Variational Automatic Channel Pruning Algorithm Based on Structure Optimization for Convolutional Neural Networks

Shuo Han[1], Yufei Zhan[2], Xingang Liu[1]

[1] School of Information and Communication Engineering, University of Electronic Science and Technology of China, China
[2] Glasgow College, University of Electronic Science and Technology of China, China
h_shuo0108@163.com, unayufei99@163.com, hanksliu@uestc.edu.cn

Abstract

Channel pruning has achieved remarkable success in solving the significant computation and memory consumption in Convolutional Neural Networks (CNN). Most existing methods measure the importance of channels with manually designed algorithms and pruning unimportant channels rely on heuristics or expertise during the processing, which are labourious and subjective. In this paper, we proposed a Variational Automatic Channel Pruning Algorithm based on structure optimization (VA-CPSO) which can automatically optimize channel numbers via channel scales in end-to-end manner through variational inference. Firstly, a weights generator controlled by channel scales is built to produce weights for various pruned structure of CNN. And then, the channel scales with truncated factorized log-uniform prior and log-normal posterior are optimized by variational inference for optimal pruning structure. Meanwhile, parameters of the weights generator are optimized synchronously. Finally, the acquired optimal structure and corresponding generated weights are deployed in the pruned CNN for further training to achieve high-performance compression. The experimental results demonstrate that our proposed VA-CPSO acquires better compression performance compared to existing pruning algorithms. The VA-CPSO achieve a compression of 34.60×, 4.28×, 1.96× and a speedup of 28.20×, 2.03×, 2.02× on LeNet-5, VGGNet, and ResNet-110 with no more than 0.5% loss of accuracy.

Keywords: Automatic channel pruning, Variational inference, Structure optimization, CNN compression, Truncated distributions

1 Introduction

As a flexible family of models, Deep neural networks [1] have achieved state-of-the-art performance in many machine learning problems. With the strong learning ability to image data, Convolutional neural networks (CNNs) have created impactful advances in many computer vision tasks including image classification [2], object detection [3], image segmentation [4], and human behavior recognition [5]. While the success of CNNs is often supported by significant computation and memory consumption. The state-of-the-art models in image classification usually have tens of millions of parameters and take billions of floating-point operations to complete the prediction of one image. “Bigger is better” makes it difficult to deploy the advanced CNNs on devices that have strong limitations with respect to memory and energy consumptions such as mobiles, laptops, and wearable devices.

Recently, many studies have shown that the deep neural network is heavily over-parametrized and can be compressed without significant loss of accuracy. Researchers are inspired by this and proposed a series of neural network compression techniques such as tensor decomposition [6], network quantization [7], knowledge distillation [8] and network pruning [9, 12], aiming to reduce the complexity of large models as much as possible while maintaining high accuracy. Network pruning is one of the effective methods which focuses on removing the redundant parameters from the network. According to the granularity of pruning, it can be divided into weight pruning and channel pruning. In weight pruning, the less salient individual weights are set to zeros with little loss to the final model quality. Though it is able to prune majority connections and achieve high compression ratio, it comes at the cost of the irregularity of the sparse computation pattern and requires specialized hardware for further acceleration. In contrast, channel pruning directly removes entire channels and corresponding feature maps which is fully supported by off-the-shelf deep learning library, and is efficient for practical implementation.

The common idea of channel pruning is to measure the contribution of channels to the model performance and remove channels below thresholds. It is effective but the definition of the filter importance and the selection of thresholds rely heavily on heuristics or expertise, which is subjective, laborious and often sub-

*Corresponding Author: Xingang Liu; E-mail: hanksliu@uestc.edu.cn
DOI: 10.3966/160792642021032202009

optimal. A recent study [18] shows that the pruned architecture itself rather than a set of inherited "important" weights is more crucial to the final model. Inspired by it, the AutoML and neural architecture search are introduced in channel pruning to find the optimal pruning structure directly. Reinforcement learning and genetic algorithm are widely used, but the "hard" searching usually result in significant computation. To avoid this problem, the Variational Bayesian is introduced in this paper to translate the search problem into the optimization problem for effective pruning.

In this paper, we propose a Variational Automatic Channel Pruning Algorithm based on structure optimization (VA-CPSO) to automatically explore the optimal pruning architecture through variational Bayesian sparsity learning for channel pruning. It regards the contraction scales of channel numbers for each layer *i.e*. the channel scales as variables with truncated factorized log-uniform prior and lognormal posterior, and introduce the variational Bayesian to optimal the distribution of them. The weights generator is built following [37] to generate weights for various pruned structure of CNN under the control of channel scales. The distributed parameter of the channel scales and the parameters in weights generator are optimized synchronously by stochastic gradient variational Bayes (SGVB). The optimal results are employed in Pruned CNN for channel pruning and we can get the pruned network by further tuning the Pruned CNN with cross-entropy loss. Also, the number of neurons in the fully connection layers can be learned in our network by neuron scales which is similar to channel scales. Our algorithm gives a general structure optimization technique in end-to-end manner, it can apply for diverse layers such as convolutional layers, fully connected layers and depthwise separable convolution for structured pruning.

Our contributions are summarized as follows:

- We propose a Variational Automatic Channel Pruning Algorithm based on structure optimization (VA-CPSO) for pruning of CNNs. It is able to automatically find the optimal channel structure through variational Bayesian sparsity learning for pruning, which avoids the tedious hyperparameter adjustment and the empirical design.
- The Bayesian Inference is introduced to optimize the distributed parameter of the channel scales, which makes the searching and the optimization of the channel structure becomes interpretable and stable, avoiding the arduous and ineffective exploration in the huge search space. And with a sparse prior, the structure tend to be light and compact.
- Extensive experiments on several datasets show the effectiveness of our algorithm. 76.7% parameters and 50.9% floating-point operations on VGGNet are reduced with only 0.37% loss of accuracy. And on ResNet-110, 48.8% parameters and 50.5% floating-point operations can be cut down with 0.39% loss of accuracy. It outperforms the state-of-the-art methods.

The remainder of the paper is organized as follows. In Section 2, the related work of network pruning are described. Section 3 presents our proposed VA-CPSO for CNN channel pruning. To verify the proposed VA-CPSO, Section 4 shows our experimental process, results as well as and the relevant analysis. Finally, we conclude the paper in Section 5.

2 Related Work

The model compression and acceleration is the hotspot in the deep neural network community recently. Extensive studies are emerging in an endless stream, including tensor decomposition [6], network quantization [7], knowledge distillation [8] and network pruning [9, 12]. Here, we focus on network pruning and provide a non-exhaustive review of related methods. The network pruning reduces the size of the deep neural network by pruning some parameters and connections. The thinking about network pruning is deepening, and different directions have emerged. We classify them into three categories: importance-based pruning, sparsity learning, and architecture search.

