Incremental classifier learning based on PEDCC-loss and cosine distance

Qiuyu Zhu1 · Zikuang He1 · Xin Ye1

Received: 2 November 2020 / Revised: 11 January 2021 / Accepted: 14 June 2021 / Published online: 26 August 2021 © The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract
Traditionally, the performance of deep convolutional neural networks relies on a large number of labeled datasets in advance. However, in real-world applications, training data are not collected at once, so an algorithm which can deal with continuous incoming data is needed. This learning method is called incremental learning, whose main problem is the catastrophic forgetting. Neural network will perform badly on the old classes after training for the new classes. To solve this problem, this paper proposes an integrated incremental network approach based on the PEDCC-loss, and the cosine distance between the sample’s output feature and the PEDCC (predefined evenly distributed class centroids). Old and new knowledge learned by neural networks are stored separately. During the training of each network, PEDCC-Loss is used to constrain the cosine distance between the output feature and their corresponding pre-defined class center. Meanwhile, the old knowledge is retained by different learning rates across the network, and the retention mode of the old samples is discussed. In test phase, the final prediction is determined by the cosine distances between PEDCC and output features across all networks. Our experiments on EMNIST, CIFAR100 and TinyImageNet datasets show that our approach can learn classes incrementally without quickly failing of performance. Compared with some existing algorithms, such as Hou, iCaRL and finetune, our approach has better performance.

Keywords Incremental learning · PEDCC-loss · Convolutional neural network · Ensemble learning
1 Introduction

The human visual system is incremental in nature, new visual information is constantly merged under the premise of retaining the known information. For example, people learn a lot about the features of new cars in the car exhibition, but they never forget the features of their own cars. Most current pattern recognition systems can only set up a batch of data training, the information of the training data label is known, and all classes of data can be obtained in a random order during the training. But with the rapid development of computer vision and artificial intelligence, we need a more flexible strategy to deal with the large amount of information with dynamic properties in real life. For this purpose, an incremental learning approach is required, which allows continuous learning of new data without forgetting what was previously learned.

The significance of studying incremental learning is twofold: on the one hand, with the rapid development of Internet information technology, massive data are accumulated in all fields. At the same time, due to copyright and privacy issues, complete original model training data cannot be retained. Therefore, how to efficiently and continuously obtain effective information from these data is an important direction of research. Incremental learning can effectively solve the needs that cannot be met by traditional batch learning. On the other hand, by studying incremental learning models, we can have a deeper understanding of the structure of biological neural networks and the learning mode of human brain, which provides a solid theoretical and technical basis for the creation of new computing models and efficient learning algorithms.

When a convolutional neural network is trained on different steps of data in batches, the network will degrade its performance in the previous task. This phenomenon is called catastrophic forgetting [16]. Therefore, the biggest challenge of incremental learning at present is catastrophic forgetting, balancing the relationship between the new classes knowledge and the old classes knowledge, that is, how not to forget the old classes knowledge while learning the new classes.

Traditional incremental learning methods, such as ensemble learning [23], train multiple single learning models and then combine them to obtain a unified integrated learning model, so as to achieve more accurate, stable and strong results. Nowadays incremental learning more use convolutional neural networks, such as LwF (learning without forgetting) [12], which proposes an increment training method based on convolutional neural network, the convolution layer parameters sharing in each step, only the linear layer is different. When new classes arrive, the linear layer is extended by using the distillation loss function and fine-tuning methods to save the knowledge learned before. Therefore, Loss function is important to retain the previous class knowledge when it learns new classes. Recent class incremental learning methods such as iCarl [17], Rebalancing [9], all use a small part of the old samples when training for new classes. We propose a different retention strategy of the old samples and a different way to save the knowledge learned before.

