Single-head lifelong learning based on distilling knowledge

YEN-HSIANG WANG¹, CHIH-YANG LIN², (Senior Member, IEEE), TIPAJIN THAIPISUTIKUL³, AND TIMOTHY K. SHIH.¹ (Senior Member, IEEE)
¹Department of Computer Science and Information Engineering National Central University, Taiwan
²Department of Electrical Engineering, Yuan Ze University, Taoyuan 32003, Taiwan
³Faculty of Information and Communication Technology, Mahidol University, Thailand

Corresponding author: Chih-Yang Lin (andrewlin@saturn.yzu.edu.tw) and Tipajin Thaipisutikul (tipajin.tha@mahidol.ac.th).

ABSTRACT Within the machine learning field, the main purpose of lifelong learning, also known as continuous learning, is to enable neural networks to learn continuously, as humans do. Lifelong learning accumulates the knowledge learned from previous tasks and transfers it to support the neural network in future tasks. This technique not only avoids the catastrophic forgetting problem with previous tasks when training new tasks, but also makes the model more robust with the temporal evolution. Motivated by the recent intervention of the lifelong learning technique, this paper presents a novel feature-based knowledge distillation method that differs from the existing methods of knowledge distillation in lifelong learning. Specifically, our proposed method utilizes the features from intermediate layers and compresses them in a unique way that involves global average pooling and fully connected layers. We then use the output of this branch network to deliver information from previous tasks to the model in the future. Extensive experiments show that our proposed model consistency outperforms the state-of-the-art baselines with the accuracy metric by at least two percent improvement under different experimental settings.

INDEX TERMS Lifelong Learning, Continuous Learning, Incremental Learning, Knowledge Distillation

I. INTRODUCTION

In the real world, humans can learn things incrementally. We might have learned skills like reading, calculating, and speaking during childhood. As we get older, we don’t easily forget what we have learned; we can continue to use those skills. However, traditional machine learning models cannot be easily programmed with the same ability to learn incrementally as a human does since it requires a new training process each time new data become available. Thus, there is a call for dynamic systems that keep learning over time.

We call this concept as lifelong-learning, also referred to as continuous learning. Lifelong learning is different from traditional machine learning since lifelong learning models gradually learn from the infinite stream of data over time in a sequential fashion. The lifelong learning approach aims to maintain the performance on a previously learned task as new tasks or domains are added. Under these assumptions, when we research lifelong learning, we face some following challenges.

Firstly, how can we apply knowledge transfer to mimic the human capability to use previous experience to make better future learning. In other words, we should design strategies to enable the lifelong learning model to use prior knowledge to enhance future learning performance efficiently. Secondly, how can we deal with the problem of catastrophic forgetting since we only can use the data from the current task to train the neural network? In particular, the backpropagation [3, 4, 5] process will modify the neural network parameters using the data that exists in different timings, leading to a data imbalance problem [6, 7, 8]. This problem causes our neural network model to perform well on new tasks but poorly on old tasks. Lastly, the neural network should be scalable enough when tasks increase to handle whole tasks completely.

To overcome the challenges, we need to achieve several goals. To begin with, the model must be able to train in a situation where the data is sequentially provided. We can only use a small amount of data from old tasks, and the storage space for the old data must be limited [9]. Then, this lifelong learning model should have the ability to classify categories of the data that appeared and were trained at any time before. Finally, the classifier of this lifelong learning model should expand dynamically to handle the growth of the classes over time.
Class-incremental lifelong learning can be divided into two sub-fields: multi-head configuration and single-head configuration. On the one hand, multi-head lifelong learning [10, 11, 12] will have task information at the testing stage; this configuration usually has a structure with multiple output layers for each task and will choose a suitable output layer based on the task label of the input during testing. Therefore, it does not need to handle the influences of the data imbalances between each task. Under this configuration, researchers focus on increasing memory utilization, which increases the proportion of shared parameters. The goal is to make the shared parameters contain as much information from the previous tasks as possible so that less memory is required to construct a multi-head lifelong learning model. This allows the neural network to converge more quickly in the training stage by transferring knowledge.

On the other hand, the researchers in the single-head lifelong learning [9, 13, 14, 15] field pay more attention to the problem of imbalanced data between each task. Hence, the model configuration does not have the tag of the task during the testing stage, the model must automatically predict which category a task belongs to when limited rehearsal storage, as shown in Figure 1. Consequently, the researchers have to deal with both catastrophic forgetting and model preference problems.

To this end, in this paper, we propose a novel feature-based knowledge distillation method to overcome the catastrophic forgetting problem in a single-head lifelong learning classification task. Unlike traditional knowledge distillation methods [16], which only distill knowledge from predictions, we also transfer the knowledge from intermediate layers to deal with model preference caused by imbalanced data by using a branch network composed of the average pooling layers and fully connected layers. In contrast to traditional distillation methods that utilized the Euclidean distance to estimate the similarity of features between teacher and student networks [15, 17, 18], we use the soft target from both predictions of the branch network and the backbone network to calculate the distillation loss and classification loss in the training stage. In this way, we eliminate the model preference problem due to the data imbalance between the old task and the new task by removing the last output layer and using the intermediate layer as a feature vector to indicate the possible output class. The feature map from intermediate layers includes more spatial information and feature details than the last output layer, which generally gives a higher probability of new classes in the new task training. Therefore, we can obtain better performance in single-head lifelong learning with our distillation method.

