Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 11, No. 1, 2023, 53-67.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Adaptive Pruning of Convolutional Neural Network

Saeed Ahmadluei[1], Karim Faez[2*] and Behrooz Masoumi[3]

1,3. Department of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran.
2. Department of Electrical Engineering, Amirkabir University of Technology, Tehran, Iran.

Article Info

Article History:
Received 10 October 2022
Revised 03 December 2022
Accepted 19 December 2022

DOI:10.22044/jadm.2022.12337.2380

Keywords:
Convolutional Neural Network, Adaptive Architecture, Pruning, Compression.

**Corresponding author: kfaez@aut.ac.ir(K. Faez).*

Abstract

Deep convolutional neural networks (CNNs) have attained remarkable success in numerous visual recognition tasks. There are two challenges when adopting CNNs in real-world applications: a) the existing CNNs are computationally expensive and memory intensive, impeding their use in edge computing; b) there is no standard methodology for designing the CNN architecture for the intended problem. Network pruning/compression has emerged as a research direction to address the first challenge, and it has proven to moderate CNN computational load successfully. For the second challenge, various evolutionary algorithms have been proposed thus far. The algorithm proposed in this paper can be viewed as a solution to both challenges. Instead of using constant predefined criteria to evaluate the filters of CNN layers, the proposed algorithm establishes evaluation criteria in online manner during network training based on the combination of each filter's profit in its layer and the next layer. In addition, a novel method has been suggested that inserts new filters into the CNN layers. The proposed algorithm is not simply a pruning strategy but determines the optimal number of filters. Training on multiple CNN architectures allows us to demonstrate the efficacy of our approach empirically. Compared to current pruning algorithms, our algorithm yields a network with a remarkable prune ratio and accuracy. Despite the relatively high computational cost of an epoch in the proposed algorithm in pruning, altogether it achieves the resultant network faster than the other algorithms.

1. Introduction

In the recent years, deep learning networks, specifically convolutional neural networks (CNNs), have achieved unprecedented success in challenging problems in vision such as classification [1, 2], face detection [3, 4], semantic segmentation [5, 6], object detection [7, 8] and all other data-rich fields such as natural language understanding [9] and speech detection [10]. This success is attributed to the hierarchical structure of CNN that was inspired by the human brain structure [11, 12]. The success of CNNs in a variety of applications is accompanied by a substantial rise in computation and parameter storage costs. A glance at network architectures such as AlexNet (8 layers) [13], VGG (18 layers) [14], GoogleNet (19 layers) [15], ResNet [16], and DenseNet [17] (a few hundred layers) reveals that they have gotten wider and deeper over time. This strategy results from the well-known rule that, in general, CNNs with greater depth and breadth are better equipped to handle complex and larger problems [16, 18]. With the advent of wider and deeper CNNs, however, the hardware requirements have elevated in a way that makes it difficult to use CNNs in edge computing. Even though it is well-known that deep neural networks have many redundant parameters that can be replaced by a more compact architecture, there is no standard method for designing the compact deep architecture for new tasks. Consequently,

many researchers tend toward network pruning and compression [19, 20]. A common practice in these studies involves comprehensive training of the network, followed by repetitive pruning (based on some predefined criterion) and fine-tuning. The essence of pruning is evaluating filters and removing the ones with the lowest score that result in minimum accuracy loss and maximum acceleration.

Typically, the input to the pruning algorithm is a pre-trained network. This being said, one must initially train the network exhaustively in order to employ these algorithms. In addition, these algorithms typically require remarkable fine-tuning epochs after pruning. In our proposed algorithm, pruning occurs during training, eliminating the requirement for a pre-trained network. However, our algorithm requires approximately *n* epochs of startup training before the pruning process can initiate. In the proposed algorithm, the number of fine-tuning epochs is approximately 0.1 times that of extant pruning algorithms. By eliminating the requirement for an entirely pre-trained network and a significant fine-tuning phase following the pruning phase, the proposed algorithm attains the resulting network faster than existing pruning algorithms.

In the current pruning algorithms, if pruning some filters and reducing the capacity of corresponding layers result in a significant loss in the network performance, there is no way to undo the pruning. Under such circumstances, irrecoverable information loss occurs within the network. Several methods such as soft pruning [22], dynamic regularization [23], and filter attenuation [24] are recommended for mitigating these conditions. In order to fully address this problem, we have proposed a novel method that measures the layer's capacity and, if necessary, inserts a new filter into that layer. Therefore, we refer to our algorithm as Adaptive Network pruning (AdapNet). Inserting a new filter that has a specific correlation with other filters also accelerates network training because stochastic gradient descent (SGD) is utilized for training the network on what it will learn in the future.

Filter evaluation methods in pruning algorithms can be categorized into two main types: (1) evaluation based on filter intrinsic property such as L1/L2 norm [25, 26], absolute value [27, 28], gradient value [29, 30], and entropy [31-33]; and (2) evaluation based on filter influence on network cost function [34-38]. The first approach is based on the outdated but currently contested notion that "magnitude equals salience" [39-41]. The second method typically has a high computational cost and is insufficiently accurate. The proposed algorithm's filter evaluation criterion is developed based on the intended task during network training. We have considered image classification as the intended task. Future research may consider the development of AdapNet for other tasks.

Filter evaluation in our algorithm is based on intermediate representations that emerge in the network layers during training. CNNs' intermediate layer function as semantic detectors, and these semantics are extremely sensitive to network architecture and training data [42-45]. We argue that intermediate-layer generated semantics can be used to determine the optimal number of filters for each layer. As the network's hierarchical structure is inspired by the human brain, it stands to reason that the number of filters in each layer may also be [11, 46]. Accordingly, we define the Intermediate Concept (IC) as the interpretability and information richness of the generated Feature Maps (FMs) in intermediate layers. Here, interpretability refers to the degree to which FMs correspond to the concept trained to the network. We introduce a quantitative measurement based on the distribution of filter activation on IC to quantify the interpretability of intermediate layer FMs. By interpretability, we do not mean interpretability for humans, as we know that in an optimal state, only a small percentage of FMs (approximately 40%) that are valuable to discriminability is interpretable for humans [40, 47]. Thus, we can criticize approaches such as [45, 48, 49] that rely on human interpretation in determining the IC to be learned by the network. We propose a novel method for extracting ICs from FMs and for quantitatively evaluating the extracted ICs' utility for the intended task.

In sum, unlike the existing pruning algorithms, AdapNet does not require a pretrained network and eliminates the substantial fine-tuning phases following pruning. In addition, it does not utilize the static filter evaluation criterion during pruning; and, it inserts a new filter based on the layer capacity evaluation. AdapNet enhances the prune ratio while maintaining the accuracy of pruned networks and is significantly faster than the existing pruning algorithms. The proposed approach has been evaluated on multiple CNNs (VGG11, ResNet50, AlexNet, Net2) over two benchmarked datasets (CFAR10/100). The simulation results demonstrate the effectiveness of AdapNet in terms of prune ratio and acceleration

in achievement of resultant network. Our primary contributions are as follows:

1. AdapNet input is not a trained CNN, and unlike other methods, the CNN does not require exhaustive training prior to pruning.
2. The evaluation criteria for filters are not static and task-independent. Rather, these criteria are developed based on the extracted IC during network training.

3. AdapNet inserts new filters into the layers based on layer capacity evaluation, as opposed to the existing methods that only prune the filters and may cause irrecoverable information loss. AdapNet is not, therefore, simply a pruning algorithm, as it determines the optimal number of filters for each layer.

2. Related Works

CNN pruning and compression is a method of finding the optimal CNN architecture in terms of network size or complexity. These methods fall into two primary categorizes with regard to filter evaluation and pruning technique: (1) intrinsic properties and (2) performance importance. In the first category, the significance of a filter is calculated by its intrinsic properties. On the other hand, a score is assigned to the filter based on its intrinsic properties, and filters with low scores are removed from the network. L1/L2 norm [25, 26], absolute value [27, 28], gradient value [29, 30], and filter entropy [31-33] are examples from the first category. Research indicates a weak correlation between L1/L2 norm-based approaches and network performance [39, 40]. In this category, the association between selected parameters (for filter evaluation) and network performance is typically validated through simulation with no formal proof.

