**: most of these methods need extra re-training stage for performance restoration. Even some need

*Equal contribution
†Corresponding author

to conduct several iterations of alternative pruning and re-training, which is very time-consuming and not friendly for operation in practice. 2) **Stability**: layer-alternative parameter optimization approach inevitably suffers from instability. Namely, parameters of hidden layers may change drastically during consecutive iterations. This results in arbitrary removal of some channels during optimization, which is NOT interpretable. This eventually causes unexpected performance degradation after pruning.

To address aforementioned issues, we re-formulate the channel pruning problem within a Bayesian probabilistic learning framework (*i.e.*, in contrast to previous deterministic methods), called *variational CNN pruning*, to yield compact, stable, interpretable and flexible compression. To avoid introducing additional parameters, the proposed probabilistic pruning framework directly operates on re-defined scaling factor in Batch Normalization (BN). Different from [27], we extend the scaling factor to include bias term, called *channel saliency*, and still keep the linear form of Batch Normalization. Instead of deterministic value computation, we regard each channel saliency as a random variable and seek a proper probabilistic distribution to model it.

Thus, unimportant layers could be pruned based on the distribution of the channel saliency. However, directly estimating these distributions involves intractable multidimensional integral in Bayesian manner. We therefore introduce a stochastic variational inference [20] to estimate the distribution of channel saliency induced by a sparse prior. Our Bayesian framework possesses the following advantages. First, by directly applying probabilistic learning on the scaling factor (*i.e.*, our method can also be represented as a *variational-pruning* layer), no additional parameter is introduced in deep CNN model learning, therefore ours can be directly plugged into any off-the-shelf deep learning package, without any requirements for special network component design. Second, the proposed method conducts channel pruning with no desire of re-training step. Therefore, computational efficiency is improved. Third, being formulated in a probabilistic way, optimization process becomes stable and interpretable. Extensive experiments on CIFAR-10 show that we can prune up to 74% channels with little accuracy loss. On ImageNet, the ResNet-50 achieves better performance than baseline while about 40% channels are removed.

2. Related Work

Tensor Decomposition. Jaderberg *et al.* [19] obtain CNNs substantial speedups via low-rank decomposition. Denil *et al.* [5] remove redundant parameters in deep learning model by using a few weight values to represent each feature map, which saves a large number of the memory and computation consumption. Techniques of matrix decomposition, SVD, is introduced to approximate the weight matrix in neural networks in [6]. In [42], Tensor Ring (TR) factorization technology is applied to compress deep neural networks. However, these methods are impracticable, because of involving the computation-expensive decomposition operations.

Network Quantization. Courbariaux *et al.* [4] propose a method to quantize the network weight with binary value. To obtain significant improvement for computation efficiency, Gupta *et al.* [10] leverage stochastic rounding to represent weight by a 16-bit wide fixed-point. In [3], network weights are hashed into different groups. For each group, shared weights are saved with same hash index. To further compress networks, Tung *et al.* [40] combine weight pruning and quantization in a single learning framework. Rastegari *et al.* [36] propose XNOR-Networks which impose XNOR and bitcount operations on convolution layer to quantize weights. Park *et al.* [34] propose a method for quantizing weights and activations based on weighted entropy. These low-bit approximation methods usually suffer form accuracy loss.

Non-structured Pruning. Inspired by neurobiology, optimal brain damage [22] and optimal brain surgeon [12] are proposed to remove parameters in networks for saving storage through an analysis of the Hessian of the loss function. However, these heuristic based methods are computationally-intensive and cannot boost running time. Han *et al.* [11] judge the importance of weights in network based on magnitude and remove redundant ones with small value. A data-free algorithm is proposed in [39] to eliminate the superfluous neurons in fully-connected layers. Lebedev *et al.* [21] boost ConvNets speed by group-wise sparsity operations. Wang *et al.* [41] propose a density-diversity penalty on weight to compress fully-connected networks. Pavlo *et al.* [32] remove unimportant kernels based on brute-force search and restore the performance of compressed networks via fine-tuning. These non-structured pruning methods [22,24,38] need special software and hardware to speedup inference, due to the irregular sparsity in weight tensor.

Structured Pruning. Wen *et al.* [43] exploit CNNs compression at different grains via imposing L_1 norm on a group of weights. Li *et al.* [23] propose an one-shot pruning and retraining strategy to compress filters across multiple layers. Liu *et al.* [27] leverage the scale factor of Batch Normalization layer [18] to remove non-efficient channels and fine-tune the pruned model to restore the comparable accuracy. He *et al.* [14] propose a method that reduces redundant channels based on LASSO regression and least square reconstruction. [29, 47] transform network pruning into an optimization problem and perform channel selection in a layer-wise manner via greedy algorithms. All these structured pruning methods are hardware friendly, i.e., boosting networks inference directly. However, most of these pruning approaches [1, 7, 16] need re-training stage to restore comparable accuracy, which increases computational complexity and is not friendly for operation. In contrast to these, our method conducts channel pruning with no desire of re-training stage, while keeping favorable performance in both compression and speedup.

