Robust Convolution Kernel Quantity Determination based on Corner Radiation Area Adaptation

Yu Liang[a], Yuyue Du[a], Siguang Li[b,c], Maozhen Li[d]

[a]*Department of Computer Science and Engineering,Shandong University of Science and Technology, Shandong ,China*
[b]*The Key Laboratory of Embedded Systems and Service Computing, Tongji University, Shanghai, China*
[c]*College of Electrical Engineering, Binzhou University, Binzhou, Shandong, China*
[d]*Department of Electronic and Computer Engineering, Brunel University London, Uxbridge, UB8 3PH, UK*

Abstract

The number of kernels in a convolutional neural network (CNN) can have a significant impact on the performance of the CNN in the aspects of accuracy and computation efficiency. However, existing approaches to determining the number of convolution kernels are mainly conducted through a manual process, which suffers from the problems of potential overfitting, instability and inefficiency. In response to these problems, this paper presents a corner radiation area adaptation (CRAA) based method to automatically determine the number of convolution kernels. CRAA is evaluated in comparison with three representative methods on multiple public data sets. Experimental results show that CRAA is robust to the number of convolution kernels which is well adapted to a specific data set and achieves a higher level of classification accuracy by 3% when spending the same period of time in classification. More importantly, CRAA reduces the computational time by 15% in comparison with the three representative approaches when reaching the same level of accuracy in classification.

Keywords: Convolutional kernel, convolutional neural network, feature extraction, corner radiation area adaptation

1. Introduction

In recent years, deep learning has attracted a widespread interest from all walks of life and promoted the rapid development of a series of applied

research in the field of artificial intelligence. As an important branch of deep learning, convolutional neural networks (CNNs) have a close connection with the research of computer vision. Through many well-known CNN models [1, 2, 3, 4, 5, 6], the field of computer vision has achieved a tremendous development. The feature learning strategy of convolution neural networks has been widely successful in a series of fields of computer vision, gradually replacing the traditional research of artificial design features and becoming a new research focus.

As one of the important hyperparameters in CNN, the number of convolutional kernels determine the runtime of CNN, the storage cost, and accuracy when training the model. At present, selecting the convolution kernels' quantity is mainly conducted manually. But it is challenging to determine a suitable quantity manually to ensure a high level of accuracy. In many cases, we can only choose suitable hyperparameters through experience, which is an inefficient process. Therefore, it is a normal practice to set an excessive number of convolution kernels to ensure a high level of accuracy. Some well known convolution neural network models, such as Residual Net [1], VGGNet [2] and other neural network models [3, 4, 5, 6] all directly specify the number of convolution kernels. However, setting a large number of convolution kernels may lead to the problem of overfitting in those CNN networks. This makes feature maps with redundant weights, which greatly reduces the training efficiency.

To solve these problems, there are mainly two approaches at present. One is network pruning, the other is convolution kernel quantity adaptation. Network pruning reduces overfitting by pruning the model structure and a great deal of further developments [7, 8, 9, 10, 11] have been proposed. Convolution kernel quantity adaptation adapts the number of convolution kernels through the features of the data set itself to avoid overfitting. However, the existing methods for convolution kernel quantity adaptation are limited by the size of a convolution kernel, leading to poor stability and a high computation overhead.

To address these challenges, this paper presents a robust convolution kernel quantity determination solution based on corner radiation area adaptation (CRAA). CRAA builds on a denoising module, a corner detection module and a corner radiation area (CRA). Here, the denoising module and the corner detection module are used to extract the points with rich features in an image and facilitate CRA extraction. CRA is a corner extension area used to find auxiliary features.

The major contributions of the paper are as follows:

- It presents CRAA to adaptively determine the number of convolution kernels.
- Furthermore, CRAA is not limited by the size of a convolution kernel. For different sizes, the output results are consistent and robust.
- The feature points obtained by CRAA have good stability in the training process of CNN.
- Compared with three representative methods on multiple public data sets, CRAA increases the accuracy level by 3% when spending the same period of time in classification, but reduces the computation cost by 15% when reaching the same level of accuracy.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 presents the CRAA and describes the steps of the CRAA. Section 4 evaluates the performance of CRAA and analyzes the experimental results. Section 5 concludes the paper and points out some future work.

2. Related Work

This section reviews some related work on the problem of overfitting in convolutional neural networks, but it first discusses the role of convolution kernels.

The work in [3] presents an improved convolutional neural network and visualizes the convolution kernel. It is found that many convolution kernels play a role in extracting image features which can be regarded as an feature extractor. The work of [12] proposes a visualization technique. This technique deeply analyzes the function of the convolution layer and the operation of the classifier. It is found that the convolution kernels learn the typical local features of images. With an increasing number of layers, the learned features become stabilized and some low level features are dropped out. The work in [13] analyzes the internal structure of CNN, theoretically interprets the powerful feature extraction capabilities of CNN and reveals the roles of the convolution kernels. In addition, we also briefly introduce the existing estimation methods against the noises. The work in [14, 15] propose two noise estimation methods for complex networks and communication networks.

