Computing and Informatics, Vol. 44, 2025, 467–492, doi: 10.31577/cai_2025_2_467

SMRFC-PDCNN: AN EFFICIENT SCENE MATCHING RECOGNITION WITH DCNN AND FEATURE CLUSTERING ON SPARK

Jingguo Dai

School of Data Science, Guangzhou Huashang College
Guangzhou, 511300 China
e-mail: 646257139@qq.com

Yimin Mao

School of Information Engineering, Shaoguan University
Shaoguan, 512000 China
e-mail: mymlyc@163.com

Abstract. Scene recognition, an AI technology based on deep learning, has been widely used in public safety, road traffic, and automatic driving, but applying it on massive data in deep convolutional neural networks (DCNNs) results in performance bottlenecks. This paper proposes SMRFC-PDCNN, an efficient scene matching recognition algorithm that addresses three specific problems: decreased accuracy of feature maps, redundant feature calculations, and low efficiency in parallel recognition. The proposed algorithm includes a feature pooling selection strategy called MI-IPSS, a feature selection strategy called DCPSO-FSS, a load balancing strategy called CCG-LBS. MI-IPSS solves the problem of de-creased accuracy of feature maps by adapting the pooling strategy based on mutual information coefficient between feature maps before and after pooling. DCPSO-FSS uses density clustering and particle swarm optimization to locate clustering parameters quickly and recognize clustered features through sampling in the fully connected layer. CCG-LBS dynamically calculates the computing overhead of feature maps and allocates data between groups according to the over-head to solve the problem of low efficiency in parallel recognition. Experimental results show that SMRFC-PDCNN has good performance and is suitable for the fast scene matching recognition process of parallelized DCNNs on large-scale datasets.

Keywords: Scene recognition, parallel DCNN, Spark framework, Smart system

1 INTRODUCTION

Scene recognition is a critical computer vision task that involves analyzing images or videos to identify objects, people, actions, and other elements in a scene [1, 2, 3, 4]. Then, we can get a better understanding of the overall semantic meaning of the scene. The process of scene recognition involves extracting significant features from the visual data and categorizing them into predefined scene types [5, 6]. These deep level scene features can predict potential behaviors and help us achieve intelligent decision-making [7, 8, 9]. The technology has played a crucial role in various fields, including public safety, road traffic, and autonomous driving [10, 11, 12, 13]. However, as storage media technology and IoT devices have advanced, the field of public scene recognition has generated numerous massive, complex, and challenging-to-handle datasets, which exhibit the "4V" characteristics of high volume, velocity, variety, and value [14]. These 4V features make it challenging to uncover the potential features of big data, thus it impact the accuracy of scene recognition.

In order to extract scene features accurately, numerous studies have integrated scene recognition with DCNNs, it has the characteristics of automatic feature extraction, flexible model structure and shared parameters, these excellent features reduce the computational cost of scene recognition and improve recognition accuracy.

1. Herranz et al. [15] proposed the "Scene Recognition with CNNs: Objects, Scales and Dataset Bias" approach. They ad-dressed the challenge of recognizing scenes with objects of varying scales by designing a CNN architecture that can model both object and scene information simultaneously. Additionally, they introduced a dataset bias adaptation method to mitigate the impact of training data bias on scene recognition performance.
2. Khan et al. [16] proposed the "Integrating Multilayer Features of Convolutional Neural Networks for Remote Sensing Scene Classification" approach, which extracts multi-layer features from images using pre-trained CNN models and combines and classifies these features to obtain the final result.
3. Zhang et al. [17] proposed the "Multi-Level Ensemble Network for Scene Recognition" approach, which captures global scene information, focuses on local scene details, and combines the outputs of these two levels for final scene classification, this approach has demonstrated superior performance compared to traditional single-network methods in scene recognition tasks.

These scene recognition methods based on DCNNs still have some problems. In the process of large-scale and real-time scene recognition, plenty of repeated scene feature maps can cause wastage of computing resources. Furthermore, the computation of redundant feature maps in DCNNs can result in overfitting of the recognition model, ultimately leading to a decline in real accuracy.

To address the aforementioned issues, we propose an efficient scene Matching recognition with DCNNs and feature clustering on Spark. The main tasks of this algorithm is as follows. Firstly, we propose a pooled selection strategy based on improved mutual information, the strategy calculates the mutual information coefficient of feature maps before and after pooling, and adaptively adjusts the next pooling strategy. Secondly, we design a feature selection strategy based on density clustering and particle swarm optimization, The strategy uses particle swarm optimization algorithm to quickly locate density clustering parameters, then identify clustering features in fully connected layers to avoid redundant computation. Meanwhile, we reduce computational costs by introducing the Spark framework into the algorithm. To address the load balancing issue of DCNNs in the Spark framework, we propose a load balancing strategy based on cluster characteristic graph. The strategy dynamically calculates the computational cost of feature maps for each node in a distributed system, and dynamically allocates data among groups based on their costs. By implementing these strategies, algorithms can effectively address the challenges of large-scale and real-time scene recognition.

2 RELATED WORK

These studies represent advancements in scene recognition algorithms that focus on multi-scale feature fusion to increase robustness and accuracy, they also leverage DCNNs to accelerate feature extraction speed. However, it is important to note that these approaches primarily focus on improving multi-scale feature extraction and fusion, and do not address potential improvements in the computation process of DCNNs, which may limit their performance in large-scale data environments [18, 19].

To deal with the challenges of DCNNs in efficiently processing large data sets, the integration of the Spark framework into DCNNs has emerged as a novel research direction. Spark, being an in-memory distributed computing framework is extensively employed in processing and analyzing big data owing to its ease of programming, real-time responsiveness, load balancing, and support for diverse data sources [20]. Numerous optimized algorithms for DCNNs scene recognition based on the Spark distributed computing framework have also been extensively investigated. For example, Wang et al. [21] proposed the SASTCNN algorithm, a distributed DCNNs recognition algorithm based on Hadoop. By creating data partitions and parallelizing the recognition of DCNNs, the algorithm achieved parallel recognition of DCNNs.

Building upon this work, Bello et al. [22] proposed the BDCNN algorithm. The BDCNN algorithm associate neurons with pooling layers and calculates the correlation between two neurons based on the Pearson correlation coefficient between variables. This enables the algorithm to dynamically select the pooling layer strategy. This method has accelerated the pooling steps of convolutional neural networks under scene recognition algorithms. Boulila et al. [23] proposed the RS-DCNN algorithm. This algorithm splits large images into smaller ones and applies a maximum

likelihood classification supervised method for data set selection. While it processes images using a distributed parallel framework, it selects data features through classification methods. Thus, the design of large images segmentation enhances the detailed features of various sizes in the scene recognition process. Mao et al. [24] introduced the PDCNNO algorithm. The PDCNNO algorithm employs the modified secant-based conjugate gradient method (CGMSE) for DCNNs. The modified CGMSE is designed to obtain local classification results, leading to fast convergence of the network model's classification results. Moreover, the algorithm utilizes the load balancing strategy (LBRLA) to regulate the load rate and obtain global classification results which implements large-scale scene recognition parallel training.

In summary, the aforementioned parallel algorithms have partially addressed the performance issues of DCNNs when dealing with large datasets. However, since the model parameters are fixed during the recognition process, the Dropout strategy of the BDCNN algorithm cannot be adjusted to accommodate changes in the large-scale data. RS-DCNN cannot handle redundant feature calculation for non-fixed class datasets. Meanwhile, the load balance strategy of PDCNNO may cause an increase in global node load overhead due to the impact of abnormal node loads. Therefore, the algorithm still cannot effectively solve the issue of low parallel recognition efficiency.

3 PRELIMINARY

This section will introduce the relevant techniques utilized in the SMRFC-PDCNN algorithm. Specifically, we will discuss mutual information, particle swarm optimization algorithm, and cosine similarity. Below are their respective definitions.

3.1 Mutual Information

Mutual information [25] is a measure used in information theory to assess the correlation between two random variables. It quantifies the reduction in uncertainty of one random variable when we know the value of another. If we have two discrete random variables, X and Y, with values x_i and y_j, respectively, and their probabilities are $p(x_i)$ and $p(y_j)$, then the mutual information between X and Y is defined as:

$$I\left(X;Y\right) = \sum\sum p\left(x_i, y_j\right) * \log \frac{p\left(x_i, y_j\right)}{p\left(x_i\right) * p\left(y_j\right)}, \tag{1}$$

where $\sum\sum$ represents the sum over all possible (x_i, y_j), and log denotes the logarithm with base 2.