Based on cosine distance and ensemble learning, this paper introduces a method to achieve class-incremental learning. The PEDCC (Predefined Evenly Distributed Class Centroids) algorithm is used to fix the weight of each network’s last linear layer to let the output features of different classes converge on their own pre-defined center. So, the new and old classes features are not interfered with each other. This step is to convert the traditional probabilistic prediction to the confidence level based on cosine distance, therefore the networks with different number of predicted classes can be compared with each other. When
training a network for new task, the network’s initial parameter is inherited from the previous network. By adding a part of old samples and adjusting the learning rate of different levels of the network, the knowledge learned from the old samples can be retained as much as possible. The final accuracy of incremental learning has been significantly improved. In the testing stage, which network should be chosen is determined by using the characteristic of the norm value and cosine confidence of network output feature. The system diagram is shown in Figs. 1 and 2, corresponding to the training phase and testing phase respectively.

For clarity, we summarize our research and contributions as follows:

– Firstly, based on the idea of ensemble learning and PEDCC-Loss, we propose a class incremental learning method via multiple network, which avoids the bias of a single network training towards new classes, and does not need to retrain the previous network each time when new classes data arrive.

– Secondly, we design a new scheme for the selection of old samples and the knowledge retention of previous networks, which can improve the accuracy of the final network selection.

– Finally, In the process of selecting the network during the prediction, we proposed a discriminant function based on cosine confidence and feature norm, which can be compared across networks.

Fig. 1 Overall architecture of incremental learning: We combine the new batch of data with the retained old samples as the training data set, for the training of the new network. The weight of the previous network is used as the initial value, and the method of controlling learning rate is adopted to retain the classification accuracy of the old classes as much as possible. In the figure, LR refers to the overall learning rate of the network in the training process.
**Testing phase.** The test image gets the output feature through each pre-trained network, the cosine distance is calculated between the output features of each network and the pre-defined center used in the training process of the network. The network with the maximum cosine distance multiplied by the corresponding feature’s norm value is the final chosen prediction network.

### 2 Related work

#### 2.1 Class-incremental learning

Xiao et al. [27] propose a kind of network that can grow in a hierarchical way, each node is composed of some clusters of similar classes. By the tree structure, when the model updates we only need to adjust some parts of the model, and the model adjustment scope can be strictly controlled. Incremental learning is realized through the growth of network, but it is facing the increasing difficulty of training large network and the difficulty of how to effectively increase network capacity. For the catastrophic forgetting problem in incremental learning of convolutional neural networks, Rusu et al. [18] present a progressive NN, which are used to solve the problem of networks adapting to new tasks. The idea is to keep all the networks of the previous class, create a new network for each new class, while retaining the low-level characteristics of the old networks. Venkatesan et al. [24] introduce phantom samples produced by GAN to retain the information of the original training samples. These Phantom samples are used to train the new deep network together with incremental samples, achieving a good result of incremental training of classes. However, this method takes a long time to train and is difficult to be applicable to the new incremental samples of old classes.

Li et al. [12] propose a method called LwF, using the distillation loss function, classification loss function and fine-tuning to retain the original model knowledge in training classes.
of new tasks, the role of the distillation loss function is to make the output of a new class approach to the output of original network trained by the previous classes, so as to keep the original network learned information. By setting the proportion of the classification loss to the distillation loss you can control whether the data is more biased towards the original or the new class. Rebuffi et al. [17] suggest that the convolutional neural network is used for feature learning and representation. The new classes samples and the previously stored old classes samples are added to the convolutional neural network for training together, so as to update the current model parameters and obtain new feature representation for all classes. In classification step, the NCM [14] is used to classify the feature vectors extracted from the sample set.

