IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 March 2024, accepted 25 March 2024, date of publication 29 March 2024, date of current version 10 April 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3382994*

RESEARCH ARTICLE

# Channel Pruning Method Based on Decoupling Feature Scale Distribution in Batch Normalization Layers

**ZIJIE QIU, PENG WEI, MINGWEI YAO, RUI ZHANG, AND YINGCHUN KUANG**
College of Information and Intelligence, Hunan Agricultural University, Changsha 410128, China

Corresponding author: Yingchun Kuang (475839672@qq.com)

This work was supported in part by the National Science Foundation of China under Grant 61972147, in part by Shanghai University IPv6 Scale Deployment Innovation Integration Demonstration Application and IPv6 Network Support Capability Improvement Project under Grant 2021-2022-JCSS-01003, and in part by the Based on 5G+AI Data-Driven 020 Integrated Immersive Teaching Platform Project under Grant 202201026.

**ABSTRACT** Pruning and compression of models are practical approaches for deploying and applying deep convolutional neural networks in scenarios with limited memory and computational resources. To mitigate the impact of pruning on model accuracy and enhance the stability of pruning (defined as the negligible drop in test accuracy immediately following pruning), an algorithm for reward-penalty decoupling is introduced in this study to achieve automated sparse training and channel pruning. During sparse training, the influence of unimportant channels is automatically identified and reduced, thereby preserving the ability of the important channels for feature recognition. First, by utilizing the gradient information learned through network backpropagation, the feature scaling factors of the batch normalization layers are combined with the gradient to determine the importance threshold for the network channels. Subsequently, a two-stage sparse training algorithm is proposed based on the reward-penalty decoupling strategy, applying different loss function strategies to the feature scaling factors of ''important'' and ''unimportant'' channels during decoupled sparse training. This approach has been experimentally validated across various tasks, baselines, and datasets, demonstrating its superiority over the previous state-of-the-art methods. The results indicate that the effect of pruning on model accuracy is significantly alleviated by the proposed method, and pruned models require only limited fine-tuning to achieve excellent performance.

**INDEX TERMS** Neural network, structured pruning, model compression, neural network lightweighting, automatic pruning, pruning stability.

## I. INTRODUCTION

Remarkable success is achieved by a Convolutional Neural Network (CNN) in the realm of computer vision because of its deeper and wider architecture. Its applications are widespread across academia and industry [1], [2], [3], [4], showcasing prowess, particularly in tasks related to image classification, detection, and semantic segmentation.

However, as tasks become more complex and demand higher precision, the computational power and memory requirements of the models also increase. This challenging deployment on edge devices, such as small unmanned rotorcrafts and fixed-wing aircraft, is observed. Consequently, the development of lightweight models has become paramount for edge devices constrained by computational limitations. Feasible deployment and effective operation within these constrained computational environments are ensured by these models.

Currently, research on model lightweighting primarily involves the following methods: 1. Model pruning; 2. Model parameter quantization [5], [6], [7]; 3. Model knowledge distillation [8], [9], [10]; 4. Lightweight model design [11], [12]. Pruning methods achieve compression by removing redundant and unimportant parameters from

The associate editor coordinating the review of this manuscript and approving it for publication was Mu-Yen Chen.

large models. Quantization methods involve quantizing high-precision parameters in the model to low precision, thereby achieving compression. Knowledge distillation methods use a large model to guide the training of a smaller model, reducing computational cost while retaining effective information from the large model. Lightweight model design methods reduce the size of feature maps by directly modifying the model architecture, thereby reducing computational cost. This study mainly focuses on model lightweighting from the perspective of model pruning.

Current network pruning techniques typically follow three steps [13], [14]: 1) pre-training or sparse training: training the original network; 2) pruning: unimportant neuron connections or channels are eliminated; and 3) fine-tuning: the pruned network is refined through fine-tuning. Significant accuracy loss after pruning is suffered by most existing pruning methods, rendering the pruned models unusable without subsequent fine-tuning. The substantial decrease in model accuracy after pruning can be attributed to the fact that determining the importance of neurons or channel feature maps is the primary focus of most pruning methods, overlooking the pruning stability [15] (defined as the negligible drop in test accuracy immediately following pruning) of the model. A network with poor pruning stability often fails to achieve accurate pruning, resulting in suboptimal accuracy performance even after fine-tuning the model.

To address the challenges mentioned above, this paper presents a decoupled sparse algorithm based on the distribution of batch normalization layers for channel pruning in convolutional neural networks(referred to as DSD). The primary objective is to enhance the quality of sparse training, ensuring that the model exhibits excellent pruning stability after sparse training, and guarantees no significant difference in accuracy performance before and after pruning. The evaluation metric for determining channel importance integrates the scaling factors of the batch normalization layer and gradients. A two-stage reward-penalty decoupled sparse training algorithm was devised to achieve precise sparse training. Comparative experiments demonstrated that DSD outperformed the state-of-the-art methods in classification and detection tasks. The main contributions of this study are summarized as follows:

1) The method devises a channel-level importance decoupling threshold that combines batch normalization layer feature scaling factors and gradients. This threshold accurately distinguishes between the important and unimportant channels.
2) Introducing a two-stage reward-penalty-based decoupled sparse training approach. This method consists of a decoupled sparse phase and decoupled fine-tuning phase. Different reward or penalty strategies are applied to channels of varying importance in each stage, enhancing the pruning stability of the model and effectively mitigating the issue of the accuracy drop associated with pruning.
3) The effectiveness and universality of the proposed method were validated through experiments on benchmark models across various renowned classification and object detection datasets.

## II. RELATED WORK

As a technique aimed at achieving a lightweight model, the focus of model pruning is fundamentally the identification of the optimal subnetwork. The research conducted by Denil et al. [16] proposed that significant redundancy exists in the weight parameters of deep neural networks. It is suggested that a comparable or even superior performance to the original network model during training can be achieved by predicting the remaining weights using only a small portion. Moreover, the hypothesis proposed by Frankle and Carbin [17] posits that, within each large-scale network, there exists a smaller subnetwork. This subnetwork requires no specific initialization and can achieve a test accuracy similar to that of the original network over similar training iterations. This hypothesis is further supported by Zhou et al. [18], who provided additional evidence for the authenticity and practicality of this concept.

Pruning methods are encompassed by two main categories: non-structured pruning [19] and structured pruning [44]. Non-structured pruning involves the removal of neural connections (weights) within a neural network, resulting in non-structural sparsity. Although high compression rates are often achieved through non-structured pruning, specific hardware or library support is required to realize tangible acceleration. In contrast, all filters are eliminated from the neural network through structured pruning, utilizing efficient standard hardware such as the Basic Linear Algebra Subprograms (BLAS) library to achieve acceleration and compression [20].

Importance assessment is the primary focus of pruning studies, as exemplified by unstructured pruning techniques referenced in [18], [19], [21], and [22]. Typically, these methods employ weight importance evaluation to determine weight pruning in models. Pruning operations typically involve zeroing model weights, thereby influencing the neural connections and simulating the outcomes of weight trimming. Although preserving the model accuracy is more readily facilitated through weight pruning, the resulting convolutional kernels and feature maps manifest sparsity interspersed with irregular zeros. Achieving lightweight model deployment becomes challenging without specific hardware or software support, owing to the observed sparseness after pruning.

Hence, a shift in research focus toward more structured methods for eliminating redundant channels has been observed, as evidenced in studies such as [23], [24], [25], [26], [27], [28], [29], [30], [31], and [32]. The distinguishing factor is found by extending the granularity of the importance assessment from the level of weights to that of the channels. Slimming [25] implemented a penalty-based channel pruning factor technique to induce matrix sparsity, but encountered

challenges in precisely detecting redundancy. In the work of [23], rank evaluation was performed on feature maps generated by specific channels, discarding those with low ranks as indicative of redundancy. However, feature mappings are approached on a per-channel basis, overlooking the interrelations between the channels. Investigations into the approach in [24] include the introduction of trainable masks related to class label post-channels to achieve redundancy analysis. Nonetheless, the applicability of this method is limited to object classification, and lacks generality.

In recent studies, model-pruning techniques have been applied to remote sensing image classification, object detection, and semantic segmentation models. The research in [33] introduced a novel frequency-domain-based filter pruning method, aligned with the human visual system, to determine filter importance based on relative low-frequency components across channels. In [34], a data-customized multi-objective optimization pruning (DMOP) framework was proposed for pruning in remote sensing scene image classification. This framework utilizes a multi-objective evolutionary algorithm (MOEA) to determine the balance between the CNN pruning ratio and capability. It also incorporates a data-customized surrogate mechanism, reducing the size of input datasets to expedite the pruning evolution by streamlining the structure of the pruned subnetwork. Furthermore, [35] proposed a structured detector sparse pruning strategy that employs channel proportion factors as representations of weight importance. This method prunes network channels and employs a teaching assistant distillation model to recover compressed network performance.

## III. METHODS

### A. NOTATIONS

Assuming a convolutional neural network architecture with a set of L convolutional layers, $L^i$ represents the $i^{th}$ layer of the model. The parameters of all convolutional kernels in the $i^{th}$ layer can be expressed as a set of three-dimensional filters, defined as $K_{L^i} = \left\{K_1^i, K_2^i, K_3^i, \ldots\ldots, K_{n_i}^i\right\} \in \mathbb{R}^{n_i \times n_{i-1} \times k_i \times k_i}$, where $K_j^i \in \mathbb{R}^{n_{i-1} \times k_i \times k_i}$ is defined as the $j^{th}$ filter of the $i^{th}$ layer, $n_i$ and $n_{i-1}$represent the number of filters in the $i^{th}$ layer and in the $i-1$ layer, respectively. The filters and channels have a one-to-one correspondence, where each channel corresponds to the output result of the convolution. $k_i$ is the filter size. The complete set of filters for a model can be defined as $\Theta = \left[K_i\right] (\forall 1 \leqslant i \leqslant L)$.

Each output of a convolutional layer is termed a feature map, defined as $F_{L^i} = \left\{F_1^i, F_2^i, F_3^i, \ldots\ldots, F_{n_i}^i\right\} \in \mathbb{R}^{n_i \times b \times h_i \times w_i}$, where the $j^{th}$ feature map(channel) of the $i^{th}$ layer is represented by $F_j^i \in \mathbb{R}^{b \times h_i \times w_i}$, where b denotes a mini-batch. $h_i$ and $w_i$ are the length and width of the feature map, respectively.