3.2 Particle Swarm Optimization Algorithm

Particle Swarm Optimization [26] is an optimization algorithm based on swarm intelligence, which originated from the study of bird predation behavior. For swarm

intelligence algorithms, the problem is seen as searching for the optimal solution in a multi-dimensional space. Each search point is called a "particle" and finds the optimal solution by continuously updating its position and velocity. The algorithm steps include initializing the particle swarm, computing the fitness function, updating particle velocity, updating particle position, updating the historical best position and global best position, and determining the stop condition. The algorithm process is as follows:

1. Initialize particle swarm: The number of particles $G = [\beta_1^0, \beta_2^0, \ldots, \beta_{N_p}^0]$ in the feasible solution space, as well as parameters such as position and velocity for each particle, while randomly generating the position and velocity for each particle.
2. Compute fitness function: Determine the fitness function based on the characteristics of the problem, and evaluate the performance of each particle. The fitness function can be a maximization or minimization problem, depending on the specific problem.
3. Update particle velocity: Update the velocity of each particle based on its historical best position and the global best position. The velocity update formula is as follows:

$$v_i^{t+1} = \omega \times v_i^t + c_1 r() \times (p_i - x_i^t) + c_2 r() \times (g_i - x_i^t), \tag{2}$$

   where ω represents particle inertia weight, v_i^t represents the velocity of the i^{th} particle in the t^{th} iteration, c_1, c_2 are the learning factor, they represent individual and social learning factors, respectively, p_i, g_i represent the best positions of the particle in the previous t iterations and the best positions of all particles in the previous t iterations, and x_i^t represents the position of the i^{th} particle in the t^{th} iteration, $r()$ is a uniform random number within the range of $[0, 1]$.
4. Update particle position: Update the position of each particle based on the updated velocity. The position update formula is as follows:

$$x_i^{t+1} = x_i^t + \omega v_i^t. \tag{3}$$

5. Update historical best position and global best position: Based on the current fitness function value, update the historical best position and global best position of each particle. If the fitness value of the current position is better than that of the historical best position, update the historical best position; if the fitness value of the current position is better than that of the global best position, update the global best position.
6. Determine stop condition: Check whether the algorithm has reached the preset stop condition, such as reaching the preset iteration number or fitness function value below a threshold. If the stop condition is met, output the global optimal solution; otherwise, return to step 3. and continue the algorithm.

3.3 Cosine Similarity

Cosine similarity [27] is a method used to measure the similarity between two vectors. The similarity value ranges from -1 to 1, with a value closer to 1 indicating that the two vectors are more similar, and a value closer to -1 indicating that the two vectors are less similar. The calculation formula is as follows:

$$\text{cosine\_similarity} = \frac{A \cdot B}{\|A\| * \|B\|}, \tag{4}$$

where $A \cdot B$ represents the dot product of vectors A and B, $\|A\|$ represents the magnitude of vector A, and $\|A\|$ represents the magnitude of vector B.

4 SMRFC-PDCNN ALGORITHM DESCRIPTION

Different from conventional scene recognition algorithms that extract and fuse features at multiple scales, this paper focuses on improving the speed and accuracy of scene recognition algorithms in large datasets. It achieves this by optimizing the recognition structure of DCNNs and using the Spark parallel computing framework. The proposed algorithm, SMRFC-PDCNN, is used in the optimization part of the DCNNs for scene recognition. The designed model structure in this paper performs parallel recognition on multiple Spark nodes, with each task node running the SMRFC-PDCNN algorithm. This algorithm calculates the mutual information coefficient of feature maps during the pooling stage to select appropriate pooling methods and improve model accuracy. It then clusters the one-dimensional feature vectors formed by the final feature maps, and samples within the clusters for computation in the fully connected layer to reduce redundant feature computations, accelerates the algorithm process, and calculates the node load during the entire process. Computation is allocated to achieve load balancing. Figure 1 provides the scene recognition model structure.

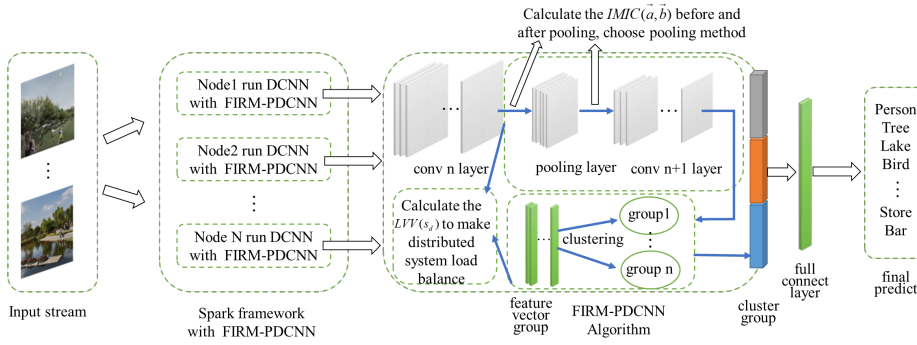


Figure 1. The flowchart of scene recognition model with SMRFC-PDCNN algorithm

4.1 Parallel Pooling Selection

In scene recognition using parallel DCNN algorithms, there is currently an issue with reduced feature map accuracy during the recognition process. In this stage, we propose a solution to this problem using a pool selection strategy based on improved mutual information (MI-IPSS). This strategy is composed of two main steps, which are discussed in detail below:

1. Feature comparison: An improved mutual information correlation coefficient, referred to as IMIC($\vec{a}, \vec{b}$) has been proposed for the purpose of calculating and comparing the IMIC($\vec{a}, \vec{b}$) value of feature maps before and after pooling in parallel during recognition. This innovative approach provides an effective indicator for selecting the optimal pool.
2. Pool selection: We are comparing two different IMIC($\vec{a}, \vec{b}$) values. One was obtained from the previous pooling layer, while the other was obtained by combining all the IMIC($\vec{a}, \vec{b}$) values from the entire distributed system.

By evaluating the difference between these two IMIC($\vec{a}, \vec{b}$) value aggregations, we can assess the effectiveness of the previous pooling strategy and determine whether it is suitable for the current datasets. This allows us to address the problem of reduced feature map accuracy and select the most appropriate pooling method for our needs.

4.1.1 Feature Comparison

To select the most suitable pooling method for a given data set, it is crucial to compare the feature maps both before and after each pooling layer in a distributed DCNN. To accomplish this, we propose an improved mutual information correlation coefficient, called IMIC($\vec{a}, \vec{b}$), to evaluate the adaptability of the current pooling method to the data set, based on the similarity between the feature maps before and after pooling. Here is the specific process we follow: We first divide the image files into several blocks, and then use OpenCV to convert the initial image data format and assign an index to each block, which is then stored in HDFS and input into the Spark job. Then, for each block, we run a Mapper and allocate a pre-trained DCNN network model to it, and input the data in the block into the network model. When the data reaches the pooling layer, the mutual information correlation coefficient IMIC($\vec{a}, \vec{b}$) for all feature maps in the Mapper is calculated, and the overall mutual information correlation coefficient IMIC($\vec{a}, \vec{b}$) of the system is aggregated. Finally, the difference between the overall IMIC($\vec{a}, \vec{b}$) aggregation value of the previous pooling and the current pooling is compared to determine the next pooling method selection.

Theorem 1. Improved mutual information correlation coefficient, IMIC($\vec{a}, \vec{b}$). Assume that $\vec{a}$ represents the one-dimensional vector representation of the feature map before pooling, and $\vec{b}$ represents the one-dimensional vector representation of

the feature map after pooling. The calculation formula for the correlation coefficient $\mathrm{IMIC}(\vec{a},\vec{b})$ is as follow.

$$\mathrm{IMIC}(\vec{a},\vec{b}) = \frac{p(\vec{a},\vec{b})}{1+d(\vec{a},\vec{b})} \times \log_2 \frac{p(\vec{a},\vec{b})}{p(\vec{a})p(\vec{b})}, \tag{5}$$

where $p(\vec{a},\vec{b})$ represents the joint probability density of $\vec{a}$ and $\vec{b}$, $p(\vec{a})$ and $p(\vec{b})$ represent the probability density of $\vec{a}$ and $\vec{b}$, respectively, and $d(\vec{a},\vec{b})$ represents the projection distance from $\vec{a}$ and $\vec{b}$.

