LETTER Communicated by Han Xu

Cross-Entropy Pruning for Compressing Convolutional Neural Networks

Rongxin Bao
rxbao@foxmail.com
Xu Yuan
david@dlut.edu.cn
Zhikui Chen
zkchen@dlut.edu.cn
Ruixin Ma
maruixin@dlut.edu.cn
School of Software, Dalian University of Technology, Dalian, Liaoning, China

The success of CNNs is accompanied by deep models and heavy storage costs. For compressing CNNs, we propose an efficient and robust pruning approach, cross-entropy pruning (CEP). Given a trained CNN model, connections were divided into groups in a group-wise way according to their corresponding output neurons. All connections with their cross-entropy errors below a grouping threshold were then removed. A sparse model was obtained and the number of parameters in the baseline model significantly reduced. This letter also presents a highest cross-entropy pruning (HCEP) method that keeps a small portion of weights with the highest CEP. This method further improves the accuracy of CEP. To validate CEP, we conducted the experiments on low redundant networks that are hard to compress. For the MNIST data set, CEP achieves an 0.08% accuracy drop required by LeNet-5 benchmark with only 16% of original parameters. Our proposed CEP also reduces approximately 75% of the storage cost of AlexNet on the ILSVRC 2012 data set, increasing the top-1 errorby only 0.4% and top-5 error by only 0.2%. Compared with three existing methods on LeNet-5, our proposed CEP and HCEP perform significantly better than the existing methods in terms of the accuracy and stability. Some computer vision tasks on CNNs such as object detection and style transfer can be computed in a high-performance way using our CEP and HCEP strategies.

1 Introduction

In recent years, deep neural networks (DNNs) have achieved great success in many fields. Convolutional neural networks (CNNs) perform especially well on image and video recognition, object detection, face alignment, and image style transfer, since AlexNet-popularized CNNs won the ImageNet

Neural Computation **30, 1–22** (2018)
doi:10.1162/neco_a_01131

Large Scale Visual Recognition Competition (ILSVRC) in 2012 (Krizhevsky, Sutskever, & Hinton, 2012). The recent trend is that higher accuracy can be achieved by making a deeper network with more parameters. For example, the Visual Geometry Group (VGG)–16 has approximately 134 million parameters and VGG-19 148 million parameters (Simonyan & Zisserman, 2014). These high-capacity networks have high memory consumption and low processing speed (Zou, Zheng, Miao, Mckeown, & Wang, 2017). This makes it complicated to employ CNNs on computationally limited platforms and timely applications such as augmented reality, self-driving, and target tracking.

Various attempts have been made to compress or accelerate neural network models by trained quantization (Rastegari, Ordonez, Redmon, & Farhadi, 2016), optimized implementation (Bagherinezhad, Rastegari, & Farhadi, 2016), decomposition (Tai et al., 2015), or pruning. Among these methods, pruning has received widespread attention recently due to its high compression rate and low accuracy loss. The concept of pruning was first proposed by LeCun, Denker, and Solla (1989), inspired by the fact that connections between biological brain neurons can be destroyed. This was a valid method able to reduce network complexity (Luo, Wu, & Lin, 2017). Besides, pruning with retraining has been proved to be effective in reducing overfitting and redundancy of a network (Sun, Wang, & Tang, 2016). Even though these two methods have demonstrated great success, there are still several problems with network pruning. For example, how to choose pruning strategies is a serious problem for different network models. In addition, there are two further problems. First, after obtaining a well-trained dense network, pruning and retraining layer by layer will take considerable time and effort, especially when the initial accuracy after pruning is relatively low. Second, different structures of multilayered compression on the same network result in unstable performance. In other words, accuracy varies greatly in different structures and often has to be tested for many times.

To solve the problem of network complexity, Srinivas and Babu (2015) proposed removing redundant neurons in a systematic way, which reduces the computational complexity of retraining. But accuracy is decreased much more than that of baseline models. For the other problem Sun et al. (2016) propose a novel neural correlation–based pruning criterion to ensure the performance of moderately sparse DeepID2+ models and improve the accuracy of face recognition. Han, Mao, and Dally (2015) and Han, Pool, Tran, and Dally (2015) calculated a trade-off curve between accuracy loss and compression rate with $L1$ and $L2$ regularization, and concluded that "the more parameters pruned away, the less the accuracy." However, these architectures are feasible only for very deep neuron networks. Although good results have been achieved on large models in these works, the performance on small models is rarely explored.

To avoid these limitations, this letter proposes a robust cross-entropy pruning strategy for compressing CNNs: CEP. Inspired by the optimal

brain damage (LeCun et al., 1989) rule of "deleting connections with small saliency," we first group connections according to their output neurons. Then each connection, is assumed to be eliminated for calculating the cross-entropy error of connections. Afterward, all connections with the error below a threshold are deleted from the dense network model by binary pruning filters that have the same shape as weight matrices. Finally, the sparse network is retrained layer by layer. Experimental results on the MNIST and ILSVRC 2012 data sets demonstrate that our proposed CEP strategy can achieve a high compression rate and low accuracy loss for deep networks. Moreover, the method also has good performance on both initial accuracy after pruning and stability for different compression structures, even for low-redundant networks. These results show that the CEP strategy can extract effective connection sets with high predictive accuracy.

The remainder of this letter is structured as follows. Section 2 introduces related work about popular pruning methods for deep networks. Section 3 describes the cross-entropy pruning strategy, which is applicable to both fully connected (FC) and convolutional layers. Four groups of experiments on three types of networks are reported in section 4. Section 5 concludes and notes our future work.

2 Related Work

Current DNN models suffer from heavy redundancy and overparameterization. For example, Denil, Shakibi, Dinh, Ranzato, and De Freitas (2013) pointed out that a neural network could be restructured with only a small fraction of its original parameters. Furthermore, recent research by Wang, Xu, You, Tao, and Xu (2016) demonstrated that deletions of around 85% weights of CNN models would not affect their accuracies. Hence, weight management for deep networks is very important, and various compression methods have been proposed. Compared to other compression methods, pruning has been proved to be effective in reducing network redundancy. In this section, we describe several popular pruning methods since the birth of pruning.

2.1 Reducing Training Error. Early pruning methods focused on reducing increased training errors. For example, LeCun et al. (1989) proposed an optimal brain damage (OBD) pruning strategy in 1989. They calculated the saliencies by computing the second derivatives for each parameter and then pruned parameters with low saliency. Similarly, the optimal brain surgeon (OBS) strategy (Hassibi & Stork, 1992; Srinivas & Babu, 2015) added an adjustment stage for the remaining parameters after each parameter was pruned. Hence, there was no need to retrain or fine-tune network models in OBS. Inspired by their work, Lebedev and Lempitsky (2016) explored pruning clustered weights by grouping sparsity regularizer. However, some complex operations (i.e., second-order derivative) needed to be calculated

in these methods. They were not suitable for current deep networks due to heavy computation consumption.