II. RELATED WORK

In this section, we will describe the techniques employed in the proposed method. The explanation is divided into two parts. In the first part, we focus on the technologies of knowledge distillation [16]. In lifelong learning, it is very useful to extract the knowledge from the model of previous tasks and transfer it to enhance the learning performance of our model. In the second part, we describe some lifelong learning methods based on knowledge distillation. These methods distill knowledge from pre-updated neural networks and fine-tune the current network. They also use special techniques to design their classifiers to avoid classification preferences resulting from imbalanced data. Last but not
least, we describe a single-head lifelong learning method based on feature-based knowledge distillation.

A. KNOWLEDGE DISTILLATION

Knowledge distillation [16] is a technique that can transfer knowledge from a neural network that has already been trained, to another network through soft probabilities. This technique is widely used in the field of model compression by condensing the knowledge in a neural network into soft probabilities. Hence, the student network can efficiently obtain knowledge from the teacher network using these soft targets. Therefore, the student network can be trained more easily. Typically, neural networks usually use a “softmax” operator to generate the output of the model. It divides the logit $z_i$ by the summation of other logits to transform a logit into a probability $q_t$. The key point of the knowledge distillation is that it will divide all logits by the hyperparameter $T$ first. Where $T$ is temperature, it is usually set to 1 in the general training situation for neural networks. However, in a knowledge distillation scenario, we will set our temperature $T$ greater than 1 to get a softer probability. The larger the temperature $T$ we set, the smoother the probability we get. Through this soft target from the teacher network, the student network can understand the correlation between classes. By using this extra information that is not contained in the ground truth, the student network can train itself more easily and perform better. In the lifelong learning field, knowledge distillation [9, 13, 14, 15] provides an effective way to transfer knowledge from a pre-updated model to a new one if only a few examples from previous tasks are available. The neural network can alleviate catastrophic forgetting by recalling knowledge that it has learned from prior tasks. For example, Farquhar and Gal [19] studied knowledge distillation in continual learning with various evaluation strategies and highlighted shortcomings of the common practice to use one dataset with different pixel permutations.

B. SINGLE-HEAD LIFELONG LEARNING

In general deep learning models perform well when all data is mixed for training. Since the model is trained task by task with the labels separate, results tend to be more predictive in new tasks and poor in old tasks [9]. Lifelong learning, the ability to learn a new task like humans without forgetting previous knowledge, is the primary method used to solve this problem. For example, Lesort et al. [20] descriptively analyzed and studied continual learning in dynamic environments, especially in robotics. Pfui lb and Gepperth [21] studied the catastrophic forgetting effect and developed a protocol for setting hyperparameters to minimize this effect.

In lifelong learning, the model is trained gradually task by task. The labels for each task are different. While training new tasks, data in the previous tasks cannot be used. Lifelong learning can fundamentally be divided into two categories: multi-head and single-head. Multi-head [10, 11, 12] makes predictions with task information and single-head makes predictions without task information. We focus on single-head lifelong learning [9, 13, 14] research, which is based on knowledge distillation.

Unlike model compression, knowledge distillation in lifelong learning will encounter a problem with class extensions. Because the classes in lifelong learning increase as time passes, the traditional knowledge distillation methods in model compression field cannot directly shift to support model training in lifelong learning. Therefore, Li et al. proposed a lifelong learning method, Learning without Forgetting (LwF) [13], which is based on knowledge distillation. They combine the soft target obtained from pre-updated neural networks and the ground truth for new tasks as training targets. By calculating the loss between this expanded target and the predictions from the new model that has already expanded, they get the classification and distillation loss at the same time. In this way, they integrate the concept of knowledge distillation into the lifelong learning field. In the absence of old task data, the neural network can obtain the information from old tasks through the soft target from the pre-updated model.

Although LwF uses the concept of knowledge distillation to acquire the knowledge from old tasks and tries to resolve the problem of insufficient samples from the old task, it still encounters the problem of model classification preference due to the data imbalance between the old task and the new task. The output layers of the model prefer to give a higher probability to new classes because of the greater sample size in the training of new tasks. Rebuffi et al. proposed iCaRL [9] to handle this dilemma. They added three components into their training process to mitigate the problem of classification preference. First, their classifier classifies the classes by nearest-mean-of-exemplars (NME) instead of using a dense layer during testing. They eliminate the classification preference problem by directly removing the fully-connected layer and replacing it by calculating the Euclidean distance between the mean of exemplars of each class and the feature vectors of the input image to determine the label of the input images. Because they adopt the nearest-mean-of-exemplars strategy for their classifier, they need to store some exemplars of each class. This rehearsal strategy is the second difference from LwF. Nevertheless, the size of the exemplar storage should be small and limited under the definition of lifelong learning. They need to remove some exemplars when new tasks occur, so they design a strategy based on herding to prioritize exemplar selection. This selection strategy is the third component that they propose.

