Research and optimization pruning method in train process: DPIREC

D.A. Novikov, D.Y. Buryak

The faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University

This work focuses on the implementation and investigation of a novel pruning method for deep convolutional neural networks, DPIREC (Dynamic Pruning by Importance of Random Excluded Channels), which enables pruning directly during training. The core idea of this method is to apply random masks to the channels of convolutional layers during each training iteration, temporarily excluding certain channels from the training process. Subsequently, the importance values of the channels used in the current step are updated. Repeating these actions over multiple training epochs allows the identification of channels with the least impact on the loss function, which can then be permanently excluded from the pruned neural network. We conducted a comprehensive analysis of approaches to assessing parameter relevance. Based on this, a new approach was proposed that takes into account the dynamics of changes in model accuracy. Comparative experiments demonstrated the superiority of the DPIREC method over several existing techniques. When pruning the ResNet18 convolutional neural network during training on the CIFAR-100 dataset by 40%, the accuracy loss was 0.89%.

Keywords: convolutional neural networks, dynamic pruning, structured pruning.

1. Introduction

Modern advancements in deep learning are driven not only by the development of numerous novel neural network architectures and the engagement of a large number of specialists but also by continuously increasing computational power. Deep neural networks can encompass dozens or even hundreds of hidden layers, with each layer containing up to hundreds of neurons, resulting in an immense number of parameters [1]. Training such networks has become feasible due to the widespread availability of powerful parallel computing devices, such as GPUs (Graphics Processing Units) and TPUs (Tensor Processing Unit) [2]. However, many users lack access to high-performance computational resources equipped with thousands of CUDA cores.

Nevertheless, it is well-established that many parameters within a neural network are redundant and exert minimal influence on the final outcome. This insight has inspired the concept of reducing model weights and operations with minimal quality degradation. An alternative approach involves leveraging only those weights and operations that contribute most significantly to the network's performance. Furthermore, a body of research has demonstrated that removing a small fraction of non-essential parameters can lead to improvements in test-set performance. The process of removing redundant parameters aids in discarding noisy weights and disrupts patterns that the network may have learned from training data but which do not generalize across the entire data distribution [3, 4].

2. Related works

Pruning of neural networks is an optimization technique aimed at identifying and removing parameters with minimal impact on the network's accuracy [5]. This process involves employing various criteria to assess parameter significance, enabling precise determination of which parameters can be excluded without noticeable degradation in the network's performance. The primary body of research on pruning is predominantly focused on convolutional neural networks

(CNNs), widely used for image classification tasks, which form the foundation for a broad range of computer vision applications.

Within this domain, several key strategies have been developed for pruning algorithms, alongside criteria that assist in identifying parameters whose removal exerts minimal impact on the overall performance of the model.

2.1. Iterative pruning

Iterative pruning methods are typically applied to pre-trained neural networks. In this approach, after selecting an appropriate criterion, a series of parameters is progressively removed from the model. The updated network is then fine-tuned over several epochs, ensuring its further refinement. This cycle is repeated until the performance of the pruned neural network declines below a predefined threshold relative to the original model, or until the desired level of parameter reduction is achieved.

2.1.1. Magnitude-based pruning

One of the most effective and widely adopted approaches to pruning is magnitude-based pruning [6]. The central idea of this criterion is the exclusion of weights from the neural network whose absolute values fall below a predefined positive threshold. For instance, convolution operations can be represented as:

$$y_i = \sum_{j=0}^{n} w_{ij} x_j, \quad i = \overline{0, m}\ ,$$

indicating that weights with the smallest absolute values contribute the least to the resulting sum. Consequently, their removal is expected to have a minimal impact on the network's overall performance. Due to its simplicity, magnitude-based pruning is frequently employed in practice; however, its effectiveness diminishes when the neural network is heavily pruned.