2.2 Magnitude-Based Pruning. In order to simplify a pruning process, some researchers have paid attention to magnitude-based pruning methods. A simple weight magnitude pruning method proposed by Gale, Vestheim, Gravdahl, Fjerdingen, & Schjolberg (2013) enhanced the weight powers method on radial basis network (RBF), for removing redundant units. Han et al. (2015), Han, Liu et al. (2016), Han, Pool et al. (2016), and Han, Pool et al. (2015) proposed a series of state-of-the-art methods in which all connections with their weights less than a threshold were deleted. Quantization and Huffman coding were then used to further compress model size. A promising compression ratio with no accuracy loss can be thus achieved. For sparse matrices after being pruned, Han, Pool et al. (2016) proposed a dense-sparse-dense training flow with no inference overhead. In spite of good results, the performance on smaller models that are more suitable for mobile devices has rarely been explored.

2.3 Importance-Based Pruning. As a recent trend, various methods of importance-based pruning have been explored too. To reduce the size of a network model, Sabo and Yu (2008) calculated the sensitivity of neurons and developed a cross-validation-based pruning algorithm in 2008. More recently, Sun et al. (2016) computed a correlation between connections and their neural activations. Moreover, by dropping 88% of connections with low correlation, the VGG-like face recognition model achieves improved accuracy. Different from weights pruning, Rueda, Grzeszick, and Fink (2017) first grouped neurons into Maxout units and then pruned neurons by removing the least active neurons sorted by a Maxout architecture. However, the architecture lacks flexibility and universality and is feasible only for very deep neuron networks. It is also a common fault of recent pruning methods.

In addition, there are various effective strategies for compressing and speeding up CNNs, including tensor decomposition (Garipov, Podoprikhin, Novikov, & Vetrov, 2016; Novikov et al., 2015) and binary/ternary weight networks (Hubara, Courbariaux, Soudry, & Yaniv, 2016; Li, Zhang, & Liu, 2016). They can be brought together with pruning for further improvement.

Much pruning work has focused on reducing the loss of prediction accuracy. However, there are still many problems in existing pruning strategies. Due to morbid pruning strategies, overly complex operations in a layer-by-layer pruning process take considerable time and effort. Moreover, the performance on smaller CNN models that is suitable for timely platforms and applications is rarely examined, which leads to the universality and stability of the existing strategies. To tackle these problems, this letter focuses on both large and small network models by proposing a robust

cross-entropy–based pruning strategy for compressing CNNs. The stability and accuracy of our proposed CEP strategy under a variety of network models have been validated by experiments reported in section 4.

3 Cross-Entropy Pruning

In this section, we describe our novel CEP strategy. We first describe the concept of traditional pruning strategies: grouping, pruning of weights and neurons, and retraining. Then we present how to prune weights by our CEP strategy in detail. We present a cross-entropy pruning algorithm for fully connected layers, which is also suitable for neural networks with only fully connected layers (e.g., MLP). Finally, we generalize our method to convolutional layers and CNN models.

3.1 Pruning Strategy. In weight pruning, we first build a convolutional neural network for a given data set, and then train the network until all of its parameters converge. Thus, the baseline dense model M_0 is formed. Next, some connections in M_0 are pruned (assign to zero) by a cross-entropy pruning filter in a layer-wise way, from the last layer to the previous layers. After each pruning in a layer L_i, a sparse model M_i is initialized by its last model M_{i-1} and retrained without updating pruned weights. Since the initial values have been trained, the whole retraining process is developed in a finely tuned way. In this way, a series of increasingly sparse models with fewer and fewer connections is obtained (Sun et al., 2016).

In general, the last few layers of a convolutional neural network are fully and locally connected layers and contain most of the weights of the model, resulting in high redundancy. In principle, weight pruning in these redundant layers would not have a greater influence on performance than convolutional layers. Hence, connections in higher layers are supposed to be pruned away first, and then other connections are pruned down layer by layer. This is because the neurons between different layers are connected. The outputs of the lower layers are used as the inputs of the higher neurons. Furthermore, higher layers have a stronger prediction ability with more complete features than lower layers do.

Given the total number of original weights $|W|$, we define the degree of sparsity as $\lambda(0 < \lambda < 1)$, which is quantified as pruned weights/original weights. Hence, a total of $\lambda|W|$ weights will be pruned from the baseline model. In principle, we intend to find a sequence of weights that make smaller contributions to the network, and the deletion of the weights will cause the minimal impact of training errors. As shown in Figure 1, we first train the network and obtain the dense baseline model. Second, we divide connections into groups in a group-wise way, according to their corresponding output neurons. Meanwhile, the number of groups is the same as that of output neurons. Finally, we delete a certain proportion of connections in each group according to our cross entropy pruning strategy and

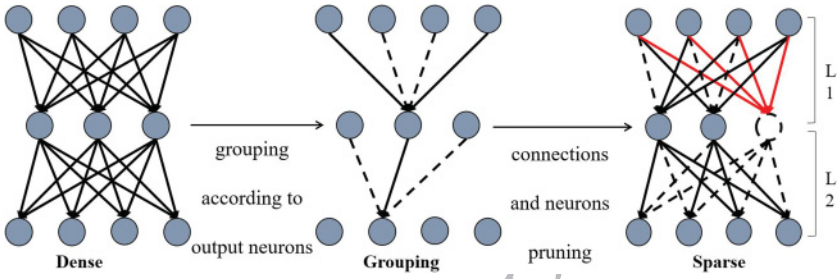

Figure 1: Pruning and retraining flow. The connections are grouped and pruned. Dotted line: pruned connections; dotted circle: pruned neurons; red line: input connections of pruned neurons.

retrain the sparse network layer by layer. In particular, if all the output connections of a neuron are pruned, the neuron and all its input connections are also pruned. In the next section, we describe in detail how to prune weights.