On the other hand, Wu et al. have proposed BiC [14], which uses the fully connected layer with a bias layer as the classifier. They employ a two-stage training mode and a special structure to their model, which contains a bias layer to handle the preference of the output layer. In the first stage, they train their backbone neural networks just like iCaRL and LwF do. After training the backbone, they train the bias correction layers with the validation set, which has the same number of samples for each class. During the second stage, the backbone should be frozen. When they train the bias cor-
rection layers with the same amount of data in each class, the bias correction layer will be able to reduce model preferences by assigning different weights to the predictions of the new and old categories. The structure of bias correction layers must be simple given the constraints of limited validation data. Therefore, the bias correction layers are composed with a linear model that only contains two parameters. This design can use the dense layers as the classifier and still achieve robust performance.

C. FEATURE-BASED KNOWLEDGE DISTILLATION IN LIFELONG LEARNING

Many traditional feature-based knowledge distillation methods have been developed [17, 18]. Hence, some researchers have started to consider whether those methods can be added to the lifelong learning model. Douillard et al. proposed PODNet [15], which added the concept of feature-based knowledge distillation into their method. In order to reduce noise interference and feature complexity, they try to combine different kinds of pooling operators to get better features. They can then use these processed features to distill the knowledge from the pre-updated network to the current model. Through experimentation, they obtained a combination of pooling operators that can effectively balance between the rigidity and plasticity of the lifelong learning network.

Although there is a growing success of lifelong learning studies to diminish the effect of catastrophic forgetting on the model performance by proposing various knowledge distillation techniques, most of them still stress the experiments under multi-head settings, which hides the actual difficulty of the problem. Also, their evaluation is mainly limited to small datasets, approximately ten classes for all tasks. We are different from the previous studies with the following aspects. First, we propose a novel knowledge distillation with a branch network in the intermediate layers. Therefore, the knowledge from the prior tasks can adequately transfer to the current model. Second, we conduct the experiments on larger datasets and many incremental learning tasks in an experimental setting. Third, we perform various ablation studies to mine the best model architecture for the single-head lifelong learning classification task.

III. PROBLEM DEFINITION

Define $\mathcal{T} = \{ \mathcal{T}^1, \mathcal{T}^2, \ldots \}$ as the dataset of each task. $X = \{ X^1, X^2, \ldots \}$ donate the data stream in a class-incremental setting, the superscript means the classes it belongs to. Because each task contains several classes, we define $S_n$, where classes $n, n+1, \ldots, m$ appear when time $t$. Then, we define dataset of each classes $X^y = \{ x^y_1, \ldots, x^y_n \}$, where $n$ means $n^{th}$ data in the dataset of class $y$.

Next, we define $S = \{ S^1, S^2 \}$ as the exemplars of each classes which is stored in a fixed size of memory, and the superscript means the classes it belongs to. Define $S^y = \{ s^y_1, \ldots, s^y_m \}$, where $m$ means $m^{th}$ data in the exemplars of class $y$. Lastly, we define $f_{\text{backbone}}(x)$ as the output of the backbone and $f_{\text{branch}}(x)$ as the prediction of the branch network for any class $y$. For the pre-updated model, we use $h$ to represent it.

As noted earlier, we focus on the Class-incremental Single-head Lifelong Learning problem. With the definition of Class-incremental Lifelong Learning problem, the total number of the classes will be increased as time passes by. Besides, the dataset $\mathcal{T}^1, \mathcal{T}^2, \ldots, \mathcal{T}^{t-1}$ won’t be accessible when time $t$. Only the exemplars $S$ which store in a fixed size storage can be used for training. The Single-head problem means that we won’t have the task label at testing stage, so we need to deal with the problem of data imbalance and predict the label of input images correctly.

In summary, given the $\mathcal{T} = \{ \mathcal{T}^1, \mathcal{T}^2, \ldots \}$, we aim to obtain learned functions $f_{\text{backbone}}^y(x)$ and $f_{\text{branch}}^y(x)$ that can predict the classes of the $X = \{ X^1, X^2, \ldots \}$ data stream correctly on both old and new tasks.

IV. KNOWLEDGE DISTILLATION WITH A BRANCH NETWORK

Intuitively, catastrophic forgetting is a result of the tuning parameters of the model, which are based on existing data. In the lifelong learning field, we only have few data that belong to old tasks. In this case, the parameters of the neural network can perform classification well in the current task after tuning. However, it loses the capability to identify the data from old tasks. Therefore, we use knowledge distillation methods like [9, 13, 14] to avoid catastrophic forgetting. In addition, we believe that feature-based knowledge distillation can provide more information about old tasks. Unlike the prediction from the last layer, the feature maps from intermediate layers contain more spatial information and feature details. Consequently, we adopt feature-based methods to alleviate forgetting as presented in Figure 2.