We further introduce some related methods for overfitting issues. The work of [7] proposes a three-stage method which works together to reduce the storage requirements of neural networks by 35 to 49 times without affecting their accuracy levels. The work in [8] proposes a CNN acceleration method in which they cut out the filters that are considered to have little effect on the output accuracy from the CNN. The work in [9] proposes a new formulation for pruning convolution kernels in neural networks. It is based on greedy standards for pruning and interweaving through back propagation fine-tuning to maintain a good generalization in the pruning network. The work in [16] proposes three convolution kernel quantity adaptation methods to extract different types of edge features which are Log edge detection count(LEDC), Canny edge detection count(CEDC) and Sobel edge detection count(SEDA). Among them, the LEDC method uses laplacian of gaussian operator [17] to segment an image set. CEDC uses the canny edge detection [18] operator to segment the images. The Sobel operator [19] uses the gradient value of the neighboring area to calculate the gradient of 1 pixel and selects it according to the absolute value. CEDC, LEDC and SEDC can process the image data sets according to the characteristics of their own operators. The three methods all aim to automatically set the number of convolution kernels and each has their own advantages and disadvantages in terms of computation time and accuracy in classification.

Although CEDC, LEDC and SEDC have certain effects, there are still some limitations that need to be addressed. Firstly, these methods are limited by the size of a convolution kernel. This is because CEDC, LEDC and SEDC all use the detection methods that are affected by the size of a convolution kernel. In CEDC, LEDC and SEDC, the number of kernels obtained using a convolution kernel size of 3×3 is different from that of using a size of 5×5. Furthermore, as the size of a convolution kernel becomes larger, the quantity of adaptation becomes higher. Hence, when the quantity is high enough, it also causes the problem of overfitting. Secondly, the features obtained through these methods do not have a good stability. According to [12], CNN learning is a gradual process. The initial layer is targeted at low level features such as edge, color and brightness. As the number of layers deepens, the learned features become more complex, and the low level features are gradually eliminated. Therefore, when the number of layers gradually increases, edge features will gradually lose, which incurs poor stability.

3. Corner Radiation Area Adaptation(CRAA)

To address the limitations mentioned in Section 2, this section presents the corner radiation area adaptation method.

3.1. The Adaptation Principle of CRAA

In CNN, a convolution kernel mainly performs the convolution operations. As pointed out in [12, 13], a convolution kernel is usually initialized in the form of a random matrix. It learns through Error Back Propagation (BP) during the network training process. Following BP operations, a convolution kernel constantly learns valuable features to increase its weight. For ease of classification, CNN brings the weights of the superposition into the score function. The larger the activation value is, the more a convolution kernel meets the requirements and the easier it is to distinguish the target image from other images.

A convolution kernel works like a filter which is used to extract the local features of the image. When the convolution layer contains only one convolution kernel, the model can only extract one feature. Obviously, the features extracted by a single convolution kernel are not able to classify images. Therefore, more features can be learned by increasing the number of convolution kernels. However, it is challenging in determining the number of kernels with an aim to achieve a high level of accuracy in classification. Setting a suitable number of convolution kernels not only enables the model to learn enough local features, but also saves computation time.

In this work, we set the number of convolution kernels based on the number of key features in an image, and those key features must always exist during the CNN network training process. Obviously, the corner points meet the above requirements. The corner points are considered to be the maximum points of curvature on the edge curve of a 2D image, and determine the local feature extraction when an image is used as input [20]. Corners have three characteristics which are the intersections between the contours, the features that are usually stable, the pixels in the area around the point have large variations in both the gradient direction and their gradient magnitudes [21]. Therefore, corners in an image are the key features that satisfy these requirements, and we use corner detection to adapt the number of convolution kernels.

In addition to the corner points, CRAA also requires a large number of auxiliary features. CNN performs a black box operation when it operates,

so it is hard to know in detail which specific features the convolution kernel has learned. If only the number of corner points is adapted to the number of convolution kernels, the features learned by the CNN through the Error Back Propagation (BP) are not necessarily from the corner points that own the most abundant feature values. This leads to the loss of corner points and has some instability. In order to compensate for the deficiencies of the black box learning process and increase the probability of CNN acquiring corner features and improve network stability, it is necessary to increase the number of convolution kernels to extract more valuable features.

In order to find pixels with attributes similar to the corner points, we focus on the area around the corner points. A corner point is a point where the gray level difference between the corner point and its surrounding area is large. However, it probably has a small number of points with similar gray values to the corner points, and these points are also likely to have the same large pixel differences as the surrounding points. Hence, the core idea of CRA is to detect the surrounding circular radiation area and find valuable points by setting the CRA at the known corner points.

3.2. The Sufficient Conditions of CRAA

Before introducing the CRAA, we first analyzes the sufficient conditions for CRAA.

If the number of adaptive convolution kernels is lower than the number of corner points, there will be too few features learned in continuous training, and some corner points with important features will not be extracted, which will lead to a low level of accuracy. Therefore, in order to ensure that the complete features of an image are extracted, the number of features extracted in CNN must be at least equal to the number of corner points detected by the corner detection module. Similarly, when CNN trains the data set, the number of convolution kernels should be greater than or equal to the maximum number of corner points when processing the entire data set. Its specific definition is given below:

$$a \geqslant max(b_1, b_2 \cdots b_n) \tag{1}$$

where a is the number of convolution kernels, max represents the maximum value in the collection. n is the total number of images in the data set, and b is the value of the number of corners in an image in the data set.