Pruning segregates the channels of the model into two groups: retained channels(filters), defined as $R_L = \left\{K_{R_1}, K_{R_2}, K_{R_3}, \ldots\ldots, K_{R_{n^1}}\right\}$, and pruned channels(filters), defined as $U_L = \left\{K_{U_1}, K_{U_2}, K_{U_3}, \ldots\ldots, K_{U_{n^2}}\right\}$. Here, R represents the importance indicator for channels(filters) and U signifies the insignificance indicator. $n^1$ and $n^2$ denote the numbers of important and unimportant channels(filters), respectively. $n$, $n^1$, and $n^2$ represent the total number of channels(filters), important channels(filters), and unimportant channels(filters), respectively, in the entire model. $R_L \cup U_L = K_L$, $R_L \cap U_L = \varnothing$, and $n^1 + n^2 = n$ are stipulated.

### B. BATCH NORMALIZATION

In convolutional neural networks, a common practice is to include a batch normalization layer [36] after the convolutional layers to enhance the efficiency of network training. The parameters of the BN (batch normalization) layer can be denoted as a set consisting of $[\tau, \beta]$ elements, where each set correlates with the channels of the feature maps from the convolutional layer. The parameters of all BN layers in the $i^{th}$ layer can be represented as $Bn_{L^i} = \left\{Bn_1^i, Bn_2^i, Bn_3^i, \ldots\ldots, Bn_{n_i}^i\right\} \in \mathbb{R}^{n_i \times 2}$, where the parameters of the $j^{th}$ BN layer in the $i^{th}$ layer are defined as $Bn_j^i \in \mathbb{R}^{1 \times 2}$.

BN involves two steps. First, it normalizes the $F_j^i$, transforming the parameter distribution in $F_j^i$ to a normal distribution with a mean of 0 and variance of 1. This normalized distribution was then reverted using the learnable parameters $\gamma$ and $\beta$. This process can be expressed by the following formula:

$$\hat{z} = \frac{F_{n_i}^i - \mu}{\sqrt{\sigma^2 + \epsilon}}, z_{out} = \tau\hat{z} + \beta. \tag{1}$$

where $\sigma$ and $\mu$ represent the mean and standard deviation of the input activation, respectively, and $\tau$ and $\beta$ denote the channel feature scaling and shifting factors for $\hat{z}$.

The normalization process reshaped the feature map into a standard normal distribution with a mean of 0 and standard deviation of 1. $\tau$ and $\beta$ restore the expressive capacity of the model. Each set corresponded one-to-one with convolutional filters, and the channel scaling factors directly reflected the restoration capability of the BN layer. Therefore, $\tau$ can be perceived as an indicator of the significance of the convolutional layer channels. When $\tau$ was closer to 0, the associated channels were less crucial and could be pruned without significantly affecting the final loss of the model. Conversely, when $\tau$ was further from 0, the corresponding channels were more critical, warranting retention.

The $\tau$ usually formed a normal distribution with a mean of zero, which did not effectively distinguish between the important and unimportant parameters. Through experimentation, it were observed that when the channel scaling factors $\tau$ of the Batch Normalization (BN) layer conformed to a decoupled distribution, this distribution accurately distinguished the redundancy present in the parameters. While ensuring that the non-redundant parameters remained unaffected, the weights of the redundant parameters were minimized, ensuring that

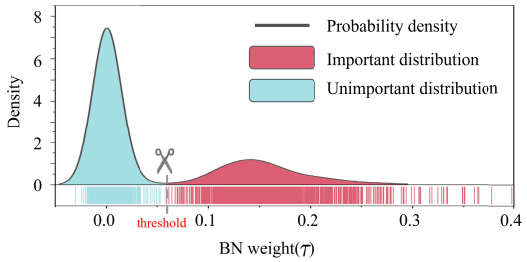


**FIGURE 1.** The distribution decoupling diagram of channel importance based on batch normalization(BN) layers scaling factors was presented in Figure (scaling factors are taken in absolute values). In the distribution chart, scaling factors clustered around 0 were categorized as "unimportant channels," while those far from 0 were identified as "important channels."

these parameters had a minimal impact on the performance of the model, as depicted in Fig. 1. The VGG model trained on the CIFAR-10 dataset [37] was employed in the experimental setup. The sparsed model achieved an accuracy of 93.8%. Post-pruning the unimportant section on the left (with a 90% parameter pruning rate), the accuracy reached 93.4%. Evidently, this distribution minimally affected the model's accuracy after pruning (i.e., maximizing stability).

The distribution of the BN layer scaling factors, suitable for pruning operations is shown in Fig. 1. It was suggested that the distribution adhered to the following guidelines: the channel scaling value $\tau$ of the BN layer was divided into two parts: $U$ (which would be pruned) converging to zero and $R$ (retained) moving away from zero.

### C. DSD

Based on the analysis in the previous section, a new two-stage model sparsity pruning method is introduced, as illustrated in Fig. 2. As shown in the figure, a reward-penalty decoupled sparse strategy was initially employed for sparse training on a baseline model. Subsequently, decoupled fine-tuning was applied to the model after the decoupled sparsity. Finally, the model underwent pruning and reconstruction using the ''pruning-torch'' framework [38].

#### 1) SPARSITY PENALTY TERM

A straightforward approach was applied to induce sparsity in the $\tau$ and $\beta$ of the BN layers, resulting in a sparse model. This method was based on the sparsity loss function introduced in [25]. The detailed formulation of the sparsity loss function is as follows.

$$Loss = \sum_{(x,y)} l\left(f\left(x, W\right), y\right) + \lambda \sum g\left(\tau, \beta\right) \tag{2}$$

The definitions of x and y represent the training input and labels, respectively, whereas W refers to the trainable weights. The first term denotes the training loss, whereas $g(\cdot)$ serves as a sparsity penalty for $(\tau, \beta)$, $\lambda$ is aimed at balancing the two components.

The L1 loss was adopted as a penalty term, formulated as follows:

$$L1\ loss\left(f\left(x, \tau, \beta\right), y\right) = ||\tau, \beta||_1 = \sum_{i}^{L} \sum_{j}^{n_i} |\tau_{i,j}| \cup |\beta_{i,j}|. \tag{3}$$

$|\cdot|$ is the absolute value. The sparse formula can be rewritten as follows:

$$Loss = \sum_{(x,y)} l\left(f\left(x, W\right), y\right) + \lambda L1\ loss\left(f\left(x, \tau, \beta\right), y\right). \tag{4}$$

The standard loss function is represented by the first term in the equation, and the second term denotes the L1 loss penalty for $\tau$ and $\beta$.

#### 2) IMPORTANCE DECOUPLING THRESHOLD

In the Slimming method [25], only the sparsity penalty function g(s) = L1 loss = |s| is utilized for $Bn_L$, where no importance analysis integration is performed to decouple the sparsity from $Bn_L$. Based on the findings from Fig. 1 in the preceding text, a revision of the sparsity penalty function $g(\cdot)$ is required by incorporating an important decoupling mechanism. The inspiration was drawn from [21]. In each iteration of the training process, the image data were fed into the model, gradients were computed using chain rule differentiation, and a metric was subsequently calculated for each $\tau$. The incorporation of gradients into the feature importance metric function allows for the consideration of neuron or channel correlations using a first-order Taylor series, resulting in a more precise measurement of importance. $\tau$ with concurrently high absolute values in both gradient and weight, received higher feature importance metrics, thereby incentivizing $\tau$ to shift away from zero. Conversely, $\tau$ with lower absolute values in both the gradient and parameter was penalized, causing $\tau$ to move toward zero. In the training process, the product of the gradient and parameter was employed as the feature importance metric. Let $T\left(x, y, \tau_{i,j}\right)$ represent the metric value for the $j^{th}$ scaling factor in the $i^{th}$ layer. The detailed formula is as follows:

$$T\left(x, y, \tau_{i,j}\right) = \left|\nabla\tau_{i,j} \times \tau_{i,j}\right|. \tag{5}$$

In the above formula, $\nabla\tau_{i,j}$ was defined as a batch of training iterations' gradients for $\tau_{i,j}$. Each time the model computed the gradients, a set of metrics represented by $T\left(x, y, \tau\right) \in T_\tau$ was generated. This set of metric values was arranged in a descending order to obtaining $T_\tau \downarrow$. The metric values were then decoupled into important and unimportant components using a threshold value $\eta$. The threshold is determined by the pruning rate $Q$, as shown in the following formula:

$$n^1 = n \times (1 - Q) \tag{6}$$

$$\eta = n^1\_th\ in\ T_\tau \downarrow. \tag{7}$$

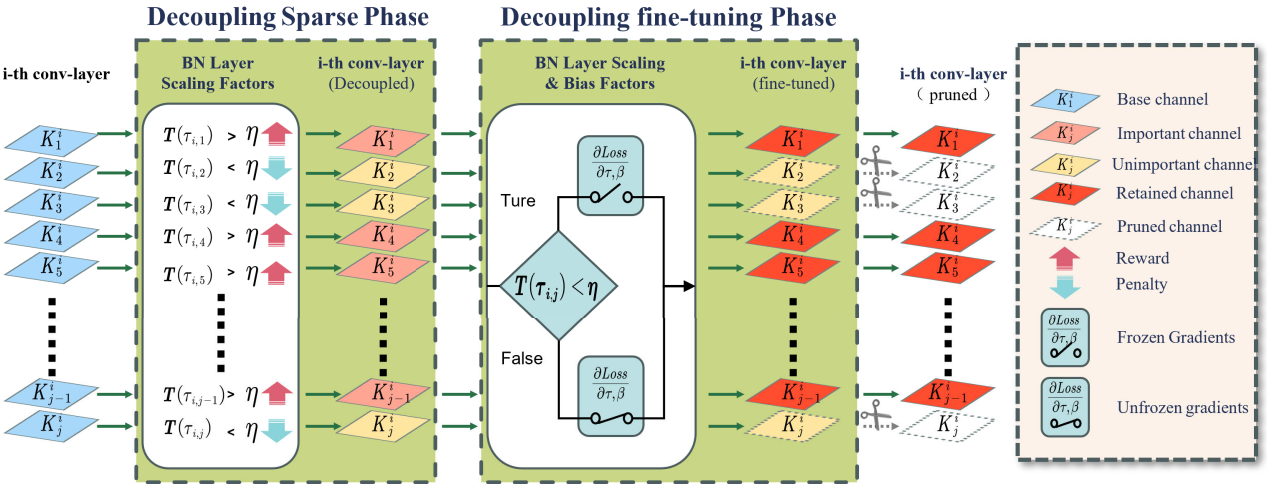


**FIGURE 2.** The DSD flow chart initiates with the "decoupling sparse" step, followed by "Decoupling fine-tuning", and concludes with channel pruning on the refined sparse model.

Based on this threshold, importance-based decoupling sparsity training can be performed on the model, rewarding the important parts and penalizing the unimportant parts.

