Providing clear pruning threshold: A novel CNN pruning method via $L_0$ regularisation

Guo Li  |  Gang Xu

North China Electric Power University, Beijing 102206, China

Correspondence
Guo Li, North China Electric Power University, Beijing 102206, China.
Email: geliguol@gmail.com

Abstract

Network pruning is a significant way to improve the practicability of convolution neural networks (CNNs) by removing the redundant structure of the network model. However, in most of the existing network pruning methods $l_1$ or $l_2$ regularisation is applied to parameter matrices and the manual selection of pruning threshold is difficult and labor-intensive. A novel CNNs network pruning method via $l_0$ regularisation is proposed, which adopts $l_0$ regularisation to expand the saliency gap between neurons. A half-quadratic splitting (HQS) based iterative algorithm is put forward to calculate the approximation solution of $l_0$ regularisation, which makes the joint optimisation problem of regularisation term and training loss function can be solved by various gradient-based algorithms. Meanwhile, a hyperparameters selection method is designed to make most of the hyperparameters in the algorithm can be determined by examining the pre-trained model. The results of experiments on MNIST, Fashion-MNIST and CIFAR100 show that the proposed method can provide a much clearer pruning threshold by widening the saliency gap, and achieve a similar or even better compression performance, compared with the state-of-the-art studies.

1 | INTRODUCTION

Convolution neural networks (CNNs) are powerful tools in various computer vision tasks [1–4]. However, CNNs have complicated network architectures and are storage massive and computation-intensive, which results in high deployment cost and limits application scenarios. As an effective method for reducing the deployment cost in both storage and computation, network compression has aroused increasing attention in the community.

Numerous ways have been proposed to compress the network model, such as weight quantisation [5, 6], low-rank decomposition [7, 8], knowledge distillation [9] and network pruning. Weight quantisation compresses the model size and inference time by weight sharing and reducing the number of bits, which does no change in network structure. Therefore, it can not remove the structure redundancy and is often used with other methods. Low-rank decomposition decomposes a large convolution kernel into a smaller kernels or a cascade of small kernels to reduce computation load. But this type of method for decomposing the convolution kernel cannot delete the branches in the parallel structure, for example, the shortcut paths in resnet. Meanwhile, matrix decomposition is often computationally expensive. Knowledge distillation uses a pre-trained model to design a smaller architecture and then train from scratch. Network pruning achieves network simplification by removing unimportant substructures (e.g. filters, channels, layers) of the network without unaccepted accuracy drop. Network pruning starts with a pre-trained model, which is easier to keep model performance than knowledge distillation.

Many of the network pruning methods focus on saliency metric [10–20]. However, the saliency metric in the pre-trained model is often gaussian distribution [13–15, 20, 21]. It is difficult to select a set of layer-wise pruning thresholds directly without sparse regularisation.
In the line of regularisation, \( l_1 \) [22–24] and \( l_2 \) [17, 25–28] norm are mainstream choices. He et al. [20] stated that the effectiveness of norm-based pruning criterion depended on two requirements: (1) the norm deviation of the filters should be large, in other words, the numerical gap of norm between the important and unimportant filters should be large. (2) the minimum norm of the filters should be small. However, applying \( l_1 \) or \( l_2 \) regularisation in a parameters matrix will let all the parameters close to zero, which could hardly increase the saliency gap and results in more difficulties in adjusting the layer-wise pruning ratios in subsequent trimming steps. It should be noted that only the unimportant neurons other than all the neurons need to be regularised to zero and the others should be free from regularisation to find better values.

In general, pruning algorithms are iteration-based, which have two iteration strategies: global iteration and layer-by-layer iteration. Global iterative pruning [17, 24] can globally consider the sensitivity of each layer to get the optimal pruning ratio, but the conventional \( l_1 \) and \( l_2 \) norm in global regularisation hardly increase the saliency difference between the important and unimportant neurons, which makes it hard to trim too many neurons in a single iteration and the trimming ratio of each layer needs to be carefully adjusted. For large depth models that require multiple pruning, the manual selection of layer-wise pruning ratio at each iteration is a huge workload. In opposite, the layer-by-layer pruning method [14, 15] can reach the best pruning result in some layers but prone to over-cutting, especially in the first few layers. Once previous layers are over-pruned, the following needs more parameters to build a stronger classifier to compensate for the loss of feature information. Because the pruning sensitivity and saliency distribution in different layers are diverse, to obtain the globally optimal solution, layer-wise regularisation coefficients and thresholds need to be carefully selected, which is often labor-intensive.

In order to solve the shortcomings of the previous methods in the saliency gap and the difficulty of parameters tuning, we propose to use \( l_0 \) norm to regularise the model parameters. \( l_0 \) regularisation constrains some parameters to 0 while having no effect on other parameters, which makes some parameters free from regularisation to widen the saliency gap. Xu et al. [29] used HQS algorithm to solve \( l_0 \) regularisation problem in the field of image filtering. However, classical HQS is hard to use in the field of network pruning directly due to the obscurity of hyperparameters and the differences in objective function of regularisation in different application areas.

In this paper, an HQS-based iteration algorithm is proposed to get the approximate solution of \( l_0 \) regularisation problem in the field of network pruning. A parameter selection method is also presented, which transforms ambiguous hyperparameters in the classical HQS algorithm into parameters that can be determined by directly observing the parameter distribution of the pre-trained model. Meanwhile, a strategy of iterative regularisation and one-shot trimming is adopted to further reduce the workload of parameters tuning: First, \( l_0 \) regularisation is applied to the pre-trained model to make the saliency distribution of neurons as sparse as possible and increase the number of zero-saliency neurons. Then, all the neurons are globally trimmed according to their saliency only once. Finally the pruned model is retrained to compensate for loss of accuracy.

