Class-Incremental Learning of Convolutional Neural Networks Based on Double Consolidation Mechanism

LEILEI JIN, HONG LIANG, AND CHANGSHENG YANG, (Member, IEEE)
School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China

Corresponding author: Hong Liang (lianghong@nwpu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61971354.

ABSTRACT Class-incremental learning is a model learning technique that can help classification models incrementally learn about new target classes and realize knowledge accumulation. It has become one of the major concerns of the machine learning and classification community. To overcome the catastrophic forgetting that occurs when the network is trained sequentially on a multi-class data stream, a double consolidation class-incremental learning (DCCIL) method is proposed. In the incremental learning process, the network parameters are adjusted by combining knowledge distillation and elastic weight consolidation, so that the network can better maintain the recognition ability of the old classes while learning the new ones. The incremental learning experiment is designed, and the proposed method is compared with the popular incremental learning methods such as EWC, LwF, and iCaRL. Experimental results show that the proposed DCCIL method can achieve better incremental accuracy than that of the current popular incremental learning algorithms, which can effectively improve the expansibility and intelligence of the classification model.

INDEX TERMS Class-incremental learning, convolutional neural networks, knowledge distillation, weight consolidation, visual recognition.

I. INTRODUCTION

Incremental learning [1], also known as lifelong learning [2] or continuous learning, aims to enable learning models to continue learning just like human beings. The way the human brain learns is incremental, e.g., if a child gets to know the tiger and lion in the zoo, he can retain this knowledge and learn more new species such as the dolphin and seal in the future. However, most machine learning systems can only learn once with the available training data, and the models obtained are often task-oriented and isolated [3]. In contrast, incremental learning represents a dynamic model learning technique that can help models continuously learn new knowledge and realize knowledge accumulation. To achieve this, incremental learning has to overcome catastrophic forgetting, namely, to learn new knowledge without forgetting the old knowledge.

According to the degree of difficulty, incremental learning can be divided into three categories [4]: task-incremental learning (Task-IL), domain-incremental learning (Domain-IL), and class-incremental learning (Class-IL). Among them, Class-IL is the closest one to human learning behavior and is the most challenging one. Due to unknowing the number of output classes, researchers have to develop different methods to model new classes while preserving pre-trained performance [5]. Rebuffi et al. [6] made a clear statement of class-incremental learning. Formally, it requires an algorithm to have the following three attributes to be qualified as class-incremental: (1) it can learn from a data stream, in which the sample data of different classes occur at different times; (2) it can provide competitive multi-class classification accuracy for the classes observed so far; (3) the computational requirements and memory footprint of the algorithm should be kept limited, or grow very slowly relative to the number of classes observed so far.

The research on incremental learning is in the ascendant. In 2000, Christophe et al. [7] proposed the notion of incremental learning, and in 2001, Polikar et al. [8] standardized the definition of incremental learning and was widely accepted. In 2007, Shen and Osamu [9] proposed a
self-organizing incremental neural network (SOINN), which is based on competitive learning and has been applied to unsupervised learning tasks. In 2009 and 2011, two incremental learning algorithms Learn++NSE [10] and Learn++NC [11] were proposed, further expanding the application scope of incremental learning in non-stationary environments. For supervised learning, the broad learning proposed by Chen and Liu [12] can update network with only a small amount of calculation, however, this algorithm is only suitable for shallow neural networks. In 2017, the DeepMind team proposed the EWC [13] method and realized the continuous learning of multiple tasks for deep neural network models. By modifying the learning rule, the model can learn new tasks without forgetting the information on old tasks. In the same year, Li and Hoiem [14] proposed another approach, namely learning without forgetting (LwF), which could be seen as a combination of Distillation Networks [15] and fine-tuning [16]. Although the EWC and LwF methods are good at task incremental learning, they are not effective for the Class-II scenario. The iCaRL method proposed by Rebuffi et al. [6] is the improvement of the LwF method, which is superior to most of the current incremental learning methods. But because historical data is needed to train the new network, the iCaRL approach is not consistent with the way humans learn.

In this paper, we propose a novel Class-IL method, DCCIL (double consolidation class-incremental learning), that allows convolutional neural networks to learn in such a way: only new class data are used to train the network while preserving the original capabilities. The DCCIL method draws on the idea of knowledge distillation in LwF and the idea of weight consolidation in EWC, and combines with the nearest-mean-of-exemplars classifier used in iCaRL to achieve better performance in Class-II scenario. Under the hood, DCCIL makes use of a convolutional neural network (CNN). In principle, the DCCIL strategy is largely architecture agnostic and could be used on top of other features or metric learning strategies. With the proposed algorithm, the training is easy and fast. The result is also very promising.

The rest of this paper is organized as follows: Section II introduces the related works of incremental learning; in Section III, we explain the details of DCCIL method; in Section IV, experimental results on sonar image dataset are presented and show that DCCIL can class-incrementally learn over a series of target recognition tasks; finally, concluding remarks are provided in Section V.

