ABSTRACT The success of convolutional neural networks (CNNs) benefits from the stacking of convolutional layers, which improves the model’s receptive field for image data but also causes a decrease in inference speed. To improve the inference speed of large convolutional network models without sacrificing performance indicators too much, a data-aware adaptive pruning algorithm is proposed. The algorithm consists of two parts, namely, a channel pruning method based on the attention mechanism and a data-aware pruning policy based on reinforcement learning. Experimental results on the CIFAR-100 dataset show that the performance of the proposed pruning algorithm is reduced by only 2.05%, 1.93% and 5.66% after pruning the VGG19, ResNet56 and EfficientNet networks, respectively, but the speedup ratios are 3.63, 3.35, and 1.14, respectively, and the comprehensive pruning performance is the best. In addition, the generalization ability of the reconstruction model is evaluated on the ImageNet dataset and FGVC Aircraft dataset, and the performance of the proposed algorithm is the best, which shows that the proposed algorithm learns data-related information in the pruning process, that is, it is a data-aware algorithm.

INDEX TERMS Model compression, pruning algorithm, attention mechanism, reinforcement learning.

I. INTRODUCTION
In recent years, deep learning technology has achieved rapid development with the support of massive data resources and large-scale high-performance computing equipment. Although deep learning has led to great breakthroughs in many related fields, the large-scale operations required for the associated calculations have become a bottleneck in the practical application of these networks. A VGG-16 network has 138.34 M parameters and requires 31 megabytes of floating-point operations for a single image classification task [1]. To apply the deep neural network model to more scenarios, compressing the neural network has become an indispensable option.

Among many neural network compression techniques, pruning is one of the popular methods for reducing model complexity [2]. Deep learning models generally have certain parameters or computational redundancy. A common technique to eliminate this redundancy is the pruning algorithm. A typical pruning process usually follows the following three steps: (1) determine the importance of weights; (2) delete unnecessary weights; and (3) fine-tune the network to recover accuracy. Among these, how to measure the importance of weights has become a core issue [3]. The simplest method is to measure it by the absolute value of the weights. Han et al. [4] proposed an iterative neural network pruning algorithm. In this method, when the weight of a connection is lower than
a certain threshold, the connection is deleted. This method can significantly reduce the number of model parameters without sacrificing too much performance. However, the structured sparse model obtained by this method is likely to cause the network to be too sparse, thereby causing discontinuities in memory addressing and difficulties in model deployment.

This paper proposes new ideas to address the problem of accuracy reduction caused by convolution kernel importance evaluation and pruning and proposes a method to preserve model capacity and accelerate model inference. The main contributions of this paper are as follows: (1) The group attention pruning (GAP) module is used to evaluate the convolution kernels of different layers, analyze the importance of different convolution kernels, and select different subsets of convolution kernels for different layers to achieve adaptive pruning. (2) Through the reinforcement learning method, different pruning policies are adopted for different datasets so that the final pruned model can adapt to different datasets, which is the so-called data-aware pruning. The method proposed in this paper does not cut the model parameters in a real sense but selects different convolution kernels for different data to participate in the operation, which can keep the model capacity and the learnable feature knowledge space unchanged. The loss of prediction accuracy will be greatly reduced after pruning, and the inference speed of the model will be greatly improved.

The contents of this paper are arranged as follows: the first part introduces the research background, the second part analyzes the limitations of current methods and introduces the general idea of our algorithm, the third part introduces the adaptive pruning algorithm in detail and describes the design of the core module, the fourth part describes the experimental verification, and the last part summarizes the methods proposed.

II. RELATED WORKS
To avoid the disadvantages of unstructured sparse models, structured pruning has gradually become a hot topic of research. The structured pruning method attempts to produce structured sparseness in the tensor of parameters; that is, pruning takes the tensor as the basic unit. According to the different pruning granularities, the structured pruning algorithm can be divided into connection-level pruning, vector-level pruning, channel-level pruning, and filter-level pruning [5]. Yao et al. [6] discussed the impact of pruning with different granularities, realized different pruning algorithms from the connection level to the filter level, and analyzed the storage capacity and hardware computing efficiency of the model at different granularities. They found that coarse-grained pruning algorithms (such as the filter-level algorithm) can save more storage space, obtain a greater compression rate and greatly help in improving hardware efficiency. This paper also focuses on coarse-grained pruning algorithms. This is because the damage to the original network structure can be reduced by this method, and the pruned model perfectly fits the existing deep learning library. Moreover, since the network structure does not undergo major changes, the pruned model can also be combined with other compression algorithms to further reduce the number of calculations.