3. Variational CNN Pruning

In this section, we propose a variational Bayesian scheme for channel pruning in convolutional networks: *Variational CNN Pruning*. First, we reformulate the Batch Normalization layer, by extending scale factor to shift term, called *channel saliency*. Instead of deterministic value, we estimate the distribution of channel saliency via variational inference. To further facilitate pruning, a sparsity-inducing prior is imposed on the channel saliency. Then we prune redundant channels based on distributions of channel saliency via a simple criterion. Our method is implemented as a *variational-pruning* layer, which can be inserted into any existing framework without special design.

3.1. Batch Normalization Revisit

Batch Normalization (*BN*) [18] has been introduced in current convolutional neural networks as a standard layer, which improves training speed and accelerates convergence process. This great performance benefits from normalizing activation distribution by using the mini-batch statistics (*i.e.*, σ_B and μ_B). x_{in} and x_{out} are represented as input and output activation map, respectively, and *B* denotes the mini-batch. *BN* layer performs normalization as follows:

$$BN(x) = \frac{x_{in} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}; x_{out} = \gamma \cdot BN(x) + \beta \qquad (1)$$

where μ_B and σ_B denote the mean and standard deviation of input activations over *B*. An affine transformation is applied on normalized activations by a linear function learnable parameter, *i.e.,* scale factor γ and shift factor β.

Batch Normalization (*BN*) has been inserted between convolution and non-linearity layer, constituting a basic cell in modern networks. γ and β are channel-wise parameters on models, with function of scaling and shifting. Parameter γ can represent the effectiveness coefficient of corresponding channels and no parameter need to be introduced. Because of this characteristic, it makes sense to choose the parameter γ as a factor to indicate the importance of convolution channel [27]. However, two significant issues are ignored here:

1) we notice the fact that parameters γ and β are independent in *BN*. Thus, eliminating unimportant channel with small γ's value is an improper manner, due to ignoring the influence of shift term. For example, a channel has zero value of γ but with a big value of β. Pruning this channel is arbitrary, because activations of this channel are not zero and still contribute effect to the next layer.

2) The value of parameter in hidden layer changes drastically after several iterations in training stage. This means that the value (*e.g.*, scale factor) is dynamical. Determining the effect of layer by only observing a value of the parameter is not sufficient. We argue that this is one significant cause for performance degradation after pruning [27].

To remedy this, we reformulate Batch Normalization layer in Equation 1 as follows:

$$x_{out} = \gamma \cdot BN(x) + \tilde{\beta}, \qquad (2)$$

$$\text{where, } \tilde{\beta} = \gamma \cdot \beta. \qquad (3)$$

We extend the scale factor γ on shift term β, while still maintaining affine function of *BN*. In the case, the parameter γ can be directly utilized as factors on channels, called *channel saliency*, because activation of each channel is totally depended on γ. Rather than through the value of γ, we prune unimportant channels based on the distribution of γ, which is estimated by variational inference (more details in Sec 3.2). We do not introduce any parameters in *BN* layer and the reformulated *BN* layer can be inserted into any existing frameworks.

3.2. Variational Inference on Channel Saliency

Instead of deterministic value, we prune channel with the distribution of channel saliency. Obviously, this method is more stable and interpretable, because the distribution contains rich information and has good mathematical property. Thus, we estimate distribution of channel saliency via Bayes rule. More details are discussed as follows:

Consider a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, x is input data and y is corresponding label. Our goal is to learn a model with parameter γ of the conditional probability $p(y|x, \gamma)$. Parameter γ is *channel saliency* defined in previous section and we leverage γ to determine the effect of each channel. After obtaining the prior knowledge of γ (prior distribution), we can inference the posterior distribution of γ with Bayes rule. However, computing such a posterior distribution $p(\gamma|\mathcal{D}) = p(\gamma)p(\mathcal{D}|\gamma)/p(\mathcal{D})$ is difficult, because the $p(\mathcal{D}) = \int p(\mathcal{D}, \gamma)d\gamma$ is a computation-intractable integral. It is hard to obtain distribution of channel saliency directly. Hence, an effectively approximated method, variational inference, is introduced to tackle this problem. Compared to MCMC, variational inference is a good choice, benefiting from its solidly theoretical property and small computation cost.