Traditional feature-based knowledge distillation such as [17, 18], use convolution operators to reshape and extract the feature maps obtained from intermediate layers. By doing this, the complexity of the feature maps can be decreased. We can also obtain features that are more representative and less noisy. However, it is not easy to train a robust convolution layer when imbalanced data exists. Consequently, the feature-based knowledge distillation method in the field of lifelong learning [15] adopts pooling layers to replace the convolutions. Afterward, it calculates the Euclidean distance or Kullback Leibler distance between these intermediate features from the pre-updated model and the current neural network as the feature-based distillation loss.

Distinct from the aforementioned methods, our system adds an extra dense layer after the pooling operator to condense the acquired features. We add a branch network at the intermediate layer of the feature extractor in order to handle the knowledge distillation problem. When the input image is processed by convolutional operators and pooling operators in the feature extractor, we can get the $w \times h \times c$ feature maps. Next, these feature maps are converted into a feature vector with a size $c$ after processing via global average pooling. This step reduces the complexity of the feature maps and obtains
a highly representative vector to replace the complex feature maps. We next add a fully connected layer to transfer this feature vector linearly into the probabilities of classes. The size of this fully connected layer is the number of classes, so it will increase as the number of categories increases. By converting the feature vector into probabilities, we can enhance its abstract representation and condense this feature vector again to make it simpler and easier to learn. This soft target from the intermediate layer helps us to obtain different scale of features information. Distilling knowledge using these probabilities allows us to achieve the effect of feature-based knowledge distillation.

The training loss will be divided into two parts, the classification and distillation loss of the backbone network $L_{backbone}$ and the branch network $L_{branch}$. First of all, we define the accessible dataset $D$ at a specific time as Equation (1) below:

$$
D = \bigcup_{y=n, \ldots, m} \{ (x, y) : x \in X^y \} \\
\cup \bigcup_{y=1, \ldots, n-1} \{ (x, y) : x \in S^y \}
$$

(1)

where classes $1, \ldots, n - 1$ belong to old tasks, so we only can use the examples in the storage $S$ at training stage. On the other hand, because classes $n, \ldots, m$ belong to current task, we can access the whole dataset when training.

The loss of the backbone $L_{backbone}$ is defined below in Equation (2). $x^i$ refers to the input image and $y^i$ is the label for $x^i$. This loss comprises classification and distillation loss. For the classes that belong to old tasks, we will calculate the binary cross-entropy loss between the outputs from the pre-updated and the current model as the distillation part. Otherwise, we use the ground truth and the output from the current model to calculate the binary cross-entropy loss as classification loss.

$$
L_{backbone} = - \sum_{(x, y) \in D} \sum_{y=n}^{m} \delta_{y=y^i} \log f_y^y(x^i) \\
+ \delta_{y\neq y^i} \log (1 - f_{y^i}^y(x^i)) \\
+ \sum_{y=1}^{n-1} h_y^y(x^i) \log f_y^y(x^i) \\
+ (1 - h_{y^i}^y(x^i)) \log (1 - f_{y^i}^y(x^i))
$$

(2)

After that, we define the loss of the branch network $L_{branch}$ as Equation (3). This loss also consists of classification and distillation loss, just like $L_{backbone}$. We only need to change the output which is obtained from the backbone network to the output from the branch network.

$$
L_{branch} = - \sum_{(x, y) \in D} \sum_{y=n}^{m} \delta_{y=y^i} \log f_y^y(x^i) \\
+ \delta_{y\neq y^i} \log (1 - f_{y^i}^y(x^i)) \\
+ \sum_{y=1}^{n-1} h_y^y(x^i) \log f_y^y(x^i) \\
+ (1 - h_{y^i}^y(x^i)) \log (1 - f_{y^i}^y(x^i))
$$

(3)

At last, we can define our final loss $L_{final}$ as Equation (4). $\lambda$ is a hyperparameter which represents the ratio between two losses.

$$
L_{final} = \lambda L_{backbone} + (1 - \lambda) L_{branch}
$$

(4)

At the testing stage, we verify the validity of our knowledge distillation through NME [9].

In the training phase, we refer to [9] and add our knowledge distillation methods into the process. Algorithm 1 presents the training process, which uses the NME as the classifier for the testing stage. In the beginning, we randomly initialize the model and empty the storage of the exemplars. When the new task occurs, the training dataset $D$ is constructed by combining data from the new task and the exemplars of the old tasks. Next, we have to expand the output layer because of the increase in classes. In practice, we expand the parameters of the output layer, but others, such as the feature extractor, remain the same as in the original model. After setting up the architecture of the network, we can start to train it by using the loss function $L_{final}$. If the current task is not the first task, we use the output of the network with pre-updated parameters and the output of the current model to calculate the loss $L_{final}$ and update the parameters of the neural network. After training, we update the exemplar set by estimating the distance between feature vectors from different exemplars. We keep the exemplar that is closer to the average of the feature vectors of all exemplars in the same class. This updating strategy for exemplars is derived from iCaRL [9]. Lastly, we save the current model at $\theta'$ temporarily, so we can use it to calculate the loss function when a new task is obtained.
V. MULTI-LAYERS DISTILLATION
If we want to distill the knowledge through multiple scales of features at the same time, we can use simple concatenation to combine the feature vectors produced by average pooling operators at different layers. This multi-layer distillation structure is shown in Figure 3.