After the CRAA method is completed, according to (1), the number of convolution kernels should be greater than or equal to the larger number between the number of CRAA operations and a, i.e.

$$d \geqslant max(a, k) \tag{2}$$

where k is the number of CRAA operations for the image. d represents the range of values that fit through CRAA.

3.3. The Description of CRAA

To implement CRAA, a total of three modules are required which are the denoising module, corner detection module and CRA module. In this framework, we mainly contribute to the denoising module and the corner radiation area. Among them, the core contribution is the corner radiation area. And the corner detection module belongs to the existing method, corner detection is mentioned in this paper as a necessary tool for CRAA. Figure 1 shows the work flow of CRAA.

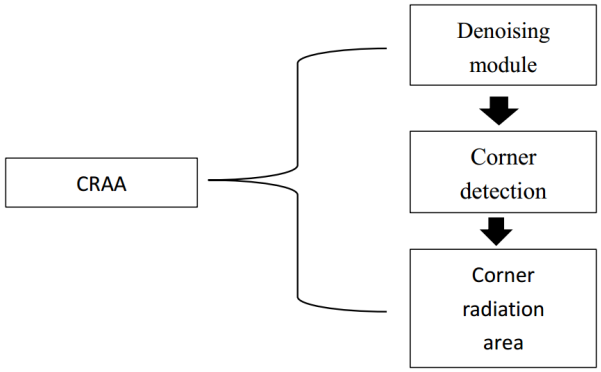

Figure 1: The workflow of CRAA.

Firstly, we perform the denoising operation. We define a sliding 3×3 window that allows the window to pass through an image. We set the distance to move continuously and traverse the image. Then, we eliminate some noise points which are meaningless. We take the value of the eight pixels around the pixel p, and calculate the pixel difference between the point p and its neighbors. When we complete the previous step, we set a threshold K and give the following formula:

$$S_{p\to x} = \begin{cases} 1 & |Q_{p\to x} - Q_p| > K \\ 0 & |Q_{p\to x} - Q_p| \leq K \end{cases} \tag{3}$$

where $S_{p\to x}$ represents the result of the value, Q represents the value of the pixel, and $S_{p\to x}$ represents the pixel in eight directions around the p. A point meets the condition of $S_{p\to x}$=0 point which indicates that the difference between the gray value of the point and the p point is small.

As shown in Figure 2, a black point means p, a grey point means the point which $S_{p\to x}$=0 and a white point means the point which $S_{p\to x}$=1. When the number of the points which meet the condition of $S_{p\to x}$=0 is 1 to 4, the p point may became a corner point. However, when the number of the points which meet the condition of $S_{p\to x}$=0 is equal to 5 and those five points are continuous, the point p cannot be called a corner but an edge, we eliminate the p points. When the points satisfying $S_{p\to x}$=0 are greater than 5, it means the degree of change around p is not large, then the point p can not be called a corner point, we eliminate the p points.

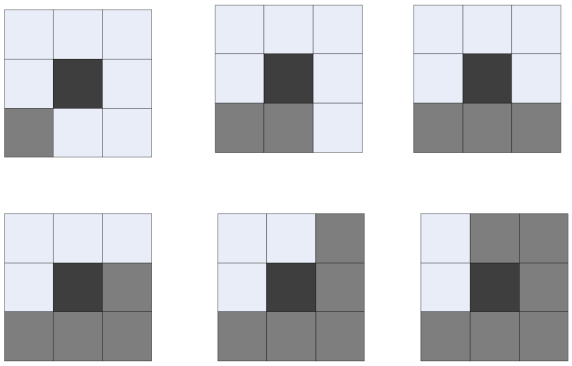

Figure 2: Corner denoising principle.

We further perform the corner detection operations and we can get the corner point through a corner detection method. In this work, we use the corner detection method proposed in [20].

During the movement, when it encounters a flat area, there will be no change in the gradient in all directions. When the edge is encountered, there is no change in the direction of the edge. However, when a corner is encountered, great changes will occur in all directions. By calculating the gradient for each pixel, if the absolute gradient values in both directions are

large, the pixel is treated as a corner point. The specific formula proposed by the work in [20] is:

$$E_{x,y} = \sum_{x,y} w_{u,v} \left[I_{x+u,y+v} - I_{u,v} \right]^2 \tag{4}$$

where $w_{u,v}$ is window function, I is pixels value, $[u, v]$ is the offset of the window, (x, y) is the corresponding pixel coordinate position in the window.

In order to facilitate the identification of corner points, the work in [20] performs Taylor formula expansion for formula (4):

$$E\left(u, v\right) \cong \begin{bmatrix} x & y \end{bmatrix} M \begin{bmatrix} x \\ y \end{bmatrix} \tag{5}$$

where M is a matrix, its expression is:

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{6}$$

Let λ_1 and λ_2 represent the eigenvalues of the matrix M. If the values of λ_1 and λ_2 are both small, the image in the Gaussian window is nearly flat. If one value is large, the other value is small, it means that the edge is detected. If both λ_1 and λ_2 are large, then the corner point is detected. Finally, the work in [20] provides another step change in order to make the results more friendly:

$$R = detM - k\left(traceM\right)^2 \tag{7}$$

where $detM = \lambda_1 \lambda_2$, $traceM = \lambda_1 + \lambda_2$, R is threshold.

When a point satisfies the condition of $R > 0$, it shows that the point is a corner point. Then, we should accumulate the number of corner points and get corner points' location. Finally, CRAA will traverse the data set, and get the total number and specific positions of the corner points that meet the conditions in each image.