Wu et al. [26] redefined the loss function (cross entropy loss function + distillation loss function) on the basis of iCaRL, and added GANS [15] to generate a few samples of the old class, which improve the generalization ability. Cross-distilled loss function is proposed in [1]. Growing classification layer for each incremental step and the cross-distilled loss is the sum of calculations between each two pairs. Dhar et al. [4] propose a distillation loss for network attention to penalty the changes in classifiers’ attention maps to retain information. A distillation loss based on Grad-CAM [19] is used to alleviate catastrophic forgetting. Hou et al. use an improved distillation loss function and a less forget loss function to maximize the distance between old and new classes, to alleviate catastrophic forgetting when training new classes.

2.2 PEDCC-loss

PEDCC-Loss [29] (Predefined Evenly Distributed Class Centroids Loss) is a classification loss function based on PEDCC, and can make the features of different classes have the largest inter-class distance and the smallest intra-class distance, so as to achieve a good classification performance. Different from traditional loss function used in the process of training neural network, in which the center of each class’s feature is uncertain, PEDCC-Loss has pre-defined centers for the features of each class, and the pre-defined centers are even distributed on feature hypersphere. The distribution diagram of pre-defined centers in 3D space is shown in Fig. 3.

The PEDCC’s goal is to generate n points that evenly distributed on the n-dimensional hypersphere surface as clustering centers, where n is the number of classes. It is generated based on the physical model with the lowest like charge energy on the hyperspherical surface. The n charge points on the hyperspherical surface have the repulsive force between each other, and the repulsive force push the charges to move. When the motion is finally balanced and the points on the hypersphere surface stop moving, n points will eventually be evenly distributed on the hyperspherical surface.

![Fig. 3](image_url) The diagram of PEDCC pre-defined centers. The figure shows 2, 4, 10, 20 number of pre-defined centers generate from PEDCC.
Through the fixed-weight of last linear layer and improved Softmax loss function, we can make the features of each classes distribute evenly on a hypersphere. The loss function is as follows:

\[ L_{PBDCC-AM} = -\frac{1}{N} \sum_{i} \log \frac{e^{s(\cos \theta_{yi} - m)}}{e^{s(\cos \theta_{yi} - m)} + \sum_{j=1, j\neq yi}^{c} e^{s \cos \theta_{j}}} \]  

(1)

\[ L_{PBDCC-MSE} = \frac{1}{N} \sum_{i=1}^{m} \| x_i - \text{pedcc}_{yi} \|^2 \]  

(2)

\[ L = L_{PEDCC-AM} + n \sqrt{L_{PEDCC-MSE}} \]  

(3)

where (1) is the AM-Softmax loss [25], \( m \) is the cosine margin between each class, and \( s \) is used to improve the rate of convergence. Equation (2) is the MSE Loss function between output feature and pre-defined center, \( x_i \) is the output feature and \( y_i \) is the corresponding label, \( \text{pedcc}_{yi} \) means the pre-defined class center of \( y_i \). PEDCC-Loss is obtained by combining them two, where \( n \geq 1 \) is a superparameter that can be adjusted. In this paper, \( n \) takes 2.

By using this loss function, we can artificially control the output feature distribution of each class. The traditional Softmax loss function only adjusts the classification layer parameters according to the current training data, does not normalize the weight and input features of the classification layer. Obviously, it is inevitable to be affected by the weight and feature modulus when calculating the cosine distance between features and weights, thus reducing the predicting accuracy. By using PEDCC-Loss, we can make the features in the feature space have the largest distance between the classes, and small inner-classes distance. More importantly, through the fixed classification layer, the network’s final prediction only depends on the cosine distances between output feature and the pre-defined centers.

The advantages of using this loss function are also obvious. If traditional cross entropy loss function is used, the network’s output is probabilistic representation. Then after the new stage network is trained, its output prediction number is the old class number and the new class number. In this case, the number of final predictions between networks is not equal, so its probability value cannot be directly compared. After PEDCC-Loss is applied, we can convert the probability representation of network prediction into the confidence representation based on cosine distance, so as to compare the network outputs with different predicted numbers. It is the main idea of this paper.