The second group is based on the significance of the performance. In these methods, the impact of the filter on network performance is evaluated, and pruning is performed in accordance with the calculated filter importance. For example, [50, 51, 37] inserted the auxiliary loss function into the intermediate layers and pruned based on the discrimination power of the FMs. However, research indicates that increasing the discrimination power of FMs in intermediate layers does not necessarily increase the discrimination of the FMs in the final layer but may even harm generalization [52-54]. The Taylor series was used in [34-36] to measure the effect of each filter on the loss function or network accuracy. Due to the large number of filters and the impossibility of calculating the Taylor series with reasonable accuracy, however, a more compact approximation was calculated using the first-order expansion. In general, measuring the impact of filters on network performance typically entails high computational costs and cannot be accomplished accurately.

Millions of parameters and computations in the network's intermediate layer extend the usefulness of the network beyond the task they were designed for and reach features that are not necessarily useful for the network's intended task [52]. On the other hand, intermediate layers of CNN behave like a semantic detector (we refer to as IC), which is sensitive to the network structure [42, 44, 45, 52]. [45] uses the evolutionary method to extract these ICs. In [48], the user identified ICs that help the network to achieve greater generalization and accuracy during training. [55, 56] extracted a single prototype for each class, whereas [57] utilized the same number of prototypes for each class. We argue that based on these ICs, accurate evaluation of filters is possible while taking the intended task into consideration.

AdapNet applies incremental clustering to FMs for IC extraction. Clustering is commonly used to compress CNNs. In [58, 59], filter clustering is used as the similarity criterion for pruning; in [59], filter pruning was performed online, whereas in [58], filter pruning was performed after training. Using the trained network as input, [58] employed k-means clustering on the kernels, utilized cluster centers as new kernels, and eliminated the remaining cluster members from the network. [59] applied clustering during training and, with modifications to the cost function, attempted to merge similar kernels into the same clusters.

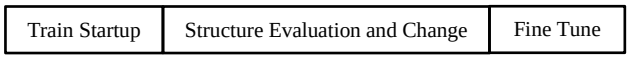
Nevertheless, online clustering methods receive two major criticisms. Firstly, the high dimensionality of data not only increases the computational time and memory requirements of algorithms but also negatively affects their performance due to the noise effect and the insufficient number of samples with respect to the ambient space dimension, commonly referred to as the curse of dimensionality. Secondly, one cannot decide between filters solely based on their similarity, as the information of each filter can be used in subsequent layers depending on its composition [60, 61]. Instead of being uniformly dispersed across the ambient space, high-dimensional data frequently lay in low-dimensional structures. Recovering low-dimensional structures in the data reduces the

computational cost, memory requirements of algorithms, and reduces the effect of high-dimensional noise in the data, leading to improved performance of inference, learning, and recognition tasks [62, 63]. [62, 64, 65] applied CNN to clustering in another sub-space. AdapNet utilizes the idea proposed in these papers. Similar to [64], a non-linear mapping into another sub-space was used to extract an adjacency matrix for use in spectral clustering. The cluster evaluation method is inspired by [44], which presents the kernel activity distribution on labels to extract the sensitivity of each kernel. It assesses each CNN convolutional unit as a solution to a binary segmentation task for each visual concept in BRODEN (Broadly and Densely Labeled Dataset assembled and applied in [44]).

Another common issue with network compression algorithms is the need for a CNN that has been pre-trained. In many previous approaches, the accepted strategy has been "pruning followed by training", which imposes an additional retraining phase to recover the accuracy degradation caused by pruning. The simplest algorithm that takes into account this issue and eliminates the fine-tuning phase is [66]. It randomly applied three compression criteria during network training: l1-norm, random selection, and filter activation. In other words, filter pruning was performed without a specific metric and based solely on one of the aforementioned criteria. In light of the simulation results presented in [66], the l1-norm is the optimal network compression criterion. In [67], an online assessment algorithm was proposed that selected a set of criteria before and during training based on cost function distribution on a selected criteria space, with specific criteria for filter pruning being selected in each layer. Online network compression was performed in [39, 68] with no regard to the network's cost function. In [39], random gates were used to prune the filter channels until the intended FLOP was reached. In [68], scale factors were introduced and used during training to reduce filter values selectively and to ultimately eliminate them from the network. Two distinct modules were presented in [69], and the algorithm alternates between pruning and recovery. However, [69] does not eliminate the fine-tuning phase; rather, it combines the fine-tuning phase with pruning.

3. Proposed Algorithm

As depicted in Figure 1, we divided network training into three distinct phases. During the train startup phase, network initialization and partial convergence of filters to the intended task occur. In the second phase, which we refer to as the "Structure Evaluation and Change Phase," AdapNet execution commences. Upon the termination of the second phase, the third phase commences, during which the network's structure is maintained, and fine-tuning occurs. In the first and third phases of all simulations, we employ 50 and 10 epochs, respectively.



All training epochs

Figure 1. Division of training in AdapNet

The input to AdapNet is a CNN with n layers and with k_i filters in layer i. During network training, ICs are extracted and filter evaluation is performed; decisions are subsequently made regarding the removal, merge or insertion of new filters in each layer. Only during the "Structure Evaluation and Change" phase do layer structure modifications and the number of filters increase or decrease. We introduce "Layer Structure Evaluation and Change Period" (LSEP) as a set of α consecutive batches. At each LSEP, evaluation and structure change are performed on each layer. After each evaluation and potential change in the layer structure, training continues with the number of LSEP batches without any changes in the network structure; this is illustrated in Figure 2.

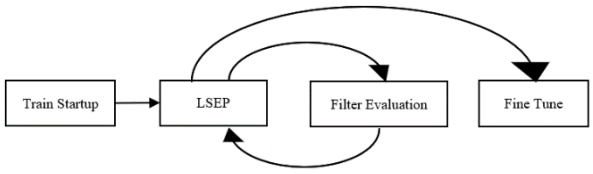


Figure 2. Training process phases in AdapNet

If fewer than k filters are pruned during n consecutive LSEPs, AdapNet terminated and the fine-tuning phase initiated. In all simulations, we set *n=2* and *k=3*. However, in extreme circumstances, we can set *k=0*. Figure 3 portrays the AdapNet filter evaluation phase pipeline that will be described in the following sections.

In each layer *l*, the activation distribution matrix of the filter is represented as $FMDist_{FC}^{l}$, where *F* is the number of filters and *C* is the number of extracted ICs in layer *l*. The matrix $FMDist_{FC}^{l}$ provides AdapNet with instructions for removing/merging/inserting layer *l* filters. The matrix $FMDist_{FC}^{l}$ is derived from $RFMDist_{FC_t}^{l}$ (Raw *FMDist*). The only difference between *FMDist* and *RFMDist* is in the number of

columns. *FMDist* selects several columns from *RFMDist* (in implementation, only *RFMDist* is maintained alongside an array of selected columns index, which introduces the *FMDist*. To keep the text straightforward, we refer to these two matrixes separately).

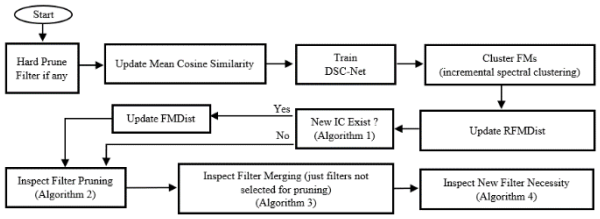


Figure 3. AdapNet pipeline of filter evaluation phase.