After extracting the corner points, the radiation area of a corner point is defined as 16 pixels around the corner point as shown in Figure 3. With the corner point a as the center, there are 16 edge pixels around it. Having the set S represents 16 points on the corner point radiation area, the variable O is used to represent the gray difference between two points, and a threshold t is set. According to the work in [21], the corner point determination condition is that the gray value of the corner point is greatly different from the gray

value of most points around it. Therefore, the majority of points around a corner point satisfying the condition of $O_S > t$. From Figure 3, it can be seen that the gray value of the triangle is greatly different from the surrounding blank area, and the gray value of the triangle itself is not much different.

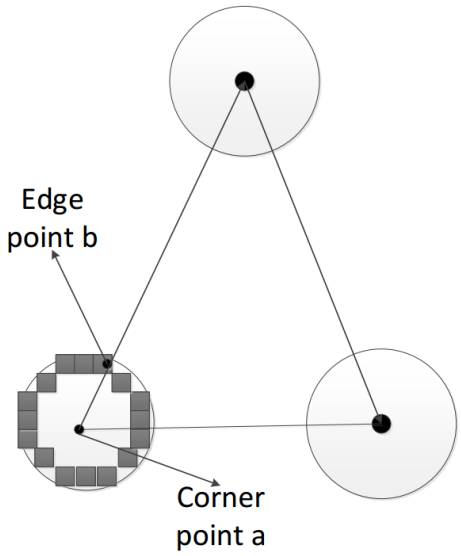

Figure 3: Corner radiation area.

Therefore, it is possible to determine the point in S where the gray difference value is smaller than the threshold t, and we extract these pixels which meet the condition. Then we can determine whether or not the variation in grayscale between these pixels and other points around them are huge. If the variation in grayscale is huge, We think this point has value and extract it. The specific description and formula of the method are as follows.

For a pixel point x of each corner radiation area, the pixel (represented by $a \to x$) at the position relevant to the corner point a may have one of the following two states:

$$S_{a \to x} = \begin{cases} h & \left| I_{a \to x} - I_a \right| \geqslant t \\ s & \left| I_{a \to x} - I_a \right| < t \end{cases} \tag{8}$$

where I_a represents the gray value of the corner point a. Consider the pixel on the circular window near the corner point. If there is a gray value in the extended pixel of the radiation area higher than I_a+t or lower than I_a-t, then the pixel point is excluded, the remaining pixel points are extracted.

We consider the A_s set as a point that satisfies the condition $S_{a\to x}$=s. The gray value of the point in the point set A_s is similar to the gray value of the corner point. If we take the point in the set A_s as the center of the circle, it still contains a large number of points with huge differences in gray values around the center point. As shown in Figure 3, the edge point b has a similar gray value with the corner point a, and around the point b still contains a large number of points that differ greatly from the gray value of the point b. Therefore, points in the set A_s may have points with more valuable features. In order to determine whether the points in the set A_s contain sufficiently valuable features, we extract the A_s set, calculate the number of conditions that the A_s set satisfies and record the number of satisfied conditions as KC. For each point in the A_s collection, we set a sliding window around this point, swipe eight directions around these points and calculate the sum of the squared differences. The eight directions are (1,0),(1,1),(0,1),(-1,1),(-1,0),(-1,-1),(0,-1),(1,-1). Through this operation, we can understand how these points change in gray when moving around. The sliding formula is given below:

$$F_{u,v}(x,y) = \sum_{\forall a,b} (I(x+a+u, y+v+b) - I(x+a, y+b))^2 \tag{9}$$

where a and b define the coordinates of the points in the sliding window area, (x,y) are the coordinates of the pixel points in the A_s collection, $I(x,y)$ represents the pixel value of (x,y), (u,v) represents the specific direction and distance of the window movement, and $F_{u,v}(x,y)$ represents the sum of the gray-scale squared differences in the eight directions near the corner point (x,y) after the window has moved in the (u,v) direction.

Then we use formula (10) to traverse the eight directions near the corner point (x,y):

$$G(x,y) = \textstyle\sum_{i=1}^{8} (F_{u,v}(x,y)) \tag{10}$$

We extract the $G(x,y)$ values for each point of the A_s set, then find the expected value H:

$$H = \frac{\sum_{i=1}^{n} G(x,y)}{8n} \tag{11}$$

For ease of classification, we set a classifier where the threshold is H. Compare the $F_{u,v}(x,y)$ values of each selected point in the A_s set in eight

directions. The specific formula is give in (12):

$$S = \begin{cases} 1 & F_{u,v}(x,y) \geqslant H \\ 0 & F_{u,v}(x,y) < H \end{cases} \tag{12}$$

S is used to determine whether the points around the set A_s meet the conditions. For a point z in the set A_s, if the number of points satisfying condition S=1 among the eight points around point z is greater than or equal to 4, the point is considered to have certain characteristics, then the values satisfying the conditions are added and a new set c_{kc} is formed, we count the number in set c_{kc}. Finally, we count the total number of the corner points and the number of set c_{kc},then we record the number of outputs as KC. CRAA is not designed to change the parameters related to a convolution kernel. As a result, the number of convolution kernels is independent of the size of a convolution kernel. The specific implementation of CRAA is detailed in Algorithm 1.