### 3) TWO-STAGE SPARSITY

To further mitigate the impact of pruning on model accuracy, efforts have been made to enhance the stability of sparse models for pruning through improved sparse training strategies. During sparse training, the influence of unimportant channels was forcefully minimized through a decoupled penalty, ensuring that their removal did not adversely affect the performance of the model. Conversely, important channels were safeguarded through a decoupled reward, preserving their original impact on the model. To achieve this, distinct strategies were employed during various stages of sparse training. Sparse training was divided into two phases, as shown in Fig. 2. The initial phase, termed "the decoupling sparsity stage," incorporated a rewarding-penalizing mechanism within the sparsity penalty function based on a threshold. In the subsequent phase, identified as "the decoupling fine-tuning stage," the model was segmented into important and unimportant components using the threshold, allowing only the significant filters to undergo parameter updates.

#### *a: DECOUPLING SPARSITY*

During the decoupling sparsity stage, the previously obtained threshold was integrated into the sparse loss function as part of the rewarding-penalizing phase. This resulted in further modification of the sparse loss function of the BN layer.

$$Loss\ \tau_p = l\left(f\left(x, W\right), y\right) + \lambda L1\ loss\left(f\left(x, \tau_{i,j}\right), y\right) \quad (8)$$

$$Loss\ \tau_r = l\left(f\left(x, W\right), y\right) - \lambda L1\ loss\left(f\left(x, \tau_{i,j}\right), y\right) \quad (9)$$

$$Loss\ \beta_p = l\left(f\left(x, W\right), y\right) + \lambda L1\ loss\left(f\left(x, \beta_{i,j}\right), y\right) \quad (10)$$

$$Loss\ \tau = \sum_{(x,y)} \sum_{i,j}^{L,n_i} \begin{cases} Loss\ \tau_p, & if\ T\left(x, y, \tau_{i,j}\right) \leqslant \eta \\ Loss\ \tau_r, & otherwise \end{cases} \quad (11)$$

$$Loss\ \beta = \sum_{(x,y)} \sum_{i,j}^{L,n_i} \begin{cases} Loss\ \beta_p, & if\ T\left(x, y, \tau_{i,j}\right) \leqslant \eta \\ l\left(f\left(x, W\right), y\right), & otherwise \end{cases} \quad (12)$$

In the above equation, $Loss\ \tau_p$, $Loss\ \tau_r$, and $Loss\ \beta_p$ represent the loss functions of $\tau$ with the added penalty regularization term, $\tau$ with the added reward regularization term, and $\beta$ with the added penalty regularization term, respectively. $Loss\ \tau$ and $Loss\ \beta$ represent decoupled sparse loss functions for $\tau$ and $\beta$, respectively. By employing threshold $\eta$, the scaling factors $\tau$ and offset factors $\beta$ of the BN layer were segmented into important and unimportant components. The unimportant components were penalized, whereas the important components were rewarded with the opposite of the penalty term.

When penalizing the unimportant components of the model, simultaneous penalization of the scaling value $\tau$ and the offset value $\beta$ is required. However, when rewarding the important components of the model, only the scaling value $\tau$ was subjected to a reward, and the offset value $\beta$ could not be simultaneously rewarded. This limitation arises because rewarding the offset value $\beta$ compromises the precision of the model's pruning. Hence, the reward mechanism does not include offset factors $\beta$. Further elaboration on this is provided in the section detailing ablation experiments.

Sparsity training was conducted on the model using the proposed sparsity loss function. After multiple iterations, a novel distribution emerged that divided the $\tau$ distribution into two segments: important and unimportant. This revised distribution converged segments deemed unimportant during network training toward zero while positioning those identified as important relatively distant from zero. Fig. 1 illustrates the distribution pattern.

#### *b: DECOUPLING FINE-TURNING PHASE*

To enhance the pruning stability of the sparse network, a decoupled fine-tuning stage was introduced following

**Algorithm 1** Pseudocode of DSD

**for** $\tau$, $\beta$ in model **do**
  $BN\_Metrics \leftarrow |\tau \times \tau\_grad|$ ▷ Computing the metric value for $\tau$.
**end for**
$t\_position \leftarrow (1 - Q) \times num\_BN$ ▷ Determining the position of the threshold,where $Q \in [0, 1]$
$top\_bn\_values \leftarrow topk(BN\_Metrics, t\_position)$ ▷ Extracting the top $t\_position$ largest elements from $BN\_Metrics$.
$BN\_Threshold \leftarrow top\_bn\_values[-1]$ ▷ Obtaining the decoupling threshold.
**for** epoch in range(epochs) **do**
  **for** $\tau$, $\beta$ in model **do**
    **if** $epoch < Epochs\ of\ decoupled\ sparsity$ **then** ▷ Decoupled Sparsity Phase.
      **if** $BN\_Metrics < BN\_Threshold$ **then** ▷ Determine reward or penalty based on $BN\_Threshold$.
        $l(f(x, W), y) + \lambda L1\ loss(f(x, \tau, \beta), y)$ ▷ Penalty
      **else**
        $l(f\ (x, W)\ , y) - \lambda L1\ loss(f(x, \tau), y)$ ▷ Reward
      **end if**
    **else** ▷ Decoupling fine-tuning.
      **if** $BN\_Metrics < BN\_Threshold$ **then**
        $no\_grad\ (l\ (f\ (x, \tau, \beta)\ , y))$ ▷ Disable gradients of $\tau$ and $\beta$ for the current update.
      **else**
        $l\ (f\ (x, W)\ , y)$
      **end if**
    **end if**
  **end for**
**end for**

the decoupled sparse stage. The decoupled fine-tuning stage involved refining the parameters near the importance threshold and eliminating excessive human interventions imposed during the reward-penalty decoupling phase through self-training of the network. The formula for this phase is as follows:

$$Loss\ \tau = \sum_{(x,y)} \sum_{i,j}^{L,n_i} \begin{cases} l\ (f\ (x, W)\ , y)\ , if\ \ T\left(x, y, \tau_{i,j}\right) > \eta \\ no\_grad\ (l\ (f\ (x, \tau)\ , y))\ , otherwise \end{cases} \tag{13}$$

$$Loss\ \beta = \sum_{(x,y)} \sum_{i,j}^{L,n_i} \begin{cases} l\ (f\ (x, W)\ , y)\ , if\ \ T\left(x, y, \tau_{i,j}\right) > \eta \\ no\_grad\ (l\ (f\ (x, \beta)\ , y))\ , otherwise \end{cases} \tag{14}$$

In this formula, $no\_grad\ (\cdot)$ represents a specific operator, indicating that gradients were not calculated when computing this function. Penalty and reward terms were no longer employed during the sparse fine-tuning phase. The gradients of the unimportant parts were frozen, and only those of the important parts were updated. This enabled the automatic correction of biases introduced by human intervention in the sparsity process, thereby effectively enhancing the stability of the model for pruning.

After decoupling the fine-tuning, the importance indicator factors $\tau$ were sorted and divided into two parts, $n^1$ and $n^2$, based on the pruning rate Q. Finally, the pruning framework [38] was employed to eliminate $n^2$ through pruning, followed by fine-tuning the pruned model.

### D. IMPLEMENTATION OF DSD

DSD was elucidated using a pseudocode representation, as shown in the pseudocode. 1. In the code, the pruning rate is denoted by Q. $\lambda$ is a sparsity parameter that balances loss and penalty terms. Function $topk\ (\cdot)$ denotes extracting the top $threshold_Position$ elements from the largest values in the list '*Metrics*.' Notably, the operator $no\_grad\ (\cdot)$ is specialized, indicating that the gradients of $\tau$ and $\beta$ are not computed when evaluating this function.

#### 1) FEATURE IMPORTANCE THRESHOLD

During training, if the gradient and weight values of a certain $\tau$ were high, their influence on the model was significant; otherwise, their influence was minimal. Therefore, the chosen importance metric is the product of the gradient and parameter [21]. (as delineated in Lines 1 and 3 of Pseudocode. 1)

Iterating the BN scaling factors within the model yields the metric set $T$. They were subsequently sorted, and the sorting metric set $T$ in descending order, resulted in $T\ \downarrow$. Utilizing pruning rate Q, an importance threshold $\eta$ can be selected from $T\ \downarrow$. Consequently, this threshold decouples metric values into two categories: essential and non-essential. (as delineated in Lines 4 and 6 of Pseudocode. 1)

#### 2) DECOUPLED SPARSITY

A two-phase strategy was adopted to enhance the precision of model pruning. In the first stage, integrated with the threshold, the L1 loss sparsity function was optimized. Using

this threshold, the scaling factors of the BN layer were partitioned into essential and non-essential components. The non-essential portion was subjected to heightened penalties, whereas the essential portion received rewards. (as delineated in Lines 10 and 14 as indicated in Pseudocode. 1)

When penalizing the non-essential components of the model, it was necessary to punish $\tau$ and $\beta$. However, only $\tau$ was rewarded when the essential components were rewarded and $\beta$ was not rewarded. This is because rewarding $\beta$ shifts the central point of the BN layer parameter distribution away from 0, thereby affecting the precision of model pruning.

#### 3) DECOUPLED FINE-TUNING

After obtaining the decoupled sparse model, the utilization of the penalty and reward terms was prohibited. The gradient of the non-essential components was frozen (set to zero), and only the gradient of the essential components was updated. This approach effectively compensated for the accuracy degradation caused by sparsification and significantly improved the precision of the model pruning. (as delineated in Lines 16 and 20 as indicated in Pseudocode. 1)

## IV. EXPERIMENTS

### A. EVALUATION METRIC

To maintain fairness and facilitate comparability of the experimental results, the detection performance of the proposed model was evaluated using the established metrics prevalent in object classification and detection methodologies. These metrics included accuracy, recall, F1 score and mean average precision (mAP).

Accuracy was measured to assess the correctness of the algorithm, whereas recall was used to evaluate the comprehensiveness of the image recognition outcomes. The F1 score served as a comprehensive metric for assessing the detection accuracy of the model, representing the harmonic mean of precision and recall. These metrics collectively contributed to a robust and holistic assessment of the model performance in object detection tasks.

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{17}$$

The formula above defined various terms: True Positive (TP) was used to denote instances where both the detection and ground truth values were positive. False Negative (FN) was used to signify the count of misclassifications where the detection result was negative while the Ground Truth was positive. False positive (FP) were used to represent the count of misclassifications where the detection result was positive, while the Ground Truth was negative. True Negative (TN) was utilized to indicate instances where the detection result and the actual value were negative, reflecting the correct identification of negative samples.

$$AP_i = \int_0^1 P(R)dR \tag{18}$$

$$mAP = \frac{1}{C}\sum_{i=1}^{C} AP_i \tag{19}$$

The Mean Average Precision (mAP) was a pivotal metric for assessing the performance of object detectors, as defined in the provided formula. The area under the precision-recall curve was quantified and averaged across all categories, offering an overall measure of the model's efficacy.