3.2 Weight Pruning. The key step of a neural network backpropagation (BP) algorithm (the key of the gradient descent algorithm) is to calculate the sensitivity of neurons. The difference among BP algorithms that use different loss functions and activation functions is dependent mainly on their sensitivity. Cross-entropy error (CEE), as a loss function, is widely used in many neural network models. CEE reveals the difference between the output and the target value of a network. Such a difference can be used to calculate the sensitivity of each neuron (Zhao, Chen, Yang, Hu, & Obaidat, 2016; Li, Zhao, Huang, & Gong, 2014). Motivated by this, we propose a cross-entropy pruning strategy assuming each weight is pruned, respectively, and then calculate CEE as losses for the trained baseline model. Finally, weights with the least CEE will be pruned.

3.2.1 Cross-Entropy of Pruning Each Weight. First, we consider fully and locally connected layers, including more connections and nonshared weights. Weights and connections have one-to-one correspondence in these layers. Given a fully connected layer L_i with i input neurons, its input is $x \in \mathbb{R}^{i \times n}$, where $n = 1, 2, \ldots, N$ denotes the number of input data. Similarly, the next layer L_{i+1} with h input neurons is given. Hence, according to the forward propagation algorithm, the inactive value of the output neuron net_h (input neuron of L_i) is

$$net_h = sum(W, x) = \sum_{i \in num\_i} W_{h,i} x_i + b_h, \tag{3.1}$$

where $W_{h,i} \in \mathbb{R}^{h \times i}$ denotes the weight matrix between L_i and L_{i+1}, and num_i the number of input neurons. The output value is activated by the activation function, such as Sigmod, tanh, ReLU and Softmax. Consider the Softmax function, for example; if its general form is given in equation 3.2, the activated neuron output value y_h is obtained in equation 3.3.

$$y_c = \varsigma(a_c) = \frac{\exp(a_c)}{\sum_{c'} \exp(a_{c'})}, \tag{3.2}$$

$$y_h = f(net_h) = \varsigma(net_h), \tag{3.3}$$

where c' denotes all parameters except for c. For each connection, it is then assumed to be pruned away, and the activation value of the corresponding neurons $\hat{Y}_{hi}$ after pruning is calculated. Hence, the calculation mode of $\hat{Y}_{hi}$ is formulated as

$$\hat{Y}_{hi} = \varsigma\left(\sum_{i \in num\_i} W'_{h,i} x_i + b_h\right), \forall h \in \{1, 2, \ldots, num\_h\}. \tag{3.4}$$

It is easy to see that the difference between y_h and $\hat{Y}_{hi}$ is whether it is to be pruned. On the basis of W, the weight of the connection that is assumed to be pruned, W'_{hi}, is set to zero. Since each connection is assumed to prune, the shapes of $\hat{Y}$ and W are the same. After obtaining y_h and $\hat{Y}_{hi}$, we calculate the cross-entropy of original output and pruned output (Huang, Li, Weng, & Lee, 2014). According to the maximum likelihood estimation, the average negative cross-entropy is

$$C_{hi} = \underset{n}{\text{avg}}(-sum(W'x)) = \underset{n}{\text{avg}}\left(-\sum_{h \in num\_h} y_h \ln \hat{Y}_{hi}\right), \tag{3.5}$$

where C_{hi} denotes the cross-entropy of pruning each weight. It serves as a major benchmark for pruning. The C_{hi} has only a positive value, which is helpful for the output of the corresponding neuron. Furthermore, the magnitude of C_{hi} represents the degree of its influence on the output of the network. Hence, from all C_{hi} for $i = 1, 2, \ldots, num\_i$, all positive values are sorted in descending order, denoted as $desC_{hi}$ for $i = 1, 2, \ldots, num\_i$. According to the degree of sparsity, $desC_{hi}$ is sampled into two parts based on coefficient ranks, having $\lambda * num\_i$ and $(1 - \lambda) * num\_i$ numbers, respectively. Specifically, considering weights with lower values, which are kept, and complementary to weights with higher values, we set L denoting the proportion of weights with lower C_{hi} values. Then $desC_{hi}$ can be divided into three parts: $\lambda * num\_i$, $\mu * num\_i$, and $(1 - \lambda - \mu) * num\_i$. The total number of pruned weights is $\lambda * num\_i$, which depends on the degree of sparsity λ.

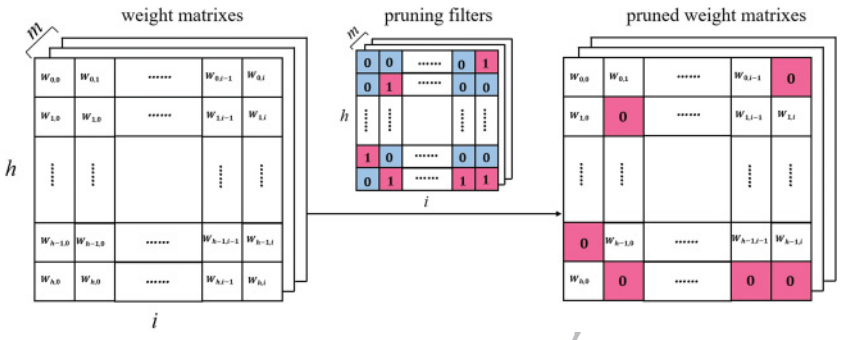

Figure 2: Pruning process for weight matrices. In pruning filters, 0 = kept (blue) and 1 = pruned (pink). Pruned weights are set to 0.

3.2.2 Pruning Filter. A pruning filter is used to identify the weights that are pruned and not being updated (Sari & Xiao, 2011). Considering a multilayer perceptron with n fully and locally connected layers, we define a pruning filter $F_m \in \mathbb{R}^{k_m \times i_m}$, which has the same shape as weight matrix $W_m \in \mathbb{R}^{k_m \times i_m}$. As shown in Figure 2, the filter matrices are filled with 0 and 1 to indicate that the weights are kept or pruned in the current layer. For each output neuron, all the weights connected to the neuron with cross-entropy below a threshold will be pruned, which will save $n \sum_m \lambda h_m i_m$ operations in forward propagation and backpropagation, respectively.

3.2.3 Backpropagation. The BP algorithm is based on the gradient descent strategy, which updates the parameters in the negative gradient direction of the target (Li, Kadav, Durdanovic, Samet, & Graf, 2016). Once a weight is pruned, it will not be updated in the BP process. This depends on the value of pruning filters F_m. Given the error E_n and learning rate η, the weights are updated as

$$W_{hi} = W_{hi} + \Delta W_{hi}, \tag{3.6}$$

$$\Delta W_{hi} = \begin{cases} -\eta \dfrac{\partial E_n}{\partial W_{hi}} & F_{hi} = 0 \\ 0 & F_{hi} = 1 \end{cases}. \tag{3.7}$$

Similar to forward propagation, the cross-entropy pruning in the BP process will save $n \sum_m \lambda h_m i_m$ operations. In the training process, the number of total operations $2n \sum_m \lambda h_m i_m$ is saved. The same shape as weight matrices is the reason that pruning filters consume a large amount of storage space. However, it exists only in the training stage and will be removed in the final

model. Hence, the running speed and size of the model will not be affected by pruning filters.