2.1.2. Pruning based on the L_1 norm

This pruning method extends to the removal of entire filters in the convolutional layers of a neural network, based on their L_1 norm [7,8] being below a specified threshold. The L_1 norm of a tensor $F \in \mathbb{R}^n$ is computed during each iteration of the pruning process as follows:

$$\|F\| = \sum_{k=1}^{n} |w_k|.$$

2.1.3. Pruning based on Euclidean distance

Another variant of this method involves removing parameters or filter tensors in convolutional layers based on their pairwise Euclidean distances [8–10] being less than a defined threshold. The Euclidean distance between any two parameter tensors P, $Q \in \mathbb{R}^n$ is calculated at each iteration of the pruning process as:

$$d(P,Q) = \sqrt{\sum_{k=1}^{n} (w_k^{(p)} - w_k^{(q)})^2}.$$

This computation yields a symmetric distance matrix $D \in \mathbb{R}^{I \times I}$:

$$D = \begin{cases} d(F_i, F_j), i \neq j \\ 0, i = j \end{cases},$$

where I – number of parameter tensors in the model, with F_i denoting the tensor at index i.

Subsequently, based on the computed distance matrix, one tensor from each pair with a distance below the threshold is removed from the model. This iterative process ensures reduction of redundant parameters while preserving the network's core functionality.

2.2. Pruning models during training

There exist algorithms that enable pruning of a neural network immediately after parameter initialization, allowing for the exclusion of redundant parameters directly during the training process. This approach relies on predefined criteria to assess parameter significance, enabling the network structure to be dynamically adapted throughout training [11, 12].

One of the primary advantages of this method is the significant time savings in creating a lightweight version of the neural network. Since the stages of training and removal of insignificant parameters occur simultaneously, the model can achieve an optimal state more rapidly, which is particularly critical in scenarios with limited time or resources.

Additionally, this approach enhances the model's generalization capability, as the removal of non-essential parameters during training can reduce the risk of overfitting. In this way, the model more effectively identifies the most important characteristics in the data, which potentially improves the quality of the final results.

2.2.1. Sparse evolutionary training

The sparse evolutionary training (SET [13]) method is based on evolutionary algorithms. The application of pruning during the network design phase, even before the start of training, significantly reduces the number of parameters in the model. This, in turn, lowers memory requirements and improves computational efficiency. Research has shown that neural networks constructed using sparse (or sparsely connected) layers exhibit excellent performance and quality comparable to traditional fully connected layers. Such layers, trained via the SET method, can replace fully connected ones without a loss in accuracy and can even feature a quadratic reduction in parameter count at the design stage. This approach involves a regular process of removing the smallest positive and largest negative weights from sparse connections. These weights are closest to zero, indicating they have minimal impact on the network's performance. After removing these weights, an equivalent number of new random connections are added to the current layer, maintaining a balanced number of connections during training. The cycle of weight removal and addition is repeated over a set number of training epochs. After training concludes, the layers are fixed at the final step of weight removal without adding new connections.

2.2.2. Trainability preserving pruning

Numerous studies emphasize the importance of preserving the trainability of neural networks, which can be disrupted by the pruning process. The trainability preserving pruning (TPP [14]) method focuses on minimizing correlations between convolutional layer filters to maintain their trainability post-removal. To achieve this, a regularization term is introduced into the loss function, penalizing filter non-orthogonality.

The authors of this method propose a more flexible constraint that does not demand perfect preservation of trainability but ensures that gradients can effectively propagate through the network without obstructions. Unlike stringent requirements for orthogonality, which mandate that singular values of the Jacobian matrix equal exactly one, TPP only requires that these values avoid extreme deviations, enabling stable training.

A critical aspect of this method is the separation of retained filters from removed ones in the target Gram matrix associated with the convolutional layer filters. All elements corresponding to removed filters are zeroed out, preserving the remaining values. Filters are pruned based on their L_1 norms, with the lowest values considered non-essential. Furthermore, it is crucial to account for batch normalization layers, as their states are also subject to training. Removing filters can alter the internal distribution of features, necessitating appropriate modifications to the statistics collected during training. To mitigate this effect, the authors suggest ordering the lists of pruned filters and introducing a penalty term for batch normalization layers, thus providing necessary regularization for these layers.

2.2.3. Adding before pruning

The adding before pruning (ABP [15]) method is an innovative approach that allows the model to focus on significant filters based on training rather than artificial criteria such as norms, ranks, or other metrics.

In this method, an attention level is introduced to improve the evaluation of filter significance. At this level, binary significance scores are assigned, helping the system more clearly determine the importance of each filter during training. Furthermore, to optimize gradient propagation at the attention level, a specialized gradient estimator was developed. This tool has proven effective in ensuring convergence of computational flows through rigorous mathematical validation, enabling more efficient training management.