In variational inference, instead of computing the true posterior distribution directly, we assume a parameterized distribution $q_\phi(\gamma)$ to approximate $p(\gamma|\mathcal{D})$. By this way, the unsolvable inference problem is cast to a tractable optimization problem. We can estimate the posterior distribution by minimizing the distance between $q_\phi(\gamma)$ and the true posterior distribution $p(\gamma|\mathcal{D})$ by Kullback-Leibler divergence, *i.e.*, $\min_\phi D_{KL}(q_\phi(\gamma)||p(\gamma|\mathcal{D})$. Minimizing the KL divergence is equivalent to maximizing the evidence lower bound (*ELBO*) as follows:

$$\mathcal{L}(\phi) = L_{\mathcal{D}}(\phi) - D_{KL}(q_\phi(\gamma)||p(\gamma)), \tag{4}$$

$$\text{where, } \mathcal{L}_{\mathcal{D}}(\phi) = \sum_{(x,y)\in\mathcal{D}} \mathbb{E}_{q_\phi(\gamma)}[\log p(y|x,\gamma)]. \tag{5}$$

The object function consists of two terms, expected log-likelihood $L_{\mathcal{D}}(\phi)$ and KL divergence. The $L_{\mathcal{D}}(\phi)$ is a reconstruction term which aims to maximize the probability of the model prediction, *e.g.*, minimizing the sum of prediction error. The KL divergence term is a regularization term, where a sparse prior will be introduced as sparsity-induced penalty on the *channel saliency* γ. Optimizing the two terms, we can take into account the performance and compression simultaneously. The trade-off between the two terms leads to a compact and effective model.

Due to the expectation in Equation 5, the gradient can not be computed directly. Following [8, 20, 31], we introduce Reparametrization Trick to obtain an unbiased differentiable minibatch-based Monte Carlo estimator of expected log-likelihood. M is the mini-batch size and N is the number of data. In this way $q_\phi(\gamma)$ can be represented as a differentiable function $\gamma = f(\phi, \epsilon)$, where $\epsilon \sim \mathcal{N}(0, 1)$. Then Equation 5 can be reformulated as follows:

$$\mathcal{L}_{\mathcal{D}}(\phi) \simeq \mathcal{L}^{\mathcal{A}}_{\mathcal{D}}(\phi) = \frac{N}{M}\sum_{m=1}^{M} \log p(y_{im}|x_{im}, \gamma_{im} = f(\phi, \epsilon)), \tag{6}$$

$$\mathcal{L}(\phi) \simeq \mathcal{L}^{\mathcal{A}}_{\mathcal{D}}(\phi) - D_{KL}(q_\phi(\gamma)||p(\gamma)). \tag{7}$$

In this way, we can solve optimization problem (Eqn. 5) by an approximation manner.

Let $\mathbf{w}$ be the weights of the neural networks. The model can be represented as $p(y|x, \mathbf{w}, \gamma)$, which is conditional on $\mathbf{w}$. Therefore, we can still keep the the evidence lower bound (*ELBO*) as object function:

$$\mathcal{L}(\phi, \mathbf{w}) \simeq \mathcal{L}^{\mathcal{A}}_{\mathcal{D}}(\phi, \mathbf{w}) - D_{KL}(q_\phi(\gamma)||p(\gamma)). \tag{8}$$

Optimizing the object function is to obtain the approximate distribution of channel saliency $q_\phi(\gamma)$ and networks weights $\mathbf{w}$. The model can be trained in an end-to-end manner [33].

3.3. KL-divergence with Sparse Prior

As shown in Equation 8, KL-divergence between the posterior and prior distributions need to be computed. A common choice of the approximate posterior is a fully factorized Gaussian distribution, formulated as follows:

$$q_\phi(\gamma) = \prod_{i=1}^{C} q(\gamma_i), \quad \gamma_i \sim \mathcal{N}(\mu_i, \sigma_i). \tag{9}$$

Our goal is to fine-tune the learnable parameters $\phi = (\mu, \sigma)$ with the object function. In order to remove channels, the approximate distribution $q_\phi(\gamma)$ should be sparse. Then the inefficient channels can be determined easily. Namely, we eliminate channels based on the mean and variance of distribution of channel saliency γ. Thus, we introduce a prior distribution as follows:

$$p(\gamma) = \prod_{i=1}^{C} p(\gamma_i), \quad \gamma_i \sim \mathcal{N}(0, \sigma_i^*), \tag{10}$$

where we fix mean to zero value. Thus, induced by this sparse prior, the channel saliency is encouraged to towards zero. Then the KL-divergence in *ELBO* can be calculated tractable as below:

$$\begin{aligned} D_{KL}(q_\phi(\gamma)||p(\gamma)) &= \sum_i D_{KL}(q_\phi(\gamma_i)||p(\gamma_i)) \\ &= \sum_i \log\frac{\sigma_i^*}{\sigma_i} + \frac{\sigma_i^2 + \mu_i^2}{2(\sigma_i^*)^2} - \frac{1}{2}. \end{aligned} \tag{11}$$

The reason of making this choice is summarized as below:

1) The selected prior distribution has sparse property which can encourage parameters γ towards zero. According to this, we can straightforwardly prune ineffective layers based on γ .