Proof. $\mathrm{IMIC}(\vec{a},\vec{b})$ represents the correlation coefficient between the feature vectors of $\vec{a}$ and $\vec{b}$. When $\vec{a}$ and $\vec{b}$ are dissimilar, $d(\vec{a},\vec{b})$ approaches infinity, causing $1/(1+d(\vec{a},\vec{b}))$ to approach zero, while the probability density $p(\vec{a},\vec{b})$ of $\vec{a}$ and $\vec{b}$ occurring simultaneously will also decrease, for this reason, the $p(\vec{a},\vec{b}) \cdot \log_2(p(\vec{a},\vec{b})/p(\vec{a})p(\vec{b}))$ will also decrease. As a result, the value of $\mathrm{IMIC}(\vec{a},\vec{b})$ will be in a low-level position.

On the contrary, when $\vec{a}$ and $\vec{b}$ are similar, $d(\vec{a},\vec{b})$ tends to zero, causing $1/(1+d(\vec{a},\vec{b}))$ to approach 1, and the probability density $p(\vec{a},\vec{b})$ of $\vec{a}$ and $\vec{b}$ occurring simultaneously will also increase, and the mutual information $p(\vec{a},\vec{b}) \cdot \log_2(p(\vec{a},\vec{b})/p(\vec{a})p(\vec{b}))$ will increase accordingly. At this time, the value of $\mathrm{IMIC}(\vec{a},\vec{b})$ is in a high-level position.

Therefore, when $\vec{a}$ and $\vec{b}$ are dissimilar, the value of $\mathrm{IMIC}(\vec{a},\vec{b})$ is small, and when $\vec{a}$ and $\vec{b}$ are similar, the value of $\mathrm{IMIC}(\vec{a},\vec{b})$ is large. It can be used as an indicator to measure the correlation coefficient of feature vectors. □

4.1.2 Pool Selection

After completing the calculation of the $\mathrm{IMIC}(\vec{a},\vec{b})$ of the system, the $\mathrm{IMIC}(\vec{a},\vec{b})$ of the entire distributed system is aggregated to obtain $\sum^{j} \mathrm{IMIC}(\vec{a},\vec{b})$ and save it to the variable sum_this, and compare it with the last accumulated value sum_last comparison to select the pooling method suitable for the current data, the specific process is as follows: First, set the pooling matrices for $1*1$, $2*2$, $3*3$, and $4*4$, and set the stride of the pooling matrix to $1, 1, 2, 2$. Initially, use a pooling matrix of size $4*4$ for computation. After that, subtract sum_this from sum_last to calculate whether there is a significant decrease in feature accuracy. If sum_this − sum_last > 0, it means that the pooling method used in this iteration is acceptable, so it is retained. If sum_this − sum_last < 0, it indicates a significant decrease in feature accuracy, so the matrix size and step size need to be reduced until the data is computed into a one-dimensional feature vector. Then, the next step of feature selection strategy, DCPSO-FSS based on density clustering and particle swarm optimization will be performed.

4.2 Parallel Feature Clustering

In scene recognition, the DCNN algorithm uses a large amount of data to parallelly compute the fully connected layer during the recognition process. However, a significant issue arises with the feature vectors of repeated class data that generates a lot of redundant computation. This problem can be particularly severe in big data environment, where excessive feature redundancy calculation can occur. To address this issue, a feature selection strategy called DCPSO-FSS has been designed. This strategy is based on density clustering and particle swarm optimization. It starts by proposing an adaptive feature selection particle swarm optimization algorithm that finds the clustering parameters of feature vectors before parallel DCNN model computing to the fully connected layer. After clustering and sampling these feature vectors, they are used for recognition in the fully connected layer. By utilizing this feature selection strategy, the DCNN algorithm can significantly reduce the computational load and processing time. Moreover, it can enhance the recognition accuracy and improve the efficiency of the entire scene recognition system. This strategy mainly includes two steps:

1. feature selection clustering: an adaptive particle speed is proposed to adaptively select particle speed based on clustering feature vectors, and a characteristic graph fitness function (CGFF) is designed to quickly obtain accurate density clustering parameters;
2. inter-group recognition of feature vectors: after completing the clustering of feature vectors in the previous stage, the data of each category is sampled for full connection operation, and the top5 accuracy of final classification is evaluated to determine the category of the entire group.

4.2.1 Feature Selection Clustering

To reduce the redundant calculations caused by the generation of a large number of repetitive data feature vectors, an adaptive particle speed algorithm (v^{t+1}) and a characteristic graph fitness function (CGFF) have been designed. These techniques aim to quickly locate the density clustering parameters (ε and $minpts$), increase the accuracy of feature clustering, and reduce the computational cost of recognition operations. The specific process is as follows: First, in the initial stage of the algorithm, the Master node reads all the feature vector collections that were parallel computed in the previous stage from HDFS. It then proceeds to determine the initialization range and number of particle swarm based on the number of feature vectors available. Once this is done, the Master node randomly generates the position and velocity vectors of the particles. Next, the algorithm calculates the fitness function value of each particle according to the CGFF. The maximum number of iterations $Maxiter$ is also set at this stage. To ensure efficient convergence, the velocity of each particle is calculated using the adaptive particle speed v^{t+1} formula. Using this formula, the velocity and position of each particle are then updated. Then, the algorithm checks

whether the maximum number of iterations $Maxiter$ has been reached or whether the CGFF value meets the predetermined accuracy requirement. If either of these conditions are met, the algorithm is stopped. This ensures that the algorithm does not run unnecessarily and saves computational resources. Finally, after obtaining the density clustering parameters ε and $minpts$, all feature vectors are clustered, and the recorded category and feature vectors are stored in HDSF to complete the feature selection clustering.

Theorem 2. Adaptive particle velocity, v^{t+1}. Given v_i^t is the velocity of the i^{th} particle in the t^{th} iteration, c_1, c_2 are the learning factors representing the individual and social learning factors, respectively. p_i, g_i represent the best positions of the particle in the previous t iterations and the best positions of all particles in the previous t iterations, respectively. x_i^t represents the position of the i^{th} particle in the t^{th} iteration, while $\vec{X}$ and $\vec{Y}$ represent the two clusters formed by feature vectors. The formula for calculating the adaptive particle velocity v^{t+1} is as follows.

$$v_i^{t+1} = v_i^t \Big/ \sqrt{\sum_{k=1}^{n} \left(\left\| \overrightarrow{X} \right\| - \left\| \overrightarrow{Y} \right\| \right)^2} + c_1 r() \times (p_i - x_i^t) + c_2 r() \times (g_i - x_i^t). \tag{6}$$

v^{t+1} is an adaptive particle velocity based on feature vectors, and the reciprocal of λ represents the Euclidean distance between the two clusters formed by $\vec{X}$, $\vec{Y}$.

Proof. When $\vec{X}$, $\vec{Y}$ are very similar, the particle swarm needs a large inertia to drive the update of the particle position, and at this time, the value of λ is large, which satisfies the requirement of a large inertia for the particle swarm. When $\vec{X}$, $\vec{Y}$ are dissimilar, it means that the particle swarm needs to reduce its inertia to search for the optimal position, and at this time, the value of λ is small, which satisfies the requirement of a small inertia for the particle swarm. Therefore, v^{t+1} can meet the requirements of velocity update in the particle swarm algorithm very well. □

Theorem 3. Fitness function of characteristic graph, $CGFF$. Given a clustering with a total number of q clusters, where c_z represents the total average value of the moduli of all feature vectors in the z^{th} cluster, and p represents the feature vectors within the cluster. $\left\| \vec{X} \right\|$ and $\left\| \vec{Y} \right\|$ represent the moduli of the two clusters formed by the feature vectors. The calculation formula for the compactness and the separability fitness function (CGFF) is as follows.

$$\text{CGFF} = \min \left(\sum_{z=1}^{q} \sum_{p \in C_z} \text{dist}(c_z, p)^2 + \sum_{z=1}^{q} \sum_{X,Y \in C_z} \frac{\vec{X} \cdot \vec{Y}}{\left\| \overrightarrow{X} \right\| + \left\| \overrightarrow{Y} \right\|} \right). \tag{7}$$