The model's capability to detect partially overlapping targets at an Intersection over Union(IoU) threshold of 0.5 was evaluated by (mAP0.5). (mAP0.5:0.95) extended this evaluation by considering IoU thresholds ranging from 0.5 to 0.95. This comprehensive assessment was accounted for by the diverse degrees of target overlap, providing a thorough understanding of the model's performance under varying conditions.

FLOPs represent floating-point-operations, indicating floating-point calculations, essentially representing the computational load. It is used to measure the complexity of the algorithms or models. The number of model parameters that gauge the scale or size of the model is referred to as ''parameters''. The calculation formula is as follows:

$$\left(2 \times C_i \times K^2 - 1\right) = \left(C_i \cdot K^2\right) + \left(C_i \cdot K^2 - 1\right) \tag{20}$$

$$FLOPs = \left(2 \times C_i \times K^2 - 1\right) \times H \times W \times C_o \tag{21}$$

The input and output channels are represented by $C_i$ and $C_o$, the convolutional kernel size is denoted by K, and the dimensions of the output feature map are indicated by H and W.

### B. EXPERIMENTAL TOOLS AND DATASETS

Experiments were conducted on three widely used datasets: CIFAR-10 [37], COCO [39] and DOTA [40], with the aim of validating the effectiveness of DSD in the domains of object classification, object detection and remote sensing imagery.

The CIFAR-10 dataset was utilized for object classification tasks and comprised ten classes, featuring 50,000 images for training and 10,000 for validation.

MS COCO, which is short of Microsoft Common Objects in Context, is regarded as one of the most popular and authoritative datasets for object detection. It comprises over 100,000 images for object detection, covering 80 categories. On average, each image contained 7.2 objects and many small-scale objects.

DOTA is a large-scale public dataset for object detection in remote sensing images that contains numerous objects with rotation, high aspect ratios, and densely arranged patterns. It consists of 2806 aerial images and 188,282 annotated instances. DOTA V1.5 included a total of 16 classes, such as

airplanes (PL), bridges (BR), large vehicles (LV) and ships (SH), among others.

In the image classification tasks, VGG-16 [41] and ResNet-56 [42] were utilized on the CIFAR-10 dataset. YOLOv5m [4] was employed in the COCO dataset for object detection tasks. In the remote sensing image detection task, a variant of YOLOv5, known as YOLOv5-obb, was used in DOTA v1.5, as referenced in [43]. It's worth noting that all experiments in this paper underwent pruning operations using the pruner [38].

### C. CONTRAST TEST

First, experiments were conducted on the classification task using classic VGG-16 and ResNet-56 model architectures on the CIFAR-10 dataset. The sparsity parameter $\lambda$ for VGGNet-16 and ResNet-56 was set to $5 \times 10^{-4}$, with a learning rate of 0.01, momentum of 0.9, weight decay of 0.0001 and images input size of 32*32 with a batch size of 64. The SGD optimizer was used in this study. The original networks underwent 300 epochs of sparse training, followed by pruning. After pruning, the models were fine-tuned for 160 epochs, utilizing the same parameters as the decoupled sparse stage. In subsequent experiments, all ResNet architecture networks omitted the channel-matching layers within the residual blocks. Retaining these layers ensured that the overall structure of the model remained unchanged and unaffected by the pruning process.

#### 1) VGG ON CIFAR-10

On VGGNet-16, the feasibility of the proposed DSD for CIFAR-10 was validated, as shown in Table. 1. The table shows a comparative analysis between DSD and other advanced pruning methods, encompassing seven channel pruning approaches (L1 [44], Slimming [25], GCN [45], HRank [23], White-Box [24], ACP [29] and ELC [26]). In Table. 1, the compression outcomes achieved by the DSD are presented, indicating a 54.84% reduction in FLOPs and a 90.1% reduction in parameters. The most substantial parameter reduction was achieved by DSD, while concurrently attaining optimal precision performance. Notably, the pruned DSD surpassed the baseline accuracy by 0.76%.

**TABLE 1. Results for pruning VGG-16 on CIFAR-10.**

| Method | Top-1 Acc(%) | FLOPs↓ (%) | Parameters↓ (%) |
|---|---|---|---|
| L1 [44] | 93.96 →93.40(-0.56) | 34.3 | 64.0 |
| Slimming [25] | 93.66 →93.8(+0.14) | 51.0 | 88.5 |
| GCN [45] | 93.1 →93.27(+0.17) | 57.29 | 85.5 |
| HRank [23] | 93.96 →92.34(-1.62) | 65.3 | 82.1 |
| White-Box [24] | 93.02→93.47(+0.45) | **76.4** | - |
| ACP [29] | 93.73→93.78 (+0.05) | 74.39 | 86.96 |
| ELC [26] | 93.56→93.25(-0.31) | 52.5 | - |
| **DSD**(ours) | **93.10→93.86(+0.76)** | 54.84 | **90.1** |

The DSD demonstrated slightly higher precision performance than the task-specific white-box method; however, its floating-point operations were 21.56% higher than those of the white-box method. Compared to ELC, which employs a similar decoupling approach and merging method, DSD excelled across all metrics. When compared to methods utilizing sparse self-training strategies (L1, Slimming,and GCN), the approach exhibited superior performance in terms of parameter reduction rate and accuracy. Moreover, the performance of FLOPs surpassed that of L1 and Slimming, marginally exceeding that of the GCN. In contrast to single-shot pruning methods such as ACP, DSD excelled in accuracy and parameter metrics while trailing slightly in FLOPs.

#### 2) ResNet-56 ON CIFAR-10

The feasibility of the proposed DSD on the ResNet-56 architecture was evaluated. In Table. 2, three distinct layer compression outcomes achieved by DSD are presented: DSD-0.6,which resulted in a 52.87% decrease in Parameters, ECL-0.65 led to a 66.66% reduction in Parameters, and ECL-0.7 showed a 71.26% reduction in parameters. Overall, as the pruning rate increased incrementally, both the parameters and FLOPs decreased gradually, leading to a corresponding decrease in model accuracy. This perspective was further substantiated in subsequent experiments using remote sensing imagery. Superior performance within the ResNet architecture was demonstrated by this method compared with the VGG architecture,which included only convolutions. At the pruning rate of 52.87%, the accuracy surpassed that of the baseline by 0.59%. With a pruning rate of 66.66%, there was a substantial 69.23% reduction in FLOPs while experiencing only a slight decrease in model accuracy (-0.34%). At a pruning rate of 71.26%, there was a 76.92% reduction in the FLOPs, but the model accuracy decreased by 1.07%.

**TABLE 2. Results for pruning RESNET-56 on CIFAR-10.**

| Method | Top-1 Acc(%) | FLOPs↓ (%) | Parameters↓ (%) |
|---|---|---|---|
| L1 [44] | 93.26→93.06 (-0.2) | 27.6 | 14.1 |
| HRank [23] | 93.26→93.52(+0,26) | 29.3 | 16.8 |
| GCN [45] | 93.72→93.85(+0.13) | 48.31 | 35.01 |
| NISP [22] | 93.26→93.29(+0.03) | 43.61 | 42.60 |
| ACP [29] | 93.92→92.56 (-1.36) | 53.65 | 49.41 |
| TPP [46] | 93.78→93.46 (-0.32) | - | 49.82 |
| WHC [47] | 93.59→93.66 (+0.07) | 54.8 | - |
| White-Box [24] | 93.26→93.54(+0.28) | 55.6 | - |
| GAL-0.8 [48] | 93.26→91.58(-1.68) | 60.2 | 65.9 |
| ELC [26] | 93.45→93.08(-0.37) | 63.8 | - |
| **DSD-0.6**(ours) | **93.95→94.54(+0.59)** | 61.53 | 52.87 |
| **DSD-0.65**(ours) | 93.95→93.61(-0.34) | **69.23** | **66.66** |
| **DSD-0.7**(ours) | 93.95→92.88(-1.07) | 76.92 | 71.26 |

On ResNet-56, DSD was compared with eight channel pruning methods (L1, Hrank, GCN, ACP, TPP, WHC, White-Box, GAL-0.8,and ELC) and a neuron-based unstructured pruning method (NISP). In comparison, relatively superior performance was exhibited by the DSD. For instance, ResNet-56's FLOPs were reduced by 55.6% with a slight increase in accuracy of 0.28% using White-Box. In contrast, DSD-0.6 lowered the FLOPs by 61.53%, achieving a 0.59% accuracy improvement. Notably, when both models

experienced nearly identical parameter decreases (66%), DSD-0.65 outperformed GAL-0.8 by 1.34% in accuracy while reducing FLOPs by 9.03%. Moreover, when DSD-0.7 demonstrated similar accuracy decreases to ACP and GAL-0.8 (-1.07% vs -1.36%, -1.68%), both the reduction rates in FLOPs and parameters for DSD-0.7 surpassed those of ACP and GAL-0.8.

#### 3) YOLOv5 ON COCO

Tests were conducted on the COCO dataset for object detection tasks, demonstrating the versatility of the DSD in this domain. YOLOv5 was selected as the test model because of its established research status and because it, provides a comprehensive basis for evaluation.

The sparse parameter $\lambda$ was set to 0.001, with a learning rate of 0.01, images input size of 640*640, and batch size of 64. SGD was employed as the optimizer, and the remaining hyperparameters followed official code specifications. Initially, the original network was subjected to 60 rounds of sparse training. Subsequently, the sparse-trained model is pruned. Post-pruning, the model experienced fine-tuning for 2000 rounds, employing identical parameter settings as the sparse training phase. During the sparsification and pruning phases, the convolutional layers preceding the residual blocks in the backbone and channel-matching layers (1*1 convolution) within the residual blocks were omitted. Preserving these layers ensured that the overall structure of the model remained unchanged and unaffected by the pruning process. This rule was consistently followed in subsequent experiments using YOLOv5.

Table. 3 illustrates the pruning outcomes of the DSD on YOLOv5m. The pruning rate of the DSD on YOLOv5m was meticulously adjusted to match the parameter count of YOLOV5s (the official lightweight version). Superior accuracy loss was exhibited by DSD when parameters were identical compared to YOLOV5s (-1.96% vs -7.7%), yet its FLOPs performance was lower than YOLOv5s (26.2M vs 17M). YOLOv4-Tiny [49] streamlined the YOLOv4 model to 6.5M parameters, slightly fewer than DSD's 7.3M, but the model's accuracy at 61.14% surpassed YOLOv4-Tiny's 42.6%. Compared with other pruning methods based on YOLOv5m (YOLOX-S [50], Eagleye-YOLOv5m [51] and PAGCP [52]), DSD demonstrated a 14.1% reduction in parameters with a -1.96% accuracy decrease, showcasing superior performance in these two aspects. DSD's FLOPs measured 26.2, slightly lower than YOLOX-S and Eagleye-YOLOv5m (26.2 vs 26.8) and higher than PAGCP's (26.2 vs 23.5).