3.2.4 CEP Algorithm for FC Layers. The cross-entropy error matrix C has the same shape as that of weight matrix W. In addition, the assumed pruned weights need to be marked in the process of calculating cross-entropy. Hence, it is necessary to design an iteration structure with three loops, where a new parameter t is used to mark pruned weights. Based on the forward propagation algorithm, the CEP algorithm is outlined in algorithm 1.

Algorithm 1 mainly describes the calculation process of cross-entropy for each connection in an FC layer. Steps 1 and 4 go through all the connections in one layer, where i and j are the number of neurons in the last layer and the next layer, respectively. Step 3 marks the connection that is assumed to be pruned. Steps 5 to 9 are designed to compute the cross-entropy of the original output and pruned output. Steps 13 to 15 are to be pruned according to the magnitude of the cross-entropy of connections in a group-wise way.

After the algorithm is executed, we obtain cross entropy-error matrix C and pruned weight matrix P. Furthermore, the size of the baseline model has already been reduced. This is because the partial values in the matrix have been replaced by binary values, which are much smaller than double-format values.

3.3 Convolutional Layer Pruning. In convolutional layers, image data are transmitted in a local sensing way, and weights in each convolution kernel are shared by local images. In convolutional layers, a 3D convolution kernel $\mathcal{K} \in \mathbb{R}^{d\times k\times k}$ consists of the number of m 2D kernel $K \in \mathbb{R}^{k\times k}$ (e.g., 5×5), where m is the channel number of the convolution kernel. Suppose $i = k \times k$ denoting the size of the input feature map. Similar to the definition in section 3.2, for each channel, the cross-entropy of weights in convolutional layer c_{hi} is calculated as follows:

$$\begin{aligned} c_{him} &= \sum_{d=1}^{num\_d} \left| \operatorname*{avg}_{n} \left(- \sum_{h=1}^{num\_h} y_{hd} \ln \hat{\mathrm{Y}}_{hid} \right) \right| \\ &= \sum_{d=1}^{num\_d} \left| \operatorname*{avg}_{n} \left(- \sum_{h=1}^{num\_h} y_{hd} \ln \frac{\exp(sum_c(W', x))}{\sum_{c'} \exp(sum_{c'}(W', x))} \right) \right|. \end{aligned} \tag{3.8}$$

Compared to c_{hi} in FC layers, each parameter in equation 3.8 adds a new dimension m, where $d = 1, 2, \ldots, num\_d$. Moreover, the total number of the saved operations is also related to d. The rank $des(c_{him})$ and pruning filter F are created to delete the connections that are similar to those created in fully connected layers.

Algorithm 1: CEP Algorithm.

Input: input of the layer: $I[j]$, original output $N[h]$

Output: $C[t][i]$, Pruned weight matrix $PW[j][i]$

```
for i = 1, 2, , num_h do
    Y=0;
    for t = 1, 2, , num_i do
        for j = 1, 2, , num_i do
            if t!=j then
                Y+ = I [j] * W [j][i];
            end
            Y+=B [i];
            C [t][i] + = (-log(Activationfunction(Y)) *  N[i]);
        end
    end
end
foreach connection corresponding to the same output neuron do
    Sort |C[:][i]| and generate pruning filters F_i;
    Prune the connections according to F_i
end
```


3.4 Why Cross-Entropy-Based Pruning Works. When using the cross-entropy loss function of neural networks, we can estimate the importance of each connection. We presented the specific formulas in the previous section. But there remain questions of how and why cross-entropy-based pruning works.

To answer these questions, we start with considering the activation function further. Assuming Softmax as an activation function, we rewrite the

cross-entropy in equation 3.9 as equations 3.2 and 3.4:

$$
\begin{aligned}
&-\sum_{h\in num\_h} y_h \ln \hat{Y}_{hi} = -\sum_{h\in num\_h} y_h \ln \frac{\exp(sum_c(W',x))}{\sum_{c'}\exp(sum_{c'}(W',x))} \\
&= -\left(\sum_{h\in num\_h} y_h sum_c(W',x)\right) - \ln\left(\sum_{c'}\exp(sum_{c'}(W',x))\right) \\
&= \begin{pmatrix} \ln(1+\sum\limits_{\substack{c'\\ c'\neq l}}\exp(sum_{c'}(W',x)-sum_l(W',x))) \\ -\sum\limits_{\substack{h\\ h\neq l}} y_h(sum_c(W',x)-sum_l(W',x)) \end{pmatrix},
\end{aligned}
\tag{3.9}
$$

where l is the index of $\max_c sum_c(W',x)$. By using the activation function, the cross-entropy formula can effectively avoid numerical overflow (LeCun, Bottou, Bengio, & Haffner, 1998). To examine the modified cross-entropy, we calculate the sensitivity of the output neurons as derivative:

$$
\begin{aligned}
&\frac{\partial}{\partial sum_c(W',x)}\left(-\left(\sum_{h\in num\_h} y_h sum_c(W',x)\right) - \ln\left(\sum_{c'}\exp(sum_{c'}(W',x))\right)\right) \\
&= -(y_h - \frac{\exp(sum_c(W',x))}{\sum_{c'}\exp(sum_{c'}(W',x))}) = -(y_h - \hat{Y}_{hi}).
\end{aligned}
\tag{3.10}
$$

Equation 3.10 is simple and intuitive. Dramatically, after the target value y_h is vectorized, the sensitivity of the output neuron is the difference between its activation value $\hat{Y}_{hi}$ and the target value y_h, which is the same as that of the mean squared error (MSE). With the sensitivity of the output neurons, the updated value of each layer ΔW can be calculated according to the chain rule. The result is that MSE and CEE are similar when we update multilayer weights.

In addition to improving MSE and speeding up convergence, cross-entropy can also measure the correlation between two variables, $P_L(s_i|x_t)$ and $P(s_i|x_t)$. Then we will analyze cross-entropy-based pruning from the perspective of correlation. Generally the cross-entropy cost function in neural networks is formalized as

$$
\begin{aligned}
C &= -\sum_{i=1}^{c} P_L(s_i|x_t)\log P(s_i|x_t) \\
P_L(s_i|x_t) &= \begin{cases} 1, & label = i \\ 0, & else \end{cases}.
\end{aligned}
\tag{3.11}
$$

Differing from other correlation analysis methods (e.g., canonical correlation analysis), which have both positive and negative correlations, cross-entropy analysis has only positive correlations in neural networks. In equation 3.11, all summed items are nonnegative. This is because $P(s_i|x_t) \in (0, 1)$ is a neuron output value. Hence, after taking a negative value of the sum, the value of cross-entropy C is definitely positive. As a result, the correlation coefficients of all connections change in the same direction, which is more suitable for pruning operations and more easily processed by a computer.