Our major contributions are as follows:

i. An iterative HQS based algorithm is firstly proposed to find the approximate solution of \( l_0 \) regularisation in the field of network pruning, which can widen the saliency gap between important and unimportant neurons and provide clear pruning threshold for each layer.

ii. A general hyperparameters selection method is presented. Some obscure parameters in HQS can be transformed into the new parameters set that can be obtained according to the observation of the pre-trained model histogram. We study the influence of each parameter in the new parameter set on the regularisation effect and give a general range for each parameter.

ii. The strategy of global regularisation and one-shot trimming does not require sensitivity analysis and the multiple manual selections of layer-wise pruning ratios. It makes the proposed method can be directly applied to various network models and datasets without many hyperparameter attempts.

The rest of the paper is organised as follows. Related works are described in Section 2. In Section 3, the proposed work method is described, including \( l_0 \) regularisation, the calculation of hyperparameters and pruning strategy. Experiments on MNIST, Fashion-MNIST, and CIFAR100 dataset are introduced in Section 4. The conclusion is drawn in Section 5.

## 2 | RELATED WORKS

### 2.1 | \( l_1 \) or \( l_2 \) regularisation based methods

Wen et al. [23] proposed a structured sparsity learning (SSL) method to regularise the substructures in network architecture. In their work, the parameters belonging to the same structure were regularised by group lasso regularisation. Liu et al. [24] introduced a scaling factor for each output channel, which was jointly trained with the network weights and constrained by \( l_1 \) regularisation. In the trimming step, all the channels with the factor smaller than a global threshold were removed. As we discussed earlier, these types of \( l_1 \) or \( l_2 \) regularisation based method are hard to offer a clear pruning threshold, which makes the selection of layer-wise pruning ratios labor-intensive and often needs iterative pruning. Luo et al. [15] proposed a filter-level pruning method called Thinet, where the filters were pruned layer-by-layer according to its saliency without sparse regularisation. He et al. [20] proposed a saliency metric called geometric median (GM) and achieved state-of-the-art results, where the saliency metric was the distance between the filter and the geometric centre of the filters set. For this type of method, the layer-wise pruning ratios are often difficult to be determined due to the gaussian distribution of saliency metric. To achieve the ideal compression ratio on large models, a lot of attempts are usually inevitable.
2.2 | L₀ regularisation based methods

Louizos et al. [30] proposed a method based on parameter prior distribution to use L₀ regularisation in network training, which can be regarded as a global pruning method with layer-wise parameter. This method can achieve network compression during network training, but its compression rate is not as good as the method based on the pre-trained model. It is already difficult to select hyperparameters from the pre-trained model, from scratch can only be more difficult. Lin et al. [31] proposed structured sparsity regularisation (SSR) that integrated the L₂, L₁, L₀ regularisation and achieved state-of-the-art performance in many models. In their work, L₀ sparse regularisation is firstly solved by alternating direction method of multipliers (ADMM), but most of the hyperparameters in their method are model-related. Moreover, the method of SSR is a layer-by-layer threshold-based pruning method. Once the filter is removed, it can not be restored. Thus, the layer-wise threshold should be selected carefully and the sensitive analysis before pruning is necessary. Meanwhile, the threshold in each layer is not controlled by an individual hyperparameter but two (α and β), which leads to the coupling of the pruning threshold and the coefficient of the regularisation term. Thus, to obtain a good result, it is necessary to select a set of suitable α, whose element number is equal to the number of layers. It should be pointed out both the selection of hyperparameters and the layer-by-layer pruning strategy are time-consuming.

3 | THE PROPOSED APPROACH

3.1 | L₀ sparse regularisation

The simplified flowchart of our regularisation algorithm is depicted in Figure 1. Conventionally, applying regularisation to a pre-trained model can be described as minimising the following objective function:

\[ L = \min_{W'} \left\{ \mathcal{L}(W') + \lambda \sum_{i=1}^{g} p(W_i) \right\}, \]  

(1)

where W' denotes all parameters of the model. \( \mathcal{L}(W') \) is the training loss function. \( \lambda \) is regularisation coefficient. \( p(\cdot) \) is the regularisation function. \( i \) and \( g \) are the layer index and the number of layers, respectively. \( W_i \) is the parameter matrix of layer, which is a 2-D parameter matrix. For a convolution layer, the shape of filter is \([c_{out}, c_{in}, h, w] \), so the shape of \( W_i \) is \([c_{out}, c_{in} \times h \times w] \).

The objective function of joint optimisation of saliency-based L₀ regularisation and training loss is:

\[ L = \min_{W'} \left\{ \mathcal{L}(W') + \lambda \sum_{i=1}^{g} \| \Phi(W_i) \|_0 \right\}, \]  

(2)

\[ \Phi(W_i) = \{ \phi_{ij}^{W} | j \in [1, s_i] \},\]

(3)

\[ \phi_{ij}^{W} = \varphi(w_{ij}), \]

\[ W' = \{ W_i | i \in [1, g] \}. \]  

(4)

where \( j \) is the filter index. \( \Phi(W_i) \) is the saliency vector, \( s_i \) is its element number and related to regular granularity. \( \varphi(\cdot) \) is the saliency function.

To simplify mathematical derivations, let filter saliency is only related to its parameters (e.g. L₁ norm), so we have the bottom equation in (3). It is worth mentioning that the saliency of filter \( \phi_{ij} \) might be not only related to its parameters \( w_{ij} \), and its definition is still an open question.
Following the HQS principle [29] to solve this non-derivable problem, the objective function (2) is rewritten as follows:

\[
L = \min_{W,T} \left\{ \mathcal{L}(W) + \frac{1}{s} \sum_{i=1}^{s} \left( \alpha \| \Phi(W_i) - \Phi(T_i) \|_2^2 + \beta \| \Phi(T_i) \|_1 \right) \right\},
\]

where \( W \) and \( T \) are two variables that need to be solved. \( \alpha \) and \( \beta \) are both regularisation coefficients with positive values. The former is related to the balance between training loss and the similarity of \( (W_i, T_i) \). The latter denotes the regularisation intensity of \( l_1 \) norm. \( T_i \) is an auxiliary variable that is directly constrained by \( l_0 \) norm and has the same shape of \( W_i \).