The main difficulty of the coarse-grained pruning algorithm arises from two aspects: the importance of the convolution kernel and the problem of model accuracy decline after pruning. The evaluation of the importance of convolution kernels is a very important part of the pruning task. After the evaluation of each convolution kernel is completed, the convolution kernels with lower importance are selected and cut in a certain proportion and order [7]. The basic idea of the current research is to first provide a large model, estimate the importance of the parameters or parameter sets in the model, and then use the model as a reference for pruning. Li et al. [8] proposed a filter pruning algorithm. They use the sum of the absolute value of each filter as a measure of the importance of the filter. If the sum of the absolute values of a certain filter is lower than a preset threshold, then the filter can be considered redundant. Lebedev and Lempitsky [9] showed that if the output of a certain filter is almost always zero, then there is enough reason to believe that the filter is redundant. Therefore, they count the output information of each filter and use the proportion of zeros in the output tensor of each filter as a measure of the importance of the filter. To quantify the importance of the filter more accurately, He et al. [10] proposed a pruning algorithm based on Lasso regression and minimum reconstruction error to measure the importance of the filter and achieved a good balance between the model size and prediction accuracy. However, when this method is used, the standard deviation of the norm is too small, so most filters have the same importance, and it is impossible to determine which filter to remove. Lin et al. [11] introduced the scaling factor to indicate the importance of the filter. They used the idea of generative adversarial networks (GANs) as a reference, introduced GANs into neural network pruning training, and achieved state-of-the-art results. This type of method does not consider the numerical impact of the dataset input on the feature map and directly evaluates the model parameters, so the evaluation speed is relatively fast, but the method cannot measure the actual contribution of each convolution kernel to the model output. Meng et al. [12] combined the advantages of structured pruning and unstructured pruning, and a method of SWP (stripwise pruning) was proposed to prune the convolution kernel. Lin et al. [13] proposed that the feature map rank can be calculated and sorted by inputting a small number of images for each layer. This method proposes that the feature map with a larger rank contains a larger amount of information, and the corresponding filter should be retained. Jia et al. [14] defined the new concept of salient energy level (SEL) to assess the parameters’ ability to distinguish importance and then proposed the novel WRGPruner method, which simultaneously considers the synthesis of three dimensions.

Most of the convolution kernel evaluation methods proposed in the current research ignore the interaction of
convolution kernels between different layers during pruning. In fact, pruning the convolution kernel of one layer will inevitably affect the pruning strategy of the next layer. To solve this problem, in this study, we designed a channel-level attention module that can automatically learn the relationship and importance between channels and generate importance weights for each channel to strengthen and suppress some channel features.

In addition, existing methods can achieve good results for specific datasets or specific tasks. We also conducted experiments to compare different convolution kernel evaluation methods. The experiment found that for some input data, the application of data-independent evaluation method A is more effective. For other input data, it is more effective to apply evaluation method B. The selection of the convolution kernel in the existing methods has nothing to do with the dataset, i.e., it is data independent. Data-independent evaluation methods rely strongly on expert experience. To solve the shortcomings of existing methods, this paper proposes a data-aware evaluation method. When evaluating the convolution kernel, a pruning decision module is trained, which can select the convolution kernel subset to prune according to the input data. In this way, the pruning method adopts different pruning strategies for different datasets so that the pruning process can be adjusted adaptively according to different datasets.

Here, we introduce some model pruning benchmark methods, which are compared with the method proposed in this paper:

1. Filter pruning (FP) [15]: This algorithm is the earliest pruning algorithm with a convolution kernel as the basic unit. The L1 norm is used to evaluate the importance of each convolution kernel in each layer, and the convolution kernel with a larger L1 norm is considered important. After the evaluation is completed, the model is pruned according to layer-by-layer pruning or the greedy global pruning method.

2. Network slimming (NSP) [16]: This is a structured convolution kernel-level network slimming method. This algorithm uses the gamma coefficient in the batch norm layer to model the importance of the channel and adds the L1 regular constraint for the gamma coefficient in the loss function so that the model is automatically sparse.

3. Channel pruning (CP) [17]: This is convolution kernel-level pruning based on alternate two-step optimization. In the first step, Lasso regression is used to select the feature channel to be pruned, and pruning the channel means pruning the corresponding convolution kernel. In the second step, the parameters of the convolution kernel are optimized to minimize the output difference after pruning and before pruning. Alternating these two steps and pruning layer by layer finally lead to a lightweight model.

4. Soft pruning (SP) [18]: In each round of baseline model training, the L2 norm is used to evaluate the importance of the convolution kernel, and the low-importance convolution kernel is set to zero. After the next round of parameter updates, the parameters of the convolution kernel are adjusted, and then, once again, the L2 norm is used to evaluate the importance of each convolution kernel, and some of the convolution kernels are reset. Until the training is completed, the final zero-weight convolution kernel and the corresponding connection are removed from the model. The paper presents the pruning ratio of each layer and zeroes the convolution kernel parameters of each layer according to the ratio. This method has a certain degree of fault tolerance. When part of the convolution kernel is set to zero, the parameter values can be readjusted in the next round.

III. DATA-AWARE PRUNING ALGORITHM BASED ON GROUP ATTENTION MECHANISM

In this section, we introduce the details of the data-aware pruning method. First, we provide a general introduction to the algorithm framework. The algorithm consists of two parts: a channel pruning method based on a group attention mechanism and a pruning policy that learns by reinforcement learning. After that, the two parts of the algorithm are introduced in detail.