II. RELATED WORK

In the domain of incremental learning, a key issue is to avoid catastrophic forgetting [17]. For example, the AlphaGo [18] trained on the Go task can only play Go and knows nothing about chess. If the AlphaGo is trained to learn chess, it would likely to forget the knowledge of Go, which we do not want to happen. Specific to the neural network, as shown in Fig. 1, the classification model continuously learns two different tasks: Dataset1 is the old task data and Dataset2 is the new task data. The learning process of a neural network is essentially the optimization process of parameters (weights and biases), and tasks are solved by constantly adjusting the network parameters. When the neural network is trained with the Dataset2, the weights learned from the old task are usually overwritten to adapt to the new task, resulting in forgetting the information related to the old task. Therefore, after learning new tasks, neural networks that even with the most advanced model structures are likely to degrade or fail completely on the old tasks.

To overcome catastrophic forgetting, researchers have come up with several solutions, the most famous of which include EWC and replay methods using knowledge distillation. This section will give a detailed introduction to EWC and knowledge distillation, and based on this, propose our DCCIL method.

A. ELASTIC WEIGHT CONSOLIDATION

In 2017, Kirkpatrick et al. proposed the elastic weight consolidation (EWC) algorithm. EWC method can help model gradually learn new tasks without forgetting the old ones, and the working principle is explained in detail below.

Inspired by the synaptic consolidation mechanism of the human brain, they found a way to tackle catastrophic forgetting. An artificial neural network is made up of connections in a way that is somewhat similar to the connections between neurons in the brain. After the network learns a task, we can calculate the importance each connection weight is to the task. When the network learns a new task, the connection weights can be protected differently depending on how important the connection is to the old task. This allows the network to learn new tasks without incurring significant computing costs while retaining what it has learned from previous tasks. We can think of the connections between nodes as connected by springs. The stiffness of the spring is proportional to the importance of the connection; a higher stiffness of the spring makes the connection weight harder to change. For this reason, the algorithm is called “elastic weight consolidation.”

How to judge the importance of parameters and how to constrain parameters is the key to implement the algorithm. In EWC, the Fisher Information Matrix (FIM) [19] is used to determine the importance of parameters for previous tasks. After the network completes the training of a task, the Fisher information value of the network parameters can
be calculated. The parameters with low Fisher values are considered redundant and can be modified to store new task information. The calculation formula of FIM is

\[ F(\theta) = E[s(\theta)s(\theta)^T] = E[\nabla_\theta \log p(x; \theta)(\nabla_\theta \log p(x; \theta))^T] \]  

(1)

where \( s(\theta) \) is a scoring function and is defined as the gradient of a log-likelihood function, used to measure the influence of a parameter on a probability distribution. We do not know the exact form of the likelihood function \( p(x; \theta) \), but we can use the empirical distribution to estimate the FIM. Given a data set \{\( x^{(1)}, \ldots, x^{(N)} \)\}, the FIM can be approximated as

\[ F(\theta) = \frac{1}{N} \sum_{i=1}^N \nabla_\theta \log p(x^{(i)}; \theta)(\nabla_\theta \log p(x^{(i)}; \theta))^T \]  

(2)

The diagonal values of FIM can reflect the reliability of the corresponding parameters, and the larger value means the more reliable the estimation is and the more information it carries about the data distribution.

Again, taking neural network continuously learns two tasks as an example, and assume that the data sets of task \( T_A \) and task \( T_B \) are \( D_A \) and \( D_B \) respectively. We can use FIM to measure the information about \( D_A \) that carried by one parameter in the network:

\[ F^A(\theta) = \frac{1}{N} \sum_{(x,y)\in D_A} \nabla_\theta \log p(y|x; \theta)(\nabla_\theta \log p(y|x; \theta))^T \]  

(3)

To constrain the network parameters, the loss function can be set as the following when the network is trained on task \( T_B \):

\[ L(\theta) = L_B(\theta) + \sum_{i=1}^Q \frac{\lambda}{2} F^A_i \cdot (\theta_i - \theta^*_A)^2 \]  

(4)

where \( L_B(\theta) \) is the cross-entropy loss function of the task \( T_B \), \( F^A_i \) is the \( i \)th diagonal element of FIM, \( \theta^*_A \) is the parameters learned on the task \( T_A \), \( \lambda \) is a hyper-parameter to balance the importance of the two tasks, and \( Q \) is the total number of the network parameters.

B. KNOWLEDGE DISTILLATION

Knowledge distillation is not originally proposed for incremental learning, but it has been widely used in this domain. Knowledge distillation is a simple and effective method for model compression and model training, which was proposed by Hinton et al. [15] in 2015. In the context of deep learning, to achieve better prediction, over-parameterized deep neural networks are often constructed. Such kind of network has very strong learning ability but requires a large amount of memory and computing resources, which is very unfavorable for deployment. However, it is empirically difficult to train a small model from scratch to achieve the equivalent performance of a large model. Fortunately, by using the knowledge distillation method, knowledge can be migrated from a large network or network cluster to a smaller network for more efficient deployment.