Proof. CGFF is a function that describes the fitness of feature map clustering. According to the principle of high intra-cluster similarity and low inter-cluster similarity, this function calculates the distance square of $\text{dist}(c_z, p)^2$. When the magnitude

of the feature vectors in the cluster is closer to the average feature vector in the cluster, the smaller the value of $\text{dist}(c_z, p)^2$ is, and the closer it is to the minimum value of CGFF. When two feature vector clusters $\vec{X}$ and $\vec{Y}$ are dissimilar, the value of $\vec{X} \cdot \vec{Y} / \left\|\vec{X}\right\| + \left\|\vec{Y}\right\|$ also becomes smaller. Therefore, CGFF meets the requirements for the fitness function of feature map density clustering and can be used to select parameters for feature map density clustering. □

4.2.2 Inter-Group Recognition of Feature Vectors

After completing feature selection clustering, the algorithm utilizes the parallel processing power of the Spark framework to perform the final recognition using fully connected layers on the clustered feature vectors. The process can be divided into several steps: Firstly, each category is randomly sampled at a ratio of 20 % and distributed to various computing nodes for fully connected operations. The purpose of this step is to obtain the feature prediction results for each category. Next, the current category recognition top5 accuracy is verified in the validation set. If the accuracy exceeds 90 %, the algorithm determines that the current clustered features belong to the same category. However, if the accuracy does not meet the predetermined value, the feature selection clustering process is repeated until the algorithm's top5 accuracy reaches the desired level. This iterative process of feature selection clustering and validation continues until the desired level of accuracy is achieved. Once the algorithm reaches the desired level of accuracy, the feature vectors are clustered together into the same category, and the recognition process is considered complete.

4.3 Feature Dynamic Load Balancing

In scene recognition, traditional distributed systems for parallel DCNN algorithms have typically used round-robin or least-connection strategies to achieve load balancing. These strategies work well for tasks with a fixed amount of computation. However, in the case of parallel DCNN recognition algorithms, deep pooling and convolution can result in significant changes in the computational load of distributed nodes. This, in turn, can lead to low parallel recognition efficiency. To address this issue, a new load balancing strategy called CCG-LBS has been developed. This strategy is based on cluster feature maps and dynamically calculates the computational cost of feature maps for each distributed node. It then allocates data among groups based on this cost, achieving dynamic load balancing. The CCG-LBS strategy works as follows: first, several DCNN models are assigned to each node, with only one model in the active state to input data for calculation. This ensures that the computing nodes reach their maximum load at the beginning. After each round of convolution and pooling, the load of each node is calculated and proposed as $\text{LVV}(s_d)$. Then, when the load is below the load threshold, a new model is enabled and new data is added for calculation on that node. The amount of input data

is LVV(s_d), and dynamic load balancing is achieved. Finally, the balanced data is calculated by each model to obtain the final recognition result.

Theorem 4. Balance load quantity, LVV(s_d). Given that FML(s_d) represents the computation load of all feature maps for the d^{th} computing node, where s_d is the feature map, FML(s_{max}) represents the maximum node load, and $\sum_{d=1}^{N} \text{FML}(s_d)$ represents the total load of all nodes, where N represents the total number of nodes. The calculation formula for the load with respect to its quantity LVV(s_d) is as follows.

$$\text{LVV}(s_d) = \text{FML}(s_{max}) - \text{FML}(s_d) + \sum_{d=1}^{N} \text{FML}(s_d)/N. \tag{8}$$

Proof. LVV(s_d) is a value used to supplement input data and balance the workload of a distributed system when there is an imbalance due to convolutional pooling operations on feature maps. The formula for LVV(s_d) can be split into two parts. The incremental part, represented by FML(s_{max})−FML(s_d), indicates the difference in workload calculation between the maximum node and the current node. This difference represents the amount by which the system's remaining nodes can still operate at maximum capacity, and can be used to fill in the missing workload of deficient nodes. In addition, when the workload of a node is below the load threshold η, the maximum capacity node is not operating at full capacity. Therefore, based on the current system average workload, $\sum_{d=1}^{N} \text{FML}(s_d)/N$ can be used to supplement the missing workload of the current node. □

4.4 Time Complexity Analysis

The SMRFC-PDCNN is composed of three stages: parallel pooling selection, parallel feature clustering and feature dynamic load balancing.

The stage of parallel pooling selection: Set the number of samples be n, the number of cluster nodes be k, and the total number of pooling layers be s. Then the time complexity of parallel pooling selection is $O(s * n^2/k + n)$.

The stage of parallel feature clustering: Set the number of samples be n, the number of cluster nodes be k, the initial population number of particle swarm optimization algorithm be p, and the overall evolution times be t. Then the time complexity of parallel feature clustering is $O(p * n^2/k + t * n)$.

The stage of feature dynamic load balancing: Set the number of samples be n and the number of cluster nodes be k. Then the time complexity of feature dynamic load balancing is $O(n^2/k)$.

The time complexity of the SMRFC-PDCNN algorithm is $O(n^2/k)$.

5 EXPERIMENTAL EVALUATION

Here, the experimental setup will be discussed, the experimental results as well as the corresponding analysis.

5.1 Experimental Setup

Experiments was performed in a Spark cluster of five nodes (1 Master node, 7 Slaver nodes). Each node contains an Intel i9 13 900 k, RTX 4080, 64 GB RAM, a 8 TB SSD, and each node is connected through a 1 000 Mb/s network. The algorithm is implemented in Java, and runs on CentOS 7.2, the JDK version is 17. The node configuration is presented in Table 1.

Node Type	Double Scan	Triangle
Primary Node	Primary Node	192.168.2.116
Secondary Node	Slaver	192.168.2.117 ∼ 120

Table 1. Node configuration information

5.2 Experimental Datasets

To verify the feasibility and effectiveness of the SMRFC-PDCNN, four datasets from different domains are used for the experiments, namely PASCALVOC2012, COCO 2012, Caltech256, and ImageNet. The details of the above datasets are shown in Table 2.

	PASCALVOC2012	Caltech256	COCO	ImageNet
Sample size	17 125	30 607	3 300 000	14 197 122
Image size	20	256	80	1 000
Categories	256 ∗ 256	256 ∗ 256	32 ∗ 32	32 ∗ 32 ∼ 160

Table 2. Experimental datasets

The PASCALVOC2012 data set provides a standard image annotation data set and a standard evaluation system for detection algorithms and learning performance. The official data set consists 17 125 records and every record consists of 20 items. This data set is widely used for small-scale scene recognition.

The Caltech256 dataset is a data set collected by the California Institute of Technology which contains 30 607 records, each including 256 items which is widely used in tasks such as scene recognition and object detection.

The COCO is a large, rich object detection, segmentation and captioning data set and can be used for scene recognition and image detection, which contains 3 300 000 records, each including 80 items. It is widely used for scene recognition and matching.

The ImageNet is a computer vision system recognition project, which is currently the largest scene recognition database in the world. It consists of 14 197 122 records, and each record includes 1 000 items.

5.3 Experimental Metrics

5.3.1 Speedup Ratio

Speed-up ratio [28] is used as a substantial indicator to compute the parallelization performance of the algorithm. The speed-up ratio is the ratio of time, defined as follows.

$$S_p = T_1/T_p, \tag{9}$$

where T_1 and T_p denote the running time of the algorithm on a single node and in parallel, respectively. The larger S_p is, the relative time spent in parallel computing is less, and the cluster efficiency is greater.

5.3.2 Top5 Accuracy

Top5 accuracy [29] refers to the accuracy of the model's top 5 predictions in a multi-classification task, by comparing the model's predicted results with the true labels, to determine whether the correct label is included among the top 5 most likely predictions.

$$\text{Top5Acc} = \frac{\text{correct}(n)}{n} \times 100\,\%. \tag{10}$$

In this case, n represents the total number of samples in the test set, correct(n) represents the number of samples in which the correct label is included among the top 5 predicted labels. If the top 5 most likely labels predicted by the model include the correct label, it is considered a correct prediction.

5.4 Data Set Texture Enhancement

To improve the reliability of the data set, texture augmentation was applied to four datasets in the experimental section. Firstly, the data was decoded from the original image file format to in-memory image data using OpenCV, and the pixel values were normalized to the range of $[0, 1]$. Secondly, Local Binary Patterns (LBP) were used to extract texture features from the initial data, capturing the texture information in the images. The calculation formula for Local Binary Patterns is as follows.

$$\text{LBP}(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c) * 2^p. \tag{11}$$

In the formula, (x_c, y_c) represents the coordinates of the central pixel, g_c represents the grayscale value of the central pixel, g_p represents the grayscale value of the p^{th} pixel in the circular neighborhood centered at the central pixel with a radius of R. P represents the number of pixels in the neighborhood, and $s()$ is the sign function. It outputs 1 when the parameter is greater than or equal to 0, and 0 when it is less than 0.