One of the key features of the ABP method is simultaneous pruning and training of filters, which facilitates the automatic elimination of redundancy in the neural network. This is achieved by reducing reliance on complex prior knowledge typically required to establish threshold criteria. Simplifying this process increases the model's flexibility and adaptability to various tasks.

Experimental results obtained during testing on standard image classification datasets show that the proposed ABP method significantly outperforms previous pruning algorithms in efficiency and accuracy. These achievements make the method particularly promising for modern machine learning and computer vision systems, where both resource efficiency and high performance are critical.

2.2.4. Graph convolutional network pruning

Graph convolutional network-based pruning (GCNP [16]) is designed to improve model compression, adapting networks for resource-constrained environments without compromising classification accuracy. The primary idea involves leveraging graph structures to make more effective decisions about which channels to retain or remove. This process consists of several key stages, from feature extraction at each layer to training a reinforcement learning-based agent.

The first step involves extracting channel features for each layer of the neural network. This information is then used to construct a graph, where the graph nodes represent channels, and edges between nodes are established based on feature similarity. The Kullback–Leibler divergence is used to evaluate this similarity, and edges are formed if the distance between channels falls below a predefined threshold.

Next, a GCN-based agent analyzes the constructed graph and synthesizes information on the importance of various channels. The agent decides which channels to prune, generating a probabilistic distribution of actions. To enhance adaptability and convergence to optimal pruning schemes, the agent is trained using the Policy Gradient method. This method enables the agent to optimize its actions based on the model's post-pruning performance.

The agent's training process is iterative. For each layer, the network generates multiple pruning schemes, selecting those that minimize the target objective as training labels. Parameter updates for the GCN are performed using backpropagation, gradually improving decision quality.

3. Dynamic pruning by importance of random excluded channels

Previously we proposed a novel neural networks pruning method, named DPIREC (dynamic pruning by importance of random excluded channels) [21]. The proposed pruning approach builds on the concept of random subset feature selection (RSFS [20]), where the classification task is constructed as an ensemble of classifiers, each utilizing random subsets of features from the dataset. Each feature is assigned a relevance score determined by the accuracy of the classifiers that use it for prediction. Pruning in this algorithm is performed dynamically during the neural network's training process.

Let F denote the set of neural network parameters (e.g., weights, filters, or channels). Each parameter $f_j \in F$ is associated with a relevance value $r_j \in \mathbb{R}$. Similarly, the set of dummy parameters Z is defined, where each dummy parameter z_j is assigned a relevance value $q_j \in \mathbb{R}$.

During each i-th iteration of the neural network training, the following steps are executed:

1. Randomly select n parameters from F using a uniform distribution to form the subset S_i.
2. Similarly, randomly select m dummy parameters from Z using a uniform distribution to form the subset Y_i.
3. Compute the loss function value L_i for the neural network, where all parameters outside S_i are excluded. Define $l_i = -L_i$.
4. The relevance values r_j for parameters $f_j \in S_i \subset F$ are updated as $r_j = r_j + l_i - E(l)$, where $E(l)$ is the expected value of the loss function (average value across previous iterations).
5. Similarly, the relevance values q_i of the dummy parameters $z_j \in Y_i \subset Z$ are updated: $q_j = q_j + l_i - E(l)$.

The weights of the neural network are updated according to standard training algorithms, involving all trainable parameters. Dummy parameters do not participate in the training process or loss computation. However, their relevance values q_j are accumulated in the same manner as for true parameters. Consequently, the accumulation of relevance values q_j for a dummy parameter z_j becomes a random process unrelated to the actual performance of the neural network.

To determine which parameters f_j significantly influence the quality of neural network training, we establish a baseline level $q_{baseline}$ using the relevance of dummy parameters q_j. True parameters are deemed significant if their relevance exceeds this baseline level, as measured by their contribution to minimizing the neural network's loss function.

Finally, to identify the subset $S \subset F$ of parameters that really surpass the baseline relevance of dummy parameters, a statistical test is performed. Specifically, the relevance r_j of a parameter f_j must satisfy:

$$p(r_j > q_{baseline}) \geq \delta, \quad \forall f_j \in S \subset F, \tag{1}$$

where δ is a fixed probability threshold. The random baseline level $q_{baseline}$ is modeled as a normal distribution of dummy relevance values $\bar{q}_j$. The probability that a parameter is more relevant than a dummy parameter is then derived from the normal distribution:

$$p(r_j > q_{baseline}) = \frac{1}{\sigma\sqrt{2\pi}} \int_0^{r_j} \exp(\frac{-(x-\mu)^2}{2\sigma^2})dx, \tag{2}$$

where μ and σ are the mean and standard deviation of q_j across all dummy parameters.