In this letter, our consideration of cross-entropy comes from likelihood. In probability science, likelihood is a measurement criterion for the difference between two distributions. This criterion is more direct, and the greater the likelihood, the closer the two distributions are. Since all data are independently and identically distributed, the likelihood of all data can be defined as the product of the likelihood of all data points:

$$L(Y, \hat{Y}) = \prod_{h \in num\_h} L(y_h, \ln \hat{Y}_{hi}). \tag{3.12}$$

For equation 3.12, the maximum likelihood estimation method is used to calculate the optimal parameters. Obviously the maximum of the likelihood function should be equal to that of the logarithmic likelihood function, and then a negative of the value is equal to the minimum of the function. According to equation 3.11, the value of y_h is 1 or 0. Hence, the likelihood function is rewritten as

$$-\ln L(Y, \hat{Y}) = -\sum_{h \in num\_h} \ln L(y_h, \ln \hat{Y}_{hi}) = -\sum_{h \in num\_h} y_h \ln \hat{Y}_{hi}. \tag{3.13}$$

In this way, we obtain the cross-entropy function used for pruning. Meanwhile, this result also proves the validity of cross-entropy to measure the difference between probability distributions.

Inspired by the OBD rule of deleting connections with small saliency, we design CEP based on the idea of minimal impact on the accuracy of the baseline model. After each connection is assumed to be pruned, the corresponding weight W_{hi} in the weight matrix is set to zero. Afterward, the cross-entropy of the assumed output and baseline output is considered the saliency of the connection on the layer.

In general, if some connections are less important, their cross-entropy is relatively small. Therefore, a smaller cross-entropy of W_{hi} means that the connection is less useful in this layer, and its removal will have the smallest effect on the accuracy; thus, it can be deleted. Meanwhile, pruning and retraining layer by layer ensure that every pruned connection is the optimal choice for the current layer.

The similarity in their updating weights between CEE and MSE has been explained. With the correlation criterion as the error function, CEP pruned networks have achieved the lower initial global error with the higher initial accuracy. The convergence speed of the sparse network is thus accelerated, and the computation cost of retraining is reduced. These advantages of our CEP will be further validated in our experiments.

4 Experiments

We evaluate the performance of our CEP on three tasks against different types of networks: MLP (multilayer perceptron) on the MNIST data set, LeNet5 on the MNIST data set and AlexNet on the ImageNet (ILSVRC 2012) data set. Unlike VGG and ResNet, which are highly redundant, they are often used to demonstrate network compression. Our three selected networks contain fewer parameters with low redundancy. In principle, it is harder to compress low-redundancy networks than high-redundancy networks with high capacity. Moreover, a wide range of compression with little accuracy loss is challenging in low-redundancy networks. This is because the percentage of useless connections is small in these networks.

In order to compress more pervasive and more practical networks for applications, we first evaluate the astringency, accuracy, and compression rate of our CEP on MLP, which contains only fully connected layers. We then examine the effectiveness of CEP on LeNet5 and AlexNet, which are typical models of low-redundancy networks, with our focus on CNNs. Finally, we explore the pruning strategy of keeping weights with small CEE and compare our strategies with other pruning strategies.

4.1 Performance Evaluation on MLP. MLP on the MNIST data set with only one hidden layer is our baseline model. The MNIST handwritten digits data set has centered input images of 28×28 pixels and 10 output categories. Hence, to keep the redundancy of the network low, we design the network structure as a 784-dimensional input layer, a 20-dimensional fully connected hidden layer, and a 10-dimensional output layer. When training the MLP model, we do not use many tricks, such as Dropout, Adagrad, and ReLU. In order to facilitate the implementation of CEP, we train the network by using Sigmod as the activation function, cross-entropy error as the loss function, and gradient descent as the BP algorithm. In addition, the learning rate η is set to 0.0001 for the training process. Finally, the accuracy of baseline MLP model converges to around 0.9737.

In addition to the input layer, the network has two fully connected layers: the hidden layer is named L1 and the output layer L2. The total number of parameters in the network is around 16,000, which is even fewer than the number of parameters in a convolutional layer of VGG-16. As a result, it is difficult to achieve a high compression rate with little accuracy loss on the network using only network pruning. Still, we intend to fully evaluate the

Table 1: Accuracy, Compression Rate, and Kept Parameters for Sparsified MLP with Various Pruning Structures.

Structure	Accuracy	Compression Rate	Kept Parameters
Baseline model	0.9737	1	15,910
25%_L1/25%_L2	0.9731	0.7505	11,940
25%_L1/35%_L2	0.9727	0.7493	11,920
30%_L1/25%_L2	0.9725	0.7012	11,156
30%_L1/35%_L2	0.9720	0.7000	11,136
35%_L1/25%_L2	0.9717	0.6519	10,372
35%_L1/35%_L2	0.9698	0.6507	10,352
40%_L1/40%_L2	0.9641	0.6008	9558

performance of our CEP in a low-redundancy network (e.g., MLP) and then generalize it to the convolutional neural network.

In the experiments, we use fractions (e.g., 0.25, 0.30, 0.35) of sparsity degree λ to evaluate the performance under different compression rates. We gradually reduce the compression rate and pruned weights according to λ layer by layer; then sparsified models $M_0, M_1, \ldots, M_i$ are trained until they converge.

As shown in Table 1, with the change of pruning structure and the increase in pruned parameters, the prediction accuracy decreases gradually. The parameters in the table (the last column of table) are the total number of trainable parameters, including weights and biases. When pruning structures are dramatically increased to 40% L1/40% L2 (the last row of table), the accuracy is further reduced to 0.9641, nearly 1% lower on the baseline model. Therefore, in the experiments, we do not compress the original model any further.

It can be seen from the balance between the accuracy and compression rate that our CEP strategy on MLP can still maintain a lower accuracy loss even if the compression rate reaches around 70%. Unlike very deep neural networks that have been highly compressed, our baseline MLP model has only three layers with around 16,000 parameters. It is found that parameters in these low-redundancy networks cannot be significantly reduced without degrading accuracy.