So far, (5) can be decomposed into optimising two parameters: \( W \) and \( T \). Firstly, we get \( \Phi(T_i) \) by using \( l_0 \) norm to constrain \( \Phi(W_i) \), which is non-derivable. And then, minimising \( \| \Phi(W_i) - \Phi(T_i) \|_2^2 \) to make every element in \( \Phi(W_i) \) as similar as possible to its counterpart in \( \Phi(T_i) \), which can be transferred to minimise \( \| W_i - T_i \|_F^2 \) or \( \| W_i - T_i \|_1^2 \). Both regularisation term are convex and derivable (only if the value is zero, the latter is not), so it can be optimised by many back-propagation-based methods, such as gradient descent. However, the auxiliary variables \( T \) limits the change of all parameters, which might be harmful to the compensation of performance loss caused by regularisation by optimising other filters. We define a variant of \( T \) called version 2 to verify this intuition at the end of Sec 3.1.1, which is shown in Figure 2.

### 3.1.1 Subproblem1: Optimising T

In objective function (5), \( T \) is only related to the second term. In (3), we have assumed that the filter's saliency is only related to its parameters, so the \( \phi_{ij}^T \) with different \( ij \) is irrelevant to each other and can be solved individually. Combining (5) and (3), the objective function of subproblem1 is:

\[
L_2 = \min_{\phi_{ij}^T} \sum_{i=1}^{\ell} \left( \alpha \| \Phi(W_i) - \Phi(T_i) \|_2^2 + \beta \| \Phi(T_i) \|_1 \right) + \min_{\phi_{ij}^T} \sum_{i=1}^{\ell} \sum_{j=1}^{g} \left[ \alpha (\phi_{ij}^W - \phi_{ij}^T)^2 + \beta h (\phi_{ij}^T) \right],
\]

where \( L_2 \) is the second term of (5). \( \phi_{ij}^W \) and \( \phi_{ij}^T \) are the \( j \)-th elements of \( \Phi(W_i) \) and \( \Phi(T_i) \), respectively.

The closed-form solution of (8) is:

\[
\phi_{ij}^T = \begin{cases} 0, & \phi_{ij}^W \leq \sqrt{\frac{\beta}{\alpha}} \\ \phi_{ij}^W, & \text{else} \end{cases},
\]

Finally, combining (12) and (10):

\[
t_{ij} = \begin{cases} 0, & t_{ij} = 0 \\ w_{ij}, & \text{else} \end{cases}.
\]

**Weight-wise regularisation**

For weight-wise pruning, the \( \phi_{ij} \) is the saliency of the \( j \)-th weight in layer \( i \), and \( w_{ij} \) is the value of the \( j \)-th weight in \( W_i \). A common choice of \( \phi_{ij} \) is the magnitude of weight value, so \( \phi_{ij} = |w_{ij}| \).

**Filter-wise regularisation**

For filter-wise and channel-wise pruning, \( \phi_{ij} \) is the saliency of the \( j \)-th filter in layer \( i \). \( w_{ij} \) denotes the parameters set that includes all the parameters in the corresponding filter, which is the \( j \)-th row vector of matrix \( W_i \). If the saliency metric is \( l_2 \) norm, \( \phi_{ij} = \| w_{ij} \|_2 \).

**Batch normalisation regularisation**

Batch normalisation is widely used in many network architectures, where every output channel \( x \) is normalised and then
mapped to $y$ by function $y = ax + b$. The scale parameter $a$ can be used to minimise the channel outputs. Inspired by previous work [24], when the convolution layer $i$ is followed by a batch norm layer, let the filter saliency $\phi_{ij}$ in (13) equal to the corresponding scale parameter $a_{ij}$.

Sparse normalisation

In general, the saliency distribution range in each layer is vastly different, which means that applying a global $r$ in (10) might cause different constrain intensity in each layer. For example, if a layer’s distribution range is smaller than $r$ in all iterations, the corresponding $T$ is always zero according to (13).

To precisely controlling the regularisation intensity in each layer as wish, normalising the saliency histogram of different layers to a similar distribution range is needed. To this end, (13) is replaced to (14).

$$\langle \phi_{ij}^{(a)} \rangle' = \frac{\phi_{ij}^{(a)}}{\sqrt{D(\Phi(W_j^{(0)}))}}$$

where the variables with superscript $(a)$ are iteration-varying, $n$ is the current iteration number. $W_j^{(0)}$ is the parameter matrix of the pre-trained model, $D$ is the variance function.

In actual situations, the saliency vector in every layer is normalised using (14) once and then multiply by a layer-wise coefficient to scale to the same range. Using the following method to let most of the saliency values in the pre-trained model be small than 1 might be more directly:

$$\langle \phi_{ij}^{(a)} \rangle' = \frac{\phi_{ij}^{(a)}}{95^a(\Phi(W_j^{(0)}))}.$$  \hspace{1cm} (15)

where $95^a(\Phi(W_j^{(0)}))$ is the 95th percentile of $\Phi(W_j^{(0)})$.

This method discards the average value of the saliency vector that might be related to the layer’s saliency, but the layer’s saliency can be obtained after the first sparse regularisation. After performing the same regularisation on each layer, the more sparse the layers, the less important.

3.1.2 Subproblem2: Optimising $W$

Solving $W^*$ is equivalent to solving the following equation, which is formed by removing irrelevant variables in (5).

$$L = \min \left\{ L(W^*) + \frac{1}{\beta} \sum_{j=1}^{n} c_j \left( \alpha \|W_j^* - T_j\|_F^2 \right) \right\}.$$  \hspace{1cm} (16)

where we add a switch $c_j$ to control the regularisation strength of each layer. Generally, $c_j = 1$. There are various algorithms to solve this kind of minimisation problem and Frobenius norm loss is convex, so it has no great influence on the original model convergence. In our work, we use gradient descent to optimise this function.