3 Class-incremental learning based on ensemble way and PEDCC

The main purpose of existing incremental learning methods like LwF and iCaRL is to preserve the knowledge of old task as much as possible, and at the same time to learn new knowledge of new task. Different from these methods, when some new classes’ data arrive, they are used to train a new network with PEDCC-Loss. In this way, different training data’s knowledge is saved in different networks. Finally, the combination of classifier is realized based on cosine distance.

3.1 Training phase

Firstly, the number \( N \) of incremental class upper limit is set according to the specific scenario. Then the pre-defined centers for each class is produced by PEDCC algorithm and set
the pre-defined centers’ dimension M at the same time. Finally, a PEDCC matrix of N rows M columns is produced. The last classification layer’s weight of neural network is initialized by PEDCC centers and will not change during training. For instance, when N₁=10, M=512, the number of initial data classes is 10, which means the training data of the first network is those data of 10 classes. The first neural network uses the first 10 pre-defined center to be the feature center, and retain the network model after training. The whole training step is shown in Fig. 1.

When a batch of new data appears, the corresponding pre-defined weights are taken out from the generated PEDCC class center matrix according to the number of classes of the new data for network training. The new network is trained by the cosine distance between output features and the predefined centers, and be saved after the training, and so on. In the process of training new network, several optimization methods are proposed for the retention strategy of the old sample and the transfer of the previous network knowledge.

First optimization method is the old sample retention strategy. Current retention strategy of incremental learning methods is similar to the method in iCaRL. Based on NCM [13], train dataset is screened by the Euclidean distance between each sample’s output feature and the corresponding class’s average feature. The nearest first n samples are selected as the remaining part of the old samples, and be trained together with the data set of new classes. Distillation loss function is also added to retain the knowledge learned from the old classes in previous network. Different from this screen method, old samples are selected randomly in this paper. The experimental result shows that, compared with the method of selecting the first n samples with the best performance in the task before, the randomly selection method of old samples in the incremental learning of multiple classification network can improve the accuracy of the final classification performance to a certain extent. We think this is due to the different distribution of samples’ latent feature learned from the network.

It can be seen from the Fig. 4 that the distribution of old samples selected according to the NCM method is very concentrated, while the distribution of the old samples feature retained by random method is similar to the overall distribution of the latent feature of the original dataset. Therefore, it is speculated that the learning of the first distribution in the training process of the network will inevitably reduce the generalization ability compared with the feature distribution of randomly selected samples. This is also demonstrated in our experiments. In this paper, our retention strategy for old samples is to reserve 8 percent samples for each category in CIFAR100, TinyImageNet, and 4 percent samples in EMNIST.

![Fig. 4](image.png)  
*(a) (b)*  
**Fig. 4** Schematic diagram of retained old sample selection method. The left picture shows the selection of the old samples which have best performance (NCM method), (b) random selection. For simplicity, we set the output feature to 2 dimensions.
The second optimization method is the retention strategy of the previous network knowledge. The convolution kernel at each layer of a convolutional neural network can be considered as a feature template. The input image is matched with the template at each layer and finally mapped to the hidden feature space, and then the image is judged by this output feature [11]. The training process can be considered as that the parameters of the template function are constantly adjusted through label information to make the final output of the network fit to the given label. It can be expressed as:

\[ p = f_1 ( f_{\ldots} ( f_n (x)) \) (4) \]

where \( p \) is the final prediction vector, \( x \) is the input image, and \( f_n \) is the template function of the \( N_{th} \) hidden layer. Since the convolution kernel at different levels of a convolutional neural network learns different information, the higher of the level, the stronger the semantic information will be [6, 28]. According to [28] the feature output by the bottom layers of convolutional neural network are some highly reusable information of lines and colors, while the feature output by the next several layers are the contour, shape and other information combined by these bottom features. Due to the high reusability of the bottom layers, we believe that the retention of more bottom information is helpful to incremental learning to a certain extent compared with high-level semantic information.