VI. EXPERIMENTS
In this section, we design several experiments to evaluate the performance of our method and compare the results against those of other renowned methods within the field of lifelong learning such as LwF [13], iCaRL [9], BiC [14], and PODNet [15]. In addition, we test the influence of the knowledge distilled from different layers and identify the architecture that obtains the most robust performance.

A. EXPERIMENTAL SETUP
1) RESEARCH QUESTIONS
We aim to evaluate the effectiveness of the proposed model by answering the following research questions: **RQ1:** Does our proposed model perform better than the state-of-the-art baseline methods in single-head lifelong learning classifi-
**Algorithm 1 Training process**

**Input:** $X^n, \ldots, X^m$ // training images of $T^t$

**Output:** prediction of the classes which exist

**Require:**
- $S = (S^1, \ldots, S^{n-1})$ // exemplar sets
- $\theta$ // current parameters
- $\theta'$ // pre-updated parameters

1: Setup training data $D$:
   $$ \bigcup_{y=m+1} \{ (x, y) : x \in X^y \} \cup \bigcup_{y=1} \{ (x, y) : x \in S^y \} $$

2: Train the network with the loss function:
   $$ L_{final} = \lambda L_{backbone} + (1 - \lambda) L_{branch} $$
   // use pre-updated parameters $\theta'$ to obtain the soft-target of classes $1 \ldots n - 1$

3: Update $S$ // update the exemplar sets
4: $\theta' \leftarrow \theta$ // update the pre-update parameters

**Questions: RQ2:** Does the number of incremental tasks affect our proposed model’s performance compared with other baselines? **RQ3:** Should we use multiple layers for feature-based knowledge distillation? **RQ4:** How does the performance of our proposed model when introducing a bias layer?

2) **DATASETS**

We adopted CIFAR-100 [22] as our experimental dataset. CIFAR-100 is a well-known dataset widely used in classification research. As the name implies, it includes 100 classes, and each category contains 600 images. A challenge with this dataset is that the resolution of the images is small; the width and height are only 32 x 32. In the experiment, we separate the dataset into several tasks during training, and each task is accessible at a different time. Afterward, we will use random cropping, random horizontal flipping, etc., and standard data augmentation methods to generate additional data.

In our experiments, we divide the datasets into several tasks with an incremental task setting. Data arrives sequentially in batches, and one batch corresponds to one task. Since CIFAR-100 has 100 classes, we experiment with a different number of tasks ranging from 5 tasks (20 classes per task), 10 tasks (10 classes per task), and 20 tasks (5 classes per task). We assume that all data becomes available simultaneously for a given task for offline training. Notably, the model cannot access data from previous or future tasks. Thus, optimizing for a new task in this setting might result in catastrophic forgetting, with significant declines in performance for old tasks.

3) **BENCHMARK PROTOCOL**

During the experiment, we refer to the evaluation methods from iCaRL [9]: we divide a classification dataset into several tasks, and the order of those classes are fixed after randomization. Unlike new tasks, which the model hasn’t learned before, the samples of previous tasks aren’t accessible. Only a few exemplars of previous tasks can be used for training. The total number of these exemplars is fixed, and they will be mixed with the data from the new task for training data. After training, we focus on the performance of the tasks that have been learned, and we use the testing data to evaluate it. In lifelong learning, the model should have the ability to distinguish between images that have been learned at any time. Therefore, we test our model after training each batch of classes. After training on whole tasks, we calculate the average accuracy on those to determine the average performance of the model at each time point. In [9], this average is called average incremental accuracy.

4) **IMPLEMENTATION**

We use Resnet-18 [23] as our backbone, and train it using the whole dataset for the new task and 2,000 exemplars for the old tasks. The storage of the exemplars is fixed, and each class has the same amount of space allocated to it. Therefore, the examples for each class will decrease as the number of categories increases. Each task will be trained for 100 epochs. The learning rate starts at 2.0 and will decrease to 0.4, 0.08, 0.016 at 50, 60, and 80 epochs, respectively. All of our methods use backpropagation with a batch size of 128 and weight decay $10^{-5}$. The loss of the backbone and the branch network, we adopt binary cross-entropy loss to evaluate the training situation. The training process is implemented under the **pytorch** framework.

**B. RESULT**

1) **EVALUATION OF DIFFERENT METHODS**

In this section, we compare the results of our method with those of other state-of-the-art lifelong learning methods, including LwF [13], iCaRL [9], BiC [14], and PODnet [15]. All models are trained on the scenarios mentioned earlier. Those methods all use knowledge distillation to avoid catastrophic forgetting. Moreover, PODnet adopts feature-based knowledge distillation to enhance the rigidity of the model. Aside from LwF, the other methods store a part of the samples as exemplars.