A. ALGORITHM FRAMEWORK

Channel-level model pruning includes three basic steps: convolution kernel importance evaluation, pruning, and fine-tuning. The algorithm framework proposed in this paper is shown in Fig. 1. Our algorithm is composed of two parts. Given a trained convolutional neural network (CNN) model (usually a large model that needs to be compressed), the GAP module is used to achieve channel pruning of the target layer, and the reinforcement learning algorithm is used to make cross-layer pruning position decisions. Between channel pruning and cross-layer pruning position decisions, the pruned model needs to be fine-tuned on the dataset.

In the data-aware pruning framework proposed in this paper, the GAP module not only realizes the convolution kernel evaluation function but also completes the pruning task. The group attention module can generate an importance score for each convolution kernel, but the difference from the traditional attention module is that it can reflect the influence of the pruning of the previous layer on the pruning of the subsequent layer. The GAP module can select different
parameters to participate in the forward and backward operations according to the different input data. Since group attention pruning does not truly delete model parameters, the capacity of the model is not reduced, and the accuracy loss caused by adaptive pruning is greatly reduced compared to the usual pruning methods. GAP adaptively evaluates the importance of each convolution kernel for different data and ignores the convolution kernel with a low score so that it does not participate in the calculation, which can effectively improve the model inference speed.

Since selecting different parameter sets for different data is complex and difficult to define, this study uses a self-learning method based on reinforcement learning to generate parameter selection decisions for each dataset. The GAP module is constructed on the large baseline model. Through training and updating the model, the GAP module updates parameter selection decisions in each round until the learning stabilizes. In this process, the pruning function of GAP changes with the training iteration. Therefore, adaptive pruning has data dependence and dynamic characteristics.

### B. GROUP ATTENTION MODULE

The success of CNNs benefits from the application of convolution kernels, which not only greatly reduces the number of model parameters but also ensures that all filters converge to different local minima of the cost function by randomly initializing the weights of different convolution kernels to extract valuable information from high-dimensional input data. However, not all feature channels play a role in feature extraction. To improve the inference speed of the model, in this study, we designed an attention module to achieve channel-level pruning. The principle is shown in Fig. 2.

The attention mechanism has been applied to model pruning [19]. Similar to previous studies, the attention module proposed in this paper first calculates the relationship coefficient between channels according to the previous convolution layer \( \alpha \). The coefficient is used as the evaluation index for channel importance. To prevent the occurrence of gradient explosion or gradient disappearance, scholars usually use the softmax activation function to normalize the relationship coefficient \( \alpha \) when using the attention mechanism, and the softmax activation function satisfies the following formula:

\[
\sigma(\alpha_v) = \frac{e^{\alpha_v}}{\sum_{u=1}^{N} e^{\alpha_u}} \tag{1}
\]

where \( \alpha_v \) represents the \( v \)-th relational coefficient variable, and \( N \) represents the number of relational variables.

When this traditional attention mechanism is adopted, there exists an inhibitory effect between features, and the relationship coefficient after normalization is generally low. To solve this problem, this paper proposes a GAP module. Considering that different convolution kernels can obtain different types of semantic concepts when extracting features, the relationship coefficients of the convolution kernels are divided into different groups, and the softmax activation function is normalized separately, which can effectively avoid the inhibition between feature groups, explore the association information within the group, and better reflect the differences between channels. Moreover, most of the previous studies only considered the importance of a convolution kernel channel and did not consider the influence of the former convolution kernel on the latter convolution kernel. Therefore, the GAP module proposed in this paper can integrate the pruning results of the previous convolution layer.
between channels, we first use global average pooling to map the number of channels, width and height of the feature map \(X^L\), respectively. To calculate the relationship coefficient between channels, we first use global average pooling to reduce the dimension of the feature map \(X^L\) and obtain the feature \(Z^L \in \mathbb{R}^C\) that only retains the dimension of the feature channel, as shown in the following formula:

\[
Z^L = \frac{1}{WH} \sum_{h=1}^{W} \sum_{w=1}^{H} X^L(h, w)
\]  

(2)

where \(X^L(h, w)\) is the channel vector of the feature map \(X^L\) at height \(h\) and width \(w\).

The relationship coefficient \(\alpha^{L-1} \in \mathbb{R}^{C \times C}\) of the L-1 layer feature map is calculated by matrix multiplication:

\[
\alpha^{L-1} = A^{L-1} \cdot Z^L
\]  

(3)

\(A^{L-1} \in \mathbb{R}^{C \times C \times C}\) is the learnable parameter for calculating the characteristic relationship coefficients of the (L-1)th layer, and \(C^{L-1}\) and \(C^L\) are the number of characteristic channels of the (L-1)th layer and the Lth layer, respectively.

To avoid the inhibitory effect between different semantic features when calculating the attention mechanism, we propose a method of group activation of feature relationship coefficients to enhance the difference between similar semantic features in different channels. Due to the dynamic changes in the importance of the convolution kernel in training, this paper uses masks for dynamic pruning. The formula for group activation and completion of feature map pruning is as follows:

\[
\hat{X}_i^{L-1} = X_i^{L-1} \cdot (1 + \sigma(\alpha_i^{L-1})), \quad i \in [1, 2, \ldots, K]
\]  

(4)

where \(\alpha_i^{L-1}\) is the \(i\)th set of relationship coefficients of the (L-1)th layer, \(X_i^{L-1}\) is the \(i\)th set of feature maps of the (L-1)th layer, and \(\hat{X}_i^{L-1}\) is the pruned feature maps of the (L-1)th layer. In this paper, \(K\) is set to 4.