Fine-tuning, as one of the common network initialization methods in deep learning, plays an indispensable role in incremental learning [9, 17], in ordinary computer vision tasks, fine-tuning can make the network converge more quickly during training [7]. In incremental learning tasks, how to use the previously pre-trained network as the carrier of learned knowledge is a very important issue. In this paper, fine-tuning and the learning rates adjustment of different layers of the network were employed to train a new network, so that the network prediction accuracy could be further improved through the retention of bottom information on the basis of fine-tuning.

### 3.2 Prediction phase

Because we retain multiple networks in this paper, the outputs of the same test sample over different networks need to be compared. If we use the traditional cross entropy loss function, the output of each network will be a probabilistic representation. Because each network prediction class is different, the output probabilities cannot be compared directly. Our whole prediction step is shown in Fig. 2.

Due to the usage of PEDCC-Loss, the linear classification layer of convolutional neural network we have trained is fixed. Moreover, the PEDCC pre-defined center is also the prediction prototype for the final network determination. Therefore, for classification, correlation is calculated between the corresponding PEDCC center and the network latent feature, namely the classification result depends on the correlation (that is cosine distance) between the latent feature and the pre-defined centers. In the linear classification layer, \( w_i \) is the weight of a specific class \( i \) defined by PEDCC’s pre-defined center, and the output feature \( x \) and \( w_i \) are normalized. According to the formula of the fully connect layer, we have:

\[ g_i(x) = w_i \cdot x = \|w_i\| \|x\| \cos \theta = \cos \theta \] (5)

where \( g_i(x) \) is the correlation of \( x \) and \( w_i \), and is equivalent to the posterior probability that the latent feature \( x \) belongs to class \( i \), that is, it is a discriminant function. \( g_i(x) \) represents the confidence that feature \( x \) belongs to class \( i \). Here the \( w_i \) in last linear layer of each class is the corresponding pre-defined center calculated by PEDCC generation algorithm. The prediction results will depend entirely on the cosine distance between the output
latent feature and the predefined center. Then, the classification results can be obtained by maximizing distance:

$$j = \arg\max_i g_i(x)$$

(6)

This classification result is consistent with the direct network classification output, but, because we turn the classification prediction into the confidence based on cosine distance, and each network has the same center for same class, it makes it easy for us to compare the output from multiple networks.

In addition to turn the probability to cosine confidence method to select the prediction network, in this paper, norm value of output feature before normalization is proposed as the final confidence judgment together with cosine distance. If the class of the test sample is in the training batch of the current network, then the value of the norm value of its output feature will be larger than that of other networks. In [22], the sample feature norm value is first used as the prediction score for face recognition. The smaller the feature norm value is, the lower the prediction confidence will be. In addition, it has also been successfully applied in out-of-distribution detection problem [20], and it is concluded that the feature norm value of in-distribution samples output by neural network is higher than that of out-of-distribution samples. Inspired by this, we believe that using the norm value of output features of samples in different networks as one of the discriminant features can improve the accuracy of final judgment. In this paper, the specific method is to take the product of the norm value of the sample feature before normalization and the cosine distance calculated by the feature as the final prediction confidence. The experimental results prove that the network can be further selected accurately, so as to improve the accuracy of the increment.

The confidence of each network is calculated through the test sample’s output feature, $z$ is the latent feature before normalization:

$$C_n = \max_i g_{ni}(x) \times ||z||$$

(7)

where $g_{ni}$ is the $i$-th cosine distance in network $n$, $C_n$ is confidence score when sample $x$ is recognized by network $n$. The final selected network $J$ is the network with the highest confidence score:

$$J = \arg\max_n (C_n)$$

(8)

The recognized label with the maximum cosine distance between the latent feature and the predefined centers in network $J$ is the final prediction result.