The method for evaluating relevance represents one of the key elements of the DPIREC approach. Therefore, in this work, we conducted an analysis of the existing method and introduced a new algorithm that accounts for changes in the loss function during the neural network training process. The advantage of the new approach was demonstrated through extensive comparative testing of DPIREC against several existing pruning methods on image recognition tasks.

4. Reflected losses approximation

One of the primary challenges associated with the DPIREC method is its inability to effectively respond to reductions in the values of the loss function during neural network training. To overcome this limitation, we propose an innovative approach based on reflected losses approximation (RLA), in which each parameter f_j from the set F is linked not to a single relevance value but to a dynamic collection of reflected loss function values $r_j = \{r_{jk} \mid -\infty < r_{jk} \leq 0\}$. Similarly, the dummy z_j parameter of the set Z will also correspond to the set of reflected loss function values $q_j = \{q_{jk} \mid -\infty < q_{jk} \leq 0\}$.

During each iteration i of the neural network training process, the following sequence of operations is executed:

1. A subset S_i of n parameters is randomly sampled from the set F using a uniform distribution.
2. Similarly, a subset Y_i of m dummy parameters is randomly selected from the set Z under the same uniform distribution.
3. The value of the loss function L_i is evaluated for the neural network, retaining only the parameters in S_i while discarding all others. Define $l_i = -L_i$.
4. The computed value l_i is added into the set of relevance values r_j corresponding to each parameter $f_j \in S \subset F$.
5. The value l_i is added to the set of relevance values q_j associated with each dummy parameter $z_j \in Y_i \subset Z$.
6. To derive a numerical measure of parameter importance, polynomial fitting is applied to the set of loss values r_j, in this approach first degree polynomials are used. The coefficient corresponding to the highest-degree term of the polynomial is used to determine the relevance $\bar{r}_j$. A similar procedure is applied to compute the relevance $\bar{q}_j$ for dummy parameters.

Following the computation of the relevance metrics $\bar{r}_j$ for true parameters and $\bar{q}_j$ for dummy parameters, a statistical evaluation analogous to the baseline approach is conducted. The decision rule defined by equations (1) and (2) is applied, with $\bar{r}_j$ and $\bar{q}_j$ substituted as the respective relevance measures. This adaptation ensures that the statistical framework remains consistent with the baseline methodology while incorporating the refined relevance metrics $\bar{r}_j$ and $\bar{q}_j$.

4.1. RLA approach advantages demonstration

Consider an example that demonstrates the advantages of the DPIREC approach based on reflected losses approximation (RLA) compared to DPIREC approach based on an iterative change in relevance:

- We define the size of the set $|F| = 3$ and `iterations_count` $= 30$.
- Then, we execute steps 1, 3 and 4 of the algorithm iteratively for `iterations_count` times.
- For the obtained sets r_1, r_2, r_3, polynomial fitting is performed.
- The parameter relevance values f_1, f_2, f_3 will be represented by the coefficients corresponding to the highest-degree term of the polynomial – $\bar{r}_1, \bar{r}_2, \bar{r}_3$.

  As can be seen from the table and graph, parameter r_1 contributes the least to the changes in the loss function and thus has the lowest relevance value, increasing the likelihood of its exclusion from the neural network model. However, the approach based on an iterative change in relevance determines parameter r_3 as the least significant for the model, which does not align with its impact on the loss function. Excluding r_3 would result in greater accuracy losses in the model compared to excluding r_1.