The small model need to be trained is called the student network, and the large model that has been trained is called the teacher network. Our goal is to make the output of the student network sufficiently close to that of the teacher network, but there is a problem with doing this directly: the Softmax output of the teacher network is similar to a one-hot vector, with one value large and the others small. In this case, the output of the teacher network provides very limited supervision information. Compared with the “hard” output which is almost one-hot, we hope the output can be more “soft,” that is, the probability distribution can be more moderate. Knowledge distillation presents an effective method. Let’s consider a generalized Softmax function:

\[ q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \]  

(5)

where \( q_i \) is the output value of the model’s \( i \)th output node before Softmax function performed, and \( T \) is the temperature parameter. It is easy to prove that \( q_i \) converges to a one-hot vector as temperature \( T \) approaches 0, and becomes softer as \( T \) gets higher.

The knowledge distillation method assists the training of the student network by introducing the soft-target related to the teacher network as a part of the total loss, to realize knowledge transfer. The whole process of knowledge distillation is shown in Fig. 2.

![FIGURE 2. Schematic diagram of knowledge distillation: (a) the training stage of the student network; (b) the test stage of the student network.](image-url)

In Fig. 2, the predicted output of the left teacher network is transformed by generalized Softmax to obtain the soft
probability distribution. The value is between 0 and 1, and the
value distribution is relatively moderate. The hard-target
is the true label of the input sample, which can be repre-
sented by a one-hot vector. The total loss is calculated as the
weighted average of the cross-entropy loss corresponding to
the soft-target and the hard-target. The larger the weighted
coefficient of the soft-target, the more the training process of
the student network depends on the teacher network. When
training the student network, use high T to make Softmax
distribution soft enough; after the training, set the temperature
T = 1 for inferring.

III. DOUBLE CONSOLIDATION CLASS-INCREMENTAL
LEARNING
For the memory function of human brain, neuroscientists
have thus far given two explanations: systems consolidation
[20] and synaptic consolidation [21]. Systems consol-
didation refers to the process of imprinting memory from
a fast-learning part of our brain into a slow-learning part,
through conscious or unconscious recall, which may occur
while dreaming. For the second mechanism, synaptic con-
solidation, it means that if some synaptic connections in
the cerebral cortex are important for previous learning tasks, the
brain will reduce the plasticity of these synapses and keep them stable, to achieve continuous learning.

In our view, the process of knowledge distillation is a
transfer of knowledge from one network structure to another and
is similar to the brain’s systems consolidation mechanism, in which knowledge acquired in the fast-learning part
is imprinted to the slow-learning part. And the idea of the EWC
method, as described in subsection II.A, is analogous to the
brain’s synaptic consolidation mechanism.

The process of memorizing knowledge in the human
brain is obviously much more complicated. It is possible
to impair memory capacity if the brain relying solely on
systems consolidation or synaptic consolidation. With this in
mind, we propose an incremental learning method combining
systems consolidation mechanism and synaptic consolidation
mechanism, that is, in the network learning process,
knowledge distillation and weight consolidation are used
at the same time, so that the model can better retain the
knowledge learned from the old target classes while learning the new classes. For this reason, we refer to the pro-
posed approach as “double consolidation class-incremental
learning” (DCCIL).

A. OVERVIEW OF THE DCCIL METHOD
In this subsection, we will elaborate on the implementation
process of the DCCIL method as a whole. As shown in Fig. 3,
DCCIL has two key points: during the training stage, the net-
work updates parameters through the double consolidation
mechanism, and learns the features of continuous multi-class
targets in the data stream; during the test stage, the DCCIL
method uses the trained convolutional neural network mod-
ule as a feature extractor: \( \varphi: \mathcal{X} \rightarrow \mathbb{R}^d \), followed by a
nearest-mean-of-exemplars (NME) classifier [6], [22] for
target classification, see subsection III.C for a detailed explanation.

During the training stage, the DCCIL method updates
the model parameters and exemplar sets using Algorithm 1 whenever a new class of data needs to be learned. In this pro-
cess, the update of the model parameters is the core part
class-incremental learning, see subsection III.B for the
specific parameter update strategy. Besides, the exemplar sets
need to be updated as a new class is added. The reduction of
the old class exemplar sets is realized by discarding exemplars randomly and only \( m \) exemplars are kept in each class.
The construction of the new class exemplar sets is realized by randomly selecting \( m \) samples from the training data of each new class.