![Training samples](image_url)

Fig. 5 Some training samples of (a) EMNIST, (b) CIFAR100 and (c) TinyImageNet. The sample of each row belongs to the same class.
Fig. 6 Experimental results of class-incremental training on EMNIST. The left picture is the 20+10+10 incremental result, and the right is the 10+10+10+10 incremental result.

4 Experimental results

The experimental data set used in this paper are EMNIST [2] with 40 classes, CIFAR100 [10] with 100 classes and TinyImageNet [3] with 200 classes. EMNIST contains 47 digits and letters, image resolution is $28 \times 28$, and there are 2400 training images and 400 test images for each class. CIFAR100 has 100 classes, image resolution is $32 \times 32$, and the number of training and test images for each class is 500 and 100. TinyImageNet has 200 classes, image resolution is $64 \times 64$, 500 training samples and 50 test samples for each class. For EMNIST, we choose the modified VGGNet [21] as our classification network. We eliminate some confusing classes such as ‘6’ and ‘b’, and only 6 is retained. For CIFAR and TinyImageNet, we choose the ResNet-18 [8] as our network structure. Few samples of these dataset are shown in Fig. 5. The network structure is shown in Tables 1 and 2.

The size of the convolution kernel is uniform in convolution layers, which is $3 \times 3$, and the stride and padding is 1. We use the pytorch1.0 framework [5] to train our neural networks for 100 epochs. The learning rate starts at 0.1 and is divided by 10 after 30, 60, 90 epochs. We train the networks using SGD with batch size 128, a weight decay parameter of 0.0005 and the momentum is 0.9. Following the original paper, the hyperparameters m and s in PEDCC-Loss are set to 10 and 0.3 respectively. The accuracy of each training step refers to the overall prediction accuracy of all classes that have been encountered so far.

The specific way to adjust the learning rates of different layers of the network is to set the learning rates of Conv0/1, Conv2, Conv3, Conv4 and the full connection layer at 0.2, 0.2, 0.5, 0.7 and 1 respectively. Conventional fine-tuning is uniform training for all levels after the weight is inherited, and the learning rates of different levels are not regulated. The learning rate setting in experiment on CIFAR100 50+50 scenario is shown in Table 3.

| Group name | Block type |
|------------|------------|
| Conv0.x    | $[3 \times 3, 64] \times 1$ |
| Conv1.x    | $[3 \times 3, 64] \times 3$ |
| Pool1      | $2 \times 2$ Max, Stride 2 |
| Conv2.x    | $[3 \times 3, 128] \times 3$ |
| Pool2      | $2 \times 2$ Max, Stride 2 |
| Conv3.x    | $[3 \times 3, 256] \times 3$ |
| Pool3      | $2 \times 2$ Max, Stride 2 |
| Fully Connected | 512 |
| Fixed Fully Connected | $N_i$ |
As can be seen from the table, if the learning rate of different layers are not constrained, the performance of incremental classification accuracy decreased obviously. Single Net Acc refers to using the network obtained by second training stage alone to predict the whole 100 classes, and Total Acc means to use two networks obtained from each stage training together to predict 100 class.

For result comparison, we use the methods of Hou [9], iCarl [17] and Finetune. Hou use less forget constraint and inter class separation to maintain the old knowledge, while iCarl use distillation loss to achieve this goal. In our experiment, the network’s structure is the same.

### 4.1 EMNIST experimental result

This section shows the performance of our method in EMNIST. We use three or four networks to do the incremental learning, the capacity of each network varies from 10 to 20
The result of class-incremental experiment on EMNIST

| Test classes | One batch accuracy | Two batches accuracy | Three batches accuracy | Four batches accuracy |
|--------------|--------------------|----------------------|-----------------------|----------------------|
| 20+20        | 98.82%             | 95.56%               | \                     | \                    |
| 20+10+10     | 98.82%             | 97.54%               | 95.43%                | \                    |
| 10+10+10+10  | 99.63%             | 96.93%               | 95.63%                | 94.00%               |
| CNN+PEDCC-Loss | 96.93%          |                      |                       |                      |

The accuracy of 40 classes classification directly trained by the network combined with PEDCC-Loss is 96.93%. It can be seen from the table that the classification accuracy of 20+20 of the two networks can reach 95.56%, which is relatively close to the convolutional neural network that is trained directly.