Therefore, the calculation process of the GAP module can be summarized as follows: (1) divide the L-th feature map \(X^{L-1}\) and the relationship coefficient \(\alpha^{L-1}\) into \(K\) groups, namely, \(X_i^{L-1}, \quad i \in [1, \ldots, K]\) and \(\alpha_i^{L-1}, \quad i \in [1, \ldots, K]\); (2) normalize each group of relationship coefficients \(\alpha_i^{L-1}\) by the softmax activation function; (3) set the weight values at less than a certain threshold \(\delta\) to zero (the calculation of the threshold \(\delta\) will be discussed in section 2.3); and (4) complete the pruning of the L-th feature map according to formula (4).

### C. DATA-AWARE CROSS-LAYER PRUNING POLICY

For a complete baseline network, each convolutional layer participates in adaptive pruning, so the GAP module simultaneously deployed in multiple network layers. When pruning is required at the i-th layer, the GAP module is constructed at the i-1-th layer, and the output weight vector is recalibrated to the output feature map of the i-th layer. In the pruning task, not every set of convolution kernels in each layer needs to be pruned. In particular, when the classification task is carried out in different datasets, the convolution kernel to be pruned is obviously different because the optimal weight of the feature extraction layer of the deep learning model changes due to the difference in data distribution among different datasets. Therefore, we must design a data-aware pruning policy.

The cross-layer pruning decision is essentially a search task. Before using the Soft Actor-Critic (SAC) algorithm, we first define the Markov decision process. The compression rate of all convolutional layers is taken as the action space \(A \in \mathbb{R}^L\), and \(L\) represents the number of convolutional layers. \(A_1 = 0.5\) indicates that the compression rate \(\varepsilon\) of the first convolutional layer is 0.5, that is, half of the convolution kernel of this layer needs to be pruned; \(A_3 = 0\) indicates that the compression rate \(\varepsilon\) of the third convolutional layer is 0, that is, the layer does not need to be pruned. The GAP module is used to sort the importance of the convolution kernels (assuming that there are \(N\) convolution kernels in a layer) and then reset the corresponding weights of \(N(1 - \varepsilon)\) convolution kernels to zero according to the compression rate \(\varepsilon\) output by the SAC, thus completing the pruning operation.

The difference between the prediction accuracy of the new network and the original network is used as one of the reward signals for SAC training. In addition, the purpose of model compression is not only to ensure that the accuracy rate does not drop too much but also to reduce the complexity of the calculation. This paper proposes the following reward function:

\[
R_t = -\frac{\text{Error} \cdot \log(\text{Param}_t)}{\log(\text{Param}_{init})}
\]  

(5)

In the above formula, \(\text{Error}\) represents the difference between the accuracy of the pruning model and the original model, \(\text{Param}_t\) represents the number of model convolution kernels of the t-th round of pruning, and \(\text{Param}_{init}\) represents the number of initial convolution kernels of the model. The reward function is sensitive to model accuracy and model calculations and can accelerate the convergence to the optimal pruning model.

The SAC state space design determines the sensitivity of the pruning policy to the data, which in turn affects whether the model obtained by pruning is related to the data. Although we do not directly take the data as input when training or using the pruning decision module, the future pruning policy can be adjusted according to the previous actions and pruning effect. Therefore, the features in the data are actually embedded in training the pruning decision module. The trained pruning decision-making module can obtain complete information of the data from the Markov decision-making process of cross-layer pruning, which is also the key to realizing data-aware...
FIGURE 3. Schematic diagram of the SAC-based cross-layer pruning policy.

The state space of SAC is \( S \in \mathbb{R}^L \). For the \( l \)-th layer, the state in the \( t \)-th round of pruning is expressed as the following tuple:

\[
s_{t+1}^{l+1} = (t, l, n, c_l, a_t)
\]

where \( l \) represents the number of layers, \( n \) represents the number of filters in the \( l \)-th layer, \( c_l \) represents the total number of channels in the current \( l \)-th layer, and \( a_t \) represents the decision variable (i.e., compression rate) of the \( t \)-th round of pruning.

Due to the large state space in this paper, to find the optimal pruning model faster, the SAC algorithm is used to add maximum entropy based on the original direct maximum reward expectation:

\[
J(\pi) = \sum_{t=0}^{T} E \left[ R(S_t, A_t), \lambda H(\pi(\cdot | S_t)) \right]
\]

H is the entropy of the action in the pruning policy state \( S_t \), and \( \lambda \) is the coefficient that balances the reward and entropy. By introducing entropy into the optimization problem, the policy can be encouraged to search for better pruning actions.