**TABLE 3. Results for pruning YOLOV5 on COCO.**

| Method | mAP@.5 (%) | mAP@.5:.95 (%) | FLOPs (G) | Parameters (M) |
|---|---|---|---|---|
| YOLOV5m [4](baseline) | 63.1 | 44.5 | 51.3 | 21.4 |
| YOLOV5s [4] | 55.4 | 36.7 | 17.0 | 7.3 |
| YOLOv4-Tiny [49] | 42.6 | 23.6 | **7.8** | **6.5** |
| YOLOX-S [50] | 58.7 | 39.6 | 26.8 | 9.0 |
| Eagleye-YOLOv5m [51] | 58.7 | 40.2 | 26.8 | 10.4 |
| PAGCP [52] | 60.7 | **41.5** | 23.5 | 7.7 |
| DSD(ours) | **61.14** | 40.94 | 26.2 | 7.3 |

#### 4) YOLOv5-obb ON DOTA

Rotational object detection tests were conducted on the remote sensing dataset DOTA, demonstrating the versatility of DSD in remote sensing detection tasks. YOLOv5-obb [43], an improved version of the YOLOv5 model tailored explicitly for rotation object detection tasks, was selected as the test model.

The sparsity parameter $\lambda$ was set to 0.001, with a learning rate of 0.01, image input size of 1024*1024, and batch size of 64. The SGD optimizer was utilized, whereas the remaining hyperparameters followed the guidelines provided in the official code. Initially, the model underwent 200 training rounds on the DOTA dataset, using the resulting model weights as the baseline. Subsequently, the original network is subjected to 60 rounds of sparse training and pruning. After pruning, the model underwent 200 rounds of fine-tuning, employing the same parameter settings as those used in the sparse training phase. It is important to note that because of the high-resolution images in the DOTA dataset, which might pose challenges for model processing, each image was segmented into several smaller images, each 1024*1024 in pixels.

Table.4 presents the pruning results of the DOTA dataset using the DSD method. Pruning operations were conducted on the model at various pruning rates to observe their impact on the DSD training and pruning technique. The performances of the after pruning and after fine-tuning were specifically differentiated. The stability of the model to pruning after sparse training could be observed from the post-pruning accuracy of the pruned model, whereas the post-fine-tuning model precision indicated the model's performance potential under the current pruning strategy and rate, highlighting the actual expressive capacity of the pruned model under these conditions.

Table.4 indicates that directly pruning the model without sparse training significantly undermined the model's expressive capacity. However, employing DSD for sparse training before pruning safeguards the expressive capability of the pruned model. At a pruning rate of 59.78%, the precision of the model was only 3% (mAP@.5) lower than the baseline. After fine-tuning, the precision of the model was nearly identical to the baseline. As the pruning rate increased, there was a gradual yet limited decline in the precision of the pruned model. At a pruning rate of 90.85%, the precision of the pruned model decreased substantially. However, after fine-tuning, the precision remained at 69% (mAP@.5). At a pruning rate of 91.97%, the pruned model became directly unusable and required fine-tuning to restore its expressive capacity. After fine-tuning, the precision was 65.3% (mAP@.5), a 3.7% decrease compared to the precision at the 90.85% pruning rate. This indicates that pruning YOLOv5-obb using the DSD strategy reached the pruning limit at a 90% pruning rate. Surprisingly, at a 59.78%

**TABLE 4.** Results for pruning YOLOV5-obb on DOTA. (YOLOV5m-obb [43](baseline) mAP@.5:72.39% mAP@.5:.95:45.97% FLOPs:46.08 Parameters:21.66).

| Pruning rate | Pruned | | After fine-tuning | | | |
|---|---|---|---|---|---|---|
| | mAP@.5 (%) | mAP@.5:.95 (%) | mAP@.5 (%) | mAP@.5:.95 (%) | FLOPs (G) | Parameters (M) |
| 56.6% (Without DSD)* | 0.00 | 0.00 | 29.8 | 13.5 | 19.51 | 9.40 |
| 59.78% | 69.4 | 40.2 | 72.4 | 44.1 | 29.54 | 8.71 |
| 67.26% | 66.9 | 35.9 | 71.4 | 43.5 | 32.93 | 7.09 |
| 83.33% | 56.7 | 25.8 | 69.5 | 41.4 | 21.40 | 3.61 |
| 90.85% | 15.6 | 5.8 | 69.0 | 40.9 | 16.58 | 1.98 |
| 91.97% | 0.0 | 0.0 | 65.3 | 36.9 | 12.22 | 1.74 |

* Pruning the baseline network directly without utilizing any sparse strategies.

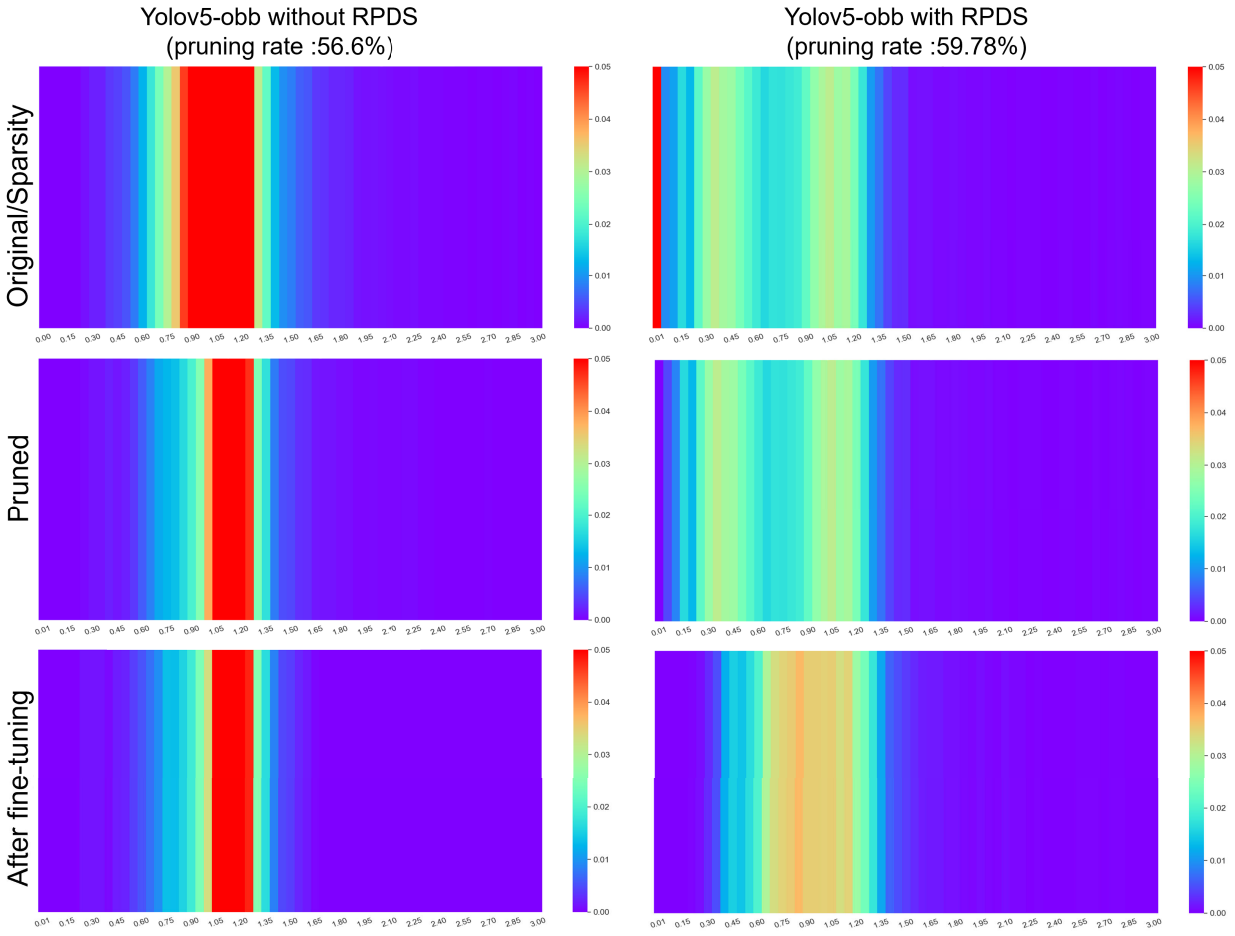


**FIGURE 3.** The heatmap illustrates the distribution changes of $\tau$ in the BN layer across different stages.

pruning rate, the FLOPs measured only 29.54G, which outperformed the FLOPs performance at pruning rates of 67.26%. It was speculated that, at a pruning rate of approximately 60%, a model structure capable of balancing FLOPs and precision was discovered by DSD.

The changes in the accuracy recovery of the pruned YOLOv5-obb model using different pruning rates during the fine-tuning stage are shown in Fig. 4. From the figure, it can be observed that when the pruning rate is 59.78%, the accuracy of the pruned model is comparable to that of the fine-tuned model, achieving satisfactory performance without fine-tuning. The trend of the accuracy recovery remained consistent for pruning rates below 90.85%, and by epoch 60, the accuracy of the pruned model could be effectively restored to an optimal state. However, when the pruning rate reached 91.97%, the detrimental effect of model

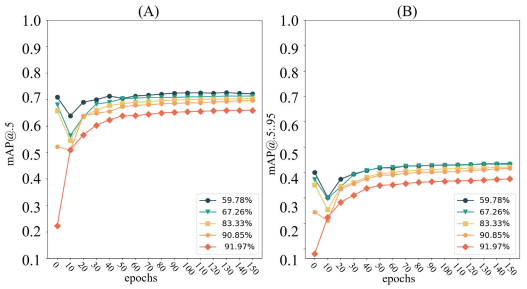


**FIGURE 4.** The accuracy measurements of network feed-forward at each epoch during the fine-tuning stage after pruning the model with different pruning rates.(The left figure (A) corresponds to (mAP@0.5), and the right figure (B) corresponds to (mAP@0.5:0.95).

pruning on accuracy became more pronounced, requiring more fine-tuning iterations to recover. It is evident that the model accuracy peaked after 90 epochs of fine-tuning.

The heatmap in Fig.3 illustrates the change in the BN layer $\tau$ distribution, and further elucidates the operational principle of pruning. The figure displays heatmaps of BN weights, showing pruning rates of 56% for BN weights pruned without using direct sparsity, and 59.78% after employing DSD to prune the model.