Algorithm 1 Network Training Based on DCCIL Method
Start with:

\[
\Theta \quad \text{// current model parameters}
\]
\[
P = (P^1, \ldots, P^{s-1}) \quad \text{// current exemplar sets}
\]
\[
\text{FIM} = (F^1, \ldots, F^{k-1}) \quad \text{// the FIM of the first k-1 tasks}
\]

Input:

\[
X^1, \ldots, X^t \quad \text{// training images of new classes s, \ldots, t}
\]
\[
M \quad \text{// the total size of exemplar sets}
\]
\[
\Theta \leftarrow \text{UpdateRepresentation}(X^s, \ldots, X^t; \text{FIM}, \Theta)
\]
\[
\text{FIM} \leftarrow (F^1, \ldots, F^k) \quad \text{// record the FIM of the kth task}
\]
\[
m \leftarrow \frac{M}{t} \quad \text{// number of exemplars per class}
\]

for \( y = 1, \ldots, s - 1 \) do

\[
P^y \leftarrow \text{ReduceExemplar}(P^y, m) \quad \text{// for old classes}
\]

end

for \( y = s, \ldots, t \) do

\[
P^y \leftarrow \text{SelectExemplar}(X^y, m) \quad \text{// for new classes}
\]

end

\[
P \leftarrow (P^1, \ldots, P^t) \quad \text{// new exemplar sets}
\]

The classification part of the model is different between the
training stage and the testing stage. For training, the learnable convolution module is followed by a fully-connected layer
with as many sigmoid output nodes as classes observed so far. When there is a new class to learn, the number of nodes in the
fully-connected layer increases accordingly. The feature vec-
tors calculated by the convolution module are \( L_2 \)-normalized before being input into the fully-connected layer. Network
parameters are grouped into a fixed number of parameters for the feature extraction and a variable number of weight
vectors. We denote the latter by \( \mathbf{w}_1, \ldots, \mathbf{w}_t \in \mathbb{R}^d \), where \( t \) is the number of classes that have been observed so far. For any class \( y \in \{1, \ldots, t\} \), the output of the fully-connected layer is:

\[
g_y(x) = \frac{1}{1 + \exp(-a_y(x))}
\]

where \( a_y(x) = \mathbf{w}_y^T \phi(x) \) and \( x \) is an example image. Although these output values can be interpreted as the probabilities that the target belongs to a certain class, for the DCCIL method, the fully-connected layer is only used to calculate the loss at the training stage to guide the parameter update, while the final classification relies on the NME classifier.

**B. NETWORK PARAMETER UPDATE**

When DCCIL obtains data \( X^s, \ldots, X^t \) for the new classes \( s, \ldots, t \), Algorithm 2 describes the routine to update network parameters step by step. Firstly, the network training set is formed with the images of new classes. Secondly, each training image is fed into the pre-update network, and the output vector \( \mathbf{q}_y^t \) of the network is recorded. Finally, the network is trained with the training set and the network parameters are updated by minimizing the constructed loss function.

**Algorithm 2 DCCIL Update Representation**

Start with:

\[\Theta\] // current model parameters

\[\text{FIM} = (F^1, \ldots, F^{k-1})\] // the FIM of the first \( k-1 \) tasks

Input: \( X^s, \ldots, X^t \) // training images of new classes

\[D \leftarrow \bigcup_{y = s, \ldots, t} \{(x, y) : x \in X^y\}\] // training set

for \( y = 1, \ldots, s - 1 \) do

\[\mathbf{q}_y^t \leftarrow g_y(x_i) \text{ for all } (x_i, \cdot) \in D\]

end

// run network training with loss function

\[L(\Theta) = \text{L}_\text{new} + \text{L}_\text{old} + \text{L}_\text{EWC}\]

In the DCCIL method, for the network to be updated, the total loss function consists of 3 items, namely classification loss, distillation loss, and weight consolidation item. The specific form is as follows:

\[
L(\Theta) = -\sum_{(x_i, y_i) \in D} \left[ \delta_{y=y_i} \log g_{y_i}(x_i) + \delta_{y \neq y_i} \log(1 - g_{y_i}(x_i)) \right] \\
+ \sum_{y=1}^{s-1} \sum_{j=1}^{N_{\text{params}}} \mathbf{F}_j^{(k-1)}(\theta_j - \theta_{y}^{(k-1)})^2
\]

where \( \delta \) is an indicator function defined as \( \delta_{[\text{true}]} = 1 \), \( g_y(x_i) \) is the output value of the network as defined in (6), \( N_{\text{params}} \) is the total number of parameters in the network, \( \mathbf{F}_j^{(k-1)} \) is the summation of the \( j \)th diagonal element of all Fisher information matrixes on the first \( k - 1 \) learning tasks, and \( \theta_{y}^{(k-1)} \) is the value of the \( j \)th network parameter after finishing training on task \( k - 1 \). The specific calculation formula of \( \mathbf{F}_j^{(k)} \)

\[
\mathbf{F}_j^{(k)} = \gamma \mathbf{F}_j^{(k-1)} + F_j^{(k)}
\]

where \( \gamma \) is a hyper-parameter used to adjust the influence of previous tasks on weight consolidation, which is set to 1 in this paper.

The loss function \( L(\Theta) \) we constructed encourages the network to output the correct image class label on the new class node (reflected in classification loss), to output the recorded soft-target on the old class output node (reflected in distillation loss), and to maintain the stability of those weights that are important to the previous tasks during the network parameter update process.

The feature representation update process described in Algorithm 2 seems complex in form, but its implementation is similar to network fine-tuning: minimizing the loss function on a new data set with the previously learned network parameters as initial value. Therefore, DCCIL can be implemented with standard end-to-end learning methods, such as the back-propagation algorithm, to update the network. And the latest network optimization methods, such as Dropout [23] and batch normalization [24], can also be applied to network training.