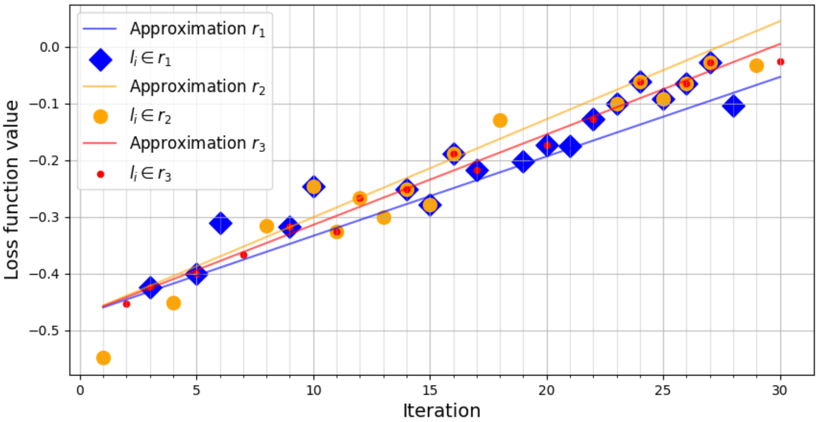

Figure 1. The points on the graph below correspond to the loss function values at each iteration. If a point was included in the set r_j of parameter f_j, it will be highlighted with the respective color associated with the parameter. The lines represent the approximation polynomials. The lines are offset to one point to emphasize the differences in the values $\bar{r}_1, \bar{r}_2, \bar{r}_3$

Table 1. Parameter – the parameter for which the relevance values were calculated, Base – the relevance value obtained using the iterative DPIREC approach, RLA – the relevance value obtained using the DPIREC RLA approach

Parameter	Base	RLA
f_1	0.786	0.014
f_2	0.844	0.017
f_3	0.628	0.016

As illustrated, the DPIREC RLA method more accurately reflects the true contribution of each parameter to the loss function, thereby facilitating more effective pruning decisions. This contrasts with the iterative DPIREC approach, which might misidentify less impactful parameters, leading to suboptimal pruning outcomes.

5. Experiments and results

5.1. Datasets

The CIFAR [17] datasets are widely used benchmarks in computer vision tasks. They are divided into two parts: CIFAR-10 and CIFAR-100, containing 10 and 100 classes, respectively. Each dataset consists of 50,000 training images and 10,000 test images with a resolution of 32×32 pixels. All images are uniformly distributed among the classes, ensuring balanced data.

During model training, data augmentation methods were applied, including random horizontal flipping and padding with cropping. For both CIFAR-10 and CIFAR-100, the padding size was set to 4 pixels. These strategies help reduce the risk of overfitting. This study does not require additional sophisticated data augmentation methods, as the primary focus is on model compression.

5.2. Network architectures

Two widely recognized convolutional neural network architectures, VGG [18] and ResNet [19], were selected for the experiments, as they serve as reliable benchmarks for evaluating model compression techniques, including pruning and other optimization methods.

The following architecture variants were used in the experiments:

- VGG-16: A classical architecture with 14.73 million parameters;
- ResNet-18: A residual block-based model containing 11.17 million parameters;
- ResNet-20: A compact model with 0.27 million parameters;
- ResNet-56: A mid-sized model with 0.86 million parameters.

These architectures were chosen due to their differences in size and complexity, allowing for an in-depth investigation of how parameters pruning affects performance.

To account for the differences in dataset scale, the architectures were adapted to minimize parameter redundancy. This ensures a balance between model performance and computational efficiency. Such adaptations facilitate a more meaningful comparison of models with varying parameter counts and structural complexities, providing deeper insights into the effects of pruning methods, such as sparsity, on classification accuracy.

5.3. Performance assessment

To investigate the effectiveness of the DPIREC RLA approach for pruning convolutional neural networks, it is necessary to conduct an experimental comparison with existing convolutional neural network pruning methods. The final accuracy of the neural network model at an equivalent level of sparsity will be used as the criterion for comparing the pruning methods. Below, we present the results of an experimental comparison of the DPIREC approach with pruning methods based on the L1 norm and Euclidean distance, which were also implemented.

Table 2. Comparison of pruning quality for various neural networks on the CIFAR-10 dataset. Network – convolutional neural network model; Method: (1) – pruning based on the L_1 norm, (2) – pruning based on Euclidean distance, (3) – iterative DPIREC based on an iterative change in relevance, (4) – DPIREC RLA; Baseline Acc. – accuracy of the model on a test sample without pruning; Pruned parameters – number of parameters excluded from the model 20%, 40%, 60% correspondingly; Acc.(%) and ↓(%) – accuracy and decrease of accuracy pruned neural network respectively