2) There is no distribution gap between $q_\phi(\gamma)$ and $p(\gamma)$. The KL divergence will be zero, when the expectation μ and variance σ of two distributions have same value. This character guarantees that the γ can be calculated accurately.

3) The KL divergence $D_{KL}(q_\phi(\gamma)||p(\gamma))$ can be computed tractable, because both distributions belong to Gaussian. Other sparsity distributions, such as Laplace distribution or log-uniform distribution, also can be used as prior distribution. However, we cannot obtain closed-form solution of the KL-divergence, due to the involved intractable integrals. Although some methods of numerical estimation can be used to estimate the KL term, this will introduce inevitable errors.

Let the variance of the both distributions to be identical, Equation 11 can be simplified as follows:

$$D_{KL}(q_\phi(\gamma)||p(\gamma)) = \sum_i k\mu_i^2 \tag{12}$$

Where k is a coefficient and inversely proportional to variance. In this case, the mean value (*i.e.,* μ) of the approximated distribution is encouraged to be small, due to imposing a L_2 norm. Hence, we can discard corresponding channels safely.

3.4. Variational Pruning on Channels

We optimize *ELBO* with the KL-divergence mentioned above to obtain the distribution of channels salience γ, where $\gamma \sim q(\gamma|\phi = (\mu, \sigma))$. Then we remove redundant channel based on the following criterion.

Based above section, obtained channel saliency γ obeys a gaussian distribution. Consider to the centrality property of gaussian, samples distribute around the expectation. When the expectation μ is close to zero and variance is small, the probability of variable γ is close to zero. Based on this idea, we eliminate redundant channels when the optimized parameters are less than thresholds, *i.e.*, $(\mu, \sigma) < (\tau, \theta)$. The pseudocode of variational constitutional neural networks pruning is illustrated in Algorithm 1.

Algorithm 1 Variational CNN Pruning

```
Input: $\mathcal{N}$ pairs data $\{(x_i, y_i)\}_{i=1}^{N}$, $C$ channels $\{\gamma_i\}_{i=1}^{C}$
Output: $\phi, \mathbf{w}$
 1: for epoch= 1 to $K$ do
 2:     $q_\phi(\gamma) = \prod_{i=1}^{C} q(\gamma_i)$
 3:     $\gamma_i \sim \mathcal{N}(\mu_i, \sigma_i)$
 4:     $\mathcal{L}(\phi, \mathbf{w}) \simeq \mathcal{L}_{\mathcal{D}}^{\mathcal{A}}(\phi, \mathbf{w}) - D_{KL}(q_\phi(\gamma)||p(\gamma))$.
 5:     Optimize : $\mathcal{L}(\phi, \mathbf{w})$
 6:     Update Parameters
 7:     for $i = 1$ to $C$ do
 8:         if $u_i < \tau$, $\sigma_i < \theta$ then
 9:             Pruning the $i$-channel
10:         end if
11:     end for
12: end for
```

The algorithm is implemented in *BN* as a special layer, called variational pruning layer. We do not introduce extra parameter here and all operations can be integrated in this layer. The variational pruning layer is easy to implement as a separate module, which can be inserted to any existing framework.

4. Experiments

In this section, we carry out extensive experiments to evaluate the performance of the proposed method on image classification task. Three representative networks, including VGG Net [37], DenseNet [17], and ResNet [13], are chosen for compression. We report the performances on CIFAR and ImageNet datasets, and compare to state-of-the-arts. All these experiments demonstrate the effectiveness of our method.

4.1. Implementation Details

Training Strategy. For CIFAR, the learning rate is set as 0.1, and divided by 10 at the 150 and 240 epochs, respectively. We train networks for 300 epochs and with batch size of 256. For ImageNet, the learning rate is set as 0.1, and divided by 10 at 60 and 90 epochs. We train them with batch size of 256 and 120 epochs for total. All these networks are optimized by stochastic gradient decent (SGD), with nesterov momentum 0.9 and weight decay 10^{-4}. Random flip and crop are applied for data augmentation on CIFAR and ImageNet datasets.

Compression Metric. Channels, Parameter and FLOPs (floating point operations) are used to measure network compression. Channels indicate the memory footprint. Parameter and FLOPs denote the storage space and computation cost, respectively. In this paper, we only count parameter and FLOPs over convolutional layer, because the proposed method focuses on channel-level compression of convolutional neural networks. In addition, we only count multiply operation for FLOPs.

Pruning Details. All models are trained from scratch as baseline. We prune unimportant channels based on the distribution of the proposed channel saliency. When the parameters of the distribution are less than thresholds, the corresponding channel will be removed from the model. The thresholds τ and θ are empirically set as 0.02 and 0.01, respectively. We set the variance of prior and approximated posterior to be identical for simplifying the KL loss, which benefits for training [8]. Different from prior arts, the proposed method does not need fine-tuning to refine the pruned model.