4. Experimental Results

4.1. Experimental Model Setup and Parameters

In order to validate the performance of CRAA, experiments were carried out using both the MNIST [22] and CIFAR-10 [23] data sets. The experimental programming language used was based on Anaconda's Python 3.6 development environment. The neural network model was built under the framework of Tensorflow [24]. Also we employed the Compute Unified Device Architecture (Cuda) acceleration service provided by Nvidia [25].

To ensure the fairness of the experiment, both CRAA and the existing methods followed the same convolution neural network model architecture and adopted the same number of training iterations. The CNN model consisted of two convolution layers and two pooling layers, followed by two full-connection layers and a softmax function. Since the MNIST data set is small, it is easy to obtain high accuracy under the training of the CNN model, and it does not require excessive iterations. So, for ease of comparison, the model only performed 5000 epoch iterations on the MNIST data set. And we performed 25000 epoch iterations on the CIFAR-10 data set. The specific model structure is shown in Figure 4.

Algorithm 1 : CRAA implementation.

Input:

Image Set J

Output:

Convolution Kernel Adaptation Number KC

1: Denoising the image according to the denoising module;
2: Calculate horizontal gradient I_x and vertical gradient I_y, initialize the value of the covariance matrix cov where $a{=}I_x^2$, $b{=}I_y^2$ and $c{=}I_xI_y$;
3: Calculate the eigenvalues λ_1 and λ_1 of the matrix M according to formula (6), the R value of each pixel is found according to formula (7);
4: Traversal of corner point set to obtain the matrix coordinates and grayscale value of corner points;
5: Get the corner radiation area extension point set and set the threshold t, exclude the points of CRA that meet $\left|I_{a\to x} - I_a\right| > t$ conditions and obtain the A_s set;
6: Set the sliding window and calculate $F_{u,v}(x, y)$ of A_s set based on formula (9);
7: Find the sum $G(x,y)$ according to equation (10);
8: Find the H value according to the formula (11), and determine by formula (12). For a point z in the set A_s, if the number of points satisfying condition $S{=}1$ among the eight points around point z is greater than or equal to 4, the point is considered to have certain characteristics, then the values satisfying the conditions are added and a new set c_{kc} is formed;
9: We count the total number of the corner points and the number of set c_{kc},then we record the number of outputs as KC;

4.2. Evaluation Criteria

The elements of the evaluations were the accuracy and running time of the CNN for the two data sets, respectively. Among them, the test set accuracy is defined as follows:

$$Accuracy = 1 - \frac{E}{S} \times 100\% \tag{13}$$

where E represents the total number of classification errors and S represents the total number of samples.

The computation efficiency formula is defined below:

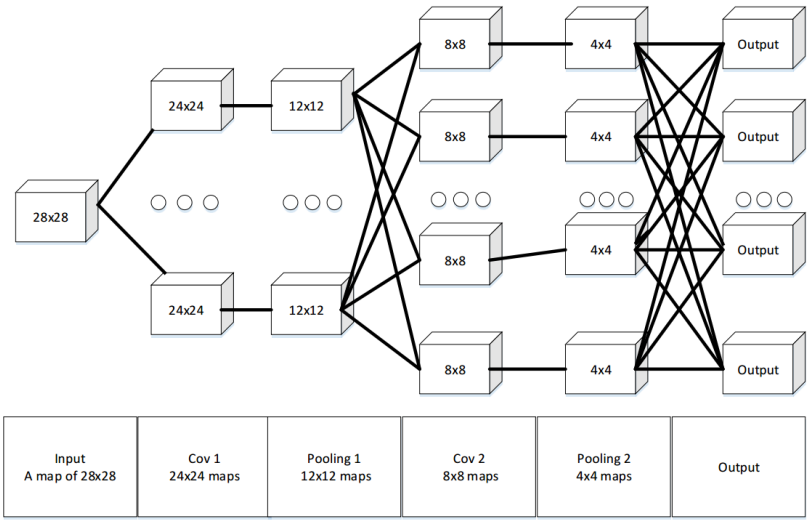

Figure 4: Convolution neural network model

$$C = \frac{t_2 - t_1}{t_2} \times 100\% \tag{14}$$

where C stands for computation efficiency, t_1 means the computation time of method 1, t_2 means the computation time of method 2.

4.3. CRAA Results

Table 1 shows the number of convolution kernels calculated by CRAA. In order to ensure that the convolution layer can extract a sufficient number of eigenvalues, the number of convolution kernels in each convolution layer is set to the number calculated by CRAA.

Table 1: The adaptation results of CRAA.

Data Set	Convolution Kernel Quantity
MNIST	35
CIFAR-10	78

Table 2 shows the accuracy levels of the CRAA that fits into the CNN model for testing. Among them, CRAA achieves 98.57% of accuracy on the

Table 2: The classification results of CRAA.

Data Set	Accuracy
MNIST	98.57%
CIFAR-10	83.12%

MNIST data set using iterations of 5000 epochs. CRAA generates 83.12% of accuracy on the CIFAR-10 data set after 25000 iterations of epochs.