Network	Method	Baseline Acc.(%)	Pruned params					
			20%		40%		60%	
			Acc.(%)	↓(%)	Acc.(%)	↓(%)	Acc.(%)	↓(%)
VGG16	(1)	92.3	91.82	−0.48	91.18	−1.12	89.64	−2.66
	(2)		91.87	−0.43	91.26	−1.04	89.84	−2.46
	(3)		92.03	−0.27	91.31	−0.99	90.27	−2.03
	(4)		92.16	−0.14	91.38	−0.92	90.38	−1.92
ResNet18	(1)	93.7	93.53	−0.17	93.28	−0.42	92.43	−1.27
	(2)		93.56	−0.14	93.29	−0.41	92.55	−1.15
	(3)		93.58	−0.12	93.40	−0.31	92.92	−0.78
	(4)		93.61	−0.09	93.43	−0.27	92.96	−0.74
ResNet20	(1)	92.1	91.76	−0.34	91.3	−0.8	89.74	−2.36
	(2)		91.84	−0.26	91.32	−0.78	89.95	−2.15
	(3)		91.92	−0.18	91.44	−0.66	90.24	−1.86
	(4)		91.98	−0.12	91.51	−0.59	90.29	−1.81
ResNet56	(1)	93.2	92.91	−0.29	92.62	−0.58	91.85	−1.35
	(2)		92.99	−0.21	92.65	−0.55	91.95	−1.25
	(3)		93.05	−0.15	92.89	−0.31	92.36	−0.84
	(4)		93.10	−0.10	92.93	−0.27	92.39	−0.81

Table 3. Comparison of pruning quality for various neural networks on the CIFAR-100 dataset

Network	Method	Baseline Acc.(%)	Pruned params					
			20%		40%		60%	
			Acc.(%)	↓(%)	Acc.(%)	↓(%)	Acc.(%)	↓(%)
VGG16	(1)	69.5	68.71	−0.79	66.93	−2.57	64.39	−5.11
	(2)		68.75	−0.75	66.98	−2.52	64.62	−4.88
	(3)		68.90	−0.61	67.63	−1.87	65.66	−3.84
	(4)		68.96	−0.54	67.76	−1.74	65.86	−3.64
ResNet18	(1)	74.32	73.94	−0.38	73.04	−1.28	71.86	−2.46
	(2)		73.99	−0.33	73.11	−1.21	71.94	−2.38
	(3)		74.08	−0.24	73.34	−0.98	72.33	−1.99
	(4)		74.16	−0.16	73.43	−0.89	72.65	−1.67
ResNet20	(1)	68.23	67.75	−0.48	66.64	−1.59	65.12	−3.11
	(2)		67.81	−0.42	66.73	−1.50	65.24	−2.99
	(3)		67.91	−0.32	66.89	−1.34	65.75	−2.48
	(4)		68.01	−0.22	67.29	−0.94	66.12	−2.11
ResNet56	(1)	72.46	72.06	−0.40	71.12	−1.34	69.81	−2.65
	(2)		72.12	−0.34	71.19	−1.27	69.91	−2.55
	(3)		72.19	−0.27	71.57	−0.89	70.34	−2.12
	(4)		72.27	−0.19	71.83	−0.63	70.66	−1.80

The DPIREC method based on reflected losses approximation (RLA) demonstrates superior performance among all reviewed methods.

- Across all architectures (VGG16, ResNet18, ResNet20, ResNet56) and datasets (CIFAR-10, CIFAR-100), DPIREC RLA ((4) in the table) achieves the lowest accuracy decrease at the same parameter pruning levels (20%, 40%, 60%).
- The advantage of DPIREC RLA becomes particularly evident at higher levels of parameter pruning. For instance, in the case of ResNet56 on CIFAR-10, with 60% pruning, the accuracy decrease is only 0.81% for DPIREC RLA, compared to 1.35% for the L1-norm-based pruning method ((1) in the table).

Accuracy decrease reduces with larger ResNet architectures

- Larger models, such as ResNet56, demonstrate less quality degradation under DPIREC RLA compared to more compact architectures like ResNet20.
- For example, with 60% pruning on CIFAR-10, the accuracy decrease for ResNet20 is 1.81%, whereas for ResNet56 it is only 0.81%. This highlights the capability of larger models to retain performance even under significant parameter pruning.

DPIREC RLA effectively considers parameters relevance

- The use of reflected losses approximation (RLA) approah enables a more precise evaluation of parameter contributions to loss reduction, resulting in more balanced and effective pruning.
- This is particularly evident when comparing DPIREC based on an iterative change in relevance and DPIREC RLA – the latter consistently outperforms the former.