Finally, Adaptive Histogram Equalization (AHE) was employed to enhance the contrast and details of local regions in the image, adapting to the grayscale distribution characteristics of different image regions. The calculation formula for Adaptive Histogram Equalization is as follows.

$$\mathrm{CDF\_eq}\,(x,y,k) = \frac{\sum_{i=0}^{k} H(x,y,i) * (L-1)}{(N*M)}. \tag{12}$$

In this formula, $H\,(x,y,k)$ represents the histogram of the local neighborhood $N\,(x,y)$, where $N\,(x,y)$ is a fixed-size rectangular window or a circular window with a fixed radius centered at (x,y). N and M represent the width and height of the neighborhood $N\,(x,y)$, respectively. k represents the intensity levels ranging from 0 to $L-1$, where L is the number of intensity levels in the image. $\mathrm{CDF\_eq}\,(x,y,k)$ represents the enhanced pixel value in the output image after adaptive histogram equalization.

5.5 Experimental Analysis

The verification of the performance of the SMRFC-PDCNN algorithm. We conduct a comparative experiment on the above experimental datasets, and compare the performance of the algorithm among the PDCNNO algorithm, the RS-DCNN algorithm and the BDCNN algorithm. We evaluate the speedup ratio, top5 accuracy, reduction of FLOPs and running time of the four algorithms, respectively.

5.5.1 Acceleration Performance Analysis

To verify the feasibility of SMRFC-PDCNN, the experiment compares the acceleration effect of SMRFC-PDCNN on the datasets of PASCALVOC2012, Caltech256, COCO and ImageNet. In addition, to ensure the accuracy of the experimental results, the average of the speedup ratio of running the algorithm 10 times was taken as the final experimental results. The Speedup radio on four data sets is presented in Figure 2.

It can be seen from Figure 2 that the speedup ratio of the SMRFC-PDCNN algorithm increases steadily with the increase of the number of nodes. When the number of nodes is 2, the SMRFC-PDCNN algorithm has little difference in the speedup ratios of the four data sets; when the number of nodes is 4, the speedup ratios of the algorithm are increased by 2.655, 2.457, 3.541 and 4.814; when the number of nodes is 8, the SMRFC-PDCNN algorithm has significantly improved in each data set, reaching 6.921, 7.384, 9.218 and 8.114, respectively.

These results are produced because the SMRFC-PDCNN algorithm designs a feature selection DCPSO-FSS strategy. The strategy proposes adaptive particle velocity and quickly finds the feature vector clustering parameters through adaptive inertia. Then, these feature vectors are clustered and sampled to the fully connected layer for recognition, which significantly improves the speedup ratio of scene recognition. As the data scale increases, the improved effect of the algorithm becomes more

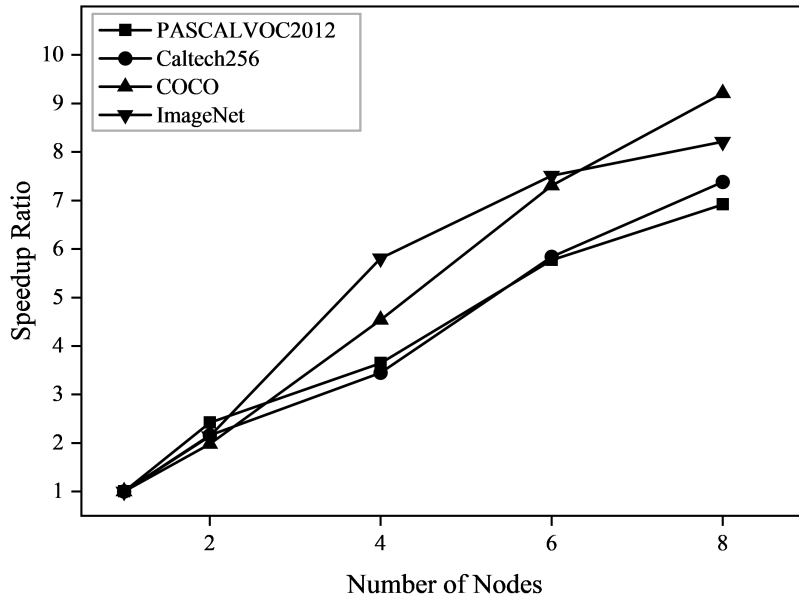


Figure 2. Speedup radio of SMRFC-PDCNN on four datasets

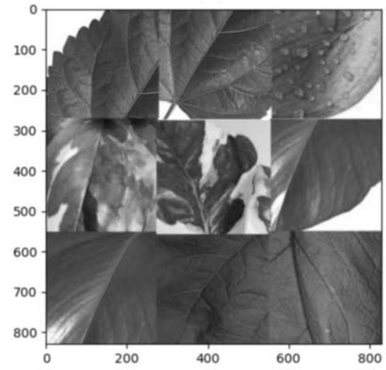
and more obvious. This also shows that the SMRFC-PDCNN algorithm is suitable for scene recognition, and the model parallel recognition of deep convolutional neural networks.

5.5.2 The Strategy of MI-IPSS Analysis

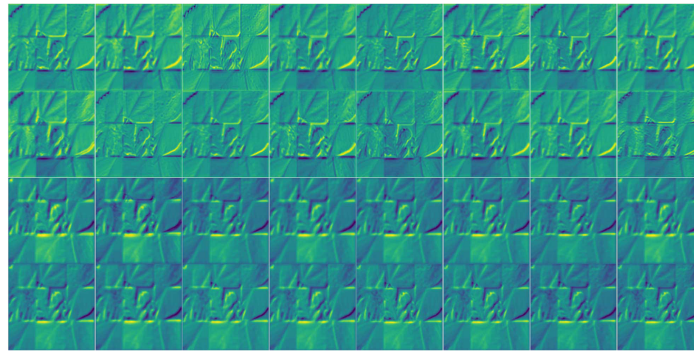
To analyze the effect of the MI-IPSS strategy on the accuracy of the feature map, we visualized the feature maps of the SMRFC-PDCNN algorithm on whether to use the MI-IPSS strategy on VGG-16 model. The visualization result is presented in Figure 3.

It can be seen from Figure 3, compared with the algorithm without using the MI-IPSS strategy, the feature information of the feature map on each convolutional layer is more abundant after the algorithm uses the MI-IPSS strategy, and as the network deepens, the strategy extracted Feature information is more efficient and centralized. We can see in Figure 3 c), after using the MI-IPSS strategy, the feature information of each feature map is relatively clear, and the light-dark boundary line in the feature information is relatively gentle compared with Figure 3 b).

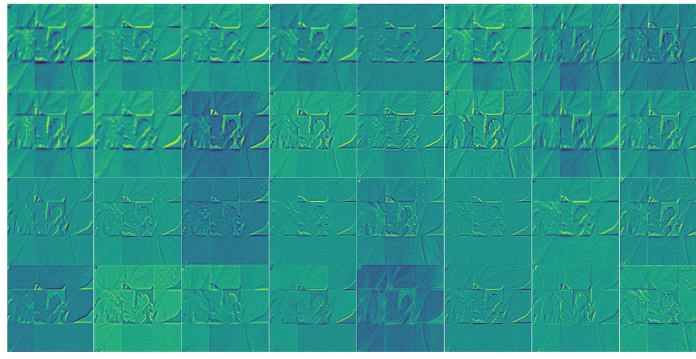
The reason for this phenomenon is that the MI-IPSS strategy selects the appropriate pooling strategy through feature comparison, and the appropriate pooling strategy enhances the ability of the SMRFC-PDCNN algorithm to extract feature information, which effectively improves the accuracy of scene recognition. In summary, the MI-IPSS strategy is feasible, and using the MI-IPSS strategy can signif-



a) The source data



b) The Feature map without MI-IPSS strategy



c) The Feature map using MI-IPSS strategy

Figure 3. The visualization results on whether to use the MI-IPSS

icantly improve the performance of the SMRFC-PDCNN algorithm for extracting feature information.

5.6 Performance of SMRFC-PDCNN

To verify the performance of the SMRFC-PDCNN algorithm. We conduct a comparative experiment on the above experimental datasets, and compare the performance of the algorithm among the PDCNNO algorithm, the RS-DCNN algorithm and the BDCNN algorithm. Evaluate the speedup ratio, top5 accuracy, reduction of FLOPs and running time of the four algorithms respectively.