The principle block diagram of the pruning policy based on SAC is shown in Fig. 3. Both the actor and critic use LSTM as the basic architecture. The critic evaluates the effect of pruning decisions made by the actor. The two constantly update their parameters in an adversarial training method, prune the intermediate model multiple times, and finally generate the optimal lightweight model. The first convolution layer of the baseline model, because it has no additional front network layer, cannot predict the convolution kernel weight of the first convolution layer. Therefore, in this study, the convolution kernel of the first convolution layer was pruned according to the predefined number. The learning of the SAC-based pruning policy needs to participate in the model pruning process. Different image data obtains different eigenvalues after the convolution operation, and the pruning produces different loss deviations on the final model output. Therefore, this pruning method is a data-related pruning method, referred to as data-aware pruning in this paper.

The proposed model pruning algorithm is summarized as follows:

1) First, an attention branch is added to the convolutional layer to be pruned in the target network according to Figure 2;
2) In the forward process of network training, the weight of the channel is obtained, and the sparsity of the channel is calculated according to the cross-layer pruning strategy in Figure 3 for sparse selection. The channels continue corresponding to more important features and removing redundant channels from the network;
3) Train and prune the network alternately until the desired pruning goal is met, and finally remove the attention branch and continue training the pruned network to complete convergence.

IV. EXPERIMENTS AND RESULTS

A. SETUP

1) DATA

The datasets used in this experiment include CIFAR-100 [20], ImageNet [21] and FGVC Aircraft [22]. The CIFAR-100 dataset is used to evaluate the pruning algorithm, and then ImageNet and FGVC Aircraft are used to evaluate the generalization of the pruned network and verify the sensitivity of the pruning algorithm to the dataset. These two datasets are all from the official version, including the training set and
test set. Before training and pruning the model, the samples of the training set and the test set are normalized. Furthermore, to enrich the training samples, data augmentation is performed on the CIFAR-100 and ImageNet datasets. Data augmentation methods include random flipping and random cropping.

2) MODEL STRUCTURE
The model structures used for the data-aware adaptive pruning experiment include VGG19 [23], ResNet56 [24] and EfficientNet [25]. The structures of these two models are relatively representative. They are typical representatives of full CNNs and neural networks with residual connections and are also widely used model structures at present. EfficientNet is currently the most representative high-performance lightweight convolutional neural network model. Instead of training EfficientNet from scratch, we prune directly on the basis of EfficientNet officially provided.

3) EVALUATION
The pruning algorithm proposed will be compared with the following baseline pruning algorithms: FP, NSP, CP and SP.

4) IMPLEMENTATION
The experimental code of the data-aware adaptive pruning algorithm is implemented by Python3 using the PyTorch deep learning framework. The hardware used for training is an NVIDIA TITAN X graphics card equipped with a cuda8.0 processor. Here, we will take VGG19 as an example to discuss the method of reconstructing the neural network structure. The pruning process of other pretraining models is similar. First, the Actor and Critic of SAC are initialized. The first round of pruning is performed according to the default parameters. Then, the model parameters are updated according to the accuracy error of the pruning model and looped in turn until the model inference speed converges to the expected target. There are a total of 16 convolutional layers in VGG19, and GAP modules need to be deployed in 15 convolutional layers. Since the number of convolution kernels in each convolution layer of the VGG model structure is large, the channel compression rate threshold is set to 50%. Different channel compression ratios are chosen according to the characteristics of different baseline models.

In the fine-tuning process of the pruning model, training is performed at a learning rate of 0.1. The maximum number of rounds of the reconstruction model is set to 200. The stochastic gradient descent algorithm is used as the optimizer to update the parameters of the reconstruction model to obtain better classification results.

B. RESULTS
1) PRUNING EFFECT ON BASELINE MODEL
The model compression effect of the benchmark network pretrained on the CIFAR-100 dataset under different pruning algorithms is shown in Table 1. All the accuracies in the table are tested according to the center cut of a single image, where B is the abbreviation for “billion”, and the speedup ratio is calculated based on the FLOP theory.

From the results in Table 1, it can be seen that whether it is VGG19 or ResNet56, the filter pruning algorithm can achieve the fastest inference acceleration, but its pruning process is more radical, resulting in a significant decrease in accuracy. Compared with other methods, the method proposed in this paper has the best performance in balancing inference acceleration and accuracy. The data-aware adaptive pruning achieved the best top-1 accuracy on the pruning model of the VGG19 baseline model and the best top-5 accuracy on the pruning model of the ResNet56 baseline model. Although channel pruning and soft pruning have also achieved high accuracy, the increase in inference speed is limited due to the addition of relatively complex calculations.

| Model       | Method | Top-1 (%) | Top-5 (%) | Params | #FLOPs | Speedup Ratio |
|-------------|--------|-----------|-----------|--------|--------|---------------|
| VGG19       | Baseline | 75.41%    | 90.58%    | 2.1×10^6 | 28.22B | 1.00x         |
|             | Ours    | 73.36%    | 88.47%    | 2.5×10^6 | 7.77B  | 3.63x         |
|             | NSP     | 70.54%    | 86.40%    | 5.4×10^6 | 9.85B  | 2.86x         |
|             | CP      | 73.05%    | 88.53%    | 9.8×10^6 | 10.25B | 2.75x         |
|             | SP      | 72.41%    | 87.53%    | 5.1×10^6 | 9.02B  | 3.13x         |
| ResNet56    | Baseline | 77.51%    | 92.41%    | 8.5×10^6 | 8.52B  | 1.00x         |
|             | Ours    | 75.58%    | 89.87%    | 4.2×10^6 | 2.75B  | 3.10x         |
|             | NSP     | 66.56%    | 76.11%    | 4.0×10^6 | 2.54B  | 3.35x         |
|             | CP      | 74.57%    | 85.54%    | 5.6×10^6 | 3.04B  | 2.80x         |
|             | SP      | 76.36%    | 88.61%    | 6.2×10^6 | 3.64B  | 2.34x         |