C_t is the total number of recognized clusters in layer l, while C is the total number of recognized ICs in that layer. As illustrated by (1), we define P_{ij} as the normalized fire probability of filter i for concept j.

$$P_{ij} = FMDist[i,i] = FMDist[i,j] / \sum_{i} FMDist[i,j] \quad (1)$$

Consequently, the row i of $FMDist_{FC}^{l}$ demonstrates the activity distribution of the fitter i on the recognized ICs in layer l. *FMDists* and *RFMDists* receive training updates until the second phase concludes. Changes to the filter quantity of each layer will cause these two matrices to be updated in accordance with the layer's structural changes. If new clusters are identified in LSEP, new columns corresponding to them will be added to the *RFMDist*, and if new ICs are identified from newly inserted clusters, new columns will be added to the *FMDist*. Section 3.1 explains the algorithm used for recognizing the IC from *RFMDist* clusters (columns) that are added to the *FMDist* (selected as the *FMDist* column). If ICs in *FMDist* columns are deemed useless during network training, they will be removed at the end of each epoch. In the *FMDist* column, valueless ICs are detected with a partially uniform distribution.

After new filter insertion or pruning/merging, the number of rows in *FMDist/RFMDist* increases or decreases in each LSEP. When the two filters P_{ix} and P_{jx} merged, the corresponding rows in *RMFDist* were removed and a new row corresponding to the new filter z with the value $P_{zx} = (P_{ix} + P_{jx})/2$ was inserted into *RMFDist*.

3.1. Intermediate concept detection

We argue that during network training, FMs are derived from the union of low-dimensional sub-spaces, which we refer to as the IC. We must learn (in an unsupervised manner) an explicit non-linear mapping of the FMs that is well-suited to a sub-space we refer to as the IC sub-space. We employ the method proposed in [64, 65], namely the subspace clustering network, to accomplish this. Deep autoencoders with a self-expressive layer between the encoder and the decoder are proposed in [64, 65] to mimic the self-expressiveness property of data. We use the self-expressiveness of FMs to generate an affinity matrix and employ it in spectral clustering, which leads to the emergence of an IC on the fly during network training.

In (2), the determination of the self-expressiveness coefficient matrix (C) is formalized as an optimization problem [65].

$$L(\theta, C) = \frac{1}{2} \| X - X'_{\theta} \|_F^2 + \lambda_1 \| C \|_p + \frac{\lambda_2}{2} \| Z_{\theta_e} - Z_{\theta_e} C \|_F^2 \quad s.t.(diag(c) = 0) \quad (2)$$

where X'_{θ} represents the data reconstructed by the auto-encoder and $||Z_{\theta_e} - Z_{\theta_e} C||_F^2$ is the self-expressiveness term. Ideally, $C_{ij} \neq 0$ only if the corresponding data points X_i and X_j are drawn from the same sub-space (that we refer to as the IC) [62]. As such, we can leverage the self-expressiveness coefficient matrix C to construct the affinity matrix for spectral clustering.

We employ the network architecture proposed for the COIL100 dataset in [65], namely DSC-Net, due to its generality and larger capacity. Using buffered FMs in the layer, the DSC-Net was trained for two epochs during each LSEP. The network learns partially non-local representations in the interior layers [70]. Hence, we utilize the layer output FMs as a training dataset for DSC-Net. DSC-Net training is conducted in every LSEP; therefore, for each DSC-Net training, we will have m training samples, as shown in (3).

$$m = LSEP \times number\ of\ \ filters\ in\ layer \times number\ of\ \ samples\ in\ each\ batch \quad (3)$$

After training DSC-NET, the affinity matrix is extracted for use in spectral clustering, as posited in [65]. In light of the fact that all training data (FMs) are unavailable in each LSEP, we use incremental k-means clustering [71] in spectral clustering eigenspace to preserve the clustering result of previous steps.

Based on the perturbation theory and spectral graph theory, the eigengap heuristic is suggested to calculate the optimal number of clusters [72,

73]. The optimal number of clusters can be determined when the Laplacian (affinity) matrix is approximately block-diagonal, which restricts the Laplacian's eigenvalue spectrum. The objective is, therefore, to select the number of clusters k in each layer such that all eigenvalues λ_1, λ_2, …, λ_k of the Laplacian matrix are extremely small but λ_{k+1} is relatively large. Each cluster center has corresponding columns in RFMDist, and RMFDist[i,j] represents the frequency with which filter i is clustered in cluster j. Each cluster resulting from spectral clustering is a candidate for the IC.

AdapNet prevents the overlap of ICs by utilizing the IC's boundary point as its margin during the ICs extraction process. To find IC boundary points, we employ the method proposed in [74]. Boundary points are data points at the margin of densely distributed data, such as a cluster. [74] has proposed the BORDER (BOundaRy points DEtectoR) method for detecting such points. BORDER utilizes the advanced database technique Gorder kNN join and the unique property of the reverse k nearest neighbor (RkNN) to find boundary points.

Algorithm 1 presents the new IC detection process.

Algorithm 1. Intermediate Concept Detection

Input:
//columns of FMDist
C ={cluster centers currently selected as the IC in layer l}
//cluster centers that are candidate to be intermediate concepts
S = {$\mathbf{C_t}$} – {**C**}
Foreach(C_i in **S**){
  if(chi-square($RMFDist_{*C_i}$) < β){
    Find $S_j \in$ **C** such that $||S_j - C_i||$ is min.
    k= sqrt(|{feature maps clustered in S_j}|)
    For all feature maps clustered in S_j calculate RkNN set.
    Border_Point ={ points that have 20% min(|RkNN|) }
    if($||C_i - \{Border\_Point\}|| > 0.75\ ||S_j - \{Border\_Point\}||$)
      {insert C_i to **C**} }

Any cluster will be selected as an IC if (1) its filter activation distribution is not uniform and (2) the distance between its center and the boundary points of the nearest IC is greater than 75% (we reach empirically to this number) of the distance between the center of the nearest IC and its boundary points. We utilize chi-square distribution to measure the uniformity of filter activation in *FMDist*, and parameter β is defined as a threshold for detecting non-uniformity.

3.2. Filter pruning and merging

The invariance in a deep neural network is equivalent to the minimality of the representation it computes. Therefore, minimizing the network layer by eliminating or merging ineffective filters reduces the network's computational and maintenance costs and increases its generalization [75]. Our intent in filter pruning and merging is to minimize the network layer size while maintaining the desired accuracy threshold.

Two criteria that can be used to evaluate each layer's filters are their relative information and diversity. These two criteria align with the concept presented in determinantal point processes [42, 76]. Consequently, we consider redundancy in the following two cases: 1. A filter is relatively valueless in its layer and for the successive layers, and 2. Other filters can mimic a filter's functionality in the subsequent layers. These two criteria must be evaluated independently. To evaluate filter value based on the first criterion, we examine the filter activity distribution in FMDist. Thus such filters are candidates for pruning if there is no statistically significant filter activity distribution on ICs based on the chi-square test and if the distribution is partially uniform. However, individual filters within and across layers play different roles in the network. Therefore, in addition to activity distribution, we measure the importance of the candidate filter from the perspective of the subsequent layer before pruning it. In order to accomplish this, we employ kernel sparsity and entropy (KSE) [61], as an indicator that represents the sparsity and information richness of FMs in a feature-agnostic manner. However, here only the information richness of the selected feature map from the perspective of the subsequent layer is essential; therefore, we set the sparsity parameter s_c in the KSE indicator (v_c) to 1 in (4).

The KSE indicator determines whether a particular filter transmits valuable data to the subsequent layer. Thus, if the KSE of a pruning candidate filter is below a certain threshold (we empirically reach a 60% KSE threshold), it will continue to be considered a pruning candidate.