To evaluate the performance of different pruning strategies on the model, we compare our CEP strategy with the correlation-based and magnitude-based pruning strategies. The comparison is conducted on the MLP dense model, with pruning 25% weights and 30% weights in both L1 and L2 fully connected layers. We sequentially prune the L2 and L1 layers and retrain the network by means of batch gradient descent algorithm. Using the three strategies, we take 28×28 square input of 60,000 MNIST handwritten images. Given that the learning rate is 0.0001 for retraining the L1 layer, we

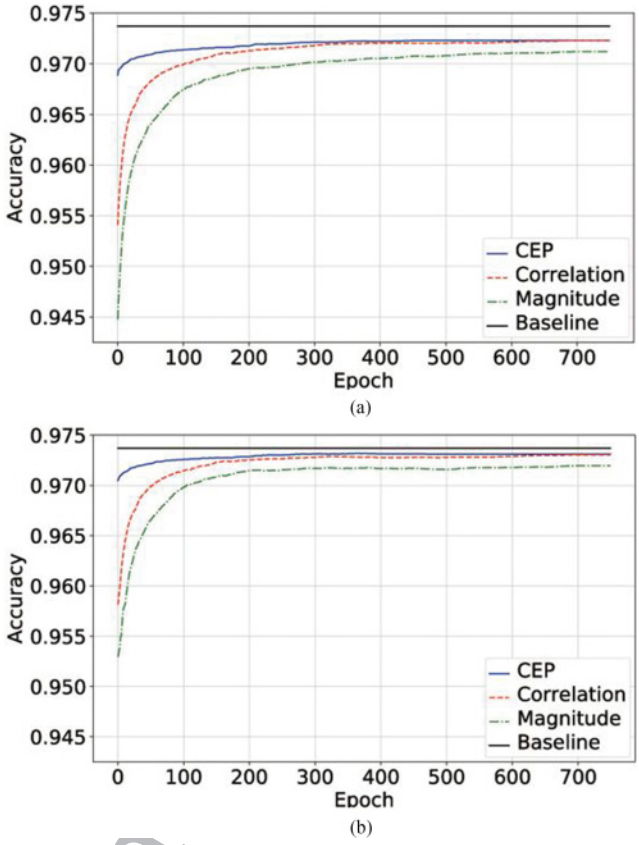

Figure 3: The accuracy curve of correlation, magnitude, and CEP, with (a) 25%_L1/25%_L2 and (b) 30%_L1/30%_L2 pruning structures and 0.0001 learning rate. The straight black line indicates the accuracy of baseline dense model.

train the three sparse models in a fixed epoch and plot the accuracy curve of three strategies in Figure 3.

As shown in Figure 3, all of the three sparse networks have converged when the training reaches 750 epochs. Our CEP strategy achieves the best performance for pruning layers L1 and L2, with 0.9731 accuracy, which is only about a 0.6% loss from the baseline model. Although the correlation-based strategy achieves almost the same accuracy as ours, its initial accuracy and convergence rate are not as good as our CEP. The obvious reason is that our CEP strategy selects the pruned weights that have the least impact on the prediction accuracy. Compared to the other two strategies, our strategy has the fastest convergence rate with the higher initial and prediction

Table 2: Accuracy, Compression Rate, and Pruned Connections When the Pruning Structure of Layers f1 and f2 Changes in the LeNet-5 Baseline Model.

Structure	Accuracy	Compression Rate	Pruned Connections
Baseline model	0.9919	1	0
50%f2	0.9912	0.9199	4800
75%f2	0.9917	0.8799	7200
50%f2_75%f1	0.9909	0.3195	40,800
75%f2_75%f1	0.9915	0.2795	43,200
75%f2_90%f1	0.9911	0.1594	50,400

Note: The number of connections is equal to weights in fully connected layers.

accuracies. Hence, the results of our experiment have sufficiently demonstrated the effectiveness of our CEP strategy.

4.2 LeNet-5 on MNIST. With promising accuracy and compression performances on MLP, we then implement our CEP strategy on a larger CNN: LeNet-5 on the MNIST data set. LeNet-5 is a convolutional neural network designed 1998 (LeCun et al., 1998), which contains two fully connected layers and two convolutional layers, with about 99.2% accuracy on the MNIST data set. For convenience, the higher fully connected layer is named f2 and the lower is named f1. In the experiment, we prune as many weights as possible in the two fully connected layers, while the bottom convolutional layers with fewer parameters are left untouched. Similar to CEP on MLP, the experiment is conducted on group-wise pruning and layer-by-layer training strategy.

It can be seen from Table 2 that LeNet-5 can be compressed to 15.94% with 0.08% accuracy reduction, pruning only on fully connected layers. Furthermore, approximately 50,000 connections in fully connected layers are deleted from the baseline model finally. One interesting result is that with the reduction of the compression rate, accuracy is sometimes improved, which is because redundancy and overfitting still exist in neural networks with higher compression rates. Once a good initialization is trained, pruning and retraining can improve the generalization of the network.

4.3 AlexNet on ILSVRC 2012. We further examine the performance of CEP on the ILSVRC 2012 data set, which contains 1.2 million training data and 50,000 test data. We use AlexNet implemented by Tensorflow as the baseline dense model, which is a convolutional neural network and achieves top-1 error of 0.4292 and top-5 error of 0.1998. There are five convolutional layers and three fully connected layers in AlexNet. For convenience, the three fully connected layers are named fc8, fc7, and fc6, from higher to lower. Similar to the implementation on LeNet-5, we leave

Table 3: Performances of Our CEP on AlexNet with Various Pruning Structures, including Top-1 Error, Top-5 Error, Compression Rate, and Number of Pruned Connections.

Structure	Top-1 Error	Top-5 Error	Compression Rate	P Connections
Baseline model	0.4292	0.1998	1	0
75%fc8_75%fc7	0.4312	0.2009	0.7432	15,654K
75%fc8_90%fc7	0.4319	0.1989	0.7019	18,171K
75%fc8_90%fc7_75%fc6	0.4329	0.1995	0.2374	46,483K
75%fc8_90%fc7_90%fc6	0.4337	0.2019	0.1445	52,145K

convolutional layers untouched. In our experiments, we calculate both the top-1 error and top-5 error of pruned sparse network models. The experimental results are shown in Table 3.