| EfficientNet | Baseline | 85.17%    | 91.74%    | 2.7×10^6 | 3.51B  | 1.00x         |
|             | Ours     | 79.51%    | 85.96%    | 2.4×10^6 | 3.08B  | 1.14x         |
|             | NSP     | 70.84%    | 77.37%    | 1.9×10^6 | 2.35B  | 1.49x         |
|             | CP      | 77.58%    | 84.61%    | 2.6×10^6 | 3.39B  | 1.03x         |
|             | SP      | 79.20%    | 87.52%    | 2.7×10^6 | 3.43B  | 1.02x         |
Since the EfficientNet model is already a lightweight model, the speedup ratio achieved by each method in this experiment is relatively small. Even so, our method still achieves the best performance in terms of performance and speed.

The results in Table 1 are the averages of 3 experiments. In the experiment, it was found that the reconstruction model sometimes obtained by the method proposed in this paper can achieve higher accuracy than the baseline model. This is because the attention mechanism has played an important role. While selecting the convolution kernel pruning, weights are recalibrated for other convolution kernels so that some of the convolution kernels with negative effects are not counted.

Fig. 4 shows the change in speedup ratio with pruning times when different pruning algorithms are adopted. The filter pruning algorithm directly sets all convolution kernels less than the threshold to zero, so it converges in the 15th round of pruning. The consequence of too fast a pruning speed is that some valuable convolution kernels cannot participate in the subsequent calculation. The speedup ratio of the pruning algorithm proposed in this paper has a faster convergence rate than other algorithms. This is due to the exploration mechanism of the SAC algorithm, which can quickly find the low-value convolution kernel for the classification task. We deploy the pruned model based on NVIDIA TITAN X graphics. For a 5-megapixel image, the inference time is approximately 50 ms, which meets the needs of most scenarios.

To more clearly reflect the necessity and superiority of the group attention pruning and cross-layer pruning policy, we carried out an ablation experiment. Without affecting the generality of the experimental results, we only conduct ablation experiments based on the VGG19 model. The results are shown in Table 2. It is not difficult to see from the results that the lack of any one module will significantly reduce the model performance. Among them, the cross-layer pruning strategy module is the key to improving the compression rate of the model because it can also have a certain pruning effect when the GAP module is not used. However, due to the lack of a GAP module, the predictive performance of the model is significantly reduced.

2) SENSITIVITY OF THE PRUNING METHOD TO DATASET

We use the pruning decision module trained on CIFAR-100 to prune on the ImageNet dataset and evaluate the performance of the model after pruning. To verify the adaptability of the pruning algorithm to the input data, it is also compared with other pruning algorithms. Table 3 shows the performance of each pruning method on the ImageNet dataset.

It can be seen in the results in Table 2 that the pruning method in this paper has the strongest generalization ability, and its performance on the ImageNet dataset is much better.
than that of other methods, indicating that the pruning decision module based on reinforcement learning can determine the pruning strategy according to the input data rather than blindly cutting out connections that have little impact on performance. The performance loss of soft pruning is more obvious, which shows that the method filters out some necessary convolution kernels in the pruning process and thus cannot extract valuable information from the new dataset. The pruning model overfits the original dataset. In the FGVC Aircraft experiment, due to the greater difference between the image distribution and CIFAR-100, the top-1 accuracy of various methods has declined to a certain extent. However, the performance of the method proposed in this paper is still the best.

3) PRUNING DISTRIBUTION VISUALIZATION
The key factor for the effectiveness of the data-aware adaptive pruning algorithm lies in the data-dependent convolution kernel selection policy, which selects different subsets of convolution kernels for different data as the pruning object. To visually observe how adaptive pruning selects convolution kernels for different data, we conducted convolution kernel distribution experiments on the pruning model. Taking the pruning model of the baseline model ResNet56 on the CIFAR-100 dataset as an example, we counted the pruning of the convolution kernel of the first convolution layer of the last two residual blocks of the model, as shown in Fig. 5. The red bar indicates that the convolution kernel was ignored for any input data, and the green bar indicates that the convolution kernel was ignored only under part of the input data. The abscissa of the figure represents the convolution kernel index, ranging from 1 to 64. The ordinate represents the number of samples. Since the test set includes 10,000 samples, the ordinate ranges from 0 to 10,000. Each bar in the figure indicates how many input samples of the convolution kernel were ignored.