Unlike network fine-tuning, however, the DCCIL approach has changed significantly in terms of loss function to overcome catastrophic forgetting. In addition to the standard classification loss item, the DCCIL method also has distillation loss item and weight consolidation item, which can stimulate the network to improve the classification accuracy of new classes without losing the discriminative information learned from previous tasks.

**C. NEAREST-MEAN-OF-EXEMPLARS CLASSIFIER**

In this paper, the nearest-mean-of-exemplars classifier is used to accomplish the final target classification. To predict the class \( y \) of a new image \( x \), the prototype vectors \( \mu_1, \ldots, \mu_t \) for each class observed so far are calculated. For class \( y \), the calculation formula of its prototype vector is

\[
\mu_y = \frac{1}{|P^y|} \sum_{p \in P^y} \phi(p)
\]

where \( |P^y| \) refers to the exemplar number of class \( y \), \( p \) is a sample in the exemplar set \( P^y \). So \( \mu_y \) is the average feature vector of all exemplars in class \( y \). For the image \( x \) to be inferred, the feature vector of the image only needs to be obtained through the convolution module, and compared with the prototype vector of each class, and then can be classified as the class corresponding to the nearest prototype vector:

\[
y^* = \arg \min_{y=1,\ldots,t} \| \phi(x) - \mu_y \|
\]

The reason we use exemplar features as the basis of classification is after considering the human visual recognition process. Although the current research results on the working
 mechanism of the human brain are limited [25], the brain can be regarded as a very accurate feature extractor and a superb painter. For example, when we hear the word “cup,” various types of cups come to mind. It is possible that our brains redrawn the cups using previously extracted and stored features about “cup.” These constructed cups serve as exemplars of “cup” that the brain stores. In other words, the brain stores not what the object looks like, but the features about the object. When we “see” an unfamiliar “cup,” the brain quickly extracts its features and, by analogy with the features of various objects previously stored in the brain, classifies the “cup” into the category of objects most similar to it, thus recognizing it as a water cup.

### IV. EXPERIMENTS AND RESULTS

In this section, we will design experiments to verify the effectiveness of the proposed DCCIL method in the Class-IL scenario, and compare the classification accuracy of DCCIL with other methods. The PC used for the experiments is configured as Centos7 operating system, Intel Core i7-9800X @ 3.80 GHz processor, RTX 2080Ti GPU, and 64 GB memory. Pytorch is used for the implementation of all algorithms in this section.

#### A. TASK PROTOCOL

Up to now, there is no agreed benchmark protocol for evaluating class-incremental learning methods. Therefore, we use the following evaluation procedure: for a given multi-class target data set, the classes are first arranged in random order and fixed. Then, each method is trained in a class-incremental manner on the available training data. After each batch of classes, the classification ability of the model is tested with the test data of classes that have already been trained, and the evaluation criterion is the classification accuracy. Convolutional neural network is trained on Echoscope sonar [26] image data set, the data set contains 9 classes sonar images of the real underwater targets including cornerstone, diver, remote operated vehicle, shipwreck, aircraft and tank wreck, the number of images for each target class is shown in Table 1.

| Target          | Sea experiments | Number of extracted images |
|-----------------|-----------------|---------------------------|
| Cornerstone     | 2009 USA        | 335                       |
| Diver           | 2010 UK         | 287                       |
| ROV             | 2011 UK         | 311                       |
| Sunked barge1   | 2012 UK         | 223                       |
| Sunked barge2   | 2012 UK         | 394                       |
| Sunked barge3   | 2012 UK         | 195                       |
| Shipwreck       | 2012 UK         | 370                       |
| Sunked plane    | 2012 UK         | 583                       |
| Sunked military tank | 2012 UK     | 217                       |

Taking the task protocol shown in Fig. 4 as an example, we divide the original Echoscope sonar image data set into 3 tasks, each of which contains 3 target classes and the order of the task and class could be set freely. 150 sonar images of each class are randomly selected from the data set to form a training set, and all the pseudo-color images are converted to gray-scale images and resized to 227 × 227 before being fed to the network.

For a fair comparison, the same convolutional neural network architecture is used for all methods, which is shown in Table 2. In the table, $k$ is the size of the convolution kernel or pooling kernel, $s$ is the stride, $d$ is the number of convolution kernel or node, $p$ is the padding parameter, and $\delta$ is the dropout rate. The number of nodes in the output layer is adjusted dynamically with the increase of the target class, and the maximum number of output nodes is 9 in this experiment.

#### B. PERFORMANCE ANALYSIS OF DCCIL METHOD

In Section III, we have described the proposed DCCIL method in detail. To verify the effectiveness of the method, experiments are carried out according to the protocol designed above. The standard back-propagation algorithm is used to train the network, and hyper-parameters are set as follows: the training iterations per task is 1000, the basic learning rate is 0.001, the weight decay parameter is 0.001, and the batch size is 45. The maximal number of exemplars is $M = 90$, that is to say, when there are at most 9 classes of targets to be classified, each class can store up to 10 exemplars, which occupy very small memory space.