5.6.1 The Runtimes of Four Algorithms

To investigate the training time consumed by SMRFC-PDCNN in multimodal data, the runtimes required for the PDCNNO, RS-DCNN, BDCNN algorithms to achieve stable training accuracy on four datasets are recorded and compared in the experiments. The experimental results are shown in Figure 4.

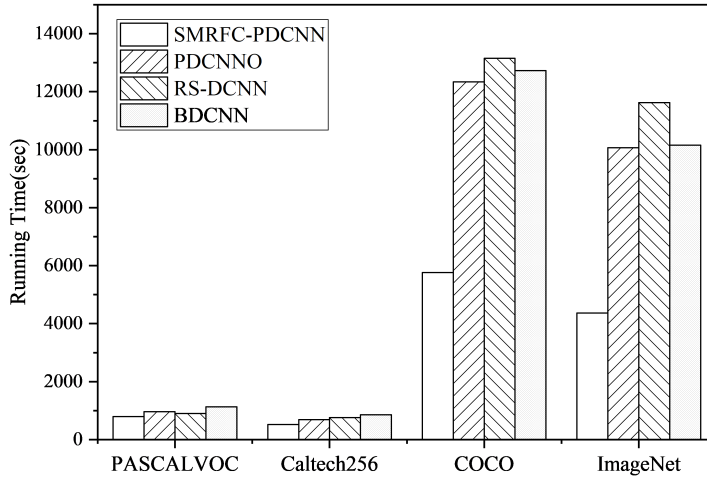


Figure 4. The training time of each algorithm on four datasets

From Figure 4, it can be seen that for small-scale datasets like PASCALVOC2012 and Caltech256, there is little difference in recognition time among the different algorithms. However, for large-scale datasets such as COCO and ImageNet, the SMRFC-PDCNN algorithm significantly outperforms PDCNNO, RS-DCNN, and BDCNN, reducing their running times by 6 581 s, 7 396 s, 6 970 s, and 5 698 s, 7 253 s, 5 789 s, respectively. Notably, the SMRFC-PDCNN algorithm demonstrates a remarkable reduction in training time as the training data size increases, outpacing PDCNNO, RS-DCNN, and BDCNN by a considerable margin. These results are clearly depicted in the graph.

These results were achieved because the SMRFC-PDCNN algorithm proposes a load balancing strategy called CCG-LBS, which is based on clustered feature maps. By dynamically calculating the computational cost of feature maps on distributed system nodes and distributing data between groups based on this cost, the algorithm achieves dynamic load balancing of data, thereby avoiding slowing down the overall speed of the algorithm due to uneven distribution of computational resources. As a result, the SMRFC-PDCNN algorithm significantly reduces running time compared to PDCNNO, RS-DCNN, and BDCNN algorithms. In conclusion, the SMRFC-PDCNN algorithm is superior to PDCNNO, RS-DCNN, and BDCNN algorithms, and is suitable for model parallelization of deep convolutional neural networks in scene recognition.

5.6.2 The Top5 ACC and FLOPs of Four Algorithms

In order to verify the accuracy and model optimization effectiveness of the SMRFC-PDCNN algorithm in a big data environment, this study computes the top5 accuracy and FLOPs of the Baseline, SMRFC-PDCNN, PDCNNO, RS-DCNN, and BDCNN algorithms using above four datasets. The Baseline refers to the VGG16 model's benchmark data under 1/8 data load. The experimental results are presented in Table 3.

Dataset	Algorithm	Top5 Acc	FLOPs	Reduction of FLOPs
PASCAL VOC2012	Baseline	87.18 %	3.26×105	–
	PDCNNO	91.75 %	2.97×105	8 %
	RS-DCNN	90.78 %	3.06×105	6 %
	BDCNN	89.56 %	3.16×105	3 %
	SMRFC-PDCNN	94.66 %	1.36×105	58 %
Caltech256	Baseline	86.81 %	7.59×106	–
	PDCNNO	90.78 %	6.31×106	17 %
	RS-DCNN	91.69 %	6.38×106	16 %
	BDCNN	87.74 %	7.29×106	4 %
	SMRFC-PDCNN	93.75 %	2.73×106	64 %
COCO	Baseline	88.38 %	1.76×107	–
	PDCNNO	91.58 %	1.41×107	20 %
	RS-DCNN	94.87 %	1.65×107	6 %
	BDCNN	93.79 %	1.49×107	15 %
	SMRFC-PDCNN	95.73 %	1.16×107	34 %
ImageNet	Baseline	91.92 %	9.72×107	–
	PDCNNO	96.24 %	7.39×107	24 %
	RS-DCNN	95.64 %	9.14×107	6 %
	BDCNN	92.48 %	8.16×107	16 %
	SMRFC-PDCNN	97.78 %	5.54×107	43 %

Table 3. The accuracy rate and FLOPs of each algorithm

From Table 3, it is evident that when dealing with relatively small datasets such as PASCALVOC2012 and Caltech256, each algorithm's FLOPs are reduced to varying degrees. Notably, compared to PDCNNO, RS-DCNN, and BDCNN algorithms, SMRFC-PDCNN algorithm has 14 %, 28 %, 19 %, and 19 %, 37 %, and 27 % lower FLOPs, respectively. On the other hand, when processing larger datasets such as COCO and ImageNet, the SMRFC-PDCNN algorithm exhibits higher top5 accuracy and lower FLOPs than other three algorithms. Specifically, the SMRFC-PDCNN algorithm's top5 accuracy is 3.07 %, 2.01 %, 6.02 %, and 3.11 %, 4.08 %, 5.31 % higher than PDCNNO, RS-DCNN, and BDCNN algorithms, respectively. Furthermore, the SMRFC-PDCNN algorithm's FLOPs are reduced by 50 %, 52 %, 55 %, and 47 %, 48 %, 60 %, respectively.

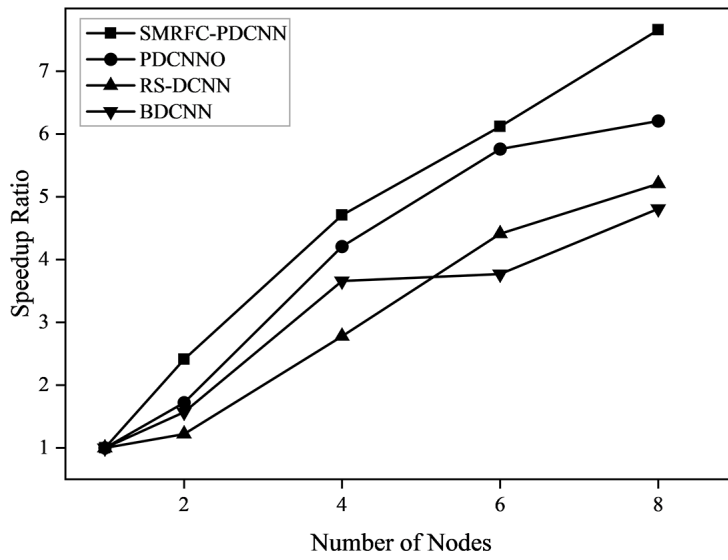
These remarkable results stem from SMRFC-PDCNN's MI-IPSS pooling selection strategy, which is based on improved mutual information. It judges the current pooling method's adaptability to the current dataset by comparing the similarity of relevant features before and after pooling. This addresses the problem of accuracy degradation of feature maps and allows the algorithm to maintain stable accuracy while significantly reducing FLOPs. Experimental data demonstrate that SMRFC-PDCNN has higher convergence speed and accuracy than the other three parallel algorithms, making it suitable for model parallelism recognition in scene recognition using deep convolutional neural networks.