According to (13) and Figure 2, when pushing the parameters in the range of regularisation radius to zero, the change of elements that out of the range is limited at the same time. On the one hand, it can protect the performance and saliency gap by preventing big value changing. On the other hand, this is not in line with the intuition that pruning would make the change of all neurons. This intuition can be tested by rewriting the (16) as follows(called version.2):

$$L = \min _{W^*} \left\{ L(W^*) + \frac{1}{\beta} \sum_{j=1}^{n} c_j \left( \alpha \|W_j^* - T_j\|_F^2 \right) \right\}.$$  \hspace{1cm} (17)

where $W_j^*$ is a subset of $W_j$, which consists of the elements that the counterpart values in $T_j$ are zero. In other words, only the elements in the range of regularisation radius are regularised to zero.

Once the output channel $j$ can be removed in the pruning step, all filters related to channel $j$ in the next layer can also be removed. To avoid the weak channel output being amplified by the filters in the next layer, we regularise the filters with the low $\ell_2$ norm and the corresponding input filters in the next layer.

3.1.3 Parameters updating and terminating condition

The coefficient $\alpha$ highly impacts the similarity between $W^*$ and $T_j$. Once we get $W^*$ from (16), $\alpha$ will be update by (18), and then resolve $\sum W^*$ in order and so on.

$$\alpha^{(n+1)} = \sigma \alpha^{(n)}, \sigma > 1.$$  \hspace{1cm} (18)

The proposed sparse regularisation algorithm is summarised in Algorithm 1, where the superscript $n \in [0, N]$ represents the number of iterations, the variable with superscript 0 is its initial value, and with $N$ is its ending value. The algorithm that use (16) in step 5 is called version.1, and (17) is called version.2.

3.2 The calculation of Hyperparameters

There are four hyperparameters in our method, which are $\alpha^{(N)}, \beta, \alpha$ and $\sigma$. To simplify, lets:

$$r^{(a)} = \sqrt{\frac{\beta}{\alpha^{(N)}}}.$$  \hspace{1cm} (19)
where \( r^{(0)} \) is the regularisation radius in iteration \( n \), and is decreases with the increasing of iteration \( n \), because:

\[
\frac{r^{(n+1)}}{r^{(0)}} = \frac{2\sqrt{\beta/\sigma\alpha^{(N)}}}{2\sqrt{\beta/\alpha^{(0)}}} = \frac{1}{\sqrt{\sigma}}, \sigma > 1.
\] (20)

According to Algorithm 1, auxiliary variable \( T^{(n+1)} \) is calculated by (14). From the mathematical point of view, when \( r^{(0)} \to 0, \| T^{(N)} \|_0 = \| W^{(N)} \|_0 \).

Meanwhile, because \( \alpha^{(N)} \) increases with \( n \), the constraint ability of \( T^{(N)} \) to \( W^{(N)} \) is gradually enhanced, according to (16), which makes more small parameters in \( W^{(N)} \) closer to 0.

Let \( r^{(0)} \) and \( r^{(N)} \) denote initialise and stopping radius, respectively. These two processes coordinate with each other and should follow the following principles:

(i) In the beginning, a large \( r^{(0)} \) can provide a wide parameter search range to find more potential, and a small \( \alpha^{(0)} \) can balance larger regularisation losses to protect model performance.

(ii) At the end of the iteration, small \( r^{(0)} \) can provide more precise norm constraints, and higher \( \alpha^{(N)} \) is helpful to increase the number of zero elements in \( W^{(N)} \).

(iii) In order to protect the model performance, the ratio of regular loss to training error (the first and second term in (2), respectively) in the final iteration should be strictly limited.

According to principles 3, let \( k \) be the ratio of regularisation loss to training loss, according to (16):

\[
\frac{1}{\sigma} \alpha^{(N)} \| W^{(N)} - T^{(N)} \|_F^2 = k \cdot \mathcal{L}(W^{(N)}). \tag{21}
\]

where \( k \) is the ratio of regularisation loss to training loss.

Assuming that model performance and training error are almost equal before and after pruning, the \( W^{(N)} \) in the right hand of (21) can be replaced by \( W^{(0)} \). According to (16) and (21), to protect the model performance, the \( \alpha^{(N)} \) should be smaller, so the overestimation of \( \| W^{(N)} - T^{(N)} \|_F^2 \) is necessary. The similarity of \( W \) and \( T \) is increasing with the iteration number, so \( \| W^{(0)} - T^{(0)} \|_F^2 > \| W^{(N)} - T^{(N)} \|_F^2 \). Thus, \( \| W^{(N)} - T^{(N)} \|_F^2 \) in the left hand of (21) is replaced by \( \| W^{(0)} - T^{(0)} \|_F^2 \). The (21) can be rewritten as follows:

\[
\frac{1}{\sigma} \alpha^{(N)} \| W^{(0)} - T^{(0)} \|_F^2 = k \cdot \mathcal{L}(W^{(0)}), \tag{22}
\]

\[
\alpha^{(N)} = \frac{k \cdot \mathcal{L}(W^{(0)})}{\| W^{(0)} - T^{(0)} \|_F^2}. \tag{23}
\]

\( T^{(0)} \) can be solved by combining (13)(19):

\[
t^{(0)}_{ij} = \begin{cases} 0, & w^{(0)}_{ij} \leq r^{(0)} \\ \sqrt{\frac{\beta}{\alpha^{(0)}}}, & \text{else} \end{cases}
\] (24)

According to principles 1, a large \( r^{(0)} \) is beneficial to expand the search range and a typical value is to cover the whole range of parameter values. Generally, \( r^{(0)} \) can be easily obtained by observing the histogram of parameters. In further, if the model parameters are normalised, it can even be independent of the model. Once \( r^{(0)} \) is defined, \( \alpha^{(N)} \) and \( \beta \) can be obtained by (23) and (24), respectively.

According to principles 2, \( r^{(0)} \) is a small value, which can also be obtained from parameter histogram. Once \( r^{(0)} \) is defined, \( \alpha^{(N)} \) can be calculated by (24). Assume the maximum number of iterations is \( N \), we have:

\[
\alpha^{(N)} = \sigma^N \alpha^{(0)}, N = \log_{\alpha^{(0)}} \frac{\alpha^{(N)}}{\alpha^{(0)}} = \frac{\ln \alpha^{(N)} - \ln \alpha^{(0)}}{\ln \sigma}. \tag{25}
\]

where the superscript \( N \) of \( \sigma \) represents the \( N \)-th power of \( \sigma \), and \( \langle N \rangle \) of \( \alpha \) denotes the maximum number of iterations.