Among them, part of the convolution kernel is selected as the pruning object (red cylinder) under any input sample, and part of the convolution kernel is selected as the pruning object (green cylinder) in only some input samples. The ordinate corresponding to some convolution kernels is 0, which means that these convolution kernels always play an important role for any input and will never be ignored. Therefore, for different input data, the convolution kernel selection strategy is different. The above observation also confirms that the pruning method proposed in this paper applies the characteristics of the image data in the pruning process.

V. DISCUSSION
The pruning method proposed in this paper can separately evaluate and select the convolution kernel for each data point in the dataset, and the less important convolution kernel does not participate in the model operation. All model parameters are retained, and the convolution kernel parameters are not deleted. In this way, the evaluation capability of the convolution kernel can be strengthened, and the loss of model accuracy can be reduced; additionally, the model inference speed can be accelerated. Although the reconstruction of the GAP module into the baseline model will introduce new parameters, the number of calculations in the fully connected layer is much smaller than that of the convolution, and the number of parameters introduced by GAP is less than one thousandth of the total parameters of the model, which can be ignored compared with convolution. Therefore, the extra storage and computation pressure introduced by the GAP model is negligible compared with the contribution of adaptive pruning.

The working process of the data-dependent data-aware adaptive pruning method proposed in this paper is similar to the dropout operation [26]. The goal of dropout is to solve the overfitting problem. There are three main differences between data-aware adaptive pruning and dropout. First, adaptive pruning selects only a part of the parameters to participate in the operation in both training and testing to improve the operation speed, while the dropout method requires all neurons to participate in the operation during testing to improve the accuracy of the model and reduce overfitting. Second, dropout sets the same neurons to zero for all input data of each training batch. Different from adaptive pruning, for different data of the same batch, different parameters are selected adaptively to participate in the operation. This means that dropout selects neurons according to the training batch, and adaptive pruning selects neurons according to the input data. Third, when dropout propagates forward in each training batch, the activation value of each neuron stops working according to a certain probability p, so the selection of neurons is a random event. Data-aware adaptive pruning is not a random selection of parameters to participate in the calculation but a learning method to generate different parameter sets for each dataset. Finally, we should also point out that one of the core innovations of our method is the application of reinforcement learning. Although reinforcement learning is widely used in neural architecture search (NAS), the design of our action space and state space is completely different. Moreover, compared with the NAS method, our method relies less on large-scale computing power and has a higher application value at this stage.

VI. CONCLUSION
A generally accepted conclusion is that deep network models are severely overparameterized [27]. Appropriate redundant parameters have a certain positive effect on model training, but they greatly slow the actual operating speed of the network. To solve the problem of too slow an actual inference speed in large networks, this paper proposes a data-aware adaptive model pruning algorithm, including a channel pruning module based on a group attention mechanism and a cross-layer pruning decision method based on reinforcement learning. Data-aware adaptive pruning is realized while maintaining the high precision of the model. The algorithm can automatically learn the relationship and importance of
channels and generate importance weights for each channel to strengthen and suppress certain channel features.

In this study, we conducted experiments on the pruning effect on the CIFAR-100 dataset and the generalization ability of the pruning model on the ImageNet dataset and FGVC Aircraft dataset and compared them with well-known channel-level pruning algorithms. We visualized the distribution of the convolution kernel of data-aware adaptive pruning and visually showed the data dependence of the proposed pruning algorithm. A GAP module combined with a pruning decision based on SAC is a model compression method with specific general applicability that is suitable for any deep learning model with a convolution kernel as the main feature extractor and different datasets with differences in data distribution. GAP has the potential to achieve good model compression performance in tasks such as pedestrian recognition, object detection, and tracking.

REFERENCES

[1] S. Ge, Z. Lao, S. Zhao, X. Jin, and X.-Y. Zhang, “Compressing deep neural networks for efficient visual inference,” in Proc. IEEE Int. Conf. Multimedia Expo (ICME), Hong Kong, Jul. 2017, pp. 667–672, doi: 10.1109/ICME.2017.8019465.

[2] A. N. Fountsop, J. L. E. K. Fenjdi, and M. Atemkeng, “Deep learning models compression for agricultural plants,” Appl. Sci., vol. 10, no. 19, pp. 6866–6880, Sep. 2020, doi: 10.3390/app10196866.

[3] M. A. Carreira-Perpinan and Y. Idelbayev, “‘Learning-compression’ algorithms for neural net pruning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Salt Lake City, UT, USA, Jun. 2018, pp. 8532–8541, doi: 10.1109/CVPR.2018.0890.

[4] S. Han, “Learning both weights and connections for efficient neural network,” in Proc. Adv. Neural Inf. Process. Syst., Montreal, QC, Canada, 2015, pp. 1135–1143.

[5] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, “Importance estimation for neural network pruning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Long Beach, CA, USA, Jun. 2019, pp. 11256–11264, doi: 10.1109/CVPR.2019.01152.

[6] Z. Yao, S. Cao, W. Xiao, C. Zhang, and L. Nie, “Balanced sparsity for efficient DNN inference on GPU,” in Proc. AAAI Conf. Artif. Intell., Hilton Hawaiian Village, HI, USA, 2019, pp. 5676–5683, doi: 10.1609/aaai.v33i01.33015676.