Importance-based pruning aims to define, find, and prune the unimportant connections from the deep neural network. Early work in weight pruning [19] and [20] determined the saliency by using all second-order derivatives of the error function to perform network pruning. More recently, Han *et al*. [9] used iterative thresholding to remove unimportant weights with small magnitudes and proposed the classical training, pruning and fine-tuning structure for pruning. Later they combined quantification and Hoffman coding after pruning to reduce the storage of sparse network [10]. The refinement for magnitude pruning techniques in [21] further increased the compression rates. In channel pruning, the unimportant filters will be removed. Li *et al*. [11] proposed a magnitude-based pruning in which l1-norm is used to measure the significance of filters. Hu *et al*. [23] pruned unimportant filters based on the sparsity of their outputs. Luo *et al*. [24] proposed the ThiNet, which pruned the filters based on statistics information computed from its next layer, not the current layer. Recently, some global-based methods for measuring the significance of channels have been proposed. Abbasi *et al*. [12] defined a filter importance index which equals the classification accuracy reduction (CAR) of the network after pruning that filter and then iteratively prunes filters based on the CAR index. Yu *et al*. [13] proposed the Neuron Importance Score Propagation (NISP) algorithm which can propagate the importance scores of final responses to every neuron in the network. The importance-based pruning has deepened the cognition to CNNs and can achieve significant compression. But the definition of the filter

importance is mainly based on empirical assumptions, which is prone to be comprehensive and subjective. Meanwhile, the thresholds of pruning for different layers are usually set manually based on heuristics or expertise, which is laborious but often sub-optimal.

Sparsity learning methods try to induce sparsity and prune the network during training. A regularization term added to the loss function is often used for sparse induction. Louizos *et al*. [15] proposed a method for l_0-norm approximation. By l_0-norm regularization, the weights are encouraged to be exactly zero and can be pruned during training. In channel pruning, Liu *et al*. [25] used l_1-norm regularization to reduce the scale factors in Batch Normalization (BN), realizing the slimming of the network. Wen *et al*. [26] proposed the Structured Sparsity Learning (SSL) method. The group Lasso regularization is introduced to directly learn a compressed structure of deep CNNs during the training. Many compression techniques based on (variational) Bayesian inference and the minimum description principle [27] emerged recently. Molchanov *et al*. [16] proposed variational dropout as a reinterpretation for Gaussian dropout training from the perspective of variational inference. They learned per-parameter dropout rates during training and pruned the weights with high dropout rates to produce high sparsity as well as strong generalization. Ullrich *et al*. [28] proposed a ”soft weight-sharing” method which applies Gaussian Mixture Distribution as a prior to encourage sparsity and clustering of weights. Louizos *et al*. [29] used the prior for inducing the sparsity of weights and used the posterior for determining the bit precisions of each weight, which achieved high compression. Similar methods are used in channel pruning. Neklyudov *et al*. [14] proposed a drop-like layer in a Baysian way to achieve structured sparsity. The novel Bayesian method leads to structured sparsity by removing elements with a low SNR. Zhao *et al*. [17] introduced the variational Bayesian to estimate the distribution of channel saliency induced by a sparse prior and pruned redundant channels based on the distributions. The variational Bayesian provides theoretical motivation for regularization and sparsification techniques and enriches the study of network compression, but the excessive variational parameters in models will greatly increase the complexity of training.

Architecture search methods have been widely adopted in neural network design and compression, leading to a compact and advanced network structure. Reinforcement learning is a common approach. NAS (Neural Architecture Search) [30] used a recurrent network to generate the model descriptions and train this RNN with reinforcement learning, learning novel architectures with high performance. This work has been further developed by PNAS [31] and ENAS [32]. Inspired by them, N2N (Network to Network) [33] established the selection mechanism based on reinforcement learning, achieving network compression through channel selection. He *et al*. [34] proposed AutoML for Model Compression (AMC) which automatically searches the design space by reinforcement learning, greatly improving the compression quality. Genetic algorithm is also used by researchers. Xie *et al*. [35] encoded the network structures and adopted the genetic algorithm to cross the large search space. They finally found the most competitive individual through recognition accuracy. Real *et al*. [36] took a similar approach to search for an efficient image recognition Network. Liu *et al*. [37] trained a PruningNet and employed the genetic algorithm to find the optimal channel pruning structure, actualizing superior performances on deep CNNs. Architecture search helps to free human labor and automatically seek for the optimal compression policy, but inefficient search algorithms often result in significant computational overhead.

Inspired by variational Bayesian sparsity learning and architecture search, the Bayesian Inference is introduced in our designed architecture search network for automatically searching and optimizing the sparse channel structure during training. Our algorithm absorbs the superiority of them and avoids the inferiority in a subtle way, achieving efficient channel selection and pruning.

3 The Proposed VA-CPSO

In this section, we introduced the proposed VA-CPSO which controls the channel numbers of each layer via channel scales for channel pruning of CNNs. The VA-CPSO algorithm is implemented on the variational structure optimization network as shown in Figure 1. It includes two parts: (1) Weights generator for producing weights under the control of channel scales. (2) Pruned CNN for predicting the input images with the generated weights to measure the quality of the corresponding channel scales. In this network, the mini-batch images and the original channel numbers of CNN are the input, the distributed parameter of and the parameters in the weights generator are network variables to be trained, and the evidence lower bound (ELBO) is the Loss for training network variables.

The channel scales are limited to 0, 1 for controlling the proportion of retained channels per layer. During training, the variational inference and stochastic gradient variational Bayes are introduced (SGVB) to train and optimize the distribution of the channel scales as well as the parameters in the weights generator. Concretely, the truncated prior and posterior distribution families are adopted to channel scales and the corresponding ELBO is minimized to acquire the probability density of the channel scales. After training, channel pruning can be implemented based on the optimal channel scales and the generated weights we obtained to get compacted CNN.

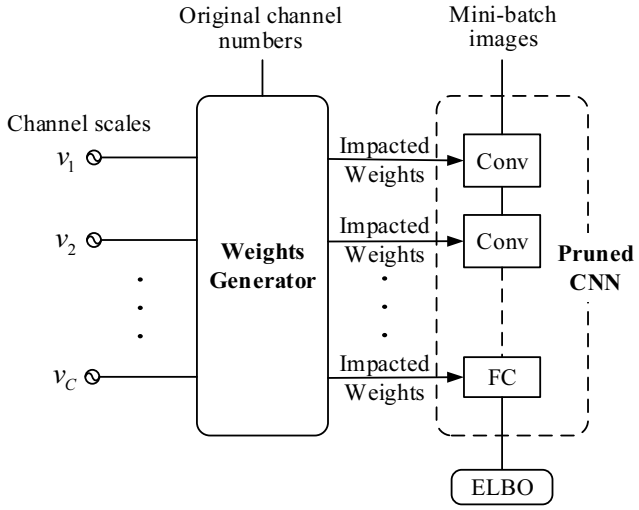


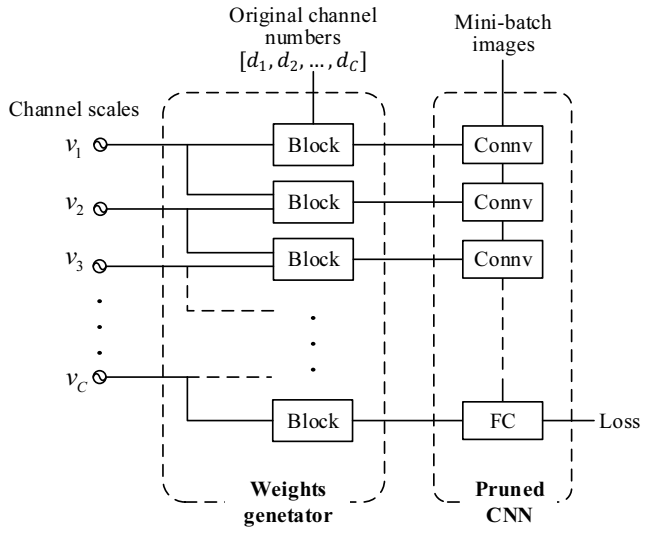
Figure 1. Variational structure optimization network for VA-CPSO

Note that there is great redundancy for the parameters of the fully connected layers (FC) in CNN, so we incorporate the FC into our structure optimization network. We deal with the neurons in FC like channels in convolution layers and find the optimal structure for pruning neurons.