We further compared CRAA with the three methods proposed in [16]: SEDC, CEDC, and LEDC. Since LEDC, SEDC, and CEDC all set the energy share,we evaluated the 80% and 95% cases whose energy share has good performance. We followed the same convolution kernel size as proposed in [16] with the first convolution kernel size being 3×3, and the second convolution kernel size being 5×5. The values listed in Table 3 are based on the MNIST data set. Conv1 and Conv2 represent convolution layer 1 and convolution layer 2 in the CNN model, respectively.

Table 3: Parameter settings on the MNIST data set.

Method	Conv1	Conv2
CEDC-80	11	32
SEDC-80	4	48
LEDC-80	5	14
CEDC-95	27	102
SEDC-95	9	204
LEDC-95	9	41

Table 4 lists the values obtained for LEDC, SEDC, and CEDC on the CIAFR-10 data set.

Due to the excessive number of algorithms involved in the experiments, the results of the comparison of the seven algorithms in one graph are too confusing. The experiment divides the comparison images into two images. One is a comparison of the CRAA method and the three algorithms with 80% energy share, and the other is a comparison between the CRAA method and the three algorithms with 95% energy share. Specific data and analysis are introduced in the following sections.

Table 4: Parameter settings on the CIFAR-10 data set.

Method	Conv1	Conv2
CEDC-80	18	59
SEDC-80	7	87
LEDC-80	6	16
CEDC-95	41	171
SEDC-95	19	283
LEDC-95	14	68

4.4. Performance of CRAA on the MNIST Data set.

Figure 5 shows a comparison of CRAA and the three algorithms with an 80% energy share on the MNIST data set. After 5000 iterations of epoch training, it can be seen that CRAA has obvious advantages compared with LEDC-80, SEDC-80 and CEDC-80 with 80% energy share.

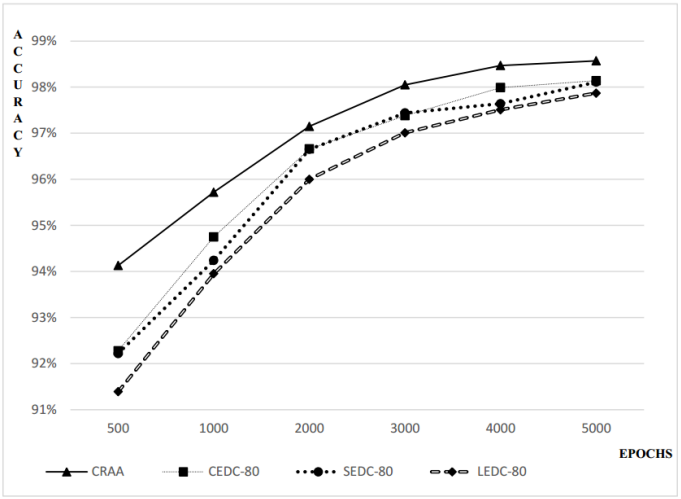

Figure 5: CRAA performance on the MNIST data set (energy share 80%).

Table 5 shows the computation times consumed among the algorithms. CRAA carried out 5000 trainings for 37s, while LEDC-80, SEDC-80, and CEDC-80 each having 30s, 31s and 23s respectively.

From the above data, we can see that CRAA does not incur much high overhead in computation but generates a higher level of accuracy.

Table 5: Training times on the MNIST data set (energy share 80%).

Method	Time
CRAA	37s
CEDC-80	30s
SEDC-80	31s
LEDC-80	23s

Figure 6 shows a comparison of CRAA with the three algorithms with a 95% energy share under the MNIST data set. It can be clearly seen that compared with CEDC-95, SEDC-95 and LEDC-95, CRAA has an accuracy rate of 98.57%. The accuracy levels of CEDC-95, SEDC-95 and LEDC-95 are 98.55%, 98.51% and 98.25%- respectively.

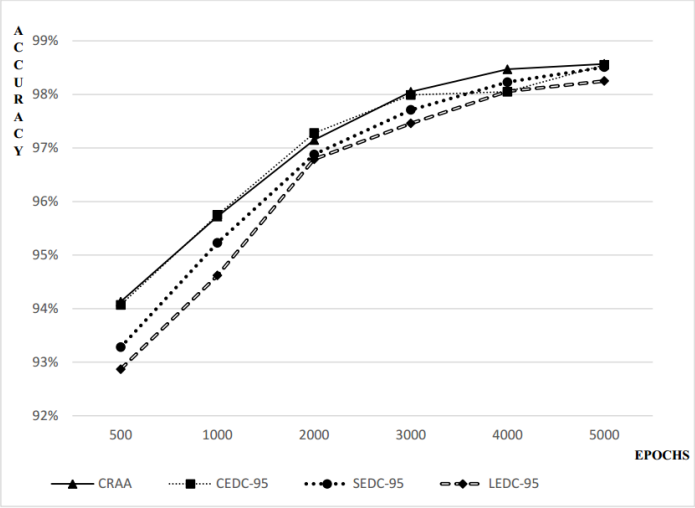

Figure 6: The performance of CRAA on the MNIST data set (energy share 95%).

It is worth noting that CRAA has a clear advantage in training time as shown in Table 6.

From Table 6, it can be seen that CRAA spends 37s for 5000 epochs. whereas SEDC and CEDC spend 64s and 51s- respectively. CRAA reduces the time by at least 27% compared with these two methods. But CRAA is slightly slower than LEDC-95.

Table 6: Training times on the MNIST data set (energy share 95%).