In HQS theory, small \( \sigma \) is good for getting close to the real solution of \( \delta \) regularisation, the typical range is \( (1, 2] \). The hyperparameter selection method is summarised in Algorithm 2.

### 3.3 Pruning

Various saliency metrics and pruning methods can be introduced to our algorithm framework. In our experiment, the
saliency metric is defined as the Frobenius norm of filters, \( l_2 \) norm of neurons, and the scale coefficients in batch normalisation layer.

As is shown in Figure 6, compared with the baseline model and conventional regularisation methods, proposed method possesses a higher saliency difference, which means that a simple pruning theory can also meet good results and can save us from the despaired loop of threshold selection and testing.

4 | EXPERIMENTS

4.1 | Experiments setup

The experiments included two parts: hyperparameters selection and pruning performance comparison. For the former, the fluctuation of pruning ratio with the wide range change of hyperparameters in pruning LeNet models on MNIST dataset are investigated and then the parameters selection principle is summarised. For the latter, the proposed method is compared with some state-of-the-art algorithms on CIFAR100 dataset, such as Liu’s [24], ThiNet [15], SSL [23], SSR [31] and GM [20].

4.2 | LeNet on MNIST

MNIST is a handwriting digital image dataset with 60k training data and 10k test data, which is suitable to implement principle test. In this section, we implement our method on LeNet and study the influence of each parameter on the saliency curve. The baseline model of LeNet-20-50-500 ("20-50-500" is the number of neurons in each layer) is same with the SSR’s (github.com/ShaohuiLin/SSR) and we use Adam to optimise loss function with default parameters (learning rate is 0.001, \( \beta_1 \) is 0.9, \( \beta_2 \) is 0.999, epsilon is \( 10^{-8} \)). In this part, the standard choice of \( k, r^{(0)}, r^{(N)}, \sigma \) are 2, 0.1, 1, 1.5, respectively. Only one parameter is changed at a time, and the others use the standard parameters.

Figure 3a and Figure 3b show the variation of saliency curves with the increase of \( k \) and \( \sigma \), respectively. The best range of \( k \) in here is [0.6, 2], but even though \( k \) is overestimated 10 times (the value is 10), the pruning ratio is reduced by less 10%. In practice, \( k \) is the last parameter to be set, just make the regularisation loss with the same order of magnitude as the network loss in the first iteration. The typical range of \( k \) is [1, 10]. With the decreasing of \( \sigma \), the number of zero elements increases and the saliency gap is also expanding. The smaller the parameter value, the slower the regular radius shrinks, and the more filters small than radius in each iteration. As is shown in (25), the number of iterations is highly related to \( \sigma \). The best range of \( \sigma \) is [1.1, 1.7], and 1.5 is a standard choice.

Figure 3c and Figure 3d show the variation of saliency curves with the increase of \( r^{(0)} \) and \( r^{(N)} \), respectively. With the increasing of \( r^{(0)} \) or the decreasing of \( r^{(N)} \), the pruning size is increased and the gap is enlarged, but the number of iterations is also increased. The pruning size is more sensitive to \( r^{(0)} \) than \( r^{(N)} \), so the increment of pruning size that brings by increasing \( r^{(0)} \) is more than \( r^{(N)} \)’s. As is shown in (25), the computing cost of increasing \( r^{(0)} \) is lower than decreasing \( r^{(N)} \) due to the...
TABLE 1  Pruning result of LeNet-20-50-500 on MNIST

| Method      | Error+ | Filters | Param. |
|-------------|--------|---------|--------|
| Baseline    | 0%     | 20-50-500 | 430k   |
| SSR default | 0.29%  | 2-37-210 | 120k   |
| SSR paper   | 0.18%  | 2-8-77   | 11k    |
| Ours        | −0.02% | 7-17-18  | 8.2k   |

1Error rate increment
Filters: The number of filters in the first three convolution layers. Param: The number of weights and biases in the model.
'SSR default' is the result of default setting and the other use the listed factor set.

characteristics of the logarithmic function. But higher $r(0)$ is risky and it is necessary to let some filters out of the regularisation radius. When the value is very large, if all the filters are included in the radius, it will make all outputs decrease at the same time, which can not widen the saliency gap. The typical range of $r(0)$ is $[0.7, 1.3]$, and the $r(N)$ is $[0.1, 0.3]$. In this section, we compare our approach with SSR [31]. Our three error rates reported here are pruned from the same sparse model whose hyperparameters $k, r(0), r(N)$ are $2, 0.9, 0.1, 1.5$, respectively. We use the official code with default setting (the regularisation factor set is $[0.4, 0.15, 0.02]$) to implement SSR, which is 'SSR default'. And we also list the result reported in SSR’s paper, which is 'SSR paper'. As we can see from Table 1, our method obtains a similar performance with the others. Because this experiment mainly focuses on parameter selection, we do not make an in-depth comparison of different methods.

4.3 LeNet on Fashion-MNIST

In this section, we use the same hyperparameters as before to carry out sparse regularisation and study the difference between version 1 and version 2. In our experiments, the error rate of the baseline model is 8.57%.

To study the difference between auxiliary variables, we increase $k$ from 0.2 to 2 and plot their respective significant curves in Figure 4. As is shown in Figure 4, with the same setting, the number of filters with a low $l_2$ norm in version 1 is slightly more than in version 2, which increases the prune size. With the same loss of pruned accuracy (before finetune), version 1 can trim more filters than version 2, and the $l_2$ norm of the important filters in the results of version 1 is slightly higher than that in the version 2. The reason is that, in the first few iterations of version 2, most of the filters are less than regularisation radius, so the $l_2$ norm of the filters out of radius is also decreased, which makes the saliency curve flatter. In version 1, the filter will maintain the original $l_2$ norm, so the curve is steeper. Therefore, version 1 can remove more filters than version 2 when the radius is reduced.