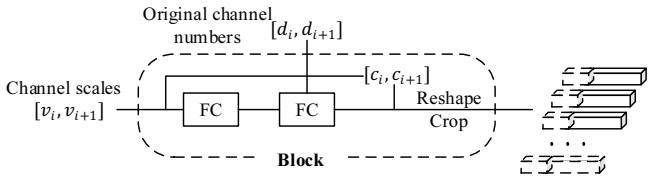
3.1 Weights Genetator

The key issue in channel pruning is how to select the channels of the convolution layers for pruning, in order that the structure of the pruned network can achieve high compression with almost lossless performance. The common methods try to pruning unimportant channels or induce sparsity via regularization as mentioned in Section 2. Recently, the development of AutoML has brought new ideas to the researchers. The architecture search for pruning attracts plenty of attention. The study [18] suggests that the pruned network structure is decisive instead of the remainder weights, which support us to find the optimal pruned network structure directly. Following [37], we design a weights generator controlled by the channel scales, which predict weights for the corresponding pruned structure. Moreover, we take the FC into account.

As shown in Figure 2, the generator consists of a series of blocks, each corresponding to a convolutional layer or FC of the CNN. And the block is comprised of two fully connected layers with 32 neurons in the middle layer. The inputs of the weights generator contain two vectors with the same length: channel scales $v=[v_1, v_2, \ldots, v_L]$ and the numbers of original channels $d=[d_1, d_2, \ldots, d_L]$. The former is the network variable composed of the layer-wise channel scales limited to 0,1 which controls the shape of the generated weights for Pruned CNN. The latter is the network input consisting of the number of channels for each convolutional layer in the original CNN for controlling the original shape of the output weights *i.e.* the number of output nodes for the last FC in each block. By multiplying the corresponding elements of the two vectors, the numbers of channels after pruning $c=[c_1, c_2, \ldots, c_L]$, can be determined by $c_i = round(d_i * v_i)$ and the final shape of the generated weights can be acquired after reshaping and cropping. Meanwhile, we append the original neuron numbers and neuron scales for each FC to the input vectors separately and generate the pruned weights of the fully connection layer in the same way.



(a) The structure of the weights generator



(b) The structure of each block

Figure 2.

In the forward propagation, the channel scales enter into the relevant blocks separately to produce weights for different layers of CNN. A Pruned CNN based on the numbers of pruned channel numbers c are established, and the weights generated by blocks are reshaped and cropped to adapt the structure of each layer in Pruned CNN. These generated weights are put into Pruned CNN to calculate the loss giving a batch of input images. In the backpropagation, the parameters in blocks and the channel scales v are updated with SGVB which will be described in detail later. Pruned CNN doesn't participate in the gradient update.

3.2 Bayesian Inference and Stochastic Gradient Variational Bayes (SGVB)

Consider a dataset $D=\{x=\{x_n\}_{n=1}^N, y=\{y_n\}_{n=1}^N\}$, x is input data, y is corresponding label and N is the number of images on this dataset. Our goal is to learn the variables v of the model with conditional

probability $p(y|x,v)$. Where the variables v are channel scales that control the numbers of channels for layers. In Bayesian learning, we can adapt the original prior knowledge after we collect some new information. In particular, if we know the prior distribution $p(v)$ and the data D arrives, we can inference that the posterior distribution is $p(v|D)=p(D|v)p(v)/p(D)$ according to the Bayes rule. However, the inference usually involves the intractable-computation multidimensional integrals like $p(D)=\int p(D,v)dv$. Hence, the approximation techniques need to be introduced.

Variational Inference is one of the most common techniques. In this approach, the posterior distribution $p(v|D)$ is approximated by a parametric distribution $q_\phi(v)$. The degree of approximation is estimated via the Kullback-Leibler divergence. From this, the intractable inference problem is converted to a tractable optimization problem. The optimal value of variational parameters ϕ can be got by minimizing the Kullback-Leibler divergence, *i.e.* $\min_\phi D_{KL}(q_\phi(v)\|p(v|D)$, which is equivalent to maximize the evidence lower bound (ELBO):

$$\begin{aligned} &L(\phi)=L_D(\phi)-D_{KL}(q_\phi(v)\|p(v|D)) \\ &\text{where } L_D(\phi)=\sum_{n=1}^{N}\mathbb{E}_{q_\phi(v)}[\log p(y_n|x_n,v)] \end{aligned} \quad (1)$$

Here, we can see why we use the ELBO in practice. The bound $L(\phi)$ plus $D_{KL}(q_\phi(v)\|p(v|D))$ equals to the marginal log-likelihood $\Sigma_{(x,y)\in D}\log(y|x)$. This marginal log-likelihood is a constant, so maximize the bound $L(\phi)$ will minimize $D_{KL}(q_\phi(v)\|p(v|D))$. The ELBO contains two parts, the expected log-likelihood $L_D(\phi)$ and the KL-divergence $D_{KL}(q_\phi(v)\|p(v))$. The former $L_D(\phi)$ acts as a reconstruction term with the goal of maximizing the log-likelihood of the model prediction. The latter acts as a regularization term which aims to draw the posterior distribution close to a sparsity prior. The ELBO takes into account the predictive performance and the sparsity of the model simultaneously, which leads to a compact and excellent model.

However, it is impossible to exactly calculate the gradient in equation (1) to achieve optimization. The stochastic gradient variational Bayes (SGVB) introduced in [38] gives a practical estimator of the lower bound and its derivatives. The basic trick is to reparameterize the random variable $v\sim q_\phi(v)$ using a differentiable function $f(\phi,\in)$ of a noise variable $\in$:

$$v=f(\phi,\in) \text{ with } \in\sim p(\in) \quad (2)$$

where the $\in$ is acquired by sampling in practice. From this, an unbiased differentiable minibatch-based Monte Carlo estimator can be formed as

$$L(\phi)\simeq L^{SCVB}(\phi)=L_D^{SGVB}(\phi)-D_{KL}(q_\phi(v)\|p(v)) \quad (3)$$

$$L(\phi)\simeq L_D^{SCVB}(\phi)=\frac{N}{M}\sum_{i=1}^{M}\log p(y_i|x_i,v=f(\phi,\in)) \quad (4)$$

$$\nabla_\phi L_D(\phi)\simeq\frac{N}{M}\sum_{i=1}^{M}\nabla_\phi\log p(y_i|x_i,v=f(\phi,\in)) \quad (5)$$

where (x_i,y_i) is an input image and its corresponding label. M is the size of a mini-batch input data and N is the total number of images involved in this training. Meanwhile, we assume that the $D_{KL}(q_\phi(v)\|p(v))$ can be either computed deterministically or be estimated approximately. In this way, we can solve the optimization problem by initializing ϕ and executing optimization methods such as Adam [39], which can be efficiently applied in deep neural networks. Once we get the optimal parameters ϕ, we will know the approximate distribution $q_\phi(v)$.

Notable is, we can update the weights $\boldsymbol{g}$ of the network as well as the variational parameters ϕ synchronously in stochastic gradient ascent. We can represent the model as $p(y|x,\boldsymbol{g},v)$. The ELBO here is

$$L^{SCVB}(\phi,\boldsymbol{g})=L_D^{SCVB}(\phi,\boldsymbol{g})-D_{KL}(q_\phi(v)\|p(v)) \quad (6)$$

$$L_D^{SGVB}(\phi,\boldsymbol{g})=\frac{N}{M}\sum_{i=1}^{M}\log p(y_i|x_i,v=f(\phi,\in),\boldsymbol{g}) \quad (7)$$

Hence, we can acquire the weights and the channel scale distribution of the network via optimizing the ELBO.