Inspired by [22], we employ soft filter pruning to avoid sudden model capacity reduction and unrecoverable information loss. Consequently, the intended filter will not be updated during the subsequent LSEP and will be physically removed (hard pruned) at the beginning of the subsequent LSEP. Algorithm 2 describes the identification of candidate filters for pruning.

$$v_C = \sqrt{\frac{s_C}{1+\alpha e_C}} \quad where\ \alpha = 1, s_C = 1 \tag{4}$$

$$e_C = -\sum_{i=1}^{N} \frac{dm(w_{i,c})}{d_C} \log_2 \frac{dm(w_{i,c})}{d_C}$$

$$dm(W_{i,c}) = sum\,of\ similarity\ for\ \ker nel\, W_{i,c}\,[61]$$

$$d_C = \sum_{i=1}^{N} dm(w_{i,c})\ \ d_C\ used\ to\ normalize\ dm$$

$$N = number\,of\ \ filter; W_{i,c} = channel\,c\,of\ \ filter\,i$$

Algorithm 2. Filter Pruning

Input:
FMDist^l = layer FMDist matrix
Γ_1 = KSE threshold
Foreach (row i in FMDist) {
if(chi-square(FM_i) > β and KSE(FM_i) < Γ_1){
Select FM_i as a candidate filter to freeze in the next L and hard prune afterward}}

On the basis of the second criterion, cosine similarity and filter activity distribution are utilized to select merging candidates. The cosine similarity matrix of each layer's FMs is calculated until the end of the second phase of network training. This matrix is reset in every LSEP and always contains the average cosine similarity of each layer's FMs. As merging candidates, filters with a cosine similarity greater than the specified threshold and a similar distribution in FMDist will be identified. In merging, the cosine similarity threshold is set at twice the average cosine similarity. Moreover, the Chebyshev distance is used to measure the differences in distribution between two marked filters. Two merged filters are removed from the network, and a new filter with the average of their values is added to both the network and RFMDist. The merging of filters is described in Algorithm 3.

Algorithm 3. Filter Merging

Input:
FM_j = Candidate filter to merge
Γ_3 = Chebyshev Distance threshold
Γ_2 = 2× mean_cosine_similarity//Cosine Similarity threshold
Foreach (row i in FMDist^l) {
if(Mean Cosine Similarity(FM_i,FM_j) > Γ_2 and Chebyshev Distance(FM_i,FM_j) < Γ_3)
{FM_i and FM_j merged}}

3.3. New filter insertion

FMDist must be diagonal in the optimal state. If this is the case, its rows are linearly independent and span the space of each layer's IC. If we identify a semi-block diagonal matrix similar to FMDist, then the block size and the number of non-zero items in the off-block diagonal item of that matrix can be utilized to determine whether or not the existing filters can code ICs in corresponding layer. Because of the high processing cost, it is not plausible to identify a similar semi-block diagonal matrix with an unknown number/size of blocks during network training. Therefore, we suffice to Algorithm 4 for determining whether or not a new filter is required in a specific layer. Our intuition suggests that 70% of each filter's activation should be focused on 30% of detected ICs in each layer; otherwise, the deviation value reveals the demand score for a new filter. To accomplish this, we use the largest gap in the normalized value of each filter activation in FMDist and divide the ICs of each filter into active and inactive groups. For each filter, two parameters are calculated, the positive value of which indicates the demand for a new filter. The parameter *P1* monitors the number of elements in the active set that must be smaller than 30% of all ICs, whereas the parameter *P2* represents the total score in the active set that must be greater than 0.7. We empirically determine these numbers and incorporate them into Algorithm 4.

Algorithm 4. Determine the necessity for new filter insertion to layer l

Input:
FMDist^l = FMDist matrix of layer l with normalized rows
n=number of intermediate concepts (column count of FMDist)
$P1_t = P2_t = 0$
foreach (row i in FMDist^l) {
Sort row items of FMDist and use the largest gap as the divider to split row items into an Active and Inactive set.
$P1 = (|\mathbf{Active}|/n) - 0.3$
$P2 = 0.7 - \sum_i Active_i$
P1=P1 / ((P1>0)? 0.7: 0.3) //*normalize P1 in [-1,0] or [0,1]*
P2=P2 / ((P2>0)? 0.7: 0.3) //*normalize P2 in [-1,0] or [0,1]*
$P1_t$=$P1_t$+P1
$P2_t$=$P2_t$+P2}
if($P1_t$>0 or $P2_t$>0) {insert new filter to layer l}

New filter initialization builds on the results reported in [77]. One filter with *P1>0* or *P2>0* is randomly selected as the master filter. Based on a specific correlation with the master filter, a new filter is generated. The network is divided into four sections to determine the correlation type. The position of the intended layer determines the correlation type between the new filter and the master filter. According to [77], the correlation between the new filter and the master filter for the first quarter layers (the first section) is inverse, whereas a rotary correlation holds in the second quarter. Scaling and translational correlation are the topics of the next two sections. During the next two LSEP, the new filter will not become a pruning/merging candidate.

4. Simulation Results

We evaluate AdapNet using four network architectures: VGG11 [14], AlexNet [13], ResNet50 [16], and a network architecture displayed in Table 1 that inspired by [56,78], which we dub as Net2. The Net2 architecture adheres to the design philosophy of the VGG network.

Table 1. Initial Architecture of Net2.

Type	Kernel	Output
Conv,BatchNorm,ReLU	5×5	256×60×60
Conv,BatchNorm,ReLU	5×5	256×56×56
Conv,BatchNorm,ReLU	5×5	256×52×52
Conv,BatchNorm,ReLU	3×3	256×50×50
Conv,BatchNorm,ReLU	3×3	256×48×48
Conv,BatchNorm,ReLU	3×3	256×46×46
Conv,BatchNorm,ReLU	3×3	256×44×44
Conv,BatchNorm,ReLU	3×3	512×42×42
Conv,BatchNorm,ReLU	3×3	512×40×40
Conv,BatchNorm,ReLU	3×3	512×38×38
Conv,BatchNorm,ReLU	3×3	512×36×36
Conv,BatchNorm,ReLU	3×3	512×34×34
MaxPool	2×2	512×17×17
Linear,BatchNorm,ReLU	-	147968×512
DropOut,Liner,BatchNor	-	512×512
ReLU,Linear	-	512× #class

Simulations on two benchmark data-sets, the mainstream dataset CIFAR-10 [79] and the more challenging dataset CIFAR-100 [79] demonstrate that our method outperforms the current methods. CFAR10 contains 32×32 images from 10 classes, 50,000 training images, and 10,000 test images, whereas CFAR100 is identical to CFAR10 except that it contains 100 classes.

Except for the number of startup epochs, which is 20 for Net2 (due to the naive architecture of Net2, 20 epochs were used for it; increasing this parameter did not improve the result) and 50 for other networks, all simulations were conducted with the same hyperparameter settings, as summarized in Table 2.

Table 2. AdapNet Hyperparameters.

β	initialize to 0.9, at each 5 epochs decrease 0.06 until Min. value: 0.18
Γ_1	(KSE threshold) 0.6
Γ_3	(Chebyshev distance threshold) 0.8

We employ stochastic gradient descent with the following parameters: a batch size of 128, weight decay of 0.01, a momentum of 0.8, and LSEP of 150. Every 15 epochs, the initial learning rate of 0.025 is divided by 2. Training is performed on an Nvidia rtx3090, and gradient accumulation is utilized due to GPU memory limitations. Instead of updating the network weights after each batch, gradient values are saved, the next batch is processed, and the new gradients are added. The weight update is then performed only after the model has processed multiple batches.

The results of training Net2 with AdapNet are displayed in Table 3.

Table 3. Result of training Net2 with AdapNet.

(Top1: the model answer must be exactly the expected answer, Top2: any of your model 2 highest probability answers must match the expected answer)

Dataset	CFAR10	CFAR100
Train Acc (%)	82	79
Top1 Acc. (%)	69	64
Top2 Acc. (%)	84	73
Prune Ratio (%)	88	89
Num of Epoch (#)	50	41

Figure 4 depicts the architecture of Net2 after training with AdapNet. On the basis of the Net2 architecture after AdapNet pruning (Figure 4), it is evident that the pruning ratio decreases as we approach the last network layer.