The first image on the left portrays the distribution heatmap of the BN weights from the pre-trained YOLOv5-obb model. It was observaed that the distribution of the BN layer $\tau$ in the original model was primarily concentrated between 0.9 and 1.30, relatively far from the origin at 0. This distribution impeded model pruning because each BN $\tau$ channel significantly affected the model output. The second and third images on the left depict the distribution of the BN layer $\tau$ after directly pruning and fine-tuning the pre-trained model. Even after pruning and fine-tuning, the distribution of BN weights remained predominantly within the 1-1.30 range. This pruning method eliminates many necessary components, severely compromising the expressive capacity of the model.

For comparison, the DSD sparsity strategy was employed to induce sparsity in the pre-trained model. By accurately identifying the importance, redundant channels were selected and normalized through decoupled sparse training, ensuring that these redundant channels did not significantly affect the performance of the model. The results after sparsity induction are depicted in the first plot on the right in Fig. 3, where DSD effectively normalizes the non-essential BN layer $\tau$ to approximately 0, highlighted in bright red in the 0-0.01 range. During pruning, the highlighted red BN layer $\tau$ and its corresponding channel components are removed, resulting in the second image on the right after pruning. Subsequent fine-tuning partially restored the expressive capacity of the model, as shown by the distribution of BN layer $\tau$ in the third image on the right.

Fig. 5 compared the detection performance on the DOTA dataset between the baseline and the pruned model (pruning rate: 59.78%). The images in the first row of the figure were represented by labeled anchor boxes, the second row showed the prediction results of the baseline model, and the third row displayed the prediction results of the model after pruning using DSD. From the comparative results in the figure, it was evident that the lightweight model obtained after pruning with DSD performed comparably to the baseline in remote sensing image recognition. Notably, the fourth column (D) presented numerous small objects (small vehicles) as targets, which posed a challenge for detection. The results indicated that, after applying DSD for sparsity induction, the pruned model exhibited an improved capability in detecting small objects, potentially surpassing baseline performance.

### D. ABLATION EXPERIMENT

#### 1) TWO-STAGE SPARSITY

Ablation experiments are conducted using the proposed two-stage sparsity strategy. The sparse decoupling stage is divided into Decoupling Sparse and Decoupling fine-tuning stages. Compression experiments on ResNet-56 using CIFAR-10 were performed by employing different sparse decoupling strategies under the same compression rate setting (52%).

**TABLE 5.** Two-stage Sparsity Ablation Experiment Based for RESNET-56 on CIFAR-10.(Pruning rate:52.87%).

| Baseline Model Top-1 Acc(%) | Strategy | Pruned Top-1 Acc(%) | After fine-tuning Top-1 Acc(%) |
|---|---|---|---|
| Resnet-56 (93.95%) | Slimming [25] | 86.95 | 91.97 |
| | Decoupling sparse | 69.59 | 93.99 |
| | Sparse & Fine-tuning* | **91.2** | **94.54** |

* Decoupling sparse and Decoupling fine-tuning

The performance of three distinct sparse decoupling strategies after model pruning is illustrated in Table.5: First,without employing Decoupling Sparse and Decoupling fine-tuning, the sparsity method aligned with Slimming's [25] approach. Slimming's sparsity method after pruning resulted in a 7% accuracy reduction compared to the baseline model, with the worst accuracy after fine-tuning, leading to a loss of 1.98%. Second,solely applying decoupling sparsity led to a significant accuracy loss after pruning, declining by 24.36% compared with the baseline. However, after fine-tuning, the accuracy surpassed that of Slimming's method, trailing the baseline by only 0.04%. Third, employing decoupling fine-tuning in the sparse decoupling stage resulted in the pruned model maintaining a relatively good accuracy, with only a 2.75% decrease. After fine-tuning, the accuracy of the model surpassed the baseline by 0.59%.

As presented in Table. 6, further experiments were conducted using Yolov5-obb on the DOTA dataset, which involves a more challenging detection task with larger and more complex data. The application of the Slimming method for model sparsity results in a significant loss of accuracy in the pruned model, rendering it unusable. Even after fine-tuning, the accuracy of the model could only be partially recovered, reaching 66.7%. With the decoupling sparse strategy, the accuracy of the pruned model was preserved to a certain extent (48.1%), and post-fine-tuning increased the

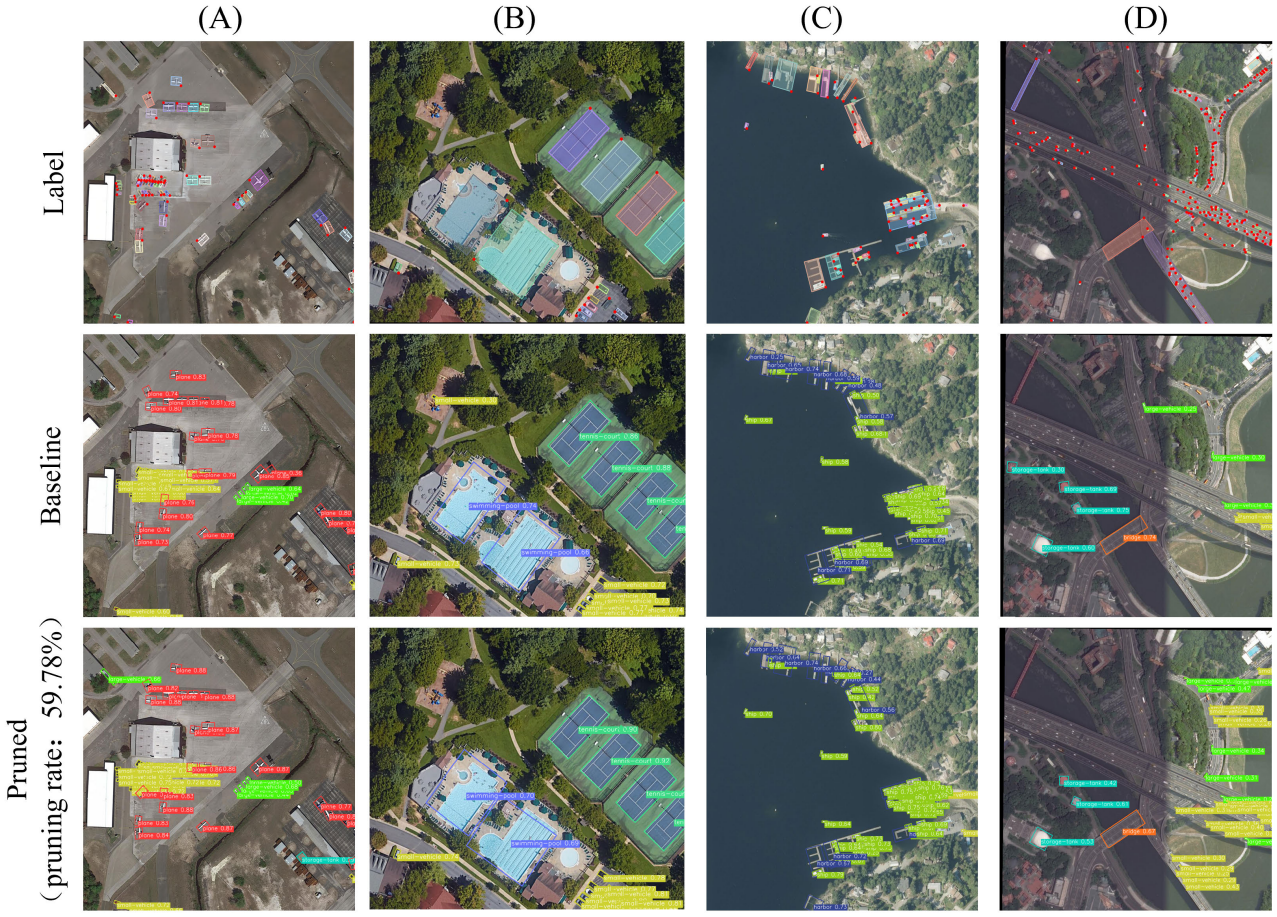


**FIGURE 5.** Comparative analysis of detection performance on the DOTA dataset.

**TABLE 6.** Two-Stage Sparsity Ablation Experiment Based on YOLOV5-obb on DOTA.(Peuning rate:85%).

| Baseline Model mAP@.5(%) | Strategy | Pruned mAP@.5(%) | After fine-tuning mAP@.5(%) |
|---|---|---|---|
| Yolov5-obb (72.39%) | Slimming [25] | 0.00 | 66.7 |
| | Decoupling sparse | 48.1 | 68.4 |
| | Sparse & Fine-tuning* | **56.7** | **69.5** |

* Decoupling sparse and Decoupling fine-tuning

accuracy to 68.4%. The best outcome was observed with the 'decoupling sparse and decoupling fine-tuning ' strategies, where a high accuracy of 56.7% was maintained by the pruned model, and post-fine-tuning achieved an accuracy of 69.5%.

The Slimming method displayed acceptable accuracy post-pruning, but its performance after fine-tuning was found to be poor. This was attributed to Slimming's simplistic regularization of scaling factors, wherein all scale factors were forcefully constrained near zero without the decoupling of essential and non-essential components. Consequently, essential channels were unavoidably pruned, leading to the loss of the model's ability to extract the necessary features and impact accuracy. However, with the inclusion of the decoupling sparse method, a noticeable enhancement in model accuracy was observed. Furthermore, with the addition of decoupling fine-tuning, there were no significant differences in the accuracy performance before and after pruning. Impressive post-fine-tuning accuracy was achieved.

Fig. 6 illustrated the accuracy measurements during each epoch of the network feedforward in the fine-tuning stage after pruning the model using various sparsity strategies. It is evident from the graph that when employing the simplest "Slimming" sparsity strategy, there is a significant loss in accuracy after pruning (i.e., poor pruning stability). Considerable time was required during fine-tuning to recover model accuracy, reaching its peak accuracy only after 130 epochs. In contrast, a decoupled sparsity method was proposed in the paper that achieved minimal accuracy loss after pruning (i.e., good pruning stability) and rapidly recovered accuracy during fine-tuning, reaching peak accuracy within 60 epochs. This indicates that the proposed sparse method enhances pruning stability by effectively distinguishing between important and unimportant components, thereby ensuring the precise removal of non-critical elements during

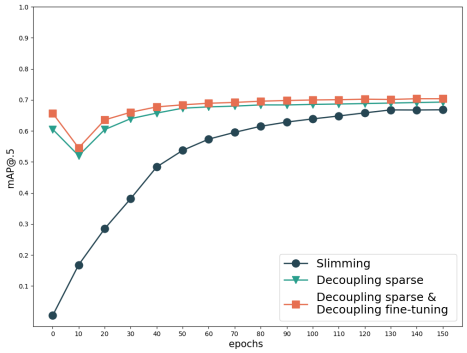


**FIGURE 6.** Two-stage Sparsity Ablation Experiment Based for Yolov5-obb on DOTA(Peuning rate:85%). The accuracy measurements of the network feed-forward during each epoch in the fine-tuning stage after pruning the model with various sparsity strategies.

pruning. Consequently, good pruning stability minimizes the accuracy loss in pruned models and significantly reduces the training cost of fine-tuning.