3.3 Variational Inference with Truncated Distributions

As shown in equation (3) and (4) the prior distributions $p(v)$ and the differentiable function $f(\phi,\in)$ of the posterior distribution $q_\phi(v)$ need to be decided. As mentioned before, the variational variables in our model are channel scales, which are limited to 0,1. Therefore, traditional unbounded distributions need to be truncated here [14]. Furthermore, we transform the variables v to the log domain $v_{in}=\ln v$ to increase the maximum truncation interval from $(0,1]$ to $(-\infty,0]$. Following that, we will analyze and infer the model in the log domain.

We use a and b to represent the left and right truncation values. And $p_{[a,b]}(v_{in})$ and $q_{\phi[a,b]}(v_{in})$ denote the truncated prior and posterior distributions in the log domain.

Before we get some information, the uniform distribution $U(a, b)$ is an appropriate choice which assumes the same initial probability for different values.

$$p_{[a,b]}(v_{\ln}) = \prod_{i=1}^{L} p_{[a,b]}(v_{\ln}^i) \quad \text{where } p_{[a,b]}(v_{\ln}^i) = U(v_{\ln}^i \mid a, b) \tag{8}$$

When we obtain the related information, the variables v_{in} have their own focus. The unimodal factorized normal distribution can give a good description. What we need is an truncated distribution. So we select the truncated factorized normal distribution as the posterior.

$$q_{\phi[a,b]}(v_{\ln}) = \prod_{i=1}^{L} q_{\phi[a,b]}(v_{\ln}^i) \quad \text{where } q_{[a,b]}(v_{\ln}^i) = tN(v_{\ln}^i \mid \mu_i, \sigma_i, a, b) \tag{9}$$

Here, L denotes the number of layers with different channel scales, and

$$tN(x \mid \mu, \sigma, a, b) = \frac{1}{Z_i\sqrt{2\pi\sigma_i^2}} \exp - \frac{(x-\mu_i)^2}{2\sigma_i^2} \quad \text{where } Z = \Phi(\beta_i) - \Phi(\alpha_i) \tag{10}$$

Here, $a_i = \frac{a-\mu_i}{\sigma_i}$, $\beta_i = \frac{B-\mu_i}{\sigma_i}$, and $\Phi(\cdot)$ is the CDF of standard normal distribution. Then, we need to identify the differentiable function $f(\phi, \in)$ for sampling and gradient descent according to the $q_{\phi[a,b]}(\ln v)$. The CDF of the truncated normal distribution is

$$F(x) = \frac{\Phi(\frac{x-\mu_i}{\sigma_i}) - \Phi(\frac{a-\mu_i}{\sigma_i})}{Z_i} \tag{11}$$

From equation (11) we have

$$x = \mu_i + \sigma_i \Phi^{-1}(Z_i F(x) +_i \Phi(a_i)) \tag{12}$$

Hence the differentiable function can be expressed as

$$v_{\ln}^i = \mu_i + \sigma_i \Phi^{-1}(Z_i \in_i + \Phi(a_i)), \in_i \sim U(\in \mid 0, 1) \tag{13}$$

Consequently, the proper probabilistic model with prior and posterior is determined and the variational inference can be executed. The KL-divergence in log domain for a single variable v_{ln}^i can be calculated as

$$\begin{aligned} &D_{KL}(q_{\phi[a,b]}(v_{\ln}^i) \,\|\, p_{[a,b]}(v_{\ln}^i)) = D_{v_{\ln}^i} \\ &= \int_a^b tN(v_{\ln}^i \mid \mu_i, \sigma_i, a, b) \ln \frac{tN(v_{\ln}^i \mid \mu_i, \sigma_i, a, b)}{U(v_{\ln}^i \mid a, b)} dv_{\ln}^i \\ &= -H(tN(v_{\ln}^i \mid \mu_i, \sigma_i, a, b)) + \ln(b-a) \\ &= \ln(\frac{b-a}{Z_i\sqrt{2\pi}\sigma_i}) - \frac{\sigma_i\phi(a_i) + \beta_i\phi(\beta_i)}{2Z_i\sqrt{2\pi}} - \frac{1}{2} \end{aligned} \tag{14}$$

Where The entorpy of the truncated normal distribution is

$$\begin{aligned} &H(tN(v_{\ln}^i \mid \mu_i, \sigma_i, a, b)) \\ &= \ln(Z_i\sqrt{2\pi}\sigma_i) \int_a^b tN(v_{\ln}^i \mid \mu_i, \sigma_i, a, b) dv_{\ln}^i \\ &\quad + \frac{1}{2\sigma_i^2} \int_a^b (v_{\ln}^i - \mu_i)^2 \mid tN(v_{\ln}^i \mid \mu_i, \sigma_i, a, b) dv_{\ln}^i \\ &= \ln(Z_i\sqrt{2\pi}\sigma_i) + \frac{a_i\phi(a_i) + \beta_i\phi(\beta_i)}{2Z_i\sqrt{2\pi}} + \frac{1}{2} \end{aligned} \tag{15}$$

Back to the linear domain $v = e^{v_{\ln}}$, the truncated interval will be $v \in [e^a, e^b]$. And the truncated prior and posterior follows truncated factorized log-uniform and log-normal distributions respectively which are expressed as follows.

$$p_{[e^a,e^b]}(v) = \prod_{i=1}^{L} p_{[e^a,e^b]}(v_i) \quad \text{where } p_{[e^a,e^b]}(v_i) = \log U_{[e^a,e^b]}(v_i) = \frac{1}{v_i(b-a)} \tag{16}$$

$$\begin{aligned} q_{\phi[e^a,e^b]}(v) &= \prod_{i=1}^{L} q_{[e^a,e^b]}(v_i) \\ \text{where } p_{[e^a,e^b]}(v_i) &= \log N_{[e^a,e^b]}(v_i \mid \mu_i, \sigma_i) \\ &= \frac{1}{v_i Z_i\sqrt{2\pi\sigma_i^2}} \exp(\frac{\ln(v_i-\mu_i)^2}{2\sigma_i^2}) \end{aligned} \tag{17}$$

It is noted that the log-uniform is the approximate distribution of floating-point format storage, so it is very appropriate as a regularization for the posterior of the channel scales which store in the floating-point format. The sparsity of the loguniform which can encourage small channel scales is also favorable for the compression and acceleration of the network. The posterior log-normal can be regarded as a special case of the log-uniform when the σ tends to infinity, thus there is no gap between prior and posterior for the variational model. Moreover, the non-negative property of the log-normal random variables makes it credible for the channel scales. Besides, the corresponding differentiable function $f(\phi, \in)$ and the KL-divergence for v is tractable by above analysis. We have

$$v_i = \exp(\mu_i + \sigma_i \Phi^{-1}(Z_i \in_i + \Phi(a_i))) \quad \text{where } \in_i \sim U(\in_i \mid 0, 1) \tag{18}$$

$$\begin{aligned} D_{KL}(q_\phi(v) \,\|\, p(v)) &= \sum_{i=1}^{L} D_{KL}(q_\phi(v_i) \,\|\, p(v_i)) \\ &= \sum_{i=1}^{L} D_{KL}(q_\phi(\ln v_i) \,\|\, p(\ln v_i)) = \sum_{i=1}^{L} D_{v_{\ln}^i} \end{aligned} \tag{19}$$

where $D_{v_{\ln}^i}$ is calculated in equation (14) In this way, given *a*and *b*, we can sampling v_i for each layer according to equation (18) and optimize the ELBO in equation (6) and (7) with KL-divergence in equation (19) *w.r.t.* the parameters μ_i and σ_i. When we obtain the optimal variational parameters μ_i^* and σ_i^*, we can get the approximate posterior distribution of channel scales from equation (17).