**Answer to RQ1:** The results on different numbers of tasks are shown in Table 1, Table 2, Table 3, and Figure 4. Based on the average accuracy of all tasks, our method outperforms others in several environments that split the dataset into different task sizes. If we divide the dataset into a few tasks, the performance of our method does not differ much from that of iCaRL and BiC. As seen in Table 1, when we only have 5 tasks, the performance is similar to the results with iCaRL and BiC. However, the average accuracy of our method shows a $2 \sim 3\%$ improvement when the number of tasks increases.

**Answer to RQ2:** Our method is based on iCaRL and adds our knowledge distillation method to it. With this small change, the average accuracy can increase by two percent on the split into 10 classes and 5 classes. PODNet always outperforms ours on the first few tasks; it performs well in the early stage regardless of the experimental situation. Nevertheless, its accuracy decreases rapidly when incremental tasks arrive.
TABLE 1. The accuracy of multi-class on CIFAR-100 with 20 classes each task

| Name   | Task1 | Task2 | Task3 | Task4 | Task5 | Avg. Accuracy |
|--------|-------|-------|-------|-------|-------|---------------|
| Ours   | 81.7  | 72.18 | 66.2  | 60.96 | 57.61 | **67.73**     |
| LwF    | 80.5  | 61.5  | 50.1  | 41    | 38.2  | 54.26         |
| iCaRL  | 81.75 | 71.55 | 65.28 | 61.31 | 57.03 | 67.38         |
| BiC    | 80.1  | 70.52 | 64    | 58.41 | 54.32 | 65.47         |
| PODnet | 80.3  | 71.8  | 59.5  | 51.5  | 44.5  | 61.52         |

TABLE 2. The accuracy of multi-class on CIFAR-100 with 10 classes each task

| Name   | Task1 | Task2 | Task3 | Task4 | Task5 | Task6 | Task7 | Task8 | Task9 | Task10 | Avg. Accuracy |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|---------------|
| Ours   | 88    | 77.9  | 71.1  | 66.52 | 63.68 | 63.03 | 61.29 | 58.2  | 56.29 | 55.82  | **64.81**     |
| LwF    | 84.4  | 68.65 | 55.6  | 47.32 | 41.4  | 38.54 | 33.23 | 29.55 | 27.12 | 25.5   | 45.15         |
| iCaRL  | 83.4  | 78.6  | 69.79 | 65.3  | 63.84 | 60.97 | 58.06 | 55.2  | 52.64 | 51.52  | 59.94         |
| BiC    | 83    | 78    | 67.07 | 63.4  | 61.5  | 60.8  | 57    | 54.1  | 50.42 | 49.3   | 59.84         |
| PODnet | 84.2  | 77    | 72.1  | 63.5  | 57.4  | 53.6  | 48.8  | 47.6  | 44.7  | 40.5   | 58.94         |

TABLE 3. The accuracy of multi-class on CIFAR-100 with 5 classes each task

| Name   | Task1 | Task2 | Task3 | Task4 | Task5 | Task6 | Task7 | Task8 | Task9 | Task10 | Avg. Accuracy |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|---------------|
| Ours   | 88.2  | 80.1  | 72.66 | 67.3  | 66.96 | 63.03 | 61.29 | 58.2  | 56.29 | 55.82  | **68.18**     |
| LwF    | 85    | 68.2  | 59.3  | 51    | 44.3  | 39    | 34.2  | 30    | 27.2  | 25.8   | 32.44         |
| iCaRL  | 83.4  | 78.6  | 69.79 | 65.3  | 63.84 | 60.97 | 58.06 | 55.2  | 52.64 | 51.52  | 34.44         |
| BiC    | 83    | 78    | 67.07 | 63.4  | 61.5  | 60.8  | 57    | 54.1  | 50.42 | 49.3   | 34.44         |
| PODnet | 84.8  | 82.8  | 76.9  | 72.5  | 69.5  | 67.7  | 62.9  | 60.5  | 57.9  | 52.5   | 34.44         |

TABLE 4. The performance of different single-layer and multi-layer architectures

| Number of classes | Single-Layer | Multi-Layers |
|-------------------|--------------|--------------|
|                   | Layer1 | Layer2 | Layer3 | Layer1 | Layer2 | Layer3 | Layer1 | Layer2 | Layer3 | Layer1 | Layer3 |
| 10                | 86.7   | 88.2   | 88     | 88.1   | 88.3   | 88.2   | 87.4   |
| 20                | 74.9   | 75.35  | 77.9   | 76.85  | 75.35  | 76.4   | 76.45  |
| 30                | 67.7   | 69.73  | 71.1   | 69.66  | 70.4   | 70.83  | 70.33  |
| 40                | 62.77  | 65.07  | 66.52  | 64.47  | 65.47  | 65.87  | 64.72  |
| 50                | 60.44  | 62.32  | 63.68  | 62.46  | 61.94  | 63.2   | 61.96  |
| 60                | 57.83  | 59.88  | 60.3   | 59.86  | 58.96  | 60.1   | 59.2   |
| 70                | 56.14  | 56.97  | 58.31  | 58.11  | 56.58  | 57.67  | 56.8   |
| 80                | 52.75  | 54.48  | 55.47  | 55.11  | 54.4   | 55.48  | 54.52  |
| 90                | 51.43  | 53.21  | 54.41  | 54.01  | 53.15  | 54.33  | 52.87  |
| 100               | 49.22  | 50.63  | 52.45  | 52.17  | 50.88  | 51.87  | 51.66  |