5.4. Comparison with other state-of-the-art solutions

This section provides a comparative analysis of the DPIREC RLA method with modern state-of-the-art approaches to neural network pruning during training, such as adding before pruning (ABP [15]) and graph convolutional network-based pruning (GCNP [16]). This comparison is essential for an objective evaluation of the effectiveness of the developed method within the context of the current state of pruning technology.

Table 4. Network – convolutional neural network model; Method – pruning method; Baseline Acc. – accuracy of the model on a test sample; Pruned Acc. – accuracy of the pruned model on a test sample; Acc. *downarrow* – accuracy decrease after pruning; Params ↓ – number of excluded parameters

Dataset	Network	Method	Baseline Acc.(%)	Pruned Acc.(%)	Acc. ↓(%)	Params ↓(%)
CIFAR-10	ResNet20	ABP	92.15	91.03	−1.12	45.1
		DPIREC RLA	92.1	91.36	**−0.64**	45
		GCNP	92.25	91.58	−0.67	38.51
		DPIREC RLA	92.1	91.57	**−0.59**	40
		GCNP	92.25	92.22	−0.33	27.42
		DPIREC RLA	92.1	91.89	**−0.21**	30
	ResNet56	ABP	93.41	93.1	−0.31	45.7
		DPIREC RLA	93.2	92.93	**−0.29**	45
		GCNP	93.72	92.75	−0.97	70.5
		DPIREC RLA	93.2	92.27	**−0.93**	70
CIFAR-100	ResNet20	GCNP	68.38	68.95	**0.57**	8.16
		DPIREC RLA	68.23	68.16	−0.07	10
		GCNP	68.38	66.14	−2.24	49.42
		DPIREC RLA	68.23	66.57	**−1.66**	50
	ResNet56	GCNP	72.86	72.22	−0.64	39.83
		DPIREC RLA	72.46	71.83	**−0.63**	40

The DPIREC RLA method demonstrates competitive performance relative to other state-of-the-art techniques. In most cases, DPIREC RLA achieves lower accuracy degradation (Acc. ↓) at comparable or better parameter pruning levels compared to the ABP and GCNP methods. For instance, with ResNet20 on CIFAR-10 and approximately 45% sparsity, DPIREC RLA reduces accuracy by just 0.64%, outperforming ABP (−1.12%) and GCNP (−0.67%).

The method performs equally well on both smaller models (e.g., ResNet20) and larger architectures (e.g., ResNet56), consistently achieving either minimal accuracy losses or competitive results at similar levels of sparsity.

6. Conclusion

We conducted a comprehensive investigation and enhancement of the previously proposed neural network pruning method DPIREC (Dynamic Pruning by Importance of Random Excluded Channels). The primary focus was on optimizing the new parameter exclusion approach and improving the overall efficiency of the method.

The following key results were achieved during the course of this research:

- A detailed analysis of the existing parameter significance evaluation criterion in the DPIREC method was carried out.

- A novel approach for evaluating parameter relevance based on polynomial approximation was proposed, enabling a more accurate consideration of the dynamics of model accuracy changes.

- Extensive experimental studies were performed on various convolutional neural network architectures (VGG, ResNet) using the CIFAR-10 and CIFAR-100 datasets.

- Results demonstrated the superiority of the improved DPIREC method over several existing pruning techniques. Specifically, when pruning 40% of the ResNet18 architecture on the CIFAR-100 dataset, the drop in accuracy was only 0.89%.

The obtained results confirm the effectiveness of the proposed improvements and suggest that this method can be recommended for optimizing neural network architectures. In future work, we plan to adapt the method for larger-scale architectures and explore its applicability in other areas of deep learning.

References

1. Gusak J., Cherniuk D., Shilova A. et al. Survey on Efficient Training of Large Neural Networks // Thirty-First International Joint Conference on Artificial Intelligence, IJCAI–22, July 23-29, 2022. DOI: 10.24963/ijcai.2022/765.

2. Wang Y., Wei G., Brooks D. Benchmarking TPU, GPU, and CPU Platforms for Deep Learning // John A. Paulson School of Engineering and Applied Sciences, Harvard University, July 24, 2019. DOI: 0.48550/arXiv.1907.10701.

3. Li Z., Chen T., Wang Z. et al. Can pruning improve certified robustness of neural networks? // Transactions on Machine Learning Research. 2022. DOI: 10.48550/arXiv.2206.07311.