5.6.3 The Speedup Ratio of Four Algorithms

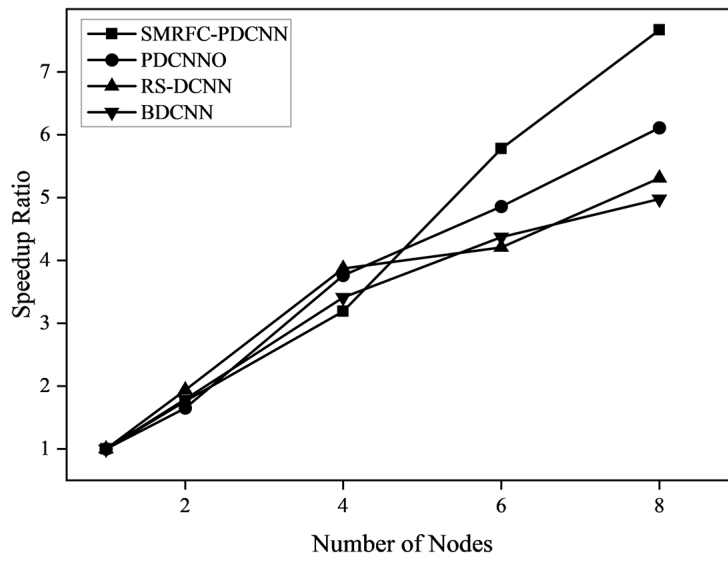
In order to compare the parallel computing performance of SMRFC-PDCNN in a large data environment, the PDCNNO, RS-DCNN, and BDCNN algorithms were tested 10 times on each of the four data sets, and the average of the speedup ratios of each algorithm with different numbers of computational nodes was used as the comparison standard. The experimental results are shown in Figure 5.

From Figures 5 a) and 5 b), it can be seen that when dealing with relatively small-scale data sets such as PASCALVOC2012 and Caltech256, the speedup ratios of the four algorithms increase slowly as the number of nodes increases. Among them, when the number of cluster nodes When it is 4, the acceleration ratio of SMRFC-PDCNN is 1.93, 1.05, 0.22, and 0.68 lower than that of the BDCNN and RS-DCNN algorithms with a low degree of parallelization; but in Figure 5 c) and 5 d), it can be seen that when dealing with relatively large data sets such as COCO and ImageNet, the acceleration ratio of the SMRFC-PDCNN algorithm is relatively large, reaching 8.98 and 9.07 when the number of cluster nodes is 8, which are higher than those of PDCNNO and RS-DCNN respectively. And the BDCNN algorithm is 1.47, 2.31, 1.89 and 1.41, 3.43, 3.39 higher.

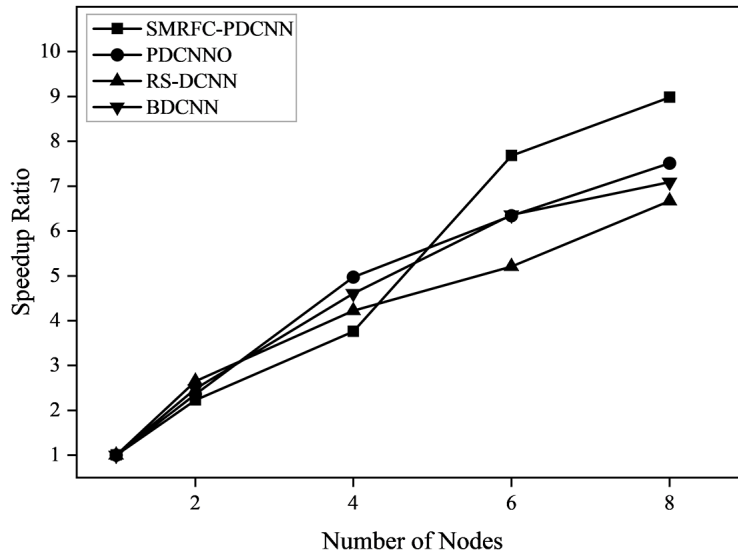
The reason for these results is that when the SMRFC-PDCNN algorithm is processing relatively small-scale data sets such as PASCALVOC2012 and Caltech256, the distribution of data to each computing node will lead to a rapid increase in the communication time overhead between nodes. The running speed improvement is extremely limited; when the SMRFC-PDCNN algorithm is dealing with relatively



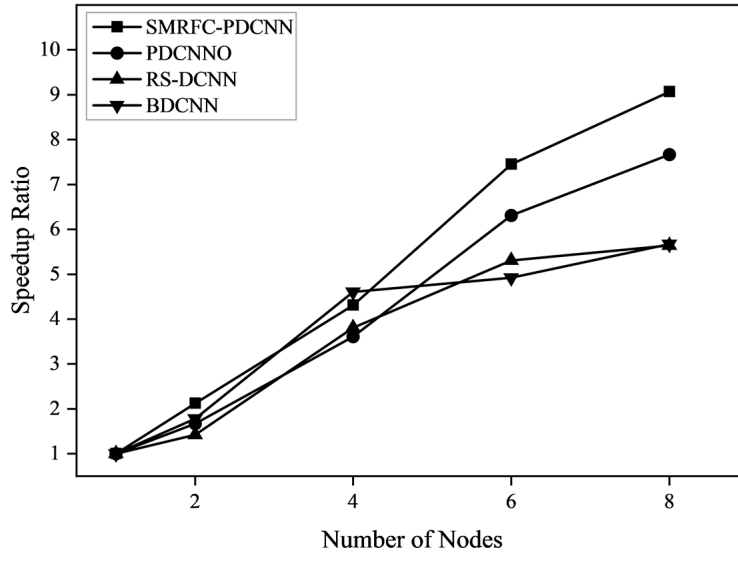
a) PASCALVOC2012



b) Caltech256



c) COCO



d) ImageNet

Figure 5. Speedup radio of SMRFC-PDCNN on four datasets

large data sets such as COCO and ImageNet, because the algorithm proposes the CGFF function in the DCPSO-FSS strategy, the particle swarm can quickly converge to find the applicable Based on the clustering parameters of the feature map, the speedup ratio of the algorithm is improved, and compared with the other four algorithms, the speedup ratio improvement effect is also more obvious. The experiment shows that the parallelization ability of SMRFC-PDCNN algorithm increases significantly with the increase of the number of cluster nodes. Enhanced, it is suitable for scene recognition, model parallel recognition of deep convolutional neural network, and has better performance.

6 CONCLUSIONS

To address the issues of decreased accuracy of feature maps, more redundant feature calculations and low efficiency in parallel recognition process of deep parallel convolutional neural networks, this paper proposes an efficient Scene Matching Recognition with DCNN and Feature Clustering on Spark, named SMRFC-PDCNN.

Firstly, a pool selection strategy MI-IPSS, based on improved mutual information, is proposed. It adaptively adjusts the next pooling strategy by calculating the mutual information coefficient of feature maps before and after pooling, thus solving the problem of feature map accuracy degradation. Secondly, a feature selection strategy DCPSO-FSS based on density clustering and particle swarm optimization is designed. This quickly locates density clustering parameters through particle swarm optimization and recognizes clustered features through feature sampling in the fully connected layer, thus solving the problem of feature redundancy. Finally, a load balancing strategy CCG-LBS based on cluster feature maps is designed. It dynamically calculates the computation cost of feature maps on each node of the distributed system and dynamically allocates data among groups based on this cost to achieve dynamic load balancing of data, thus solving the problem of low parallel recognition efficiency.

To verify the performance of the SMRFC-PDCNN algorithm, relevant experiments were designed and the SMRFC-PDCNN algorithm was compared with the PDCNNO algorithm, the RS-DCNN algorithm, and the BDCNN algorithm on four datasets, namely COCO, ImageNet, PASCALVOC2012, and Caltech256. The experimental results and analysis demonstrate that the SMRFC-PDCNN algorithm is suitable for the field of scene recognition and the model parallel recognition of deep convolutional neural networks compared with other algorithms.

In the future, we will work on solving the defects of SMRFC-PDCNN. For example, the scene recognition algorithm proposed in this article is based on training and optimization of discrete images, and it does not take into account relevant work in the time dimension. Therefore, future research can start from this aspect and build more accurate and meaningful scene recognition algorithm models.

Acknowledgement

This work was supported by the Special Project in Key Fields of Colleges and Universities in Guangdong Province (No. 2023ZDZX4069).

REFERENCES

[1] López-Cifuentes, A.—Escudero-Viñolo, M.—Bescós, J.—Miguel, J. C. S.: Attention-Based Knowledge Distillation in Scene Recognition: The Impact of a DCT-Driven Loss. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 33, 2023, No. 9, pp. 4769–4783, doi: 10.1109/TCSVT.2023.3250031.

[2] Rafique, A. A.—Gochoo, M.—Jalal, A.—Kim, K.: Maximum Entropy Scaled Super Pixels Segmentation for Multi-Object Detection and Scene Recognition via Deep Belief Network. Multimedia Tools and Applications, Vol. 82, 2023, No. 9, pp. 13401–13430, doi: 10.1007/s11042-022-13717-y.

[3] Kragel, J. E.—Voss, J. L.: Temporal Context Guides Visual Exploration During Scene Recognition. Journal of Experimental Psychology: General, Vol. 150, 2021, No. 5, pp. 873–889, doi: 10.1037/xge0000827.