Method	Time
CRAA	37s
CEDC-95	64s
SEDC-95	51s
LEDC-95	35s

4.5. Performance of CRAA on the CIFAR-10 Data Set

We further evaluated the performance of CRAA on the CIFAR-10 data set in comparison with LEDC, SEDC and CEDC.

Figure 7 shows the comparison results of LEDC, SEDC, CEDC with 80% energy in the CIFAR-10 data set space. After 25,000 iterations of epoch training, the accuracy of CRAA on testing the CIFAR-10 data set reaches 83.12% which is better than CEDC-80 (80.52%), SEDC-80 (79.21%), and LEDC-80 (66.42%).

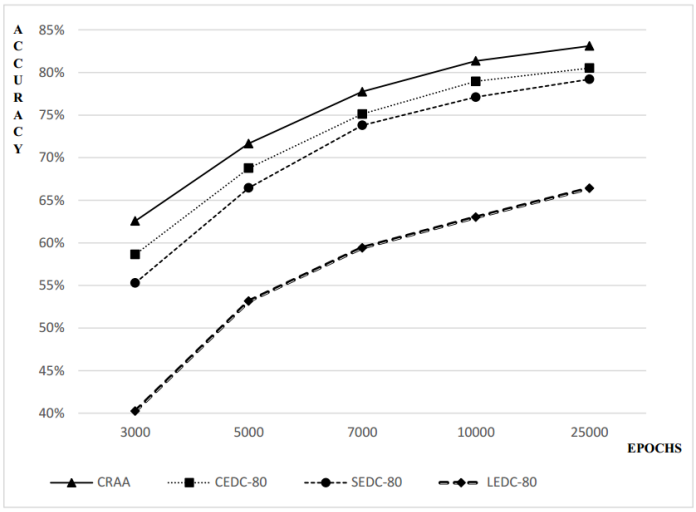

Figure 7: Performance of CRAA on the CIFAR-10 data set (energy share 80%).

For training time, as shown in Table 7, CRAA is comparable to the three algorithms.

It can be found that under the CIFAR-10 data set, CRAA has certain advantages in accuracy compared to CEDC-80 and SEDC-80, and it is very

Table 7: Training times on the CIFAR-10 data set (energy share 80%).

Method	Time
CRAA	85min40s
CEDC-80	84min54s
SEDC-80	86min50s
LEDC-80	78min25s

similar in terms of training time. Compared with LEDC-80, CRAA has a great advantage in accuracy.

Figure 8 shows a comparison of LEDC, SEDC, and CEDC with CRAA with 95% energy share under the CIFAR-10 dataset. We can see that after 25,000 iterations of training, CRAA is slightly inferior to CEDC-95 in accuracy. The accuracy rates of CRAA and CEDC-95 are 83.12% and 83.79%, respectively. Compared with SEDC-95 and LEDC-95, CRAA has certain advantages in accuracy, among which SEDC-95 and LEDC-95 are 81.23% and 78.12%, respectively.

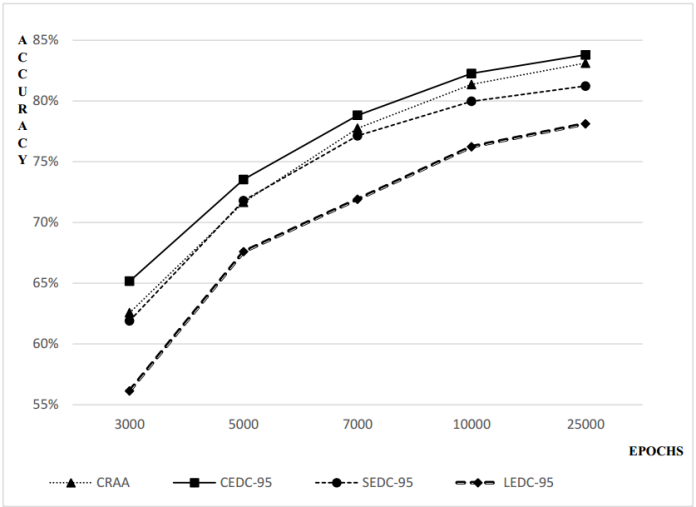

Figure 8: Performance of CRAA on the CIFAR-10 dataset (95% energy share).

For the training time, as shown in Table 8, CRAA spends less time than both CEDC-95 and SEDC-95, but is slightly slower than LEDC-95.

Table 8: Training times on the CIAFR-10 data set (energy share 95%).

Method	Time
CRAA	85min40s
CEDC-95	100min29s
SEDC-95	95min05s
LEDC-95	84min23s

4.6. The Major Advantages of CRAA

To sum up, in the case of the MNIST data set, CRAA's disadvantage in terms of training time is not so obvious compared to the three 80% energy share methods. But CRAA has a better accuracy rate. Compared with the three methods with a 95% energy share, CRAA has an advantage in training time with a similar level of accuracy to CEDC-95. Compared with SEDC-95, CRAA has advantages in both training time and accuracy. In comparison with LEDC-95, CRAA accuracy has an advantage. Under the CIFAR-10 data set, CRAA has certain advantages in accuracy compared to CEDC-80 and SEDC-80, and it is very similar in terms of training time. Compared with LEDC-80, CRAA has a great advantage in accuracy. CRAA is comparable to CEDC-95 in accuracy, but has certain advantages in training time. CRAA has certain advantages in both training time and accuracy compared with SEDC-80 and LEDC-80.