For 10+10+10+10 experiment in the table, the number of networks is increased to 4, and the total recognition rate finally drops to 94.00%. This is because the increase in the number of networks will inevitably bring more interference factors in the feature selection stage. However, it can be seen that the recognition rate of the overall 40 classes is maintained at a high level under different combinations.

4.2 TinyImageNet experiment result

The strategy in this section is the same as 5.1, TinyImageNet is divided into different combinations, and the number of networks increases to 5 (Table 5).

The accuracy of 200 classification directly trained by the network combined with PEDCC-Loss is 59.04%. It can be seen from the table that as the number of networks increased, the accuracy declined, as shown in the previous section.

The experimental results are compared with the results of Hou, iCarl and Finetune on TinyImageNet, which is shown in the Fig. 7.

4.3 CIFAR100 experiment result

This section uses the same strategy as is mentioned before, we divide the CIFAR100 into different combinations, and the number of networks increases to five, the result is shown in Table 6.

The accuracy of 100 classifications based on PEDCC-Loss is 74.86%. From the table, we can see that the 100 classifications accuracy of 50+50 can reach 70.74%, the incremental classification of 100 is also close to the traditional classification result. It shows that the
Fig. 7  Experimental results of class-incremental training on TinyImageNet. The left picture is the 50+50+50+50 incremental result, and the right is the 40+40+40+40+40 incremental result.

Table 6  The result of class-incremental experiment on CIFAR100

| Test classes   | One batches accuracy | Two batches accuracy | Three batches accuracy | Four batches accuracy | Five batches accuracy |
|----------------|----------------------|----------------------|------------------------|-----------------------|-----------------------|
| 50+50          | 79.02%               | 70.74%               | \                      | \                     | \                     |
| 25+25+25+25    | 82.82%               | 74.44%               | 68.38%                 | 66.05%                | \                     |
| 20+20+20+20+20 | 82.00%               | 74.87%               | 70.53%                 | 66.72%                | 64.88%                |
| CNN+PEDCC-Loss | 74.86%               | \                    | \                      | \                     | \                     |

Fig. 8  Experimental results of class-incremental training on CIFAR100. The left picture is the 25+25+25+25 incremental result, and the right is the 20+20+20+20+20 incremental result.

Fig. 9  CIFAR100 Experimental results of two old sample retention methods, the left is the four-stage incremental result, the right is the five-stage incremental result.
integrated incremental network can achieve real incremental learning classification with a fair good result.

The result with comparison is shown below:

It can be seen from the Fig. 8 that the method in this paper improves significantly compared with the other three methods in different incremental stages. Meanwhile, we also compare the results of different old sample retention methods, as shown in the Fig. 9.

It is shown that the old sample selection strategy proposed in this paper can indeed improve the accuracy of different stages of incremental learning to some extent. As for the adoption of latent feature norm value in the prediction stage, we also conducted a comparative test, and the results are shown in the Fig. 10.

Although the algorithm has high recognition accuracy, it has high computing and network complexity. With the arrival of each batch of training data, a new network is used for training, so the computing complexity of networks is going to be n times higher than individual network.

5 Conclusion and discussion

In this paper, we propose a class incremental learning method based on integration of multiple independent networks. In each step training, a randomly selected method is used to select a small portion of the previous classes samples. Different learning rate for different layers are selected to preserve the reusable knowledge, and confidence score based on cosine distance replaces the probability representation for the final prediction. The experiments on EMNIST, CIFAR100 and TinyImageNet datasets show good results.