4.2. Results on CIFAR-10

The purpose of our approach is pruning redundant channels for saving storage space and computation consumption. We conduct our experiments on CIFAR-10 dataset with three classic deep networks: VGG, DenseNet and ResNets. Channels, parameters and FLOPs are used to measure the performance of the pruning models. The experiment results are reported in Table 1.

VGG Net. For VGG net, the 16-layer (13-Conv + 3FC) model is adopted to perform on CIFAR-10 dataset. We remove 62% channels while keeping the accuracy at 93.18%, which is slightly lower than baseline. Notably, more than 70% of the parameters are reduced and nearly 40% computation is saved. This greatly facilitates VGG model, popular backbone for object detection and semantic segmentation, to deploy on mobile devices.

DenseNet. For DenseNet, we remove 60% inefficient channels with only 1% drop of accuracy. This is extremely meaningful for DenseNet which consumes a vast amount of memory, because the reduction of channels will decrease memory footprint directly. We also save more than half stor-

Model	Accuracy	Channels	Pruned	Parameters	Pruned	FLOPs	Pruned
VGG-16 Base	93.25%	4224	-	14.71M	-	313M	-
VGG-16 Pruned	93.18%	1599	62%	3.92M	73.34%	190M	39.10%
DenseNet-40 Base	94.11%	9360	-	1.04M	-	282M	-
DenseNet-40 Pruned	93.16%	3705	60%	0.42M	59.67%	156M	44.78%
ResNet-20 Base	92.01%	1808	-	0.21M	-	8.9M	-
ResNet-20 Pruned	91.66%	1114	38%	0.17M	20.41%	7.5M	16.47%
ResNet-56 Base	93.04%	4496	-	0.57M	-	22.3M	-
ResNet-56 Pruned	92.26%	2469	45%	0.46M	20.49%	17.8M	20.30%
ResNet-110 Base	93.21%	8528	-	1.12M	-	42.4M	-
ResNet-110 Pruned	92.96%	3121	63%	0.66M	41.27%	26.9M	36.44%
ResNet-164 Base	93.58%	12560	-	1.68M	-	62.4M	-
ResNet-164 Pruned	93.16%	3238	74%	0.73M	56.70%	31.8M	49.08%

Table 1. Accuracy and pruning ratio on CIFAR-10. We count pruned channels, parameters and FLOPs over different deep models, and the accuracy of pruned models are reported without retraining stage. We train these models from scratch without pruning as baseline in our experiments.

Model	Accuracy	Channels	Pruned	Parameters	Pruned	FLOPs	Pruned
VGG-16 Base	73.26%	4224	-	14.71M	-	313M	-
VGG-16 Pruned	73.33%	2883	32%	9.14M	37.87%	256M	18.05%
DenseNet-40 Base	74.64%	9360	-	1.04M	-	282M	-
DenseNet-40 Pruned	72.19%	5851	37%	0.65M	37.73%	218M	22.67%
ResNet-164 Base	75.56%	12560	-	1.68M	-	62.4M	-
ResNet-164 Pruned	73.76%	6681	47%	1.38M	17.59%	45.4M	27.16%

Table 2. Accuracy and pruning ratio on CIFAR-100. We count pruned channels, parameters and FLOPs on VGG-16, DenseNet-40 and ResNet-164.

age space by reducing 60% of parameters and boost computation by decreasing about 45% FLOPs. Although structure of DenseNet is pretty compact, our method still exerts pruning on channels. We consider this benefits from the proposed variational pruning method.

ResNets. For ResNets, we adopt four different structures on CIFAR-10, including ResNet-20, ResNet-56, ResNet-110, and ResNet-164. ResNets for CIFAR-10 have three stages of residual block, and each stages followed by downsampling layer to resize the scale of feature maps. As shown in the table 1 and Figure 2, we note that the pruned channels increase obviously form ResNet-20 to ResNet-164, and the reduction ratio raises from 38% to 74%. The proposed method has achieved notable result on ResNet-164, by removing 57% parameters and 49% FLOPs. As a common choice, we increase the layers to improve the performance of models. However, this operation will involve more redundant layers at the same time. The purpose of our method is eliminating redundant channels. Therefore, with more layers, the effect of our method is more obvious.