### 2) REWARD AND PUNISHMENT STRATEGIES

From the Table. 7, the experiments demonstrated an ablation study on the impact of batch normalization (BN) layer scaling factors on the reward-penalty strategy. Experiments conducted on the CIFAR-10 dataset using ResNet-56 indicated that the strategies with superior performance were NP+EP (Slimming) and ER+NP. It can be observed that the use of the ER+NP strategy yields optimal pruning performance, with no significant accuracy loss in the pruned model. After fine-tuning, the accuracy of the model was restored to its best performance.

**TABLE 7.** The investigation of reward-penalty strategies for scaling factors($\tau$).(Resnet-56 on CIFAR-10,pruning rate: 52.97%).

| Baseline Top-1 Acc(%) | Strategy* | Pruned Top-1 Acc(%) | After fine-tuning Top-1 Acc(%) |
|---|---|---|---|
| Resnet-56 ( 93.95%) | EP | 0.56 | 79.59 |
| | ER | 50.2 | 81.74 |
| | NR | 0.45 | 80.57 |
| | NP+EP | 86.95 | 91.97 |
| | NP | 40.35 | 80.9 |
| | ER+NP | **91.2** | **93.61** |

* E: Essential N: Non-essential P: Penalize R: Reward

From the Tabel. 8, the experiments conducted with YOLOv5-obb on the DOTA dataset are particularly representative. In the cases of the EP, ER, and NR strategies, the sparse condition of the model makes pruning impractical, rendering the pruned models unable to function correctly. For the NP+EP and NP strategies, a notable accuracy loss was observed in the pruned models. However, after fine-tuning, the accuracy of the model was partially recovered, reaching 66.7% and 68.3%, respectively. The ER+NP strategy, incorporating importance decoupling, effectively preserved the accuracy of the pruned model (56.7%), and

**TABLE 8.** Investigation of reward-penalty strategies for scaling factors($\tau$).(Yolov5-obb on DOTA;pruning rate:80%).

| Baseline mAP@.5(%) | Strategy* | Pruned mAP@.5(%) | After fine-tuning mAP@.5(%) |
|---|---|---|---|
| Yolov5-obb ( 72.39%) | EP | 0.00 | 0.05 |
| | ER | 0.00 | 0.06 |
| | NR | 0.00 | 0.03 |
| | NP+EP | 0.00 | 66.7 |
| | NP | 29 | 68.3 |
| | ER+NP | **56.7** | **69.5** |

* E: Essential N: Non-essential P: Penalize R: Reward

after fine-tuning, it exhibited the best performance among all the strategies, achieving 69.5%.

### 3) IMPORTANCE EVALUATION INDEX

Further investigation was conducted on the influence of the feature importance evaluation metrics on pruning quality, and the experimental results are presented in Table. 9. The table represented $\tau$ as the scaling factors of the BN layer, $\nabla\tau$ indicated the gradient value of the scaling factors $\tau$, and $|\cdot|$ represented its absolute value. In the experiments, a pruning rate of 65% was maintained by employing a decoupling sparse strategy that rewarded the important components while penalizing the unimportant components. The Decoupling fine-tuning strategy involves gradient updates for the significant components and freezing gradients for the insignificant components.

**TABLE 9.** Importance assessment indicator ablation experiment. (Resnet-56 on CIFAR-10,pruning rate: 65%).

| Baseline Top-1 Acc(%) | $\lvert\cdot\rvert$ | $\nabla\tau$ | $\tau$ | Pruned Top-1 Acc(%) | After fine-tuning Top-1 Acc(%) |
|---|---|---|---|---|---|
| Resnet-56 (93.95%) | ✓ | ✓ | | 11.25 | 92.05 |
| | ✓ | | ✓ | 38.01 | 93.3 |
| | | ✓ | ✓ | 16.90 | 92.71 |
| | ✓ | ✓ | ✓ | **46.63** | **94.11** |

The Table. 9 demonstrated that using $|\nabla\tau \times \tau|$ as the feature importance metric yielded the most favorable outcomes, with a post-pruning model accuracy of 46.63% and an accuracy of 94.11% post-fine-tuning. It was conversely, utilizing $|\nabla\tau|$ and $\nabla\tau \times \tau$ as feature importance indicators led to poor outcomes, with post-pruning model accuracies of only 11.25% and 16.9%, respectively, and fine-tuning accuracies of 92.05% and 92.71%. Using $|\tau|$ as a metric produced a decent outcome that was slightly lower than $|\nabla\tau \times \tau|$, with a post-pruning model accuracy of 38.01% and a fine-tuning accuracy of 93.3%.

Further validation was conducted using YOLOv5-obb with the DOTA dataset.Table. 10 demonstrates that employing $|\nabla\tau \times \tau|$ as the feature importance indicator yields the best results, with a pruning accuracy of 56.7% and a fine-tuned accuracy of 69.5% for the pruned model. Other importance assessment strategies failed to accurately evaluate the importance of the model channels, resulting in a pruning accuracy

**TABLE 10. Importance assessment indicator ablation experiment. (Yolov5-obb on DOTA,pruning rate: 80%).**

| Baseline mAP@.5(%) | $\lvert\cdot\rvert$ | $\nabla\tau$ | $\tau$ | Pruned mAP@.5(%) | After fine-tuning mAP@.5(%) |
|---|---|---|---|---|---|
| Yolov5-obb ( 72.39%) | ✓ | ✓ | | 0.0 | 67.2 |
| | ✓ | | ✓ | 0.0 | 66.8 |
| | | ✓ | ✓ | 0.1 | 68.3 |
| | ✓ | ✓ | ✓ | **56.7** | **69.5** |

of 0% for the pruned models, and the accuracy recovery of the fine-turned model was also unsatisfactory. In conclusion, the incorporation of gradients into the feature importance evaluation metric function, considering the first-order Taylor series to assess the relevance of neurons or channels, was found to provide a more accurate measure of importance.

#### 4) OFFSET FACTORS ABLATION EXPERIMENT

Table. 11 illustrates the ablation study of the sparsity strategy concerning the offset factor $\beta$ using ResNet-56 on CIFAR-10. From the experimental outcomes with ResNet-56 on CIFAR-10, it was evident that various $\beta$ strategies significantly impacted the accuracy of the pruned models. However, after fine-tuning, the accuracy of the model did not exhibit a pronounced difference compared to the baseline model. This suggests a relatively minor impact of offset factors on pruning accuracy in small-scale datasets, such as CIFAR-10. The scaling factor $\tau$ served as the importance metric for the model channels, whereas the offset factor $\beta$ did not reflect the channel importance. Varied offset metrics significantly compromise the pruning accuracy of the model, yet their impact on the accuracy of pruned models remains limited. Moreover, owing to the small size of the dataset, the model can restore itself to a functional state within a limited number of training iterations.

**TABLE 11. The investigation of reward-penalty strategies for Offset factors($\beta$). (on CIFAR-10,pruning rate: 65%).**

| Baseline Top-1 Acc(%) | Strategy* | Pruned Top-1 Acc(%) | After fine-tuning Top-1 Acc(%) |
|---|---|---|---|
| Resnet-56 (93.95%) | / | 24.55 | 93.730 |
| | ER+NP | 0.00 | 93.840 |
| | EP | 16.77 | 93.4 |
| | ER | **92.47** | 93.25 |
| | NR | 13.06 | 93.32 |
| | NP+EP | 74.72 | 93.67 |
| | NP | 46.63 | **94.11** |

* E: Essential N: Non-essential P: Penalize R: Reward /: Do nothing

To further analyze the impact of various strategies on model pruning, tests were conducted on a larger-scale remote sensing dataset, DOTA. The results obtained using the Yolov5-obb model on the DOTA dataset are presented in Table. 12. In the large-scale DOTA dataset, $\beta$ had a more pronounced effect on model pruning. Except for the NP strategy, all the other strategies significantly affected the post-pruning accuracy of the model. Except for the NP strategy, the operational capacity of the pruned models was lost, whereas the sparse models obtained using the NP strategy retained their operational capability, maintaining an accuracy of 56.7% (mAP@.5). After fine-tuning, with the exception of the NR strategy, which failed to restore the accuracy of the model, the other strategies exhibited accuracy recovery. Among them, the ''Do nothing'' and ER strategies showed relatively poorer accuracy recovery, reaching 64.5% (mAP@.5) and 66.7% (mAP@.5), respectively. The EP strategy demonstrated the best recovery accuracy, reaching 69.5% (mAP@.5). Based on the results presented in Table. 11 and the Table. 12, the NP strategy exhibited promising performance across two different tasks and datasets.

**TABLE 12. The investigation of reward-penalty strategies for Offset factors($\beta$). (on DOTA,pruning rate: 80%).**

| Baseline mAP@.5(%) | Strategy* | Pruned mAP@.5(%) | After fine-tuning mAP@.5(%) |
|---|---|---|---|
| Yolov5-obb ( 72.39%) | / | 0.00 | 64.5 |
| | ER+NP | 0.00 | 68.9 |
| | EP | 0.00 | 68.5 |
| | ER | 0.00 | 66.7 |
| | NR | 0.00 | 0.00 |
| | NP+EP | 0.00 | 68.7 |
| | NP | **56.7** | **69.5** |

* E: Essential N: Non-essential P: Penalize R: Reward /: Do nothing

## V. DISCUSSION

This paper further addresses three crucial issues in channel pruning:

1) Channel importance determination
2) The impact of model pruning on model accuracy performance
3) The pruning stability of sparse models

Feasible solutions are provided for these issues to guide researchers in further improvements. The method presented in this study has three advantages.

1) In the importance determination, the scaling factors (weights) and gradients of the BN layer produced during model training are directly used, eliminating the need for additional calculations. This straightforward approach accurately determines the importance of the channels and decouples them through a threshold.
2) The proposed reward-penalty sparse algorithm can normalize ''unimportant channels'' while protecting ''important channels'' from the impact of sparse training.
3) It was demonstrated that models with good pruning stability can resist the losses caused by pruning.
4) The pruned model can achieve good performance without the need for fine-tuning. After simple fine-tuning, the accuracy of the model can be further improved.

However, the approach presented in this study has certain limitations. First, the channel importance determination method based on BN layers only applies to convolutional neural networks with BN layers and cannot be easily extended to popular transformer models. Second, although the PyTorch

computing framework supports this method, its applicability to hardware that does not support PyTorch requires further validation. These limitations should be addressed in future research.