To objectively evaluate the incremental learning performance of the DCCIL method, we tested two other network learning methods with the same model structure, namely joint learning and fine-tuning. Neither of these learning methods conforms to the specification of incremental learning, but their classification results can be used as the baseline for incremental learning method.

Fine-tuning learns the tasks separately according to the order of the tasks and without taking any measures to prevent catastrophic forgetting. We can also think of the fine-tuning method as learning a new multi-class classifier by fine-tuning the parameters of the network previously learned. Since fine-tuning does nothing to avoid the catastrophic forgetting that occurs during incremental learning, we can consider the experimental results of fine-tuning as a lower bound.

In contrast, joint learning can be seen as an upper bound for incremental learning. When learning each task, the joint learning method always uses the data of all tasks so far to train the model. Since all the historical data are involved in each task, it is equivalent to relearn the historical task while learning the new task. Therefore, the joint learning method
TABLE 2. Hyper-parameter settings of the used incremental learning model.

| Layer | Operation | Hyper-parameter setting | Input data dimension | Output data dimension |
|-------|-----------|------------------------|----------------------|----------------------|
| 1     | Convolution | $k = 3; p = 1; s = 2; d = 16$ | $227 \times 227 \times 1$ | $114 \times 114 \times 16$ |
|      | RelU       | —                      | 114$\times$114$\times$16 | 114$\times$114$\times$16 |
| 2     | Average Pooling | $k = 3; s = 2$ | 114$\times$114$\times$16 | 57$\times$57$\times$16 |
| Layer | Convolution | $k = 3; p = 1; s = 2; d = 32$ | 57$\times$57$\times$16 | 29$\times$29$\times$32 |
| 3     | RelU       | —                      | 29$\times$29$\times$32 | 29$\times$29$\times$32 |
| Layer | Convolution | $k = 3; p = 1; s = 2; d = 64$ | 29$\times$29$\times$32 | 15$\times$15$\times$64 |
| 4     | RelU       | —                      | 15$\times$15$\times$64 | 15$\times$15$\times$64 |
| Layer | Convolution | $k = 2; p = 0; s = 2; d = 64$ | 15$\times$15$\times$64 | 8$\times$8$\times$64 |
| 5     | RelU       | —                      | 8$\times$8$\times$64 | 8$\times$8$\times$64 |
| Layer | Fully-connected | $d = 600$ | 1$\times$1$\times$4096 | 1$\times$1$\times$600 |
| 6     | RelU       | —                      | 1$\times$1$\times$600 | 1$\times$1$\times$600 |
| Layer | Dropout   | $\delta=0.5$          | 1$\times$1$\times$600 | 1$\times$1$\times$600 |
| Layer | Output     | $d_{\max} = 9$        | 1$\times$1$\times$600 | 1$\times$$d_{\text{classes}}$ |

The experimental results are shown in Fig. 5. The value of each point in the figure is the recognition accuracy across all classes observed up to a certain time point. We can observe that the accuracy of the DCCIL method is significantly better than the fine-tuning method, while the performance gap with the joint learning method is small, with a recognition accuracy of 87.44% across the 9 classes. As expected, the overall recognition accuracy of the joint learning method is the best and up to 89.90% because it uses training data on all tasks. Besides, because the fine-tuning method does nothing to prevent catastrophic forgetting, the accuracy across the 9 classes is only 33.08%, which means that the knowledge learned from the first two tasks (contain 6 classes) is forgotten and only the last 3 classes can be effectively classified.

In Fig. 6, we show the recognition accuracy of each task after the network has learned all the 3 tasks. We can observe that for the first two tasks, the classification results of DCCIL are much better than the fine-tuning method and close to the accuracies of the joint learning method. For DCCIL method, the accuracy of the second task is lower than that of the first task, because the second task is more difficult, which also results in the accuracy reduction of the joint learning method. For the third task, the classification results of DCCIL are similar to those of the fine-tuning method and the joint learning method. As it is the most recent learning task, the recognition accuracies of the 3 methods are relatively high.

In addition to the recognition accuracy, we also compared DCCIL, fine-tuning and joint learning methods from several other aspects. Table 3 shows the advantages and disadvantages of the 3 methods in various aspects.

Based on the experimental results and the information in Table 3, it can be seen that the DCCIL method proposed in this paper can be well applied to the Class-IL scenarios. If we
want to increase the recognition classes for existing network models, the recognition accuracy of the DCCIL approach is close to joint learning. Moreover, DCCIL does not require historical data to participate in training, so the network update process is efficient.

The DCCIL method uses the NME classification strategy in the classification part, which requires each class to store a certain number of exemplars. In order to minimize memory footprint, the exemplar set for each class contains up to 10 sonar images in the experiment above. Here, in order to figure out the impact of the memory budget $M$ on the incremental learning performance, we change the size of the exemplar sets and the total number of exemplars increases from 45 to 450. With the learning model and other hyper-parameters unchanged, the experimental results are shown in Figure 7. The value of each point in the figure is the recognition accuracy across all 9 classes.