As shown in Table 3, with the reduction of the compression rate, the variation trends of top-1 error and top-5 error are even better than the top-5 error of the baseline model in rows 4 and 5. After pruning all three fully connected layers, more than 52 million connections are deleted from the baseline dense model. Furthermore, it is still a 0.4337 top-1 error rate with only about 14% of the connections. It is found that parameters in fully connected layer fc6 are the most redundant, because the pruned connections are several times that of the other two layers and the error rates do not rise significantly in the last two rows of Table 3.

4.4 Keep Weights with the Highest CEE. A selection of pruning strategies is one of the most important factors in pruning performance. The key concept of our CEP strategy considers pruning connections with the smallest CEE. In reality, keeping partial connections with the smallest weights or correlations to connected neurons is helpful for the sparse model (Sun et al., 2016; Han et al., 2015). Motivated by this, we have explored how to keep a majority of low CEE weights as well as a small number of weights with the highest CEE.

In the experiment, the new method is HCEP and μ is set to 5%. For example, 50% weight pruning in a layer denotes that we deleted 45% of the lowest CEE connections and 5% of the highest CEE connections. Moreover, we compare the novel HCEP strategy with OBD, magnitude pruning, correlation pruning, and our CEP strategy on LeNet-5 with the MNIST data set. All parameters in the LeNet-5 network are the same as those in our experiments in section 4.2. On the basis of our experimental results, we compare in Table 4 the performances on three pruning structures: 50%f2_75%f1, 75%f2_75%f1, and 75%f2_90%f1.

In the last two rows of Table 4, we verify that keeping a small portion of weights with the highest CEE might improve accuracy. When the pruning

Table 4: Comparison between our CEP, HCEP and Other pruning Strategies on LeNet-5.

Pruning Structure	OBD	Magnitude	Correlation	CEP	HCEP (5%)
50%f2_75%f1	99.12	98.65	99.03	99.09	99.10
75%f2_75%f1	99.01	98.63	99.13	99.15	99.11
75%f2_90%f1	98.97	98.43	98.87	99.11	99.13

structure comes to 50%f2_75%f1 and 75%f2_90%f1, the accuracy of HCEP is slightly higher than that of CEP. However, this is not always true and needs to be finely tuned according to the pruning structure.

Although OBD is better than the other four methods in the 50%f2_75%f1 pruning structure, its performance declines significantly when the number of pruned connections increases in f2 and f1 layers, lacking stability. In contrast, compared to other methods, our CEP and HCEP achieve higher accuracy and better stability (with accuracy 0.9912 ± 0.0003 on the MNIST). Our results also show the possibility of improving the accuracy by keeping some weights with extremum value and pruning them.

4.5 Method Comparisons. Besides the above comparison, we take data-free (Srinivas & Babu, 2015), SSL (Wen, Wu, Wang, Chen, & Li, 2016), magnitude (Han et al., 2015), correlation (Sun et al., 2016), and OBD (LeCun et al., 1989) methods as examples and compare top-1 error, compression rate, accuracy degradation, compressed size, as well as the retraining cost of AlexNet on the ILSVRC 2012 data set with our CEP and HCEP. For network compression methods, compression rate and accuracy are the most important criteria to verify the quality of the method. Compressed size and retraining cost are also key benchmark metrics for pruning. For each method, considering the balance between the accuracy and compression rate, we compare the best experimental results that have been confirmed by previous work. The compared results of AlexNet on the ILSVRC 2012 data set in Table 5 come from the experimental section of the corresponding papers.

To make the comparisons in a more comprehensive and convincing way, for each method, we recover the best results reported in the corresponding papers as much as possible: data free with fractions (0.66, 0.94, 0.83), magnitude with fractions (0.91, 0.91, 0.75), correlation with fractions (0.75, 0.75, 0.50), OBD with fractions (0.75, 0.75, 0.75), and our CEP and HCEP with fractions (0.75, 0.90, 0.90) for three fully connected layers. In the experiment, HCEP deletes 5% of connections with the highest CEE, and it prunes the same number of connections with CEP. Therefore, the compression rate and compressed size of CEP and HCEP methods are the same in Table 5.

Multiply-and-accumulate (MAC) operations in fully connected and convolutional layers occupy over 99% of total operations in DNNs. Therefore,

Table 5: Key Benchmark Metrics of Our CEP and HCEP versus Other Mainstream Methods of AlexNet Trained on ILSVRC 2012.

Method	Top-1 Error	Compression Rate	Accuracy Degradation	Compressed Size (in millions)	Retraining Cost ($\times 10^{12}$ MACs)
Data free	0.4440	0.6511	0.0148	39.72	$\approx$0
SSL	0.4467	0.9748	0.0175	59.46	3.67
Magnitude	0.4349	0.1303	0.0057	7.95	6.15
Correlation	0.4483	0.2910	0.0191	17.75	6.31
OBD	0.4402	0.2746	0.0110	16.75	5.50
CEP	0.4337	0.1445	0.0045	8.81	3.46
HCEP (5%)	0.4352	0.1445	0.0060	8.81	3.48

we use this benchmark to measure the consumption of retraining. In general, there are about 3.7×10^8 MACs operations in the process of forward propagation of AlexNet. These operations are mainly concentrated on the calculation of convolutional layers. In addition, the time-consuming and MAC operation of backward propagation are roughly three times that of forward propagation. Due to the different scales of the data set, the retraining costs reported in Table 5 are the MACs operations of only one image with 227×227 pixel.

Since the parameters in fully connected layers account for over 95% of total parameters in AlexNet, the pruning fractions of the fully connected layers will directly affect the compression rate and compressed size. However, SSL aims to regularize the filter and channel in convolutional layers for speeding up DNNs, which leads to a rather high compression rate. Furthermore, sparse convolution layers reduce the amount of computation, thus accelerating the retraining process.

As a state-of-the-art pruning strategy, magnitude-based pruning keeps ahead in both the top-1 error and compression rate. Although a good compression rate is achieved, its error rate is much worse than the best result reported in other work when pruning is used only in FC layers. Meanwhile, the initial accuracy after pruning is relatively low, which leads to slow convergence in layer-by-layer retraining. Therefore, MACs operations are rather huge for only one image. And that is also the reason the MACs operations of correlation-based pruning are huge. In contrast, the data-free method adds an adjustment stage for remaining parameters after each neuron is pruned. Hence, there is no need for retraining or fine-tuning network models in the method.

Compared with other methods, our CEP and HCEP perform first-class under multiple benchmarks. We have achieved around 85% group-wise sparsity for FC6-FC8 (the best-pruned network is reported in Table 5). By a reasonable compression structure, experimental results validate that our

proposed methods have good robustness and extensibility. With an acceptable accuracy degradation for the baseline model with about 0.43 top-1 error, our methods achieve a reasonable compression on AlexNet.