Average Accuracy: 61.98 63.58 64.81 64.08 63.54 64.39 63.59

Ultimately, the average accuracy of PODNet is worse than that of our method on the split into 5, 10, and 20 classes. It is thus evident that the distillation method proposed in this paper can effectively transfer knowledge from a pre-updated model to the current model using different scales of features. Because LwF doesn’t use a few samples from the old tasks for fine-tuning, its results are inferior to those of other methods.

2) FEATURE-BASED KNOWLEDGE DISTILLATION WITH MULTIPLE-LAYERS

Answer to RQ3: In this section, we analyze the impact of different branch structures on performance. In Table 4, we can see the performance of different structures of the branch network. We can obtain the best performance by distilling knowledge from a single 3rd layer. For single-layer distillation, the 3rd layers can provide more meaningful features than others. Compared to other features from shallow layers, there is much less noise while some of the spatial information is retained. Therefore, this architecture performs better than...
other single-layer structures. For the multi-layer structure, we can get the best result by distilling the knowledge from the features obtained from the 2\textsuperscript{nd} and 3\textsuperscript{rd} layers. This result confirms that we must strike a balance between rigidity and plasticity. If a model distills from so many layers, it may become excessively rigid and lose the ability to learn new tasks well. In addition, the complexity of the branch network would increase, making it difficult to train under the definition of lifelong learning.

3) THE PERFORMANCE OF OUR PROPOSED MODEL WITH A BIAS LAYER

Answer to RQ4: Beyond the difference in branch network structure, we use different classifiers, such as BiC [14], for our experiments. Algorithm 2 shows the training process, which adopts the output layer with a bias layer. Unlike Algorithm 1, this process is divided into two parts. The first part freezes the bias layer and trains the backbone network. The next part trains the bias layer to adjust the model preference problem under the condition of the untrainable backbone network. Before training the bias layer, we sample the data belonging to the new task, so the number of the training data in each class is the same. Next, we use this balanced data to train our model. Therefore, the bias layer adjusts the preference of the backbone model through its simple parameters. The experimental result, which splits the dataset into 5 tasks, is shown in Figure 5. We can see that the performance of the network improves by about 1 \sim 2\% at any time point after adding our method. The results show that our feature-based knowledge distillation method is applicable to different lifelong learning strategies and can improve their performance.

C. CONCLUSIONS

We proposed a novel distillation method to deal with the catastrophic forgetting problem in the field of lifelong learning. We add a special branch network, which integrates average pooling and a dense layer, to the intermediate of our backbone network. In this way, the model can learn more knowledge from previous tasks without old task data. Moreover, the distribution of the output from the branch network can effectively compress feature information from intermediate layers. By leveraging this compression method, we can greatly reduce the complexity of feature maps and enable the model to easily learn knowledge from previous tasks with the prediction. We compare our methods to other lifelong learning methods and demonstrate that this knowledge distillation model of the branch network can increase performance by approximately 2\%. Moreover, this branch structure can be easily applied to different kinds of networks. The network can be easily modified as well to become more powerful in the field of lifelong learning.

REFERENCES

[1] G. I. Parisi, “Continual lifelong learning with neural networks: A review,” in Neural Networks, vol. 113, pp. 54-71, 2019.
[2] A. Awasthi, “Continual Learning with Neural Networks: A Review,” in CoDS-COMAD ’19: Proceedings of the ACM India Joint International Conference on Data Science and Management of Data, pp. 362-365, 2019.
[3] H. Nielsen, “Theory of the backpropagation neural network,” in International 1989 Joint Conference on Neural Networks vol. 1, pp. 593-605, doi: 10.1109/IJCNN.1989.118638.
[4] D. Rumelhart, “Learning representations by back-propagating errors,” in Nature vol. 323, pp. 533-536, 1986.
[5] P. J. Werbos, “Backpropagation through time: what it does and how to do it,” in IEEE vol. 78, no. 10, pp. 1550-1560, Oct. 1990, doi: 10.1109/5.58337.
[6] H. He and E. A. Garcia, “Learning from Imbalanced Data,” in IEEE Transactions on Knowledge and Data Engineering vol. 21, no. 9, pp. 1263-1284, Sept. 2009, doi: 10.1109/TKDE.2008.239.

[7] Y. Sun, “CLASSIFICATION OF IMBALANCED DATA: A REVIEW,” in International Journal of Pattern Recognition and Artificial Intelligence vol. 23, no. 4, pp. 687-719, 2009.

[8] G. Haixiang, “Learning from class-imbalanced data: Review of methods and applications,” in Expert Systems with Applications vol. 73, pp. 220-239, 2017.