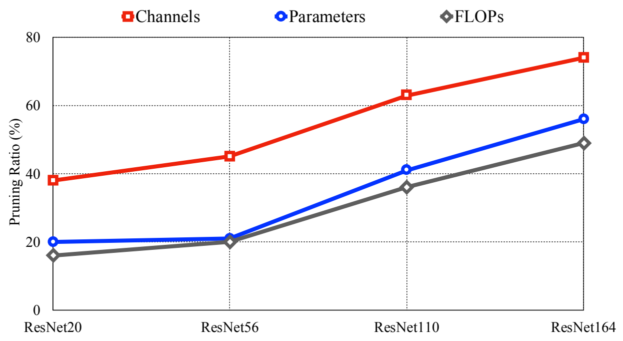
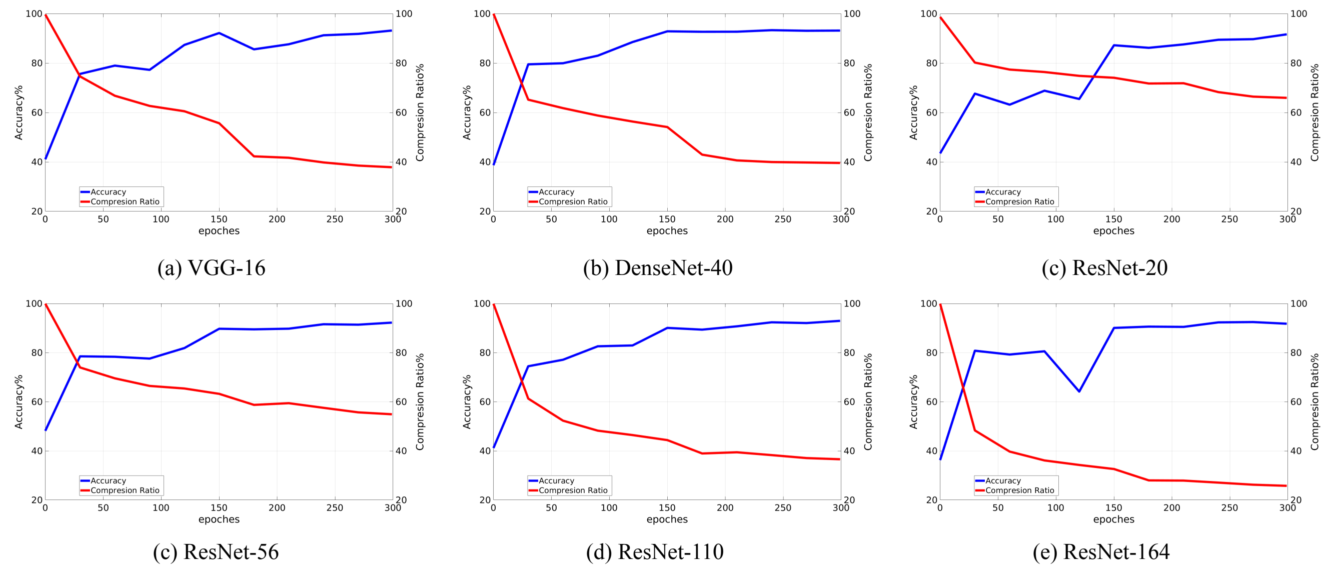


Figure 2. ResNets performed on CIFAR-10. We compare channels, parameters and FLOPs on four different layers of ResNet network. Best view in color.

4.3. Results on CIFAR-100

As illustrated in Table 2, the proposed method is evaluated on CIFAR-100 for three deep networks, *e.g.*, VGG-16, DenseNet-40 and ResNet-164. We note that our method obtain the better performance on deeper model. The removed



(a) VGG-16 (b) DenseNet-40 (c) ResNet-20

(c) ResNet-56 (d) ResNet-110 (e) ResNet-164

Figure 3. Compression and Accuracy Curves. We show the details of compression process about VGG-16, DenseNet-40 and ResNets. Best view in color.

Model	Top-1	Top-5	Channels	Pruned
ResNet-50 Base	75.1%	**92.8%**	26560	-
ResNet-50 [29]	72.8%	91.1%	18592	30%
ResNet-50 Ours	**75.2%**	92.1%	15920	**40%**

Table 3. Performance and Comparison on ImageNet.

Dataset	Model	Accuracy	Channels	Params
CIFAR-10	Dense40* [27]	89.5%	60%	54%
	Denset40 Ours	**93.1%**	60%	**59%**
	Res164* [27]	47.7%	60%	34%
	Res164 Ours	**93.1%**	**74%**	**56%**
CIFAR-100	Dense40* [27]	67.7%	**40%**	35%
	Dense40 Ours	**72.1%**	37%	**38%**
	Res164* [27]	48.0%	40%	13%
	Res164 Ours	**73.7%**	**47%**	**17%**

Table 4. Comparison with other method on CIFAR dataset. $*$ denotes pruning without fine-tuning stage.

channels raise from 32% in VGG-16 to 47% in ResNet-164. We consider deep models contain more redundant channels, and the proposed method is sensitive to these. Removing these non-essential channels is useful for saving memory, storage and computation.

4.4. Results on ImageNet

ImageNet dataset is a large scale image recognition benchmark. The dataset contains 1.2 million images for training and 50000 images for validation. All these images come from 1000 different categories.