We also compare the proposed method with SSR in Table 2. For SSR, high factor means high cutting threshold. For example, when the factor in layer1 from 0.3 up to 0.4, the number of filters from 18 down to 10. The selection of regularisation factor set is not an easy job. For this pre-train model, if the factor of layer3 greater than 0.3, all the filters in layer3 are removed, but for MNIST dataset, the paper [31] reported that the factor set can be set to $[0.5, 0.5, 0.5]$. Comparing the last one of SSR’s and the last one of Ours, the proposed method can obtain lower accuracy loss in the case of high compression ratio.

4.4 VGG19 on CIFAR-100

In this section, we apply our method in the variant of VGG19 on CIFAR-100 dataset and compare with Liu’s work [24], SSR [31], SSL [23], ThiNet [15] and GM [20]. The variant is made by Liu et al. [32] and can be obtained in here (github.com/Eric-mingjie/rethinking-network-pruning). The main difference between it and the classic VGG19 is that it adds a batch
normalisation layer after each convolutional layer. Therefore, we can sparse the scale coefficients of the batch normalisation layers following the convolution layer or the filter’s parameters itself.

4.4.1 Scale coefficients

Liu et al. [24] applied $l_1$ regularisation to the scale factors in batch normalisation layer, and used a global threshold in the trimming step for simplicity. For a fair comparison (although it is a little unfair to us), in this part, we first use our method to compress the model (we can get the layer-wise ratio from here) and then implement Liu’s method with the same layer-wise pruning ratio. In this part, the hyperparameters $\lambda_1, r^{(0)}, r^{(N)}, \sigma$ are 4, 1, 0.1, 1.4, respectively, which is similar to the setting in MNIST dataset. All the models are finetuned with same setting (learning rate is 0.01 and multiplied by 0.1 every 40 epochs) and the best accuracy is listed in Table 3.

As is shown in Table 3, with the same pruning ratio, our method has a significant advantage in pruned accuracy over the wide saliency gap. Specially, when the ‘Size’ is less than 20.93, Liu’s models have completely lost its classification ability. Although the accuracy is low to 0.97%, it can still recover to an acceptable level after retraining, but there is still a gap with our method. High pruned accuracy means the high confidence of the optimal layer-wise pruning ratio. The inappropriate pruning ratio set will make the model still have a large loss of accuracy after retraining, which will be discussed in the next section. We can also find that our finetune accuracy is better when the model size is smaller than 26.47. In ‘Ours-$L_4$’, some filters in the first few layers are removed, which makes the FLOPs is significantly less than other methods.

Global threshold-based pruning method is also implemented, but its accuracy is not listed in Table 3, because the filters in layer 10 to 14 are all removed when the pruned size is less than 29.11 as is shown in Figure 5b. The number of filters retained by this method at different compression level in each layer is compared with ours in Figure 5. In Figure 5b, we can find that most of the filters in layer 9-14 are removed, but layer 8 and 15 have little change. However, in Figure 5a, as the compression level increases, these layers still retain a certain number of filters. To illustrate this problem, we depict the scale coefficients in layer 10 and 15 in Figure 6a and Figure 6b. As we know, the filters have low scale coefficient in batch normalisation layer can be removed, and we find that the scale coefficients in layer 10 are much lower than that in layer 15. Following global threshold-based pruning method, to obtain a better compression rate, most of the filters in layer 10 to 14 are removed, so the inter-layer feature transmission is highly blocked, which leads serious performance drop. Comparing with keeping the layer with a small number of filters, removing the whole layer might be a better choice, but it is hard to practices. As is shown in Figure 5b, following the global threshold principle, all the filters from layer 10 to 14 can be removed, it is hard to decide the number of layers should be retrained, because removing all of them will cause serious performance problems and the depth of network is important. On the other hand, if we use a layer-wise pruning ratio, as is shown in Figure 6c and Figure 6d, it is difficult to decide pruning ratio when using $l_1$ and $l_2$ regularisation.

We use different regularisation terms to regularise the model, and then record the respective saliency curves in the Figure 6. As is shown in Figure 6, $l_1$ and $l_2$ regularisation are failed to widen the saliency gap in this instance, even the $l_2$ regularisation can hardly change the distribution of parameters, so it is very difficult to find an ideal pruning threshold without many attempts. Comparing with them, proposed method can make more parameters to be zero and widen the saliency gap.

In Figure 6, it is worth noting that $l_1$ regularisation looks better than $l_2$, however, in (16), the distance between $W$ and $T$ is reduced by minimising $\|W - T\|_2^2$, so we also implement a version of $\|W - T\|_1$ (it is ‘Ours-$L_4$’ in Table 3). To study the difference in saliency caused by different distance definitions, we use different distance definitions for sparse regularisation, while keeping other parameters unchanged, and then draw the saliency curve in Figure 7. As is shown in Figure 7, in terms of pruning speed, the $l_1$ version is better than $l_2$. To explain, from the perspective of regularisation, when the parameter is small, the parameter converges to 0 faster under the $l_1$ norm. And Table 3 shows that when the number of pruned parameters is 19.06% of pre-trained, pruning step does not cause any loss of accuracy, but the finetune accuracy is worse than standard Frobenious version. This result is also predictable. The $l_1$ regularisation can set a parameter to zero but $l_2$ regularisation will prevent the parameter from becoming too small. When the filter is set to 0, back propagation will not change its parameters, which is equivalent to hard removal. Meanwhile, the $l_1$ norm converges faster, which is prone to over-pruning.

| Method  | Size  | Sparse | Pruned | FineTuned | FLOPs |
|--------|-------|--------|--------|-----------|-------|
| Baseline | 100%  | –      | 72.96% | 72.96%    | 100%  |
| Liu et al. | 33.20% | 72.85% | 19.91% | 73.33%    | 72.51% |
| Liu et al. | 26.47% | 72.85% | 4.42%  | 72.91%    | 67.95% |
| Liu et al. | 20.93% | 72.85% | 0.97%  | 72.57%    | 64.16% |
| Liu et al. | 17.78% | 72.85% | 0.97%  | 72.14%    | 61.79% |
| Ours-1   | 33.20% | 72.65% | 70.53% | 72.94%    | 72.51% |
| Ours-1   | 26.47% | 72.65% | 68.59% | 72.92%    | 67.95% |
| Ours-3   | 20.93% | 69.03% | 62.97% | 73.16%    | 64.16% |
| Ours-3   | 17.78% | 69.03% | 41.08% | 72.52%    | 61.79% |
| Ours-$L_4$ | 19.06% | 72.00% | 72.07% | 72.41%    | 41.27% |

Size: The percentage of model parameters retained after pruning.
Sparse: The accuracy after model sparsification.
Pruned: The accuracy after pruning.
FineTuned: The accuracy after fine-tune.
Ours-$L_4$: A special version, where the distance between $W$ and $T$ is reduced by minimising $\|W - T\|_1$.