Type	Kernel	Output	Type	Kernel	Output
conv	5*5	64/64/129	conv	5*5	64/64/127
conv	5*5	64/64/142	conv	5*5	64/64/198
conv	5*5	64/64/173	conv	5*5	64/64/214
conv	3*3	32/32/173	conv	3*3	32/32/214
conv	3*3	32/32/214	conv	3*3	32/32/251
conv	3*3	32/32/236	conv	3*3	32/32/249
conv	3*3	32/32/236	conv	3*3	32/32/255
conv	3*3	32/32/298	conv	3*3	32/32/327
conv	3*3	32/32/318	conv	3*3	32/32/347
conv	3*3	32/32/318	conv	3*3	32/32/347
conv	3*3	32/32/324	conv	3*3	32/32/452
conv	3*3	32/32/341	conv	3*3	32/32/468

(a) (b)

Figure 4. Architecture of Net2 After training with AdapNet on CFAR10 (a), CFAR100 (b).

This indicates the extraction of numerous ICs as we progress to the deeper network layers, given the initial architecture of Net2 (Table 1), which consists of an equal number of filters in all network layers. It is known that CNNs extract new abstract features based on previously extracted ones. Consequently, more valuable ICs were extracted by AdapNet in the deeper layer as is evident from the simulation results of Net2.

Tables 4, 5, and 6 present the results of training VGG11, AlexNet, and ResNet50 with AdapNet, respectively. The number of epochs in the "*Structure Evaluation and Change Phase*" (AdapNet's second phase) is indicated in column "Num of Epoch". This number must be increased by 60 epochs to account for startup and fine-tuning.

Table 4. Result of training VGG11 with AdapNet.

Dataset	CFAR10	CFAR100
Train Acc. (%)	93	87
Top1 Acc. (%)	88	85
Top2 Acc. (%)	92	90
Prune Ratio (%)	87	79
Num of Epoch (#)	53	49

Table 5. Result of training AlexNet with AdapNet.

Dataset	CFAR10	CFAR100
Train Acc. (%)	86	82
Top1 Acc. (%)	84	79
Top2 Acc. (%)	89	81
Prune Ratio (%)	81	78
Num of Epoch (#)	52	32

Table 6. Result of training ResNet50 with AdapNet.

Dataset	CFAR10	CFAR100
Train Acc. (%)	92	91
Top1 Acc. (%)	89	87
Top2 Acc. (%)	94	89
Prune Ratio (%)	89	59
Num of Epoch (#)	49	29

5. Discussion and Analysis

- It is difficult to compare the results of pruning algorithms because there is no standard methodology [21]; AdapNet is no exception. However, AdapNet's prune ratio falls within an acceptable range. Tables 7, 8, and 9 provide a comparison of AdapNet results for VGG11, ResNet50, and AlexNet on CFAR10 with those of other algorithms.
  Based on comparisons summarized in tables 7, 8, and 9, AdapNet has a remarkable prune ratio and accuracy relative to other algorithms.
- Due to the high computational cost of AdapNet's epoch (during pruning) in contrast to other algorithms. Thus, relying only on the number of epochs is not a proper measure to compare the speed of AdapNet with others. However, the epoch number of AdapNet is about four times less than other algorithms (Table 7). Extant pruning algorithms require a completely pre-trained network at the outset and extensive fine-tuning epochs at the conclusion. AdapNet requires only a small number of epochs during startup and fine-tuning. In all simulations, we used 50 (20 for Net2) epochs at the startup phase and only 10 epochs in the fine-tuning. So, overall AdapNet achieves the resultant network faster than other algorithms.

Table 7. Comparison of AdapNet and some other algorithms for VGG11 on CFAR10.

Algorithm	Epoch	Batch size	Prune ratio(%)	Acc. (%)
HRank [80]	330	128	82	92
LDFM [81]	160	128	87	93
Zaho *et al.* [82]	300	256	73	93
SSS [83]	240	64	73	93
AdapNet	**53**	128	**87**	92

Table 8. Comparison of AdapNet and some other algorithms for RestNet50 on CFAR10.

Algorithm	Epoch	Batch size	Prune ratio (%)	Acc. (%)
HRank [80]	for each layer, retrain for 30 epochs after pruning	128	68	93
NISP [84]	50 epochs fine tuning required after pruning	-	87	93
He *et al.* [85]	150	128	-	91
Li *et al.* [61]	220	128	65	76
AdapNet	**49**	128	**89**	94

Table 9. Comparison of AdapNet and some other algorithms for AlexNet on CFAR10.

Algorithm	Epoch	Batch size	Prune ratio (%)	Acc. (%)
NISP [84]	90 epochs fine tuning required after pruning	-	75	85
DENNC [86]	40 epochs fine tuning required after pruning	-	40	12% decreased
AdapNet	**52**	128	**81**	89

- Filter evaluation methods in the existing pruning algorithms can be categorized into two approaches: (1) evaluation based on filter value in its layer, and (2) evaluation based on the filter's utility for the next layer. Given that both approaches are valid, the filter in its layer has a particular value according to the information it extracts. It also has a

special value according to the information extracted by it that the next layer uses. In AdapNet novel method is suggested, which is the combination of these two approaches. In this way, the filter evaluation method of AdapNet is more precise than the current pruning algorithm. The accuracy of AdapNet in filter evaluation led to the remarkable prune ratio and high accuracy of the resultant network.

- In addition to filter pruning, AdapNet merges similar filters using cosine similarity. On the other hand, if two filters have similar content and stimulations on ICs, regardless of filter evaluation, AdapNet merges them and generates a single filter. This allows AdapNet to achieve a higher pruning ratio than competing algorithms. To test the impact of the merge module in AdapNet, we disabled it and ran AdapNet without merging on VGG11. Table 10 presents the results of this simulation. A notable issue is that the prune ratio decreased by approximately 6%. In other words, nearly 6% of AdapNet's pruning power can be attributed to filter merging.

Table 10. Result of AdapNet on VGG11 without filter merging.

Dataset	CFAR10	CFAR100
Train Acc (%)	95	86
Top1 Acc. (%)	89	84
Top2 Acc. (%)	93	91
Prune Ratio (%)	80	75
Num of Epoch (#)	60	53

- AdapNet does not immediately prune filters with a low score that it identified. These filters are frozen during the subsequent LSEP, and their information is moved to other filters. Lastly, they are pruned at the beginning of the subsequent LSEP. In this way, AdapNet prevents irrecoverable information loss caused by filter pruning, a prevalent problem in many extant algorithms. This approach maintains the network's accuracy during training, and the resulting network is more accurate than other algorithm outcomes. On the basis of simulation results, the effectiveness of filter freezing is evident. To test it, we turned off filter freezing and ran AdapNet on VGG11. The outcome of this simulation is provided in Table 11. Based on this simulation, the prune ratio remains nearly unchanged, whereas network accuracy decreases by approximately 3%. This result demonstrates the effect of soft pruning on the transfer of knowledge from candidate pruning filters to other filters prior to hard pruning.

Table 11. Result of AdapNet on VGG11 without soft pruning.

Dataset	CFAR10	CFAR100
Train Acc (%)	94	87
Top1 Acc. (%)	85	86
Top2 Acc. (%)	90	87
Prune Ratio (%)	87	80
Num of Epoch (#)	54	48

- AdapNet evaluates the required capacity of each layer according to the extracted ICs on that layer. If filters of the layer are not sufficiently able to discriminate extracted ICs of that layer, a new filter with a specific correlation is inserted into that layer. The insertion of a new filter with a specific correlation accelerates training and enhances the network's accuracy. Our intuition is that the new filter correlation is what the network will learn with stochastic gradient descent in future epochs; however, we provide it to the network early enough. This strategy accelerates AdapNet achievement for the final network. To test the effects of the new filter insertion, we disabled the filter insertion module and ran AdapNet on VGG11. The results of this simulation are presented in Table 12. The network's accuracy and prune ratio decreased according to simulation results, while the number of epochs increased. This finding can be interpreted as AdapNet ignoring the pruning of valueless filters due to high KSE and non-uniform distribution on ICs as a consequence of not inserting new filters.