Despite the promising results, class incremental learning is far from solved. The final performance is still lower than joint training, i.e. with all training samples of all classes available at the same time. In future work we plan to reduce the interaction between different network output features to improve the class incremental performance, and to study an incremental learning method of single network based on PEDCC.

References

1. Castro FM, Marín-Jiménez MJ, Guíl N, Schmid C, Alahari K (2018) End-to-end incremental learning. In: Proceedings of the european conference on computer vision (ECCV), pp 233–248
2. Cohen G, Afshar S, Tapson J, Van Schaik A (2017) EMNIST: Extending MNIST to handwritten letters. In: 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, pp 2921–2926
3. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. IEEE, pp 248–255
4. Dhar P, Singh RV, Peng KC, Wu Z, Chellappa R (2019) Learning without memorizing. In: Proceedings of the IEEE conference on computer vision pattern recognition, pp 5138–5146
5. Facebook (2021) Pytorch. Available online: https://pytorch.org/. Accessed 3–12 Jan 2020
6. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587
7. He K, Girshick R, Dollár P (2019) Rethinking imagenet pre-training. In: Proceedings of the IEEE international conference on computer vision, pp 4918–4927
8. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
9. Hou S, Pan X, Loy CC (2019) Learning a unified classifier incrementally via rebalancing. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol 2019, pp 831–839
10. Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images
11. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
12. Li Z, Hoiem D (2017) Learning without forgetting. IEEE Trans Pattern Anal Mach Intell 40(12):2935–2947
13. Mensink T, Verbeek J, Perronnin F, Csurka G (2012) Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In: European conference on computer vision. Springer, Berlin, Heidelberg, pp 488-501
14. Mensink T, Verbeek J, Perronnin F, Csurka G (2012) Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In: European conference on computer vision, vol 488-501. Springer, Berlin, Heidelberg
15. Mirza M, Osindero S (2014) Conditional generative adversarial nets. arXiv: 1411.1784
16. Polikar R et al (2001) Learn++: an incremental learning algorithm for supervised neural networks. IEEE Trans Syst Man Cybern C Appl Rev 31(4):497–508
17. Rebuffi SA, Kolesnikov A, Sperl G, Lampert CH (2017) icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2001–2010
18. Rusu AA, Rabinowitz NC, Desjardins G, Soyer H, Kirkpatrick J, Kavukcuoglu K, Hadsell R et al (2016) Progressive neural networks. arXiv:1606.04671
19. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D (2017) Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision, pp 618–626
20. Shalev G, Adi Y, Keshet J (2018) Out-of-distribution detection using multiple semantic label representations. Adv Neural Inf Process Syst 31:7375–7385
21. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
22. Taigman Y, Yang M, Ranzato MA, Wolf L (2015) Web-scale training for face identification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2746–2754
23. Valentini G, Masulli F (2002) Ensembles of learning machines. In: Italian workshop on neural nets. Springer, Berlin, Heidelberg, pp 3-20
24. Venkatesan R, Venkateswara H, Panchanathan S, Li B (2017) A strategy for an uncompromising incremental learner. arXiv:1705.00744
25. Wang F, Cheng J, Liu W, Liu H (2018) Additive margin softmax for face verification. IEEE Sig Process Lett 25(7):926–930
26. Wu Y, Chen Y, Wang L, Ye Y, Liu Z, Guo Y, Fu Y et al (2018) Incremental classifier learning with generative adversarial networks. arXiv:1802.00853
27. Xiao T, Zhang J, Yang K, Peng Y, Zhang Z (2014) Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In: Proceedings of the 22nd ACM international conference on multimedia. ACM, pp 177-186
28. Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: European conference on computer vision. Springer, Cham, pp 818–833
29. Zhu Q, Zhang P, Wang Z, Ye X (2019) A new loss function for CNN classifier based on predefined evenly-distributed class centroids. IEEE Access 8:10888–10895

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.