3.4 A Summary of Proposed Algorithm

Based on above section, we can achieve our VA-CPSO. And by training the weights generator and the Pruned CNN successively, the optimal channel structure and the high-performance pruned CNN are achieved.

To train the weights generator, we sample channel scales $v=[v_1, v_2, \ldots, v_L]$ according to equation (18) with constants *a*, *b* which are given at first and variables μ_i and σ_i which are initialized in the beginning and updated during training. Then, we use the sampled *v* for forward propagation in the weights generator and the Pruned CNN. The ELBO can be calculated according to equation (6) and (7) at the end of the Pruned CNN with batches of images as inputs. The parameters in the weights generator, the generated weights, and the channel scales are updated using SGVB in backword propagation. From this, the optimization network achieved end-to-end traning. After training, we get the approximate posterior distribution of channel scales as well as the trained weights generator. Then, a determined value for each channel scale needs to be selected from its distribution. The expectation is a common choice. We have

$$\begin{aligned}
&\mathbb{E}_{v_i}[\log N_{[e^a,e^b]}(v_i \mid \mu_i, \sigma_i)] \\
&= \int_{e^a}^{e^b} \frac{1}{Z_i\sqrt{2\pi\sigma_i^2}} \exp(\frac{\ln(v_i-\mu_i)^2}{2\sigma_i^2}) dv_i \\
&= \frac{e(\sigma_i^2/2+\mu_i)}{Z_i} \int_{\frac{b-\mu_i}{\sigma_i}}^{\frac{b-\mu_i}{\sigma_i}} \frac{1}{\sqrt{2\pi}} \exp(-\frac{(t-\sigma_i)^2}{2}) dt \\
&= \frac{e(\sigma_i^2/2+\mu_i)}{Z_i}[\Phi(\frac{b-\mu_i-\sigma_i^2}{\sigma_i}) - \Phi(\frac{a-\mu_i-\sigma_i^2}{\sigma_i})]
\end{aligned} \tag{20}$$

The expectation of channel scales is the optimal channel structure we learned from the variational network. And the detailed Variational Automatic Channel Pruning algorithm is described in Algorithm 1.

Algorithm 1. Variational Automatic Channel Pruning

1. **Input:** Truncated bound *a*, *b*, *N* pairs data
   $D=(x_i, y_i)_{i=1}^N$ with batch size *M*, Original channel numbers $d=[d_1, d_2, \ldots, d_L]$
   Output: The optimal channel scales v^*, The generated weights $\boldsymbol{w}$
2. Initialize variational variables $\phi=[\mu, \sigma]$, parameters in weights generator $\boldsymbol{g}$;
3. **for** *epoch in 1 to K* **do**
4. Sample $\in_i \sim U(\in| 0,1)$ for *L* layers independently
5. compute $v_i = \exp(\mu_i + \sigma_i \Phi^{-1}(Z_i \in_i + \Phi(a_i)))$
6. $D_{KL}(q_\phi(v) \| p(v)) = \sum_{i=1}^{L} d_{v_{\ln}^i}$ in equation (14)
7. $L_D(\phi, \boldsymbol{g}) = \frac{N}{M}\sum_{i=1}^{M} \log p(y_i \mid x_i, v_i, \boldsymbol{g})$
8. $L(\phi, \boldsymbol{g}) = L_D(\phi, \boldsymbol{g}) - D_{KL}(q_\phi(v) \| p(v))$
9. Update $\boldsymbol{g}$ and $\phi=[\mu, \sigma]$ via maximizing $L(\phi, \boldsymbol{g})$
10. Get the $\phi^*=[\mu^*, \sigma^*]$ and $\boldsymbol{g}^*$
11. $v^* = \frac{e(\sigma_i^{*2}/2+\mu_i^*)}{z_i^*}[\Phi(\frac{b-\mu_i^*-\mu_i^{*2}}{\sigma_i^*}) - \Phi(\frac{a-\mu_i^*-\sigma_i^{*2}}{\sigma_i^*})]$
12. Obtain the corresponding prunded weights $\boldsymbol{w}$ through weights generator with v^* and $\boldsymbol{g}^*$
13. **return** $v^*=[v_1^*, v_2^*, \ldots, v_L^*]$, $\boldsymbol{w}$

Now, the learned channel structure can be applied for channel pruning. Inputting the expectation of the channel scales into the trained weights generator, the initial weights for Pruned CNN can be predicted, and then the optimal Pruned CNN can be built.

The Pruned CNN continues to be fine-tune with cross-entropy loss until convergence. Thus, the compact CNN after channel pruning can be acquired for resource-constrained scenarios.

4 Experimental Results

In this section, we evaluate the proposed algorithm on image classification tasks to demonstrate the effectiveness. The experiments are carried out on some common networks including Lenet-5 [40], VGGNet [41] and, ResNets [2]. We use Pytorch to carry out our VA-CPSO for pruning. All experiments are running on Nvidia 1080Ti GPU. The experiments have achieved encouraging results.

4.1 Experimental Settings

Implementation procedure: Two stages need to be performed for channel pruning in our algorithm. In the first stage, we execute the minibatch-based SGVB for training the weighs generator and optimizing the posterior distribution of channel scales *i.e.* the variational variables $\phi = [\mu, \sigma]$. Then, the expectation of the distribution is calculated as the optimal channel scales and the generated weights of the optimal channel scales are obtained through the trained weights generator. In the second stage, we build the Pruned CNN according to the channel scales and employ the generated weights on it. The Pruned CNN continues to be trained until convergence.

Compression Metric: Parameters and Floating-point operations (FLOPs) are widely-used to measure the computation and memory consumption of CNNs. Parameters include the convolutional layer and fully connected layer parameters. The ratio of the parameters in the network before and after pruning is called Compression Ratio (CR), which judges the effect of the compression algorithm. FLOPs include floating-point addition and floating-point multiplication. Because of the regularity of CNN, They can be count simply. Then the theoretical Speedup (Sp) of the compression algorithm can be got, which is proportional to the acceleration of CNN. In this paper, we only take account of the floating-point multiplication.

Training details: The truncated bound a and b are the hyperparameters in our model. As the channel scales v are limited to 0, 1, the ln$v$ are limited to $-\infty$, 0. In practice, the channel scales can not be 0 which will cause the disappearance of the layer, so we set $v \in [0.05, 1]$. In addition, we use a weight coefficient w_r on the regularization term of the ELBO in order to observe its effect on pruning. The momentum is set to 0.9 and the weight decay is set to 0.0001 in all experiments. The other details are discussed in the following sections.

4.2 LeNet-5 on MNIST

We evaluate our VA-CPSO with LeNet-5 on MINIST [40]. There are two convolutional layers and two fully connected layers in LeNet5 with 20-50-500 channels/neutrals structure. When training the VA-CPSO, we set different learning rates for the parameters in the weights generator $\boldsymbol{g}$ and the variational variables $\phi = [\mu, \sigma]$ with Adam[39] optimizer. The former is set to 0.005 and the latter is set to be 0.0001. When retraining the Pruned CNN, the learning rate is set to be 0.005 with Adam [39] optimizer. We explore various weight coefficient w_r in VA-CPSO with the batch size 512 for 60 epochs. The PrunedCNNs are trained with the batch size 512 for 40 epochs in the second step. Larger w_r results in higher pruning rates in our algorithm.