Table 12. Result of AdapNet on VGG11 without new filter insertion.

Dataset	CFAR10	CFAR100
Train Acc (%)	93	94
Top1 Acc. (%)	88	89
Top2 Acc. (%)	91	90
Prune Ratio (%)	81	74
Num of Epoch (#)	61	68

- AdapNet trains the DSC-Net network in two epochs for each LSEP in order to extract the affinity matrix. Spectral clustering is performed on the basis of the

extracted affinity matrix, and AdapNet must retain the clustering result and RFMDist matrix until the algorithm's termination. Therefore, compared to other methods, AdapNet requires more memory and processing power. Nevertheless, the high execution cost of AdapNet is justifiable in light of the necessity of a pretrained network and the large number of fine-tuning epochs required by other algorithms.

- Based on the simulation results, there is a minor difference between train and test accuracy in the AdapNet resultant network, which may suggest that AdapNet precludes overfitting during pruning. This issue is related to the independence of AdapNet from backpropagation. AdapNet operates based on the extracted deep representation that we refer to as ICs. At the beginning of each epoch, worthless ICs are detected and eliminated from FMDist based on the filter's activation distribution on ICs. AdapNet maintains the proper distance between extracted ICs and discourages overlap based on extracted IC boundary points. Therefore, AdapNet prevents overfitting during pruning by eliminating ineffectual ICs and preventing IC overlap.
- When the training dataset contains numerous classes, multiple clusters and ICs are identified. As a result of the extraction of numerous useful ICs, the prune ratio is low, as demonstrated in Tables 4 and 6 for CFAR100.
- There is a tension between batch size and the speed and stability of the learning in model training. Regardless of the effect that the batch size has on network training, different batch size in AdapNet has an impact on ICs extraction. AdapNet utilizes incremental k-means clustering in ICs extraction. If incremental k-means clustering converges to the same solution despite different batch sizes, then the batch size has minor to no effect on AdapNet's result. However, the result of k-means won't necessarily be the same in each run. Depending on the initial centroid and the order of training data k-means will converge to different solutions. Thus, due to the non-deterministic behavior of k-means, the definite and noteworthy effect of batch size in the result of AdapNet is not expected. To inspect we conducted a simulation with different batch sizes that result reported in Table 13. Decreasing the batch size led to an increase in noise effect in k-means clustering and instability in ICs extraction, and this led to a decline in the prune ratio and growth epoch number, as presented in Table 13. Due to the impact of batch size in the training process, the attribution of all variations to ICs extraction of AdapNet maybe not be sound. However, based on simulation result decreasing the batch size to 16 has destructive effect on AdapNet.

Table 13. Result of AdapNet on VGG11 (CFAR10) with different batch size.

Batch size	16	32	64	128
Train Acc (%)	89	93	92	93
Top1 Acc. (%)	84	82	89	88
Top2 Acc. (%)	86	88	90	92
Prune Ratio (%)	63	88	86	87
Num of Epoch (#)	Manually terminated in 100	86	61	53

6. Conclusion and future works

The proposed algorithm is a step toward eliminating trial and error from the CNN architecture design process. We compared our algorithm's results with those of pruning/compression algorithms. In contrast to many current pruning methods, the proposed algorithm operates online and has three significant improvements from existing algorithms: (1) filter evaluation is based on the combination of filter profit in its layer and the next layer; (2) similar filters are merged to compress the layer if certain conditions are met; and (3) network layer capacity is evaluated during pruning, and if a new filter is required, it is inserted into the layer. The simulation results demonstrate the algorithm's effectiveness. Our algorithm achieves a higher prune ratio and accuracy, as well as a quicker network formation.

For a future version of the algorithm, we consider three challenges: (1) determining the optimal filter size based on FMs' entropy; (2) dynamically varying the number of batches in LSEP and, thereby, enhancing the algorithm's speed and its filter evaluation accuracy, and (3) evaluating the impact of the filter in the last network layer.

Conflicts of interest

The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] N. Elyasi and M. Hosseini Moghadam, "Classification of Skin Lesions by Tda Alongside Xception Neural Network," J. AI Data Min., vol. 10, no. 3, pp. 333–344, 2022.

[2] F. Salimian Najafabadi and M. T. Sadeghi, "AgriNet: a New Classifying Convolutional Neural Network for Detecting Agricultural Products' Diseases," J. AI Data Min., vol. 10, no. 2, pp. 285–302, 2022.

[3] R. Ranjan, V. M. Patel, and R. Chellappa, "HyperFace: A Deep Multi-Task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 1, pp. 121–135, Jan. 2019.

[4] H. Filali, J. Riffi, I. Aboussaleh, A. M. Mahraz, and H. Tairi, "Meaningful Learning for Deep Facial Emotional Features," *Neural Process. Lett. 2021*, pp. 1–18, Sep. 2021.

[5] M. Alam, J.-F. Wang, C. Guangpei, L. Yunrong, and Y. Chen, "Convolutional Neural Network for the Semantic Segmentation of Remote Sensing Images," *Mob. Networks Appl. 2021 261*, vol. 26, no. 1, pp. 200–215, Feb. 2021.

[6] J. Guo, J. Yang, H. Yue, H. Tan, C. Hou, and K. Li, "CDnetV2: CNN-Based Cloud Detection for Remote Sensing Imagery with Cloud-Snow Coexistence," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 1, pp. 700–713, Jan. 2021.

[7] X. Zhang, G. Chen, K. Saruta, and Y. Terata, "A Guideline for Object Detection Using Convolutional Neural Networks," *Lect. Notes Electr. Eng.*, vol. 572 LNEE, pp. 157–164, 2020.

[8] BoukercheAzzedine and HouZhijun, "Object Detection Using Deep Learning Methods in Traffic Scenarios," *ACM Comput. Surv.*, vol. 54, no. 2, Mar. 2021.

[9] P. Wang, Q. Wu, C. Shen, A. Dick, and A. Van Den Hengel, "FVQA: Fact-Based Visual Question Answering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 10, pp. 2413–2427, Oct. 2018.

[10] N. Takahashi, M. Gygli, and L. van Gool, "AENet: Learning Deep Audio Features for Video Analysis," *IEEE Trans. Multimed.*, vol. 20, no. 3, pp. 513–524, Mar. 2018.

[11] N. Kruger *et al.*, "Deep hierarchies in the primate visual cortex: What can we learn for computer vision?," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1847–1871, 2013.

[12] Y. Bengio, "Learning Deep Architectures for AI," *Found. Trends® Mach. Learn.*, vol. 2, no. 1, pp. 1–127, Nov. 2009.

[13] KrizhevskyAlex, SutskeverIlya, and H. E., "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[14] S. Liu and W. Deng, "Very deep convolutional neural network-based image classification using small training sample size," *Proc. - 3rd IAPR Asian Conf. Pattern Recognition, ACPR 2015*, pp. 730–734, Jun. 2016.

[15] C. Szegedy *et al.*, "Going deeper with convolutions," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June-2015, pp. 1–9, Oct. 2015.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 770–778, Dec. 2016.

[17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 2261–2269, Nov. 2017.

[18] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[19] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration," *Artif. Intell. Rev. 2020 537*, vol. 53, no. 7, pp. 5113–5155, Feb. 2020.

[20] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the State of Neural Network Pruning?," *Proc. Mach. Learn. Syst.*, vol. 2, pp. 129–146, Mar. 2020, Accessed: Sep. 29, 2021. [Online]. Available: https://github.com/jjgo/shrinkbench.

[21] X. Chen, J. Mao, and J. Xie, "Comparison Analysis for Pruning Algorithms of Neural Networks," *Proc. - 2021 2nd Int. Conf. Comput. Eng. Intell. Control. ICCEIC 2021*, pp. 50–56, 2021.