![FIGURE 7. The recognition accuracies of the DCCIL method for different memory budgets M.](image)

As can be seen from Fig. 7, with more exemplars, the recognition accuracy of the DCCIL method is gradually improved. The accuracy of the network is 86.97% when there are only 5 exemplars for each class, and 88.63% when there are 50 exemplars for each class. This is because the more exemplars of each class, the more accurate the feature mean of each class can be obtained, which is more conducive to the image inference. With the recognition system allowing more exemplars to be stored, the target recognition accuracy of the DCCIL method is expected to improve further.

C. METHODS COMPARISON

We carry out extensive experiments to compare the class-incremental learning performance of the proposed DCCIL method with several popular incremental learning methods, which are briefly introduced below.

1) Elastic weight consolidation (EWC): by adding a weight consolidation term in the loss function to reduce changes to the important network parameters for old tasks. See subsection II.A for a detailed introduction.

2) Learning without forgetting (LwF): add a loss item about replayed data to the loss function of the current task: $L_{\text{total}} = L_{\text{current}} + L_{\text{replay}} + R$. There are two main differences between the DCCIL method and the LwF method in terms of the parameter update. First, after learning the old task, the DCCIL method needs to calculate the Fisher information matrix to measure the importance of each parameter to the old task. Second, the loss function of the DCCIL method is different from that of the LwF method—the third item of LwF is a regularization item, while the DCCIL is weight consolidation item. The implement of the LwF method in this experiment refers to the paper [14].

3) Incremental classifier and representation learning (iCaRL): the characteristic of this method is that for the model update, part images of the old tasks will be selected and mixed with the new task data for network training, to retain the discriminative information of historical classes. The implement of the iCaRL method in this experiment refers to the paper [6], and the number of exemplars is also set to 90. The DCCIL method proposed in this paper and the iCaRL method both require exemplars to help network classification, but there are 3 obvious differences: 1) DCCIL method does not require exemplars to participate in network training; 2) In the classification stage, exemplars used by DCCIL method are randomly selected, while exemplars in iCaRL method are selected by complex clustering algorithm based on herding [27]; 3) DCCIL method refers to the memory mechanism of the human brain and restricts the updating of network parameters at the same time through weight consolidation and knowledge distillation, while the iCaRL method only uses knowledge distillation. Therefore, compared with the iCaRL approach, DCCIL approach has 3 advantages in terms of implementation: 1) Historical data does not need to participate in the new task, so network training is more efficient; 2) Avoid the tedious process of exemplar selection and update; 3) To overcome the catastrophic forgetting problem from the perspective of the loss function so that the algorithm is easy to generalize and has better application perspective.

All methods use the same convolutional neural network structure designed in subsection IV.A. Different from DCCIL and iCaRL, the last layer of EWC and LwF are both Softmax output layer. The main hyper-parameters in each method are set in the same way: the Adam algorithm is used for parameter optimization with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, each model is trained 1000 iterations with an initial learning rate of 0.001 and a batch size of 45. The experimental results are
As can be seen from Table 4, EWC and LwF have poor learning ability in the Class-IL scenario, while the iCaRL and the proposed DCCIL method can both achieve good incremental recognition accuracy. Specifically, the LwF method using knowledge distillation has a higher recognition accuracy than the EWC method relying solely on weight consolidation. In the 4 methods, the accuracy of DCCIL method is the best, since the DCCIL method uses both knowledge distillation and weight consolidation to train the network, so as to better preserve the knowledge learned from historical tasks while learning new tasks. In addition, the NME classifier is used to classify targets in the prediction stage, which is robust to changes in feature expression.

V. CONCLUSION

To overcome the catastrophic forgetting that often occurs in neural networks during continuous learning, this paper proposes a new class-incremental learning method named DCCIL. The DCCIL method draws on the memory mechanism of the human brain, which uses knowledge distillation and weight consolidation strategies at the same time during the network training stage, and adopts the nearest-mean-of-exemplars classifier to recognize targets in the prediction stage. Experimental results on the sonar image data set show that the average recognition accuracy of the DCCIL method can achieve 87.44%, which is close to the recognition accuracy of non-incremental joint learning method and better than the popular incremental learning methods such as iCaRL, LwF, and EWC. Besides, with the increase of target classes, the performance of DCCIL decreases slowly, indicating that the method has strong conservation ability for the knowledge of historical tasks. Although the experiments are carried out on a sonar image data set, the DCCIL method is indeed universal and can be applied to the Class-IL scenarios with optical images or medical images.