Table 1 shows the pruning results with various w_r. In the table, the Sp is speedup and the CR is the Compression ratio as explained above. VA-CPSO(i) represents the results with $w_r = i$. Model LeNet-5 is the baseline and the other are different pruning methods on LeNet-5. We can see that when the w_r is set to normal 1, the optimal structure is 13-32-325 which realizesbetter accuracy than the original LeNet-5 with 0.02 accuracy advance. The VA-CPSO find the optimal structure of the LeNet-5 for channel pruning. As we turn up the weight coefficient w_r, the more compact networks can be obtained with negligible loss of accuracy. When the w_r is set to 2, 5 and 8, the LeNet5 yields a compression ratio of 9.50×, 24.19× and 34.60× respectively and the speedup of 8.71×, 18.61× and 28.20× with no more than 0.21 percent loss of accuracy compared to the base. The VA-CPSO gets higher performance compared with other methods. For example, compared with recent research GAL [48], VA-CPSO (2) obtains higher speedup and compression ratio (8.71× vs 5.71×, 9.50× vs 8.17×) with the same error (0.86%), and VA-CPSO (5) achieves 18.61× and 24.19× speedup and compression ratio, which are higher 15.04× and 15.63× in GAL with only more than 0.03% accuracy loss.

4.3 VGGNet on CIFAR-10

We then evaluate the VA-CPSO with VGGNet on CIFAR10 [42] to verify its effectiveness in deep architectures. Since the data in CIFAR-10 are RGB images with a size of 32×32 instead of the original input RGB images of VGGNet with size 224×224, Zagoruyko applies a slightly modified version of VGGNet on CIFAR-10 [43] which contain 13 convolutional layers and 2 fully connected layers. We use this model with Batch Normalization after each convolutional layer here. The initial learning rates for the parameters of weights generator g and the variational variables $\phi = [\mu, \sigma]$ are set to be 0.01 and 0.0005 separately. During retraining, the initial learning rates is set to be 0.05. All the learning rate is divided by 10 per 50 epochs. We train the VA-CPSO for 100 epochs with the batch size 256 and retrain the Pruned CNN 100 epochs with the same batch size.

Table 1. Pruning results of LeNet-5 on MNIST

Model	Error%	Channels	#Flops (Sp)	#Param. (CR)
LeNet-5	0.80	20-50-500	3.99M (1.00×)	0.65M (1.00×)
SSL [26]	1.00	3-12-500	0.31M (12.7×)	0.16M (4.20×)
SBP [14]	0.86	3-18-283	0.34M (11.6×)	0.13M (4.98×)
NISP [13]	0.82	10-25-250	1.27M (3.15×)	0.17M (3.94×)
GAL-0.01 [48]	0.86	10-15-198	0.70M (5.71×)	0.08M (8.17×)
GAL-0.05 [48]	0.90	4-13-121	0.26M (15.04×)	0.04M (15.63×)
VA-CPSO (1)	0.78	13-32-325	1.74M (2.29×)	0.27M (2.39×)
VA-CPSO (2)	0.86	6-16-162	0.49M (8.71×)	0.07M (9.50×)
VA-CPSO (5)	0.93	4-10-100	0.21M (18.61×)	0.03M (24.19×)
VA-CPSO (8)	1.01	3-8-87	**0.14M (28.20×)**	**0.02M (34.60×)**

As shown in Table 2, the error reaches the minimum when the w_r is 1 with 1.26× speedup and 1.78× compression. With setting w_r to 2 and 4, we achieve 1.67×, 2.03× speedup and 3.38×, 4.28× compression with minor loss of performance. Our algorithm achieves the highest speedup and pruning rate with the lowest error comparing with other methods as we increase the w_r. *e.g.* VA-CPSO obtains 93.05% accuracy with 4.28× compression and 2.03× speedup, which is better than 93.05% accuracy with 3.81× compression and 1.71× speedup in SSS [44]. The results suggest that our VA-CPSO with suitable w_r can greatly compact the large CNN, which improves the universality of VGG model.

Table 2. Pruning results of VGGNet on CIFAR-10

Model	Error%	#Flops (Sp)	#Param. (CR)
VGGNet	6.04	313.73M (1.00×)	14.98M (1.00×)
L1 [11]	6.6	206.00M (1.52×)	5.40M (2.77×)
SSS [44]	6.37	199.93M (1.57×)	4,99M (3.00×)
SSS [44]	6.98	183.13M (1.71×)	3.93M (3.81×)
Zhao *et al.* [17]	6.82	190.00M (1.65×)	3.92M (3.81×)
VA-CPSO (1)	**6.35**	249.97M (1.26×)	8.40M (1.78×)
VA-CPSO (2)	6.41	187.64M (1.67×)	4.43M (3.38×)
VA-CPSO (3)	6.95	**154.12M (2.03×)**	**3.50M (4.28×)**

4.4 ResNets on CIFAR-10

ResNets are a series of networks with the shortcut. They also can be pruned by our algorithm.Here, we adopt ResNet20, ResNet-32, ResNet-44, ResNet-56, and ResNet-110 on CIFAR-10 to evaluate our VA-CPSO. ResNets for CIFAR-10 have three stages of residual blocks with feature maps of different numbers (16, 32 and 64) and sizes (32×32, 16×16 and 8×8). The numbers of residual blocks in each stage are the same. And the parameter-free, identity shortcut layers with zero paddings are used for the increasing feature maps between neighboring stages. The residual block consists of two convolutional layers. Since the shortcut layers are parameter-free, the second layers of residual blocks in one stage must have the same number of channels, so they share one channel scale when training. There is no limit for the first layers of residual blocks, therefore they have their own channel scales for pruning.

In ResNets, we set the initial learning rates for the parameters of weights generator $\boldsymbol{g}$ and the variational variables $\phi = [\mu, \sigma]$ as 0.1 and 0.0005. And the initial learning rates is set to be 0.3 for retaining Pruned CNN. The learning rate is divided by 10 per 50 epochs. We train the VA-CPSO for 100 epochs with the batch size 128 and retrain the Pruned CNN 150 epochs with the same batch size.

Table 3 shows our pruning results. The best accuracy is realized when the w_r is 1, which suggested that our algorithm enables ResNets to learn better structures. As w_r increases, the networks tend to be compact. At the w_r of 4, a balance is achieved between the accuracy and compression ratio of ResNets. At this point, the ResNets achieves outstanding compression and acceleration performance with negligible accuracy loss of no more than 0.51 percent. For example, 2.02× speedup and 1.96× compression are achieved in ResNet-110 with a 0.39 percent accuracy drop.

In a scenario where the equipments are limited and requirements for accuracy is not so high, setting the w_r to 6 will achieve an amazing compression performance with no more than 1.53 percent accuracy loss. We can see that the ResNet-56 and the ResNet-110 are accelerated 2.65×, 2.82× and are compressed 2.57×, 2.59× with 1.34, 0.86 percent accuracy drop. Campared to other methods, our VA-CPSO outperforms the state-of-the-art algorithms on ResNet20, 32, 44, 56, 110. For example, VA-CPSO achieves a higher speedup and compression ratio than GAL [48] in ResNet-56 and ResNet-110.