[22] Y. He, X. Dong, G. Kang, Y. Fu, C. Yan, and Y. Yang, "Asymptotic Soft Filter Pruning for Deep Convolutional Neural Networks," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3594–3604, Aug. 2020.

[23] L. Cai, Z. An, C. Yang, and Y. Xu, "Softer Pruning, Incremental Regularization," in: *Proc. 2020 25th International Conf. on Pattern Recognition (ICPR)*, 2021, pp. 224-230.

[24] M. Mousa-Pasandi, M. Hajabdollahi, N. Karimi, S. Samavi, and S. Shirani, "Convolutional Neural Network Pruning Using Filter Attenuation," in: *Proc. 2020 IEEE International Conf. on Image Processing (ICIP)*, 2020, pp. 2905-2909.

[25] Z. Wang, C. Li, and X. Wang, "Convolutional neural network pruning with structural redundancy reduction," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 14908–14917, 2021.

[26] PeiSongwen, WuYusheng, GuoJin, and QiuMeikang, "Neural Network Pruning by Recurrent Weights for Finance Market," *ACM Trans. Internet Technol.*, vol. 22, no. 3, pp. 1–23, Jan. 2022.

[27] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient ConvNets," in: *Proceedings of the 5th International Conf. on Learning Representations (ICLR).* Nov. 2017. Toulon, France.

[28] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning Both Weights and Connections for Efficient Neural Networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, 2015, pp. 1135–1143*.

[29] X. Liu, B. Li, Z. Chen, and Y. Yuan, "Exploring Gradient Flow Based Saliency for DNN Model Compression," in *Proceedings of the 29th ACM International Conference on Multimedia*, New York, NY, USA: Association for Computing Machinery, 2021, pp. 3238–3246.

[30] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 11256–11264, Jun. 2019.

[31] C. H. Sarvani, M. Ghorai, S. R. Dubey, and S. H. S. Basha, "HRel: Filter pruning based on High Relevance between activation maps and class labels," *Neural Networks*, vol. 147, pp. 186–197, Mar. 2022.

[32] M. Soltani, S. Wu, J. Ding, R. Ravier, and V. Tarokh, "On the information of feature maps and pruning of deep neural networks," *Proc. - Int. Conf. Pattern Recognit.*, pp. 6988–6995, 2020.

[33] C. Hur and S. Kang, "Entropy-based pruning method for convolutional neural networks," *J. Supercomput. 2018 756*, vol. 75, no. 6, pp. 2950–2963, Nov. 2018.

[34] Y. Si and W. Guo, "Application of A Taylor Expansion Criterion-based Pruning Convolutional Network for Bearing Intelligent Diagnosis," *2020 Glob. Reliab. Progn. Heal. Manag. PHM-Shanghai 2020*, Oct. 2020.

[35] C. Yu, J. Wang, Y. Chen, and X. Qin, "Transfer channel pruning for compressing deep domain adaptation models," *Int. J. Mach. Learn. Cybern. 2019 1011*, vol. 10, no. 11, pp. 3129–3144, Sep. 2019.

[36] Z. Huang, L. Li, and H. Sun, "Global biased pruning considering layer contribution," *IEEE Access*, vol. 8, pp. 173521–173529, 2020.

[37] B. Wang, F. Ma, L. Ge, H. Ma, H. Wang, and M. A. Mohamed, "Icing-EdgeNet: A Pruning Lightweight Edge Intelligent Method of Discriminative Driving Channel for Ice Thickness of Transmission Lines," *IEEE Trans. Instrum. Meas.*, vol. 70, 2021.

[38] T. Xu *et al.*, "CDP: Towards Optimal Filter Pruning via Class-Wise Discriminative Power," in *Proceedings of the 29th ACM International Conference on Multimedia*, New York, NY, USA: Association for Computing Machinery, 2021, pp. 5491–5500.

[39] Z. Chen, T. B. Xu, C. Du, C. L. Liu, and H. He, "Dynamical Channel Pruning by Conditional Accuracy Change for Deep Neural Networks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 2, pp. 799–813, Feb. 2021.

[40] A. Gonzalez-Garcia, D. Modolo, and V. Ferrari, "Do Semantic Parts Emerge in Convolutional Neural Networks?," *Int. J. Comput. Vis. 2017 1265*, vol. 126, no. 5, pp. 476–494, Oct. 2017.

[41] Y. Le Cun, Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal Brain Damage," *Adv. Neural Inf. Process. Syst.*, vol. 2, pp. 598--605, 1990, Accessed: Sep. 18, 2022. [Online]. Available: http://130.203.136.95/viewdoc/summary?doi=10.1.1.32.7223.

[42] Z. Wang, W. Hong, Y. P. Tan, and J. Yuan, "Pruning 3D Filters for Accelerating 3D ConvNets," *IEEE Trans. Multimed.*, vol. 22, no. 8, pp. 2126–2137, Aug. 2020.

[43] Y. Zhang, Y. Yuan, and Q. Wang, "ACP: Adaptive Channel Pruning for Efficient Neural Networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2022, pp. 4488–4492.

[44] B. Zhou, D. Bau, A. Oliva, and A. Torralba, "Interpreting Deep Visual Representations via Network Dissection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2131–2145, Sep. 2019.

[45] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, "Variational convolutional neural network pruning," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 2775–2784, Jun. 2019.

[46] R. Q. Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried, "Invariant visual representation by single neurons in the human brain," *Nat. 2005 4357045*, vol. 435, no. 7045, pp. 1102–1107, Jun. 2005.

[47] D. Bau, J.-Y. Zhu, H. Strobelt, A. Lapedriza, B. Zhou, and A. Torralba, "Understanding the role of individual units in a deep neural network," *Proc. Natl. Acad. Sci.*, vol. 117, no. 48, pp. 30071–30078, Dec. 2020.

[48] C. Li, M. Z. Zia, Q. H. Tran, X. Yu, G. D. Hager, and M. Chandraker, "Deep Supervision with Intermediate Concepts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1828–1843, Aug. 2019.

[49] C. Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-Supervised Nets," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 2015, vol. 38, pp. 562–570, [Online]. Available: https://proceedings.mlr.press/v38/lee15a.html.

[50] Z. Zhuang *et al.*, "Discrimination-Aware Channel Pruning for Deep Neural Networks," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 883–894.

[51] Z. Hou and S. Y. Kung, "A discriminant information approach to deep neural network pruning," *Proc. - Int. Conf. Pattern Recognit.*, pp. 9553–9560, 2020, doi: 10.1109/ICPR48806.2021.9412693.

[52] E. Saraee, M. Jalal, and M. Betke, "Visual complexity analysis using deep intermediate-layer features," *Comput. Vis. Image Underst.*, vol. 195, p. 102949, Jun. 2020.

[53] A. S. Morcos, D. G. T. Barrett, N. C. Rabinowitz, and M. Botvinick, "On the importance of single directions for generalization," *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.*, Mar. 2018.

[54] J. Ukita, "Causal importance of low-level feature selectivity for generalization in image recognition," *Neural Networks*, vol. 125, pp. 185–193, May 2020.

[55] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9911 LNCS, pp. 499–515, 2016.

[56] H. Peng and S. Yu, "Beyond softmax loss: Intra-concentration and inter-separability loss for classification," *Neurocomputing*, vol. 438, pp. 155–164, May 2021.

[57] H. M. Yang, X. Y. Zhang, F. Yin, and C. L. Liu, "Robust Classification with Convolutional Prototype Learning," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 3474–3482, Dec. 2018.

[58] S. Son, S. Nah, and K. M. Lee, "Clustering Convolutional Kernels to Compress Deep Neural Networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11212 LNCS, pp. 225–240, 2018.

[59] Z. Zhou, W. Zhou, H. Li, and R. Hong, "Online Filter Clustering and Pruning for Efficient Convnets," *Proc. - Int. Conf. Image Process. ICIP*, pp. 11–15, Aug. 2018.