To further evaluate the effect of variational pruning on large scale dataset, we perform our method on ImageNet dataset for ResNet-50. From Table 3, we note that the pruned model performs better than baseline in Top-1 accuracy, while 40% channels have been removed. This demonstrates the pruned model is compact and efficient, and is meaningful for reducing memory footprint. Compared to [29], our methods obtains higher accuracy and eliminates more channels.

4.5. Comparison with Other Method

As shown in Table 4, we compare the proposed method with [27], which utilizes the deterministic value of scale factor to remove channels. Unlike this, we eliminate unimportant channels based distributions and do not need fine-tuning stage to restore performance.

To keep comparison fair, we use the results of [27] with no fine-tuning stage. For ResNet-164, we exceed [27] almost two times of accuracy and obtain higher compression performance in Parameters and FLOPs on CIFAR-10. After pruning, the accuracy of [27] drops drastically. The poor performance suffers from eliminating unimportant channels based on deterministic value. Because weights in hidden layers change straightly during training process, removing channels based on deterministic value is arbitrary. In contrast to this, our method is more robust and maintains accuracy after compression, which benefits from pruning based on distribution.

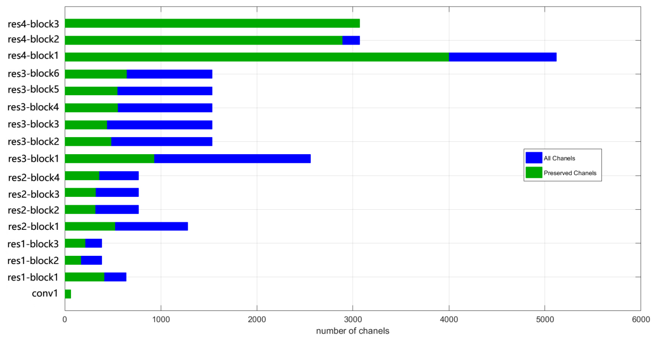


Figure 4. ResNet-50 on ImageNet. Statistics of preserved channels in 16 residual blocks and the first convolutional layer. Best view in color.

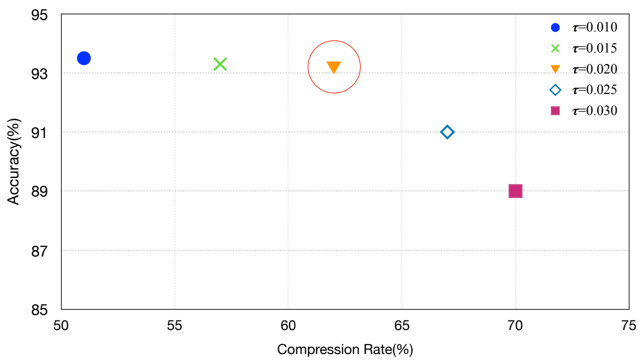


Figure 5. Sensitivity Analysis. We report compression rate and accuracy at different value of threshold τ. Best view in color.

4.6. Analysis

4.6.1 Compression and Accuracy

As illustrated in Figure 3, accuracy and compression curves are given about 6 networks, which perform on CIFAR-10 dataset. The compression ratio is defined as follows:

$$\#compression\ rate = \frac{\#preserved\ channels}{\#all\ channels} \tag{13}$$

We note that the accuracy is increasing, while the compression rate is decreasing. The proposed method removes redundant channels and improves accuracy at the same time. We consider this benefits from the loss (Eqn 8). Namely, the KL term focuses on channel shrinking and the expect-log term updates the weights for prediction. The trade-off, between the two constraint item, leads the model to be a compact and effective one.

4.6.2 Channel Pruning Analysis

As shown in Figure 4, we reveal the pruning details about ResNet-50 on ImageNet dataset. We account the preserved channels about 16 residual-blocks and the first convolutional layer. The pruned channels mainly concentrate on the middle stage of the model, and channels have few reduction at both ends. This phenomenon is caused by that channels at the last block contain abundant semantic information, which is crucial for image recognition. So it is hard to remove these channels. Interlayer-channels include more detailed information. Some of these are non-efficient and redundant. Removing these channels has little impact on the performance of model. This proves that our variational pruning method could determine the importance of channels, namely, maintaining efficient channels while removing superfluous channels. The superior character of our method leads to the result that removing a large number of channels with considerable margin of baseline, and further demonstrates the fact that some of parameters and activations in deep model are invalid and cumbersome.

4.6.3 Sensitivity Analysis

We choose 5 different value of parameter τ (threshold for mean), and perform VGG-16 on CIFAR-10 with these parameters. As illustrated in Figure 5, with the increase of τ, the compression rate is increased from 51% to 70% and accuracy is dropped down from 93.5% to 89%. This means more compact model will sacrifice more accuracy. When we tune the τ below 0.02, a little accuracy is obtained but the compression rate drops drastically. When we turn up the threshold for smaller size model, the performance becomes worse. Therefore, to achieve a trade-off between compression and accuracy, we empirically set τ as 0.02.