Table 3. Pruning results of ResNets on CIFAR-10

Model	Error%	#Flops (Sp×)	#Param. (CR×)
ResNet-20	7.8	40.55M (1.00×)	0.27M (1.00×)
Zhao et al. [17]	8.34	32.27M (1.27×)	0.22M (1.20×)
LCCL [45]	8.32	26.10M (1.55×)	–
SFP [46]	9.17	24.30M (1.67×)	–
FPGM [47]	8.91	24.30M (1.67×)	–
VA-CPSO (1)	**7.53**	33.91M (1.18×)	0.23M (1.17×)
VA-CPSO (2)	7.94	26.80M (1.51×)	0.18M (1.49×)
VA-CPSO (4)	8.39	20.11M (2.01×)	0.14M (1.92×)
VA-CPSO (6)	9.24	**15.3M (2.65×)**	**0.12M (2.20×)**
ResNet-32	7.18	68.86M (1.00×)	0.46M (1.00×)
LCCL [45]	9.26	47.60M (1.47×)	–
SFP [46]	7.92	40.30M (1.71×)	–
FPGM [47]	7.69	40.30M (1.71×)	–
FPGM [47]	8.07	32.30M (2.13×)	–
VA-CPSO (1)	**7.11**	56.37M (1.22×)	0.39M (1.19×)
VA-CPSO (2)	7.31	45.53M (1.51×)	0.31M (1.51×)
VA-CPSO (4)	7.58	34.11M (2.02×)	0.23 M (1.99×)
VA-CPSO (6)	8.7	**19.88M (3.46×)**	**0.14M (3.34×)**
ResNet-44	6.97	97.17M (1.00×)	0.66M (1.00×)
MIL [22]	7.49	63.30M (1.53×)	–
VA-CPSO (1)	**6.86**	78.85M (1.23×)	0.55M (1.19×)
VA-CPSO (2)	7.07	64.22M (1.51×)	0.43M (1.51×)
VA-CPSO (4)	7.31	48.11M (2.02×)	0.33M (2.00×)
VA-CPSO (6)	8.18	**33.37M (2.91×)**	**0.23M (2.83×)**
ResNet-56	6.17	125.49M (1.00×)	0.85M (1.00×)
L1-A [11]	6.9	11.20M (1.12×)	0.77M (1.10×)
L1-B [11]	6.94	90.90M (1.38×)	0.73M (1.16×)
NISP [13]	6.99	70.76M (1.77×)	0.49M (1.74×)
SFP [46]	7.74	59.40M (2.11×)	–
GAL-0.6 [48]	6.62	78.30M (1.60×)	0.75M (1.13×)
GAL-0.8 [48]	8.42	49.99M (2.51×)	**0.29M (2.93×)**
Hrank [49]	6.83	62.72M (2.00×)	0.49M (1.73×)
VA-CPSO (1)	**6.09**	103.78M (1.21×)	0.72M (1.17×)
VA-CPSO (2)	6.28	82.90M (1.51×)	0.56M (1.50×)
VA-CPSO (4)	6.68	61.66M (2.04×)	0.42M (2.02×)
VA-CPSO (6)	7.51	**47.40M (2.65×)**	0.33M (2.57×)
ResNet-110	5.97	252.89M (1.00×)	1.72M (1.00×)
L1-A [11]	6.45	213.00M (1.19×)	1.68M (1.02×)
L1-B [11]	6.7	155.00M (1.16×)	1.16M (1.48×)
NISP [13]	6.99	142.17M (1.78×)	0.98M (1.76×)
SFP [46]	6.62	150.00M (1.68×)	–
GAL-0.1 [48]	6.41	205.7M (1.22×)	1.65M (1.04×)
GAL-0.5 [48]	7.26	130.2M (1.94×)	0.95M (1.81×)
Hrank [49]	6.64	105.70M (2.39×)	0.70M (2.45×)
VA-CPSO (1)	**5.89**	202.8M (1.25×)	1.41M (1.22×)
VA-CPSO (2)	6.09	167.01M (1.51×)	1.14M (1.50×)
VA-CPSO (4)	6.36	125.08M (2.02×)	0.88M (1.96×)
VA-CPSO (6)	6.83	**89.56M (2.82×)**	**0.66M (2.59×)**

5 Conclusion

In this paper, we propose a Variational Automatic Channel Pruning Algorithm based on structure optimization (VA-CPSO) to compress and accelerate CNNs by automatically optimizing the channel numbers in CNNs based on Bayesian inference. We reformulate the channel numbers and the node numbers as important parameters called channel scales, and adopt the truncated factorized log-uniform prior and log-normal posterior for channel scales to make the variational model. The weights generator is designed for producing tensors with corresponding size as weight parameters of various pruned structure of CNN under the control of channel scales. During training, the channel scales as well as the parameters of the weights generator are optimized synchronously through variational inference and stochastic gradient variational Bayes (SGVB). Finally, the optimal

channel structure and the corresponding generated weights are employed in the pruned CNNs for further training to achieve high-performance compacted CNN. Our algorithm avoids the tedious hyperparameter adjustment and empirical design. And the extensive experiments demonstrate the outstanding performance of our algorithm.

Noted that the selections of prior and posterior are exploratory, therefore other sparse prior and posterior families on channel scales will be considered in future work. It is also interesting to explore more advanced weights generator structure to achieve better performance. Furthermore, we plan to apply our method to more applications such as gait recognition and human behavior recognition.

Acknowledgments

This work is supported by the National Key Research and Development Project under grant 2018YFB17002402, and also partly supported by the National Natural Science Foundation of China under grant 61872404 and the Aplied Basic Research Key Programs of Science and Technology Department of Sichuan Province on the grant 2018JY0023.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, *Nature*, Vol. 521, No. 7553, pp. 436-444, May, 2015.

[2] B. Yao, H. Zhou, J. Yin, G. Li, C. Lv, Small Sample Image Recognition Based on CNN and RBFNN, *Journal of Internet Technology*, Vol. 21, No. 3, pp. 881-889, May, 2020.

[3] C.-H. Yeh, C.-Y. Lin, K. Muchtar, RGB-D Abandoned Object Detection Based on GrabCut Using Kinect, *Journal of Internet Technology*, Vol. 18, No. 4, pp. 927-933, July, 2017.

[4] C. Liu, L. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, L. Fei- Fei, Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation, *IEEE Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, June, 2019, pp. 82-92.

[5] C. Dai, X. Liu, J. Lai, P. Li, H. Chao, Human behavior deep recognition architecture for smart city applications in the 5g environment, *IEEE Network*, Vol. 33, No. 5, pp. 206-211, September-October, 2019.

[6] X. Zhang, J. Zou, X. Ming, K. He, J. Sun, Efficient and accurate approximations of nonlinear convolutional networks, *IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, June, 2015, pp. 1984-1992.

[7] S. Lin, R. Ji, C. Chen, D. Tao, J. Luo, Holistic CNN Compression via Low-rank Decomposition with Knowledge Transfer, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 41, No. 12, pp. 2889-2905, December, 2019.

[8] J. Bae, D. Yeo, J. Yim, N. Kim, C. Pyo, J. Kim, Densely distilled flow-based knowledge transfer in teacher-student framework for image classification, *IEEE Transactions on Image Processing*, Vol. 29, pp. 5698-5710, April, 2020.

[9] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, *Advances in Neural Information Processing Systems*, Montreal, Quebec, Canada, December, 2015, pp. 1135-1143.

[10] S. Han, H. Mao, W. J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, *International Conference on Learning Representations*, San Juan, Puerto Rico, May, 2016, pp. 1-14.

[11] H. Li, A. Kadav, I. Durdanovic, H. Samet, H. P. Graf, Pruning filters for efficient convnets, *International Conference on Learning Representations*, Toulon, France, April, 2017, pp. 1-13.