[60] S. Yu, K. Wickstrom, R. Jenssen, and J. Principe, "Understanding Convolutional Neural Networks with Information Theory: An Initial Exploration," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 1, pp. 435–442, Jan. 2021.

[61] Y. Li *et al.*, "Exploiting kernel sparsity and entropy for interpretable CNN compression," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 2795–2804, Jun. 2019.

[62] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, 2013.

[63] B. McWilliams and G. Montana, "Subspace clustering of high-dimensional data: a predictive approach," *Data Min. Knowl. Discov. 2013 283*, vol. 28, no. 3, pp. 736–772, May 2013.

[64] M. Liu, Y. Wang, and Z. Ji, "Self-Supervised Convolutional Subspace Clustering Network with the Block Diagonal Regularizer," *Neural Process. Lett. 2021*, pp. 1–27, Aug. 2021.

[65] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid "Deep Subspace Clustering Networks," in: *Proceedings of the 31st International Conf. on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA*, 2017, pp 23–32.

[66] S. Roy, P. Panda, G. Srinivasan, and A. Raghunathan, "Pruning Filters while Training for Efficiently Optimizing Deep Learning Networks," *Proc. Int. Jt. Conf. Neural Networks*, Jul. 2020.

[67] Y. He, Y. Ding, P. Liu, L. Zhu, H. Zhang, and Y. Yang, "Learning Filter Pruning Criteria for Deep Convolutional Neural Networks Acceleration," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 2006–2015, 2020.

[68] Z. Zhou, W. Zhou, R. Hong, and H. Li, "Online Filter Weakening and Pruning for Efficient Convnets," *Proc. - IEEE Int. Conf. Multimed. Expo*, vol. 2018-July, Oct. 2018.

[69] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri, "Acceleration of Deep Convolutional Neural Networks Using Adaptive Filter Pruning," *IEEE J. Sel. Top. Signal Process.*, vol. 14, no. 4, pp. 838–847, May 2020.

[70] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.

[71] B. Aaron, D. E. Tamir, N. D. Rishe, and A. Kandel, "Dynamic incremental K-means clustering," *Proc. - 2014 Int. Conf. Comput. Sci. Comput. Intell. CSCI 2014*, vol. 1, pp. 308–313, 2014.

[72] U. von Luxburg, "A tutorial on spectral clustering," *Stat. Comput. 2007 174*, vol. 17, no. 4, pp. 395–416, Aug. 2007.

[73] L. Rosasco, M. Belkin, and E. De Vito, "On Learning with Integral Operators," *J. Mach. Learn. Res.*, vol. 11, no. 30, pp. 905–934, 2010, Accessed: Sep. 30, 2021. [Online]. Available: http://jmlr.org/papers/v11/rosasco10a.html.

[74] C. Xia, W. Hsu, M. L. Lee, and B. C. Ooi, "BORDER: Efficient computation of boundary points," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 3, pp. 289–303, Mar. 2006.

[75] A. Achille and S. Soatto, "Emergence of invariance and disentanglement in deep representations," *2018 Inf. Theory Appl. Work. ITA 2018*, Oct. 2018.

[76] L. Decreusefond, I. Flint, N. Privault, and G. L. Torrisi, "Determinantal Point Processes," *Bocconi Springer Ser.*, vol. 7, pp. 311–342, 2016.

[77] H. Wang, P. Chen, and S. Kwong, "Building Correlations between Filters in Convolutional Neural Networks," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3218–3229, Oct. 2017.

[78] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-Margin Softmax Loss for Convolutional Neural Networks," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, 2016, pp. 507–516.

[79] A. Krizhevsky and A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009, Accessed: May 10, 2022. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.222.9220.

[80] M. Lin *et al.*, "Hrank: Filter pruning using high-Rank feature map," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1526–1535, 2020.

[81] H. Pan, Z. Chao, J. Qian, B. Zhuang, S. Wang, and J. Xiao, "Network pruning using linear dependency analysis on feature maps," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2021-June, pp. 1720–1724, 2021.

[82] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, "Variational convolutional neural network pruning," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 2775–2784, Jun. 2019.

[83] Z. Huang and N. Wang, "Data-Driven Sparse Structure Selection for Deep Neural Networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11220 LNCS, pp. 317–334, 2018.

[84] R. Yu *et al.*, "NISP: Pruning Networks Using Neuron Importance Score Propagation," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 9194–9203, Dec. 2018.

[85] Y. He, X. Zhang, and J. Sun, "Channel Pruning for Accelerating Very Deep Neural Networks," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2017-Octob, pp. 1398–1406, Dec. 2017.

[86] T. Wu, X. Li, D. Zhou, N. Li, and J. Shi, "Differential Evolution Based Layer-Wise Weight Pruning for Compressing Deep Neural Networks," *Sensors 2021, Vol. 21, Page 880*, vol. 21, no. 3, p. 880, Jan. 2021.

هرس تطبیقی شبکه‌های عصبی کانولوشن

سعید احمدلوئی[۱]، کریم فائز[۲،*] و بهروز معصومی[۳]

[۱و۳] گروه مهندسی کامپیوتر و فناوری اطلاعات، واحد قزوین، دانشگاه آزاد اسلامی، قزوین، ایران.

[۲] دانشکده مهندسی برق، دانشگاه صنعتی امیرکبیر، تهران، ایران.

ارسال ۲۰۲۲/۱۰/۱۰؛ بازنگری۲۰۲۲/۱۲/۰۳؛ پذیرش ۲۰۲۲/۱۲/۱۹

چکیده:

شبکه‌های کانولوشن موفقیت‌های چشمگیری در حوزه بینایی ماشین کسب کرده‌اند. با این وجود استفاده از این شبکه‌ها در مسائل دنیای واقع همراه با دو چالش عمده است: (۱) استفاده از آنها مستلزم توان پردازشی و حافظه بالا است، (۲) در حال حاضر روشی برای طراحی معماری این شبکه‌ها برای مسئله خاص وجود ندارد. فشرده‌سازی/هرس‌کردن این شبکه‌ها به عنوان راهکاری جهت مواجه با چالش اول مطرح شده و نسبتا به نتایج رضایت‌بخشی نیز دست‌یافته شده است. برای مواجه با چالش دوم تاکنون صرفا برخی الگوریتم‌های تکاملی ارائه شده است. الگوریتم ارائه شده در این مقاله را می‌توان به عنوان راهکاری جهت مواجه با هر دو چالش به صورت همزمان در نظر گرفت. در الگوریتم پیشنهادی برای ارزیابی فیلترها از معیار ثابت و از پیش تعریف شده خاصی استفاده نمی‌شود، به طوریکه معیار ارزیابی فیلترها به صورت برخط در طی آموزش شبکه بر اساس ترکیب میزان مطلوبیت فیلتر در لایه خودش و لایه بعدی شکل می‌گیرد. همچنین راهکار جدیدی پیشنهاد شده است که در صورت لزوم فیلتر جدید نیز به لایه‌ها اضافه می‌شود. بنابراین الگوریتم پیشنهادی صرفا یک الگوریتم هرس کردن نیست بلکه تعداد فیلترهای بهینه هر لایه را نیز تشخیص می‌دهد. با انجام شبیه‌سازی‌های متنوع روی معماری‌های مختلف کارکرد الگوریتم پیشنهادی مورد ارزیابی قرار گرفته است. در مقایسه با روش‌های موجود، الگوریتم پیشنهادی به دقت و نرخ هرس نسبتا بالاتری دست‌یافته، و علی‌رغم هزینه پردازشی بالاتری که در هر epoch دارد با توجه به همزمانی آموزش و هرس، در کل الگوریتم پیشنهادی سریعتر از سایر الگوریتم‌ها شبکه نتیجه را ارائه می‌دهد.

کلمات کلیدی: شبکه‌های عصبی کانولوشن، معماری تطبیقی، هرس کردن، فشرده‌سازی.