*Ours-$L_4$:* A special version, where the distance between $W$ and $T$ is reduced by minimising $\|W - T\|_1$. The number behind ‘Ours’ is the number of repetitions of sparse regularisation.
**Figure 5** Number of retained filters in each layer at different compression levels. (a): Ours; (b): Liu’s

**Figure 6** Saliency curves in layers 10 and 15 obtained by using different regularisation methods. (a) and (c): The scale coefficient in layer 10 and its normalised result; (b) and (d): The scale coefficient in layer 15 and its normalised result. L1 regularisation is used in Liu’s method.

**Figure 7** Scale coefficients in layers 10 and 15. (a): Layer 10; (b): Layer 15. ‘L1’ denotes that the difference between W and T is minimised by minimising $\|W - T\|_1$, ‘L2’ is $\|W - T\|_2^2$. The values behind ‘L1’ and ‘L2’ are the numbers of regularisation iterations.
In contrast, the $l_2$ regularisation can keep the parameter value at a low value for a long time so that the filter deleted by mistake can be restored.

### 4.4.2 Filter’s parameters

Table 4 shows the result of SSL, ThiNet, GM, SSR and ours. The layer-wise pruning ratio used by 'SSL', 'ThiNet' and 'GM' are the same as we used in the previous section, except for the results marked with "*".

In SSL the regularisation is applied to the $l_1$ norm of filter and its input channels. So we re-regularise the pre-trained model and the accuracy of sparse model is 72.69. The characteristics and defects of SSL are similar to Liu’s method. Comparing with the Liu’s result in Figure 3, under the same compression level, the results of SSL are slightly better than Liu’s. It is unclear whether this is caused by differences in regularisation or differences in saliency metric. 'SSL*' is the result of global threshold-based pruning method. In a similar model size, the FLOPs of SSL* are lower than ours. (In our method, when the size is 26.47%, FLOPs are 67.95%. In SSL*, the size is 27.66% and FLOPs are 65.05%). But its accuracy, model size and FLOPs are significantly worse than others result, because many filters in layer 4-8 are removed by mistake.

ThiNet and 'GM' are regularisation-free methods, so we directly apply them to the pre-trained model. The layer-wise pruning ratio in these two methods needs to be set before pruning. These two methods are both pruning according to their respective saliency metrics, but ThiNet updates the value of retained parameters to minimise the reconstruction error in further. So the Pruned accuracy of ThiNet are higher than of GM is predictable. 'GM*' simply pruned all the weighted layers with the same pruning ratio at the same time in the filter pruning step. However, the filters in layers 4 to 8 in this model are critically important and have less room to be pruned, so the result is expected. SSR removes more filters in the first few layers but keeps more in the later. So it removes more FLOPs and less number of parameters. We also apply our method to the $l_2$ norm of filters. In order to obtain a more sparse model, we regularise multiple times in the regularisation stage and obtain the sparse models with 72.08% and 72.13% accuracy. In the pruning stage, all filters with $l_2$ norm less than 0.01 are removed. As is shown in the penultimate line of Table 4, the pruned model can be reduced to 16.8% of the sparse model nearly without accuracy loss. With the similar finetuned accuracy, ours model size and FLOPs are significant less than the other methods.

In Figure 8, we draw the saliency curves of the last layer under different saliency measures for the pre-trained, SSL and ours sparse model, respectively. As is shown in Figure 8a, the saliency distribution in the pre-trained model is often average regardless of saliency metric. The conventional $l_1$ and $l_2$ norm are hard to widen the saliency gap, which is not a big problem when the number of zero filters is large. If not, the smooth saliency curve can not provide a clear pruning threshold, just like Figure 6c and Figure 8b. Figure 8c shows that our method can significantly increase the saliency gap of 'L1', 'L2' and geometric median (GM). The BN curve has a jump before 500, which is the same as other metric.

In Figure 9, we draw the mean of the absolute value of the filter parameters under different regularisation methods before retraining. SSL applies $l_1$ regularisation to all the parameters, so the mean value is obvious less than the pre-trained model. As is shown in Figure 9b, when we remove all filters with $l_2$ norm less than 0.2, the model size is 33.73% but accuracy is 57.98%, which is worse than using ours pruning ratio (size is 33.20% and accuracy is 72.45). Our method has both lower low values and higher high values in layer 8-15, so the value gap is significant. Thus, even all filters with l2 norm less than 0.01 are removed.

### 5 CONCLUSION

A novel CNNs pruning method via $l_0$ regularisation is proposed, in which an HQS based iterative algorithm is proposed to calculate the approximation solution of $l_0$ regularisation for the model parameters. Thus, it can be jointly optimised with training loss by gradient-based methods. The experiment results showed that compared with $l_1$ and $l_2$ regularisation, the proposed method could increase the numerical difference between the important and unimportant neurons, which means the proposed method can provide a very clear pruning threshold and is beneficial to the threshold selection of iterative pruning. For the testing datasets, there was only 0.01% accuracy loss after removing more than 80% of the model parameters, which indicates that although the strategy of multi-regularisation and one-shot pruning is used, the model accuracy and compression ratio can still be competitive.
**FIGURE 8**  Saliency curves of the last layer under different saliency metrics and regularisation methods. (a): Pre-trained; (b) SSL; (c) Ours. BN denotes the scale coefficients in batch normalisation layer. L1 and L2 are the norm of filter. GM and ThiNet represent the saliency metric used by the corresponding method.