[7] R. Wu, A. Singla, M. R. Jahanshahi, E. Bertino, B. J. Ko, and D. Verma, “Pruning deep convolutional neural networks for efficient edge computing in condition assessment of infrastructures,” Comput.-Aided Civil Infrastruct. Eng., vol. 34, no. 9, pp. 774–789, Sep. 2019, doi: 10.1111/mice.12449.

[8] 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.

[9] V. Lebedev and V. Lempitsky, “Fast ConvNets using group-wise brain damage,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 2554–2564, doi: 10.1109/CVPR.2016.280.

[10] Y. He, X. Zhang, and J. Sun, “Channel pruning for accelerating very deep neural networks,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Venice, Italy, Oct. 2017, pp. 1398–1406, doi: 10.1109/ICCV.2017.155.

[11] S. Lin, R. Ji, C. Yang, 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), Long Beach, CA, USA, Jun. 2019, pp. 2790–2799, doi: 10.1109/CVPR.2019.00290.

[12] F. Meng, H. Cheng, K. Li, H. Luo, X. Guo, G. Lu, and X. Sun, “Pruning filter in filter,” 2020, arXiv:2009.14410.

[13] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, and Y. Tian, “HRank: Filter pruning using high-rank feature map,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Seattle, WA, USA, Jun. 2020, pp. 1526–1535.

[14] F. Jia, X. Wang, J. Guan, H. Li, C. Qiu, and S. Qi, “WRGPruner: A new model pruning solution for tiny salient object detection,” Image Vis. Comput., vol. 109, May 2021, Art. no.104143, doi: 10.1016/j.imavis.2021.104143.

[15] J.-H. Luo, J. Wu, and W. Lin, “ThiNet: A filter level pruning method for deep neural network compression,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Venice, Italy, Oct. 2017, pp. 5058–5066, doi: 10.1109/ICCV.2017.541.

[16] 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), Venice, Italy, Oct. 2017, pp. 2736–2744, doi: 10.1109/ICCV.2017.298.

[17] H. Peng, J. Wu, S. Chen, and J. Huang, “Collaborative channel pruning for deep networks,” in Proc. Int. Conf. Mach. Learn., Long Beach, CA, USA, 2019, pp. 5113–5122.

[18] Y. Shi, L. Nguyen, S. Oh, X. Liu, and D. Kazum, “A soft-pruning method applied during training of spiking neural networks for in-memory computing applications,” Frontiers Neurosci., vol. 13, p. 405, Apr. 2019, doi: 10.3389/fnins.2019.00405.

[19] S. Zhang, G. Wu, J. Gu, and J. Han, “Pruning convolutional neural networks with an attention mechanism for remote sensing image classification,” Electronics, vol. 9, no. 8, p. 1209, Jul. 2020, doi: 10.3390/electronics9081209.

[20] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[21] O. Russakovsky, J. Deng, and H. Su, “ImageNet large scale visual recognition challenge,” Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.

[22] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, “Fine-grained visual classification of aircraft,” 2013, arXiv:1306.5151.

[23] K. Simoneny and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. Int. Conf. Learn. Represent., San Diego, CA, USA, 2015, pp. 1–14.

[24] S. Wu, S. Zhong, and Y. Liu, “Deep residual learning for image steganalysis,” Multimedia Tools Appl., vol. 77, no. 9, pp. 10437–10453, 2018, doi: 10.1007/s11265-017-4440-4.

[25] M. Tan and Q. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in Proc. Int. Conf. Mach. Learn., Long Beach, CA, USA, 2019, pp. 6105–6114.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[27] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, “Model compression and acceleration for deep neural networks: The principles, progress, and challenges,” IEEE Signal Process. Mag., vol. 35, no. 1, pp. 126–136, Jan. 2018, doi: 10.1109/MSP.2017.2765695.
YI XIANG received the M.S. degree in control engineering and control theory from Yanshan University, Beijing, China, in 2005, and the Ph.D. degree in control theory and engineering from Chongqing University, Chongqing, China, in 2010. He is currently a Professor with the School of Intelligent Technology and Engineering, Chongqing University of Science and Technology, Chongqing. His research interests include computer vision, SLAM, and artificial intelligence.

JIANQUAN WU received the B.S. degree in computer science and technology from Shenyang Ligong University, Shenyang, China, in 2001, and the M.S. degree in software engineering from Chongqing University, Chongqing, China, in 2009. He is currently a Lecturer with the School of Intelligent Technology and Engineering, Chongqing University of Science and Technology, Chongqing. His research interests include digital image processing and artificial intelligence.

JINLIANG SHI received the M.S. degree in industrial automation from the Beijing University of Science and Technology, Beijing, China, in 1992, and the Ph.D. degree in mechanical engineering from Chongqing University, Chongqing, China, in 2009. He is currently a Professor with the Chongqing University of Science and Technology, Chongqing. His research interests include automation control and control theory.

YING WU received the M.S. and Ph.D. degrees in instrument science and technology from Chongqing University, Chongqing, China, in 1999 and 2002, respectively. She is currently a Professor with the School of Intelligent Technology and Engineering, Chongqing University of Science and Technology, Chongqing. Her research interests include micro/nano technology, smart sensor and systems, and artificial intelligence.