5 Conclusion

Currently CNNs are limited by huge network models with high inference costs. In this letter, we have presented CEP, a pruning method for reducing the number of parameters in CNNs. Our proposed method calculates the expected cross-entropy error for each connection and uses filters to prune weights in a group-wise way. With little accuracy loss, CEP has achieved around a 0.16 compression rate on LeNet-5 (on MNIST) and 0.14 on AlexNet (on ILSVRC 2012). Furthermore, we have explored keeping a small portion of weights with the highest CEE. The comparison experiments show that our methods have higher accuracy and stability under different pruning structures. As such, our method facilitates the possible implementation of CNNs on computationally limited platforms and timely applications.

In the future, we would like to combine our CEP strategy with tensorizing neural networks for further acceleration. An alternative for error-based weight selection strategies is also expected to be explored.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under grant 2016YFD0800300.

References

Bagherinezhad, H., Rastegari, M., & Farhadi, A. (2016). LCNN: Lookup-based convolutional neural network. In *Proceedings of the IEEE Conference Computer Vision and Pattern Recognition* (pp. 860–869). Washington, DC: IEEE Computer Society.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., & De Freitas, N. (2013). Predicting parameters in deep learning. *University of British Columbia*, pages 2148–2156.

Gale, S., Vestheim, S., Gravdahl, J. T., Fjerdingen, S., & Schjolberg, I. (2013). RBF network pruning techniques for adaptive learning controllers. In *Proceedings of the IEEE Convention on Robot Motion and Control* (pp. 246–251). Piscataway, NJ: IEEE.

Garipov, T., Podoprikhin, D., Novikov, A., & Vetrov, D. (2016). *Ultimate tensorization: Compressing convolutional and FC layers alike*. arXiv:1611.03214.

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. *ACM Sigarch Computer Architecture News*, *44*(3), 243–254.

Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *Fiber*, *56*(4), 3–7.

Han, S., Pool, J., Narang, S., Mao, H., Gong, E., Tang, S., . . . Tran, J. (2016). *DSD: Dense-sparse-dense training for deep neural networks*. arXiv:1607.04381.

Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems*, *15* (pp. 1135–1143).

Hassibi, B., & Stork, D. G. (1992). Second order derivatives for network pruning: Optimal brain surgeon. In C. L. Giles, S. J. Hanson, & J. D. Cowan (Eds.), *Advances in neural information processing systems*, *5* (pp. 164–171). San Mateo, CA: Morgan Kaufmann.

Huang, Z., Li, J., Weng, C., & Lee, C. H. (2014). Beyond cross-entropy: Towards better frame-level objective functions for deep neural network training in automatic speech recognition. In *Proceedings of the 15th Annual Conference of the International Speech Communication Association* (pp. 2236–2239). Baixas, France: International Speech Communication Association.

Hubara, I., Courbariaux, M., Soudry, D., Yaniv, R. E., & Bengio, Y. (2016). Binarized neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, L. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems*, *29*. Red Hook, NY: Curran.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *25* (pp. 1097–1105). Red Hook, NY: Curran.

Lebedev, V., & Lempitsky, V. (2016). Fast convnets using group-wise brain damage. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2554–2564). Piscataway, NJ: IEEE.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

LeCun, Y. L., Denker, J. S., & Solla, S. A. (1989). Optimal brain damage. In D. S. Touretzky (Ed.), *Advances in neural information processing systems*, *2* (pp. 598–605). San Mateo, CA: Morgan Kaufmann.

Li, F., Zhang, B., & Liu, B. (2016). Ternary weight networks. arXiv:1605.04711.

Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2016). Pruning filters for efficient ConvNets. arXiv:1608.08710.

Li, J., Zhao, R., Huang, J. T., & Gong, Y. (2014). Learning small-size DNN with output-distribution-based criteria. In *Proceedings of the 15th Annual Conference of the International Speech Communication Association* (pp. 1910–1914). Baixas, France: International Speech Communication Association.

Luo, J. H., Wu, J., & Lin, W. (2017). Thinet: A filter level pruning method for deep neural network compression. arXiv:1707.06342.

Novikov, A., Podoprikhin, D., Osokin, A., & Vetrov, D. (2015). Tensorizing neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems*, *28*. Red Hook, NY: Curran.

Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proceedings of the European Conference on Computer Vision* (pp. 525–542). Cham: Springer.

Rueda, F. M., Grzeszick, R., & Fink, G. A. (2017). Neuron pruning for compressing deep networks using maxout architectures. In *Proceedings of the German Conference on Pattern Recognition* (pp. 177–188). Cham: Springer.

Sabo, D., & Yu, X. H. (2008). A new pruning algorithm for neural network dimension analysis. In *Proceedings of the IEEE International Joint Conference on Neural Networks* (pp. 3313–3318). Piscataway, NJ: IEEE.

Sari, A., & Xiao, J. (2011). Efficient Levenberg-Marquardt minimization of the cross-entropy error function. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 1–8). Piscataway, NJ: IEEE.

Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition*. arXiv:1409.1556.

Srinivas, S., & Babu, R. V. (2015). *Data-free parameter pruning for deep neural networks*. arXiv:1507.06149.

Sun, Y., Wang, X., & Tang, X. (2016). Sparsifying neural network connections for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4856–4864). Piscataway, NJ: IEEE.

Tai, C., Xiao, T., Zhang, Y., Wang, X., & Weinan, E. (2015). *Convolutional neural networks with low-rank regularization*. arXiv:1511.06067.

Wang, Y., Xu, C., You, S., Tao, D., & Xu, C. (2016). Cnnpack: Packing convolutional neural networks in the frequency domain. In D. D. Lee, M. Sugiyama, U. V. Luxburg, L. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems*, *29*. Red Hook, NY: Curran.

Wen, W., Wu, C., Wang, Y., Chen, Y., & Li, H. (2016). Learning structured sparsity in deep neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, L. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems*, *29*. Red Hook, NY: Curran.

Zhao, L., Chen, Z., Yang, Z., Hu, Y., & Obaidat, M. S. (2016). Local similarity imputation based on fast clustering for incomplete data in cyber-physical systems. *IEEE Systems Journal*, *12*, 1610–1620.

Zou, L., Zheng, J., Miao, C., Mckeown, M. J., & Wang, Z. J. (2017). 3D CNN based automatic diagnosis of attention deficit hyperactivity disorder using functional and structural MRI. *IEEE Access*. doi:10.1109/ACCESS.2017.2762703.

Received February 3, 2018; accepted June 13, 2018.