[4] Masood, S.—Ahsan, U.—Munawwar, F.—Rizvi, D. R.—Ahmed, M.: Scene Recognition from Image Using Convolutional Neural Network. Procedia Computer Science, Vol. 167, 2020, pp. 1005–1012, doi: 10.1016/j.procs.2020.03.400.

[5] Handali, J. P.—Schneider, J.—Gau, M.—Holzwarth, V.—vom Brocke, J.: Visual Complexity and Scene Recognition: How Low Can You Go? 2021 IEEE Virtual Reality and 3D User Interfaces (VR), 2021, pp. 286–295, doi: 10.1109/VR50410.2021.00051.

[6] Wang, S.—Yao, S.—Niu, K.—Dong, C.—Qin, C.—Zhuang, H.: Intelligent Scene Recognition Based on Deep Learning. IEEE Access, Vol. 9, 2021, pp. 24984–24993, doi: 10.1109/ACCESS.2021.3057075.

[7] Kochakarn, P.—De Martini, D.—Omeiza, D.—Kunze, L.: Explainable Action Prediction Through Self-Supervision on Scene Graphs. 2023 IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 1479–1485, doi: 10.1109/ICRA48891.2023.10161132.

[8] Hu, Y.—Dong, B.—Huang, K.—Ding, L.—Wang, W.—Huang, X.—Wang, Q. F.: Scene Text Recognition via Dual-Path Network with Shape-Driven Attention Alignment. ACM Transactions on Multimedia Computing, Communications and Applications, Vol. 20, 2024, No. 4, Art. No. 107, doi: 10.1145/3633517.

[9] Ghosh, J.—Talukdar, A. K.—Sarma, K. K.: A Light-Weight Natural Scene Text Detection and Recognition System. Multimedia Tools and Applications, Vol. 83, 2024, No. 3, pp. 6651–6683, doi: 10.1007/s11042-023-15696-0.

[10] Middya, A. I.—Kumar, S.—Roy, S.: Activity Recognition Based on Smartphone Sensor Data Using Shallow and Deep Learning Techniques: A Comparative Study. Multimedia Tools and Applications, Vol. 83, 2024, No. 3, pp. 9033–9066, doi: 10.1007/s11042-023-15751-w.

[11] Heller, L. M.—Elizalde, B.—Raj, B.—Deshmukh, S.: Synergy Between Human and Machine Approaches to Sound/Scene Recognition and Processing: An Overview of ICASSP Special Session. CoRR, 2023, doi: 10.48550/arXiv.2302.09719.

[12] Song, X.—Liu, C.—Zeng, H.—Zhu, Y.—Chen, G.—Qin, X.—Jiang, S.: Composite Object Relation Modeling for Few-Shot Scene Recognition. IEEE Transactions on Image Processing, Vol. 32, 2023, pp. 5678–5691, doi: 10.1109/TIP.2023.3321475.

[13] Song, S.—Wan, J.—Yang, Z.—Tang, J.—Cheng, W.—Bai, X.—Yao, C.: Vision-Language Pre-Training for Boosting Scene Text Detectors. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 15660–15670, doi: 10.1109/CVPR52688.2022.01523.

[14] Zhou, B.—Lapedriza, A.—Khosla, A.—Oliva, A.—Torralba, A.: Places: A 10 Million Image Database for Scene Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 40, 2018, No. 6, pp. 1452–1464, doi: 10.1109/TPAMI.2017.2723009.

[15] Herranz, L.—Jiang, S.—Li, X.: Scene Recognition with CNNs: Objects, Scales and Dataset Bias. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 571–579, doi: 10.1109/CVPR.2016.68.

[16] Khan, S. H.—Hayat, M.—Bennamoun, M.—Togneri, R.—Sohel, F. A.: A Discriminative Representation of Convolutional Features for Indoor Scene Recognition. IEEE Transactions on Image Processing, Vol. 25, 2016, No. 7, pp. 3372–3383, doi: 10.1109/TIP.2016.2567076.

[17] Zhang, L.—Li, L.—Pan, X.—Cao, Z.—Chen, Q.—Yang, H.: Multi-Level Ensemble Network for Scene Recognition. Multimedia Tools and Applications, Vol. 78, 2019, No. 19, pp. 28209–28230, doi: 10.1007/s11042-019-07933-2.

[18] Li, E.—Xia, J.—Du, P.—Lin, C.—Samat, A.: Integrating Multilayer Features of Convolutional Neural Networks for Remote Sensing Scene Classification. IEEE Transactions on Geoscience and Remote Sensing, Vol. 55, 2017, No. 10, pp. 5653–5665, doi: 10.1109/TGRS.2017.2711275.

[19] Zhou, K.—Zhang, M.—Wang, H.—Tan, J.: Ship Detection in SAR Images Based on Multi-Scale Feature Extraction and Adaptive Feature Fusion. Remote Sensing, Vol. 14, 2022, No. 3, Art. No. 755, doi: 10.3390/rs14030755.

[20] Junaid, M.—Wagan, S. A.—Qureshi, N. M. F.—Nam, C. S.—Shin, D. R.: Big Data Predictive Analytics for Apache Spark Using Machine Learning. 2020 Global Conference on Wireless and Optical Technologies (GCWOT), 2020, pp. 1–7, doi: 10.1109/GCWOT49901.2020.9391620.

[21] Wang, Q.—Zhao, J.—Gong, D.—Shen, Y.—Li, M.—Lei, Y.: Parallelizing Convolutional Neural Networks for Action Event Recognition in Surveillance Videos. International Journal of Parallel Programming, Vol. 45, 2017, No. 4, pp. 734–759, doi: 10.1007/s10766-016-0451-4.

[22] Bello, M.—Nápoles, G.—Sánchez, R.—Bello, R.—Vanhoof, K.: Deep Neural Network to Extract High-Level Features and Labels in Multi-Label Classification Problems. Neurocomputing, Vol. 413, 2020, pp. 259–270, doi: 10.1016/j.neucom.2020.06.117.

[23] Boulila, W.—Sellami, M.—Driss, M.—Al-Sarem, M.—Safaei, M.—Ghaleb, F. A.: RS-DCNN: A Novel Distributed Convolutional-Neural-Networks Based-Approach for Big Remote-Sensing Image Classification. Computers and Electronics in Agriculture, Vol. 182, 2021, Art. No. 106014, doi: 10.1016/j.compag.2021.106014.

[24] Mao, Y.—Zhang, R.—Cao, W.: Parallel Deep Convolutional Neural Network Optimization Algorithm Based on Big Data. Computer Application Research, Vol. 38, 2021, No. 05, pp. 1416–1421, doi: 10.19734/j.issn.1001-3695.2020.04.0112.

[25] Kim, J. H.—Lee, D. S.—Lee, S. H.: Density Change Adaptive Congestive Scene Recognition Network. The International Journal of Advanced Smart Convergence, Vol. 12, 2023, No. 4, pp. 147–153.

[26] Qiao, J.—Wang, G.—Yang, Z.—Luo, X.—Chen, J.—Li, K.—Liu, P.: A Hybrid Particle Swarm Optimization Algorithm for Solving Engineering Problem. Scientific Reports, Vol. 14, 2024, No. 1, Art. No. 8357, doi: 10.1038/s41598-024-59034-2.

[27] Steck, H.—Ekanadham, C.—Kallus, N.: Is Cosine-Similarity of Embeddings Really About Similarity? Companion Proceedings of the ACM Web Conference 2024 (WWW '24), 2024, pp. 887–890, doi: 10.1145/3589335.3651526.

[28] Sun, X. H.—Lu, X.: The Memory-Bounded Speedup Model and Its Impacts in Computing. Journal of Computer Science and Technology, Vol. 38, 2023, No. 1, pp. 64–79, doi: 10.1007/s11390-022-2911-1.

[29] Sahu, A.—Das, P. K.—Meher, S.: High Accuracy Hybrid CNN Classifiers for Breast Cancer Detection Using Mammogram and Ultrasound Datasets. Biomedical Signal Processing and Control, Vol. 80, 2023, Art. No. 104292, doi: 10.1016/j.bspc.2022.104292.

Jingguo Dai is Professor at the Guangzhou Huashang College. His research interests include network and information security, graphics and image processing, and machine learning.

Yimin Mao is Professor at the Shaoguan University, Ph.D. supervisor. Her main research interests include data mining and artificial intelligence.