[9] Sylvestre-Alvise Rebuffi, “iCaRL: Incremental Classifier and Representation Learning,” in IEEE Conference on Computer Vision and Pattern Recognition pp. 2001-2010, 2017.

[10] J. Kirkpatrick, “Overcoming catastrophic forgetting in neural networks,” in PNAS vol. 114, no. 13, pp. 3521-3526, 2017.

[11] A. A. Rusu, “Progressive Neural Networks,” in ArXiv vol. abs/1606.04671, 2016.

[12] S. Ebrahimi, “Adversarial Continual Learning,” in Vedaldi A., Bischof H., Brox T., Framh JM. (eds) Computer Vision – ECCV 2020. ECCV 2020. Lecture Notes in Computer Science vol. 12356, 2016.

[13] Z. Li and D. Hoiem, “Learning without Forgetting,” in IEEE Transactions on Pattern Analysis and Machine Intelligence vol. 40, no. 12, pp. 2935-2947, 1 Dec. 2018, doi: 10.1109/TPAMI.2017.2773081.

[14] Sylvestre-Alvise Rebuffi, “Large Scale Incremental Learning,” in IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 374-382, 2019.

[15] A. Douillard, “PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning,” in Vedaldi A., Bischof H., Brox T., Framh JM. (eds) Computer Vision – ECCV 2020. ECCV 2020. Lecture Notes in Computer Science vol 12365, pp. 86-102, 2020.

[16] G. Hinton, “Distilling the Knowledge in a Neural Network,” in NIPS, 2014.

[17] J. Kim, “Paraphrasing Complex Network: Network Compression via Factor Transfer,” in NIPS’18: Proceedings of the 32nd International Conference on Neural Information Processing Systems pp. 2765-2774, 2018.

[18] S. Park, “Feature-level Ensemble Knowledge Distillation for Aggregating Knowledge from Multiple Networks,” in ECAI, 2020.

[19] S. Farquhar and Y. Gal, “Towards robust evaluations of continual learning,” arXiv preprint arXiv:1805.09733, 2018.

[20] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Diaz-Rodriguez, “Continual learning for robotics,” arXiv preprint arXiv:1907.00182, 2019.

[21] B. Puth and A. Gepperth, “A comprehensive, application-oriented study of catastrophic forgetting in dnn,” arXiv preprint arXiv:1905.08101, 2019.

[22] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” 2009.

[23] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 770-778, doi: 10.1109/CVPR.2016.90, 2016.

[24] YEN-HSIANG WANG received the B.S degrees from National University of Kaohsiung (NUK), Kaohsiung City, Taiwan and M.S degree from the Department of computer science and information engineering from National Central University (NCU), Taoyuan City, Taiwan. His research interests include computer vision, image processing, lifelong learning and deep learning.

CHIH-YANG LIN (Senior Member, IEEE) is the chief of the Global Affairs Office and a professor with the Department of Electrical Engineering, Yuan-Ze University, Taoyuan, Taiwan. He has authored or coauthored over 150 papers in international conferences and journals with high citations, and received Best Paper Awards from Pacific-Rim Conference on Multimedia (PCM) in 2008, Best Paper Awards and Excellent Paper Award from Computer Vision, Graphics and Image Processing Conference in 2009, 2013, and 2019, and Best Paper Award from the 6th International Visual Informatics Conference 2019 (IVIC’19). He has served as a session chair, publication chair, or workshop organizer on many international conferences, including AHFE, ICCE, ACCV, IEEE Multimedia Big Data, ACM IHMM,Sec, APSIPA, and CVGIP. He is also a regular Reviewer of the IEEE Transactions on Image Processing, the IEEE Transactions on Circuits and Systems for Video Technology, the IEEE Transactions on Multimedia, IEEE Access, and many other prestigious Elsevier journals. His research fields include computer vision, machine learning, deep learning, image processing, big data analysis, and the design of surveillance systems.

TIPAJIN THAPI/SUTIKUL received the master’s degree (Hons.) in the research path from The University of Sydney (USYD), Sydney, NSW, Australia in 2012 and received the Ph.D. degree from the Department of Computer Science and Information Engineering, National Central University, Chung-Li, Taiwan in 2021. She is currently an instructor with the Faculty of Information and Communication Technology (ICT), Mahidol University, Salaya, Thailand. Her research mainly focuses on machine learning, applied intelligence, data mining, and social network analysis.

TIMOTHY K. SHIH (Senior Member, IEEE) currently a Distinguished Professor and the Vice Dean of College of EECS with National Central University, Taiwan. He is also the Director of the Innovative AI Research Center. His current research interests include multimedia computing, computer-human-interaction, and distance learning. He is a Fellow of the Institution of Engineering and Technology (IET). In addition, he is a Senior Member of ACM. He is currently the Associate Editor of the IEEE TRANSACTIONS ON LEARNING TECHNOLOGIES, ACM Transactions on Internet Technology, and the IEEE TRANSACTIONS ON MULTIMEDIA.

YEN-HSIANG WANG