Furthermore, CRAA is not limited by the size of a convolution kernel. For different sizes, the output results are consistent and robust. Compared with CRAA, the three methods of CEDC, LEDC and SEDC increase with the size of convolution kernels. Therefore, the three compared methods have certain limitations compared to CRAA.

Last but not least, the features that are adapted by the CRAA method are more in line with the features obtained after the CNN training, and the features that are adapted through the CRAA method are highly stable features. With the increase in the number of CNN training layers, this feature is not easy to lose, and The probability of being a core feature is even higher.

5. Conclusion

In this paper, we have presented CRAA for robust convolution kernel quantity determination. Compared with three existing methods on both the

MNIST and CIFAR-10 data sets, CRAA achieved a higher accuracy level by 3% when spending the same amount of time in training. On the other hand, CRAA reduced the training time by at least 15% when the same level of accuracy was achieved.

A future work will investigate the techniques as proposed in [26, 27, 28, 29, 30] to approximate the corner radiation area to speed up the computation of CRAA so that CRAA can adapt the number of convolution kernels more quickly. At the same time, we are trying to combine network pruning with CRAA to enable CNN to better address the problem of overfitting with higher stability [31, 32] in convolution kernel quantity adaptation.

References

[1] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 770–778.

[2] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, Computer Science (2014).

[3] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: International Conference on Neural Information Processing Systems, 2012, pp. 1097–1105.

[4] X. Sun, P. Wu, S. C. H. Hoi, Face detection using deep learning:an improved faster rcnn approach, Neurocomputing (2018).

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[6] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, IEEE Transactions on Pattern Analysis & Machine Intelligence 39 (2015) 1137–1149.

[7] S. Han, H. Mao, W. J. Dally, A deep neural network compression pipeline: Pruning, quantization, huffman encoding, Neural Information Processing Systems 13 (2015).

[8] H. Li, A. Kadav, I. Durdanovic, H. Samet, H. P. Graf, Pruning filters for efficient convnets, in: International Conference on Learning Representations, 2016.

[9] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, in: International Conference on Learning Representations, 2016.

[10] S. H. Yang, Y. P. Chen, An evolutionary constructive and pruning algorithm for artificial neural networks and its prediction applications, Neurocomputing 86 (2012) 140–149.

[11] N. Yu, S. Qiu, X. Hu, J. Li, Accelerating convolutional neural networks by group-wise 2d-filter pruning, in: International Joint Conference on Neural Networks, 2017, pp. 2502–2509.

[12] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European Conference on Computer Vision, volume 8689, 2013, pp. 818–833.

[13] Y. Sun, X. Wang, X. Tang, Deeply learned face representations are sparse, selective, and robust, Computer Vision and Pattern Recognition (2014) 2892–2900.

[14] J. Hu, Z. Wang, F. E. Alsaadi, T. Hayat, Event-based filtering for time-varying nonlinear systems subject to multiple missing measurements with uncertain missing probabilities , Information Fusion 38 (2017) 74–83.

[15] J. Hu, Z. Wang, S. Liu, H. Gao, A variance-constrained approach to recursive state estimation for time-varying complex networks with missing measurements, Automatica 64 (2016) 155–162.

[16] Y. Wen, T. Yu, Y. Ling, Method for determining the number of convolution kernel via edge detection approach, Application Research of Computers (2017).

[17] A. R. Lamichhane, C. S. Chen, The closed-form particular solutions for laplace and biharmonic operators using gaussian function, Applied Mathematics Letters 46 (2015) 50–56.

[18] W. C. Zhang, Y. L. Zhao, T. P. Breckon, L. Chen, Noise robust image edge detection based upon the automatic anisotropic gaussian kernels, Pattern Recognition 63 (2016) 193–205.

[19] Y. Lang, D. Zheng, An improved sobel edge detection operator, in: IEEE International Conference on Computer Science and Information Technology, 2010, pp. 67–71.

[20] C. Harris, A combined corner and edge detector, Proc Alvey Vision Conf 1988 (1988) 147–151.

[21] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, in: European Conference on Computer Vision, 2006, pp. 430–443.

[22] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (1998) 2278–2324.

[23] Y. Abouelnaga, O. S. Ali, H. Rady, M. Moustafa, Cifar-10: Knn-based ensemble of classifiers, in: International Conference on Computational Science and Computational Intelligence, 2017.

[24] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, Tensorflow: a system for large-scale machine learning, 2016, pp. 265–283.

[25] W. M. Hwu, C. Rodrigues, S. Ryoo, J. Stratton, Compute unified device architecture application suitability, Computing in Science & Engineering 11 (2009) 16–26.

[26] X. Bai, Z. Wang, L. Zou, F. E. Alsaadi, Collaborative fusion estimation over wireless sensor networks for monitoring co2 concentration in a greenhouse, Information Fusion 42 (2018) 119 – 126.

[27] L. Wang, Z. Wang, Q.-L. Han, G. Wei, Event-based variance-constrained h-infinity filtering for stochastic parameter systems over sensor networks with successive missing measurements, IEEE Transactions on Cybernetics 48 (2018) 1007–1017.

[28] L. Zou, Z. Wang, Q.-L. Han, D. H. Zhou, Ultimate boundedness control for networked systems with try-once-discard protocol and uniform