**FIGURE 9**  Mean distribution of filter parameters before retraining. (a): Pre-trained; (b)SSL; (c)Ours. The picture depicts the mean distribution of each layer of filters. The larger the circle, the denser the distribution. In (b), we remove all filters with $l_2$ norm less than 0.2 and use the broken line to represent the pruning threshold of each layer. ‘Acc’ in label denotes the accuracy before retraining. In (c), $l_2$ norm pruning threshold is 0.01.

**REFERENCES**

1. Krizhevsky, A., et al.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105. MIT Press, Cambridge (2012)
2. Zhang, K., et al.: Learning deep cnn denoiser prior for image restoration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3929–3938. IEEE, Piscataway (2017)
3. Xie, J., et al.: Image denoising and inpainting with deep neural networks. In: Advances in Neural Information Processing Systems, pp. 341–349. MIT Press, Cambridge (2012)
4. Gatys, L., et al.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2414–2423. IEEE, Piscataway (2016)
5. Gong, Y., et al.: Compressing deep convolutional networks using vector quantization, arXiv preprint arXiv:14126115 (2014)
6. Chen, W., et al.: Compressing neural networks with the hashing trick. In: Proceedings of the 32Nd International Conference on International Conference on Machine Learning, vol. 37, pp. 2285–2294. Springer, Cham (2015). http://dl.acm.org/citation.cfm?id=3045118.3045361
7. Denton, E., et al.: Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in Neural Information Processing Systems, pp. 1269–1277. MIT Press, Cambridge (2014)
8. Rigamonti, R., et al.: Learning separable filters. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2754–2761. IEEE, Piscataway (2013)
9. Hinton, G., et al.: Distilling the knowledge in a neural network, arXiv preprint arXiv:150302531 (2015)
10. LeCun, Y., et al.: Optimal brain damage. In: Advances in Neural Information Processing Systems, pp. 598–605. MIT Press, Cambridge (1990)
11. Hassibi, B., Stork, D.: Second order derivatives for network pruning: Optimal brain surgeon. In: Hanson, S., Cowan, J., Giles, C., (eds) Advances in Neural Information Processing Systems, vol. 5, pp. 164–171. Morgan-Kaufmann, San Mateo (1993). http://papers.nips.cc/paper/647-second-order-derivatives-for-network-pruning-optimal-brain-surgeon.pdf
12. Srinivas, S., Babu, R.: Data-free parameter pruning for deep neural networks, arXiv preprint, arXiv:150706149 (2015)
13. Li, H., et al.: Pruning filters for efficient convnets, arXiv preprint, arXiv:160808710 (2016)
14. Hu, H., et al.: Network trimming: A data-driven neuron pruning approach towards efficient deep architectures, arXiv preprint, arXiv:160703250 (2016)
15. Luo, J., et al.: Thinet: A filter level pruning method for deep neural network compression. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5058–5066. IEEE, Piscataway (2017)
16. Ding, X., et al.: Approximated oracle filter pruning for destructive cnn width optimization. arXiv preprint, arXiv:190504748, 2019
17. Molchanov, P., et al.: Pruning convolutional neural networks for resource efficient inference, arXiv preprint, arXiv:161106440 (2016)
18. Molchanov, P., et al.: Importance estimation for network pruning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 11264–11272. IEEE, Piscataway (2019)
19. Dong, X., et al.: Learning to prune deep neural networks via layer-wise optimal brain surgeon. In: Advances in Neural Information Processing Systems, pp. 4857–4867. Curran Associates, New York (2017)
20. He, Y., et al.: Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4340–4349. IEEE, Piscataway (2019)
The objective function is:

\[ L = \min_{W'} \left\{ \mathcal{L}(W') + \lambda \sum_i \| \Phi(W'_i) \|_0 \right\}. \tag{A.7} \]

Let the exact solution of (A.7) is \( \bar{T} \). To ensure that the parameter \( W' \) can converge to the exact solution \( \bar{T} \) in subproblem2, the \( T \) should be equal to \( \bar{T} \). Under this ideal conditions, the subproblem2 is:

\[ L = \min_{W'} \left\{ \mathcal{L}(W') + \sum_i \left( \| W'_i - \bar{T}_i \|_F^2 \right) \right\}. \tag{A.8} \]

Dividing the second item into two parts:

\[ \sum_i \left( \| W'_i - \bar{T}_i \|_F^2 \right) = \sum_i \left( \| W'_{i \neq 0} - \bar{T}_{i \neq 0} \|_F^2 + \| W'_{i = 0} - \bar{T}_{i = 0} \|_F^2 \right). \tag{A.9} \]

where the parameters with subscript \(( \neq 0)\) are retrained and \((= 0)\) are removed, so \( \bar{T}_{i = 0} = 0 \).

Because the \( T \) we get is inaccurate, so the \( \bar{T}_{i \neq 0} \) and \( T_{i = 0} \) are both possible inaccurate. Considering the following function:

\[ L = \min_{W'_{(\neq 0)}} \left\{ \mathcal{L}(W'_{(\neq 0)}) + \| W'_{(\neq 0)} - \bar{T}_{(\neq 0)} \|_F^2 \right\}. \tag{A.10} \]

where the real solution of \( W'_{(\neq 0)} \) is \( \bar{T}_{(\neq 0)} \). In this equation, if the second regularisation term is removed, it can also reach the best result by minimising training loss. Let the \( W'_{(\neq 0)} \) free to find better value without any regularisation, it is possible to reach \( \bar{T}_{(\neq 0)} \). This is the reason why we proposed auxiliary variable version2.
More importantly, the previous analysis is based on the premise that $T_{\neq 0}$ and $T_{\neq 0}$ are the same set of parameters. In other words, the pruning ratio needs as accurate as possible. In proposed algorithm, we use a wide regularisation radius in the beginning to find more potential and gradually shrink to a narrow range to improve pruning accuracy in the end, which can provide a clear pruning threshold in each layer and finally improve the accurate of pruning ratio.