5. Conclusion

We propose a variational pruning method for removing unimportant channels in convolutional neural networks. We reformulate the Batch Normalization layer to obtain a new parameter, called channel saliency, which can be leveraged to measure the effect of channel. To avoid the deterministic value, we estimate the distribution of channel saliency via Bayes rule, then a simple yet effective criterion is used to prune redundant channels. Our method conducts channel pruning with no desire of re-training stage and can be implemented as separated module which is flexible and transplantable. The extensive experiments on CIFAR and ImageNet demonstrate the outstanding performance of the proposed method.

Acknowledgements

This work was supported by National Science Foundation of China (U1611461,61521062). This work was partly supported by STCSM(18DZ1112300,18DZ2270700). This work was also partially supported by joint research grant of SJTU-BIGO LIVE, joint research grant of SJTU-Minivision, and China's Thousand Talent Program.

References

[1] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems*, 13(3):32, 2017.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

[3] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pages 2285–2294, 2015.

[4] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[5] Misha Denil, Babak Shakibi, Laurent Dinh, Nando De Freitas, et al. Predicting parameters in deep learning. In *Advances in neural information processing systems*, pages 2148–2156, 2013.

[6] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014.

[7] Abhimanyu Dubey, Moitreya Chatterjee, and Narendra Ahuja. Coreset-based neural network compression. In *Proceedings of the European Conference on Computer Vision*, pages 454–470, 2018.

[8] Patrick Esser, Ekaterina Sutter, and Björn Ommer. A variational u-net for conditional appearance and shape generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8857–8866, 2018.

[9] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[10] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pages 1737–1746, 2015.

[11] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[12] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171, 1993.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

[15] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[16] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *International Conference on Learning Representations*, 2016.

[17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 448–456. JMLR. org, 2015.

[19] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference. BMVA Press*, 2014.

[20] Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

[21] Vadim Lebedev and Victor Lempitsky. Fast convnets using group-wise brain damage. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2554–2564. IEEE, 2016.

[22] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[23] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *International Conference on Learning Representations*, 2017.

[24] Chen Lin, Zhao Zhong, Wu Wei, and Junjie Yan. Synaptic strength for convolutional neural network. In *Advances in Neural Information Processing Systems*, pages 10170–10179, 2018.

[25] Jinxian Liu, Bingbing Ni, Yichao Yan, Peng Zhou, Shuo Cheng, and Jianguo Hu. Pose transferrable person reidentification. In *CVPR*, pages 4099–4108, 2018.

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[27] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017.

[28] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[29] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[30] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision*, pages 116–131, 2018.

[31] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org, 2017.

[32] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *International Conference on Learning Representations*, 2017.

[33] Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, and Dmitry P Vetrov. Structured bayesian pruning via log-normal multiplicative noise. In *Advances in Neural Information Processing Systems*, pages 6775–6784, 2017.

[34] Eunhyeok Park, Junwhan Ahn, and Sungjoo Yoo. Weighted-entropy-based quantization for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[35] Bo Peng, Wenming Tan, Zheyang Li, Shun Zhang, Di Xie, and Shiliang Pu. Extreme network compression via filter group approximation. In *Proceedings of the European Conference on Computer Vision*, pages 300–316, 2018.

[36] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

[38] Sanghyun Son, Seungjun Nah, and Kyoung Mu Lee. Clustering convolutional kernels to compress deep neural networks. In *Proceedings of the European Conference on Computer Vision*, pages 216–232, 2018.

[39] Suraj Srinivas, R Venkatesh Babu, and Supercomputer Education. Data-free parameter pruning for deep neural networks.

[40] Frederick Tung and Greg Mori. Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7873–7882, 2018.

[41] Shengjie Wang, Haoran Cai, Jeff Bilmes, and William Noble. Training compressed fully-connected networks with a density-diversity penalty. 2016.

[42] Wenqi Wang, Yifan Sun, Brian Eriksson, Wenlin Wang, and Vaneet Aggarwal. Wide compression: Tensor ring nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9329–9338, 2018.

[43] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016.

[44] Yichao Yan, Bingbing Ni, Zhichao Song, Chao Ma, Yan Yan, and Xiaokang Yang. Person re-identification via recurrent feature aggregation. In *ECCV*, pages 701–716, 2016.

[45] Yichao Yan, Bingbing Ni, and Xiaokang Yang. Predicting human interaction via relative attention model. In *IJCAI*, pages 3245–3251, 2017.

[46] Yichao Yan, Jingwei Xu, Bingbing Ni, Wendong Zhang, and Xiaokang Yang. Skeleton-aided articulated motion generation. In *ACM MM*, pages 199–207, 2017.

[47] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018.

[48] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7370–7379, 2017.