REFERENCES

[1] V. Losing, B. Hammer, and H. Wersing, “Incremental on-line learning: A review and comparison of state of the art algorithms,” Neurocomputing, vol. 275, pp. 1261–1274, Jan. 2018, doi: 10.1016/j.neucom.2017.06.084.
[2] G. I. Parisi, J. Tani, C. Weber, and S. Wermter, “Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization,” Frontiers Neurobotics, vol. 12, pp. 1–15, Nov. 2018, doi: 10.3389/fnbot.2018.00078.
[3] C. Küding, E. Rodner, A. Freytag, and J. Denzler, “Fine-tuning deep neural networks in continuous learning scenarios,” in Computer Vision. Cham, Switzerland: Springer, 2016, pp. 588–605.
[4] G. M. van de Ven and A. S. Tolias, “Three scenarios for continual learning,” 2019, arXiv:1904.07734. [Online]. Available: http://arxiv.org/abs/1904.07734
[5] W. Liu, H. Yang, Y. Sun, and C. Sun, “A broad neural network structure for class incremental learning,” in Proc. Int. Symp. Neural Netw., Minsk, Belarus, 2018, pp. 229–238.
[6] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “iCaRL: Incremental classifier and representation learning,” 2016, arXiv:1611.07725. [Online]. Available: http://arxiv.org/abs/1611.07725
[7] G. Christophe, “A note on the utility of incremental learning,” AI Commun., vol. 13, no. 4, pp. 215–223, Feb. 2004, doi: 10.1117/12.384880.
[8] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, “Learn++: An incremental learning algorithm for supervised neural networks,” IEEE Trans. Syst., Man, Cybern. C, Appl. Rev., vol. 31, no. 4, pp. 497–508, Nov. 2001, doi: 10.1109/5326.983933.
[9] S. Furao and O. Hasegawa, “An incremental network for on-line unsupervised classification and topology learning,” Neural Netw., vol. 19, no. 1, pp. 90–106, Jan. 2006, doi: 10.1016/j.neunet.2005.04.006.
[10] M. Mühlbaier, A. Topalis, and R. Polikar, “Learn++: NC: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes,” IEEE Trans. Neural Netw., vol. 20, no. 1, pp. 152–168, Jan. 2009, doi: 10.1109/TNN.2008.2008326.
[11] R. Elwell and R. Polikar, “Incremental learning of concept drift in non-stationary environments,” IEEE Trans. Neural Netw., vol. 22, no. 10, pp. 1517–1531, Oct. 2011, doi: 10.1109/TNN.2011.2160459.
[12] C. L. P. Chen and Z. Liu, “Broad learning system: An effective and efficient incremental learning system without the need for deep architecture,” IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 1, pp. 10–24, Jan. 2018, doi: 10.1109/TNNLS.2017.2716952.
[13] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, “Overcoming catastrophic forgetting in neural networks,” Proc. Nat. Acad. Sci. USA, vol. 114, no. 13, pp. 3521–3526, Mar. 2017, doi: 10.1073/pnas.1611835114.

[14] Z. Li and D. Hoiem, “Learning without forgetting,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 12, pp. 2935–2947, Dec. 2018, doi: 10.1109/TPAMI.2017.2773081.

[15] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” 2015, arXiv:1503.02531. [Online]. Available: http://arxiv.org/abs/1503.02531

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Columbus, OH, USA, Jun. 2014, pp. 580–587.

[17] R. M. French, “Catastrophic forgetting in connectionist networks,” Trends Cognit. Sci., vol. 3, no. 4, pp. 128–135, May 1995, doi: 10.1016/S1364-6613(99)01294-2.

[18] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, “Mastering the game of go without human knowledge,” Nature, vol. 550, no. 7676, pp. 354–359, Oct. 2017, doi: 10.1038/nature24270.

[19] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber, “Compete to compute,” in Proc. Int. Conf. Neural Inf. Process. Syst., New York, NY, USA, 2013, pp. 2310–2318.

[20] S. C. Mednick, D. J. Cai, T. Shuman, S. Anagnostaras, and J. T. Wixted, “An opportunistic theory of cellular and systems consolidation,” Trends Neurosci., vol. 34, no. 10, pp. 504–514, Oct. 2011, doi: 10.1016/j.tins.2011.06.003.

[21] M. K. Benna and S. Fusi, “Computational principles of synaptic memory consolidation,” Nature Neurosci., vol. 19, no. 12, pp. 1697–1706, Oct. 2016, doi: 10.1016/j.nn.2016.06.003.

[22] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, “Distance-based recognition and communication engineering. His research interests include sonar image processing, pattern recognition, and machine learning.

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

[24] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015, arXiv:1502.03167. [Online]. Available: http://arxiv.org/abs/1502.03167

[25] J. Hawkins, “Special report: Can we copy the brain?—What intelligent machines need to learn from the neocortex,” IEEE Spectr., vol. 54, no. 6, pp. 34–71, Jun. 2017, doi: 10.1109/MSP.2017.7934229.

[26] A. Davis and A. Lugsdin, “High speed underwater inspection for port and harbour security using coda echoscope 3-D sonar,” in Proc. OCEANS, Washington, DC, USA, 2005, pp. 2006–2011.

[27] M. Welling, “Herding dynamical weights to learn,” in Proc. 26th Ann. Int. Conf. Mach. Learn., Montreal, QC, Canada, 2009, pp. 14–18.