4. Harutyunyan H., Reing K., Steeg G.V. et al. Improving Generalization by Controlling Label-Noise Information in Neural Network Weights // Proceedings of the 37th International Conference on Machine Learning, ICML'20, July 13, 2020. Article 381. P. 4071–4081. DOI: 10.5555/3524938.3525319.

5. Cheng H., Zhang M., Shi J.Q. A Survey on Deep Neural Network Pruning-Taxonomy, Comparison, Analysis, and Recommendations // IEEE Trans. Pattern Anal. Mach. Intell. 2024. Vol. 46, no. 12. P. 10558–10578. DOI: 10.1109/TPAMI.2024.3447085.

6. Li H., Kadav A., Durdanovic I. et al. Pruning Filters for Efficient ConvNets // 5th International Conference on Learning Representations, ICLR 2017, April 24-26, 2017, Toulon, France. DOI: 10.48550/arXiv.1608.08710.

7. He Y., Lin J., Liu Z. et al. AMC: AutoML for Model Compression and Acceleration on Mobile Devices // Computer Vision – ECCV 2018. Vol. 11211. Springer, Cham, 2018. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-01234-2_48.

8. Lee J., Park S., Mo S. et al. Layer-adaptive Sparsity for the Magnitude-based Pruning // International Conference on Learning Representations, Vienna, Austria, May 4, 2021. DOI: 10.48550/arXiv.2010.07611.

9. He Y., Zhang X., Sun J. Channel Pruning for Accelerating Very Deep Neural Networks // 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, October 22-29, 2017. DOI: 10.1109/ICCV.2017.155.

10. Louizos C., Welling M., Kingma D.P. Learning Sparse Neural Networks through L_0 Regularization // International Conference on Learning Representations (ICLR), 2018. DOI: 10.48550/arXiv.1712.01312.

11. Zhu M., Gupta S. To prune, or not to prune: exploring the efficacy of pruning for model compression // International Conference on Learning Representations (ICLR), October 5, 2017. DOI: 10.48550/arXiv.1710.01878.

12. Frankle J., Carbin M. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks // International Conference on Learning Representations (ICLR), March 4, 2019. DOI: 10.48550/arXiv.1803.03635.

13. Mocanu D.C., Mocanu E., Stone P. et al. Scalable Training of Artificial Neural Networks with Adaptive Sparse Connectivity inspired by Network Science // Nature Communications. 2018. Vol. 9. Article 2383. DOI: 10.1038/s41467-018-04316-3.

14. Wang H., Fu Y. Trainability Preserving Neural Pruning // International Conference on Learning Representations (ICLR), July 25, 2022. DOI: 10.48550/arXiv.2207.12534.

15. Tian G., Sun Y., Liu Y. et al. Adding Before Pruning: Sparse Filter Fusion for Deep Convolutional Neural Networks via Auxiliary Attention // IEEE Transactions on Neural Networks and Learning Systems. 2021. Vol. 36, no. 3. P. 3930–3942. DOI: 10.1109/TNNLS.2021.3106917.

16. Yang C., Liu H. Channel pruning based on convolutional neural network sensitivity // Thirty-First International Joint Conference on Artificial Intelligence, IJCAI–22, Nanjing, China, October 1, 2022. Vol. 507. P. 97–106. DOI: 10.24963/ijcai.2022/431.

17. Krizhevsky A., Hinton G. Learning Multiple Layers of Features from Tiny Images // Technical Report, Toronto, Ontario, April 8, 2009. URL: `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`

18. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // International Conference on Learning Representations, September 4, 2014. DOI: 10.48550/arXiv.1409.1556.

19. He K., Zhang X., Ren S. et al. Deep Residual Learning for Image Recognition // IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 27-30, 2016, P. 770–778. DOI: 10.1109/CVPR.2016.90.

20. Pohjalainen J., Räsänen O., Kadioglu S. Feature selection methods and their combinations in high-dimensional classification of speaker likability, intelligibility and personality traits // Computer Speech & Language. 2015. Vol. 29. P. 145–171. DOI: 10.1016/2013.11.004.

21. Novikov D.A., Buryak D.Y. Neural network pruning method during training process // Russian Supercomputing Days, September 25-26, 2023, Moscow, Russia. P. 53–63. DOI: 10.29003/m3478.978-5-317-07070-0.