[12] R. Abbasi-Asl, B. Yu, Structural compression of convolutional neural networks based on greedy filter pruning, *arXiv preprint* arXiv:1705.07356, March, 2017.

[13] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, L. S. Davis, Nisp: Pruning networks using neuron importance score propagation, *IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, June, 2018, pp. 9194-9203.

[14] K. Neklyudov, D. Molchanov, A. Ashukha, D. P. Vetrov, Structured bayesian pruning via log-normal multiplicative noise, *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, December, 2017, pp. 6775-6784.

[15] C. Louizos, M. Welling, D. P. Kingma, Learning sparse neural networks through L_0 regularization, *International Conference on Learning Representations*, Vancouver, BC, Canada, April, 2018, pp. 1-13.

[16] D. Molchanov, A. Ashukha, D. Vetrov, Variational dropout sparsifies deep neural networks, *International Conference on Machine Learning*, Sydney, Australia, August, 2017, pp. 2498-2507.

[17] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, Q. Tian, Variational convolutional neural network pruning, *IEEE Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, June, 2019, pp. 2780-2789.

[18] Z. Liu, M. Sun, T. Zhou, G. Huang, T. Darrell, Rethinking the value of network pruning, *International Conference on Learning Representations*, New Orleans, LA, USA, May, 2019, pp. 1-21.

[19] Y. LeCun, J. S. Denker, S. A. Solla, Optimal brain damage, *Advances in Neural Information Processing Systems*, Denver, CO, USA, November, 1989, pp. 598-605.

[20] B. Hassibi, D. G. Stork, Second order derivatives for network pruning: Optimal brain surgeon, *Advances in Neural Information Processing Systems*, Denver, CO, USA, November, 1992, pp. 164-171.

[21] Y. Guo, A. Yao, Y. Chen, Dynamic network surgery for efficient dnns, *Advances in Neural Information Processing Systems*, Barcelona, Spain, December, 2016, pp. 1379-1387.

[22] X. Dong, J. Huang, Y. Yang, S. Yan, More is less: A more complicated network with less inference complexity, *IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July, 2017, pp. 1895-1903.

[23] H. Hu, R. Peng, Y. W. Tai, C. K. Tang, Network trimming: A data-driven neuron pruning approach towards efficient deep architectures, *arXiv preprint* arXiv: 1607.03250, July, 2016.

[24] J. Luo, H. Zhang, H. Zhou, C. Xie, J. Wu, W. Lin, Thinet: Pruning cnn filters for a thinner net, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 41, No. 10, pp. 2525-2538, October, 2019.

[25] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, *IEEE International Conference on Computer Vision*, Venice, Italy, October, 2017, pp. 2755-2763.

[26] W. Wen, C. Wu, Y. Wang, Y. Chen, H. Li, Learning structured sparsity in deep neural networks, *Advances in Neural Information Processing Systems*, Barcelona, Spain, December, 2016, pp. 2074-2082.

[27] G. E. Hinton, D. van Camp, Keeping the neural networks simple by minimizing the description length of the weights, *Sixth Annual Conference on Computational Learning Theory*, Santa Cruz, CA, USA, July, 1993, pp. 5-13.

[28] K. Ullrich, E. Meeds, M. Welling, Soft weight-sharing for neural network compression, *International Conference on Learning Representations*, Toulon, France, April, 2017, pp. 1-16.

[29] C. Louizos, K. Ullrich, M. Welling, Bayesian compression for deep learning, *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, December, 2017, pp. 3288-3298.

[30] I. Bello, B. Zoph, V. Vasudevan, Q. V. Le, Neural optimizer search with reinforcement learning, *International Conference on Machine Learning*, Sydney, NSW, Australia, August, 2017, pp. 459-468.

[31] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L. J. Li, L. Fei-Fei, A. Yuille, J. Huang, K. Murphy, Progressive neural architecture search, *arXiv preprint* arXiv: 1712.00559, July, 2018.

[32] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, J. Dean, Efficient neural architecture search via parameters sharing, *arXiv preprint* arXiv: 1802.03268, February, 2018.

[33] A. Ashok, N. Rhinehart, F. Beainy, K. M. Kitani, A. Ashok, N. Rhinehart, F. Beainy, K. M. Kitani, N2n learning: Network to network compression via policy gradient reinforcement learning, *arXiv preprint arXiv*: 1709.06030, December, 2017.

[34] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, S. Han, Amc: Automl for model compression and acceleration on mobile devices, *European Conference on Computer Vision*, Munich, Germany, September, 2018, pp. 815-832.

[35] L. Xie, A. Yuille, Genetic CNN, *IEEE International Conference on Computer Vision*, Venice, Italy, October, 2017, pp. 1388-1397.

[36] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, A. Kurakin, Large-scale evolution of image classifiers, *International Conference on Machine Learning*, Sydney, NSW, Australia, August, 2017, pp. 2902-2911.

[37] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, K. T. Cheng, J. Sun, Metapruning: Meta learning for automatic neural network channel pruning, *IEEE International Conference on Computer Vision*, Seoul, South Korea, November, 2019, pp. 3295-3304.

[38] D. P. Kingma, T. Salimans, M. Welling, Variational dropout and the local reparameterization trick, *Advances in Neural Information Processing Systems*, Montreal, Quebec, Canada, December, 2015, pp. 2575-2583.

[39] D. Kingma, J. Ba, Adam: A method for stochastic optimization, *International Conference on Learning Representations*, San Diego, CA, USA, May, 2015, pp. 1-15.

[40] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278-2324, November, 1998.

[41] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, *International Conference on Learning Representations*, San Diego, CA, USA, May, 2015, pp. 1-14.

[42] K. Filonenko, R. Wisnovsky, M. Cheriet, *Learning multiple layers of features from tiny images*, Ph. D Thesis, University of Toronto, Toronto, Canada, 2012.

[43] S. Zagoruyko, 92.45% on cifar-10 in torch. http://torch.ch/blog/2015/07/30/cifar.html, 2015.

[44] Z. Huang, N. Wang, Data-driven sparse structure selection for deep neural networks, *European Conference on Computer Vision*, Munich, Germany, September, 2018, pp. 317-334.

[45] X. Dong, J. Huang, Y. Yang, S. Yan, More is less: A more complicated network with less inference complexity, *IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July, 2017, pp. 1895-1903.

[46] Y. He, G. Kang, X. Dong, Y. Fu, Y. Yang, Soft filter pruning for accelerating deep convolutional neural networks, *International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, July, 2018, pp. 2234-2240.

[47] Y. He, P. Liu, Z. Wang, Z. Hu, Y. Yang, Filter pruning via geometric median for deep convolutional neural networks acceleration, *IEEE Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, June, 2019, pp. 4335-4344.

[48] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, D. Doermann, Towards optimal structured cnn pruning via generative adversarial learning, *IEEE Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, June, 2019, pp. 2785-2794.

[49] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, L. Shao, Hrank: Filter pruning using high-rank feature map, *IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, June, 2020, pp. 1526-1535.

Biographies

Shuo Han is currently working toward the M.S. degree in School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. Her research interests include deep learning, model compression and so on.

Yufei Zhan is a senior student at Glasgow College, UESTC. She is extremely intrested in the area of human behaviour recognition (HBR), machine learning, etc.

Xingang Liu is a professor with the school of ICE, UESTC. His research interests include HBR, deep learning, video codec, and so on. He has published more than 90 academic papers in related magazines, journals, and conferences. Dr. Liu is an IEEE Senior Member, and a Fellow of IET. He won the IEEE TCSC Award for Excellence for Early Career Researchers in 2016 because of his achievement on scalable coding for multimedia signals.