The future developments in this field are currently being explored, which include: 1) structured pruning methods under the transformer architecture. A recent study [53] has proposed the addition of a learnable mask component in transformer models, using binary mask variables and their saliency scores to automatically determine the importance of self-attention head modules. This opens up the possibility of applying the reward-penalty decoupling sparse method proposed in this research to the transformer architecture; 2) researching more effective methods for addressing loss function penalties. While significant progress has been made with existing technologies, the loss function may still be too simplistic; 3) incorporating additional metrics such as parameter count and model runtime into the channel importance determination function can effectively enhance the pruning model's generalization requirements across different scenarios.

## VI. CONCLUSION

This study proposes an innovative structured pruning technique to compress neural networks without compromising accuracy. By implementing the reward-penalty decoupled sparse training method on the channels of the convolutional network during sparse training, the loss of model accuracy caused by pruning was successfully minimized. This method demonstrates exceptional efficacy in convolutional networks and intricate architectures that incorporate residual blocks. Moreover, it shows commendable generalizability when applied to object detection models. This method surpasses several state-of-the-art (SOTA) approaches, including VGG-16 and ResNet-56 on CIFAR-10 for object classification tasks and YOLOv5 on the COCO dataset for object detection tasks. In addition, the noteworthy validation of the DOTA dataset highlights the effectiveness and applicability of this method in remote sensing. For instance, at a pruning rate of 59.78%, the lightweight model derived using DSD achieved the same accuracy as the baseline model. Even at an impressive compression rate of 90.85%, the pruning method yielded a lightweight model with only a marginal 3.39% decrease in accuracy compared to the baseline model. In future endeavors, it is hoped that this technique can be extended beyond convolutional architectures to create new opportunities for lightweight models in emerging high-performance models.

## REFERENCES


[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[3] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.

[4] G. Jocher, K. Nishimura, T. Mineeva, and R. Vilariño. (2020). *YOLOv5*. Accessed: Jan. 10, 2021. [Online]. Available: https://github.com/ultralytics/yolov5

[5] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1," 2016, *arXiv:1602.02830*.

[6] R. Sayed, H. Azmi, H. Shawkey, A. H. Khalil, and M. Refky, "A systematic literature review on binary neural networks," *IEEE Access*, vol. 11, pp. 27546–27578, 2023.

[7] B. Martinez, J. Yang, A. Bulat, and G. Tzimiropoulos, "Training binary neural networks with real-to-binary convolutions," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–11.

[8] Z. Yang, Z. Li, X. Jiang, Y. Gong, Z. Yuan, D. Zhao, and C. Yuan, "Focal and global knowledge distillation for detectors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 4633–4642.

[9] L. Zhang, C. Bao, and K. Ma, "Self-distillation: Towards efficient and compact neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4388–4403, Aug. 2022.

[10] B. Zhao, Q. Cui, R. Song, Y. Qiu, and J. Liang, "Decoupled knowledge distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11943–11952.

[11] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: More features from cheap operations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1577–1586.

[12] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "SkipNet: Learning dynamic routing in convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 409–424.

[13] G. Ding, S. Zhang, Z. Jia, J. Zhong, and J. Han, "Where to prune: Using LSTM to guide data-dependent soft pruning," *IEEE Trans. Image Process.*, vol. 30, pp. 293–304, 2021.

[14] Z. Liu, X. Zhang, Z. Shen, Y. Wei, K.-T. Cheng, and J. Sun, "Joint multi-dimension pruning via numerical gradient update," *IEEE Trans. Image Process.*, vol. 30, pp. 8034–8045, 2021.

[15] B. Bartoldson, A. Morcos, A. Barbu, and G. Erlebacher, "The generalization-stability tradeoff in neural network pruning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 20852–20864.

[16] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. De Freitas, "Predicting parameters in deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds. Curran Associates, 2013, pp. 2148–2156.

[17] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2018, *arXiv:1803.03635*.

[18] H. Zhou, J. Lan, R. Liu, and J. Yosinski, "Deconstructing lottery tickets: Zeros, signs, and the supermask," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'AlchéBuc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 3597–3607.

[19] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 1135–1143.

[20] Y. He and L. Xiao, "Structured pruning for deep convolutional neural networks: A survey," 2023, *arXiv:2303.00566*.

[21] X. Ding, X. Zhou, Y. Guo, J. Han, and J. Liu, "Global sparse momentum sgd for pruning very deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 6382–6394.

[22] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "NISP: Pruning networks using neuron importance score propagation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9194–9203.

[23] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao, "HRank: Filter pruning using high-rank feature map," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1526–1535.

[24] Y. Zhang, M. Lin, C.-W. Lin, J. Chen, Y. Wu, Y. Tian, and R. Ji, "Carrying out CNN channel pruning in a white box," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 7946–7955, 2023.

[25] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2755–2763.

[26] J. Wu, D. Zhu, L. Fang, Y. Deng, and Z. Zhong, "Efficient layer compression without pruning," *IEEE Trans. Image Process.*, vol. 32, pp. 4689–4700, 2023.

[27] T. Wu, C. Song, P. Zeng, and C. Xia, "Cluster-based structural redundancy identification for neural network compression," *Entropy*, vol. 25, no. 1, p. 9, Dec. 2022.

[28] C. L. Kuo, E. E. Kuruoglu, and W. K. V. Chan, "Neural network structure optimization by simulated annealing," *Entropy*, vol. 24, no. 3, p. 348, Feb. 2022.
[29] Y. Zhang, Y. Yuan, and Q. Wang, "ACP: Adaptive channel pruning for efficient neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2022, pp. 4488–4492.
[30] J. Hu, P. Lin, H. Zhang, Z. Lan, W. Chen, K. Xie, S. Chen, H. Wang, and S. Chang, "A dynamic pruning method on multiple sparse structures in deep neural networks," *IEEE Access*, vol. 11, pp. 38448–38457, 2023.
[31] M. Jeon, T. Kim, C. Lee, and C.-H. Youn, "A channel pruning optimization with layer-wise sensitivity in a single-shot manner under computational constraints," *IEEE Access*, vol. 11, pp. 7043–7055, 2023.
[32] J. Jeon, J. Kim, J.-K. Kang, S. Moon, and Y. Kim, "Target capacity filter pruning method for optimized inference time based on YOLOv5 in embedded systems," *IEEE Access*, vol. 10, pp. 70840–70849, 2022.
[33] C. Zhang, C. Li, B. Guo, and N. Liao, "Neural network compression via low frequency preference," *Remote Sens.*, vol. 15, no. 12, p. 3144, Jun. 2023.
[34] Z. Hu, M. Gong, Y. Lu, J. Li, Y. Zhao, and M. Zhang, "Data customization-based multiobjective optimization pruning framework for remote sensing scene classification," *IEEE Trans. Geosci. Remote Sens.*, pp. 1–14, 2023.
[35] C. Deng, D. Jing, Z. Ding, and Y. Han, "Sparse channel pruning and assistant distillation for faster aerial object detection," *Remote Sens.*, vol. 14, no. 21, p. 5347, Oct. 2022.
[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[37] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," Univ. Toronto, Tech. Rep., pp. 1–60, 2009.
[38] G. Fang, X. Ma, M. Song, M. Bi Mi, and X. Wang, "DepGraph: Towards any structural pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 16091–16101.
[39] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 740–755.
[40] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "DOTA: A large-scale dataset for object detection in aerial images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3974–3983.
[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[43] Y. Qing, W. Liu, L. Feng, and W. Gao, "Improved YOLO network for free-angle remote sensing target detection," *Remote Sens.*, vol. 13, no. 11, p. 2171, Jun. 2021.
[44] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–13.
[45] D. Jiang, Y. Cao, and Q. Yang, "On the channel pruning using graph convolution network for convolutional neural network acceleration," in *Proc. 31st Int. Joint Conf. Artif. Intell.*, Jul. 2022, pp. 3107–3113.
[46] H. Wang and Y. Fu, "Trainability preserving neural pruning," in *Proc. 11th Int. Conf. Learn. Represent.*, 2023, pp. 1–21.
[47] S. Chen, W. Sun, and L. Huang, "WHC: Weighted hybrid criterion for filter pruning on convolutional neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5.
[48] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, "Towards optimal structured CNN pruning via generative adversarial learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2785–2794.
[49] C.-Y. Wang, A. Bochkovskiy, and H. M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13024–13033.
[50] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO series in 2021," 2021, *arXiv:2107.08430*.
[51] B. Li, B. Wu, J. Su, and G. Wang, "EagleEye: Fast sub-net evaluation for efficient neural network pruning," in *Proc. ECCV*, Glasgow, U.K. Cham, Switzerland: Springer, 2020, pp. 639–654.
[52] H. Ye, B. Zhang, T. Chen, J. Fan, and B. Wang, "Performance-aware approximation of global channel pruning for multitask CNNs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 8, pp. 10267–10284, 2023.
[53] F. Yu, K. Huang, M. Wang, Y. Cheng, W. Chu, and L. Cui, "Width & depth pruning for vision transformers," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, 2022, pp. 3143–3151.

**ZIJIE QIU** is currently pursuing the master's degree with the College of Information and Intelligence, Hunan Agricultural University, Changsha, China. His research interests include pattern inspection and model lightweight and their applications in agriculture.

**PENG WEI** is currently pursuing the master's degree with the College of Information and Intelligence, Hunan Agricultural University, Changsha, China. His research interests include model pruning, deployment, and their application in agriculture.

**MINGWEI YAO** is currently pursuing the master's degree with the College of Information and Intelligence, Hunan Agricultural University, Changsha, China. His research interests include crowd counting and semi-supervised learning.

**RUI ZHANG** is currently pursuing the master's degree with the College of Information and Intelligence, Hunan Agricultural University, Changsha, China. His research interests include semi supervised object detection and its application in agriculture.

**YINGCHUN KUANG** received the Ph.D. degree in land resources and information technology from Hunan Agricultural University, in 2012. She is engaged in teaching and research with Hunan Agricultural University and serves as a Supervisor for the master's students. Her research has been dedicated to the study of smart agriculture and intelligent control. She has been involved in a national major support program and has led or participated in nearly 20 projects, including the Hunan Provincial Natural Science Foundation, the Key Projects of the Provincial Science and Technology Department, and the Key Research Projects of the Provincial Education Department. She has published 29 articles, with four being indexed in EI and ISTP databases. She has authored one textbook, coauthored one book, and has applied for and been granted more than ten national patents and software copyrights. She has received awards, such as the one Provincial Science and Technology Progress Award, the two University-Level Science and Technology Progress Awards, and recognition for teaching achievements and outstanding teaching quality at the university level.