Learning Adaptive Embedding Considering Incremental Class

Yang Yang, Zhen-Qiang Sun, Hengshu Zhu, *Senior Member, IEEE*, Yanjie Fu, Yuanchun Zhou, Hui Xiong, *Fellow, IEEE*, and Jian Yang, *Member, IEEE*

Abstract—Class-Incremental Learning (CIL) aims to train a reliable model with the streaming data, which emerges unknown classes sequentially. Different from traditional closed set learning, CIL has two main challenges: (1) Novel class detection. The initial training data only contains incomplete classes, and streaming test data will accept unknown classes. Therefore, the model needs to not only accurately classify known classes, but also effectively detect unknown classes; (2) Model expansion. After the novel classes are detected, the model needs to be updated without re-training using the entire previous data. However, traditional CIL methods have not fully considered these two challenges. First, they are always restricted to single novel class detection within each phase and embedding confusion caused by unknown classes. Besides, they ignore the catastrophic forgetting of known categories in model update. To this end, we propose a semi-supervised style Class-Incremental Learning without Forgetting (CILF) method, which aims to learn adaptive embedding for processing novel class detection and model update in a unified framework. In detail, CILF designs to regularize classification with decoupled prototype based loss, which can improve the intra-class and inter-class structure significantly, and acquires a compact embedding representation for novel class detection in result. Then, CILF employs a learnable curriculum clustering operator to estimate the number of semantic clusters via fine-tuning the learned network, in which curriculum operator can adaptively learn the embedding in self-taught form. Therefore, CILF can detect multiple novel classes and mitigate the embedding confusion problem. Last, with the labeled streaming test data, CILF can update the network with robust regularization to mitigate the catastrophic forgetting. Consequently, CILF is able to iteratively perform novel class detection and model update. We verify the effectiveness of our model on four streaming classification tasks, and empirical studies show the superior performance of the proposed method.

Index Terms—Class-incremental learning, Novel class detection, incremental model update

1 Introduction

Most traditional machine learning methods assume that the training and testing data share the same label space, and various methods have achieved significant success in different applications [1], [2], [3], [4], [5]. However, many applications are non-stationary considering that the real-world is dynamically changing, which causes unknown classes emerge sequentially when processing the streaming data. For example, driverless cars need to identify unknown objects in the received streaming images, face recognition needs to distinguish unseen personal pictures in the received monitoring image and so on. This is defined as *Class-Incremental Learning (CIL)* in literature. In detail, as shown in Fig. 1, CIL includes two pivotal tasks: novel class detection (NCD) and incremental model update (IMU). The main difficulty of NCD is to effectively distinguish known and unknown classes as shown in Fig. 1a. Meanwhile, the IMU aims to update the model with newly labeled instances from unknown classes after novel class detection as shown in Fig. 1b, in which the main difficulty is the forgetting of known classes when updating model. CIL needs to conduct the NCD and IMU iteratively with the streaming data.

To address the NCD issue, zero-shot learning (ZSL) is first proposed [6], [7], which aims to classify instances from unknown categories by merely utilizing seen class examples and semantic information about unknown classes. However, the basic assumption behind the standard ZSL methods is that test data only contain unknown classes. Thereby, generalized zero-shot learning (GZSL) [8], [9], [10] is proposed, in which the test data include known and unknown classes simultaneously. Nevertheless, both ZSL and GZSL assume that semantic information (for example, attributes or descriptions) of the unknown classes is given during training phase, which is limited to detect with prior knowledge. To solve this problem,

- *Yang Yang and Jian Yang are with the Nanjing University of Science and Technology, Nanjing, Jiangsu 210094, China, and also with the PCA Lab, Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, and Jiangsu Key Lab of Image and Video Understanding for Social Security, School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, Jiangsu 210094, China. E-mail: {yyang, csjyang}@njust.edu.cn.*
- *Zhen-Qiang Sun is with the Nanjing Normal University, Nanjing, Jiangsu 210023, China. E-mail: enderman19980125@outlook.com.*
- *Hengshu Zhu is with the Baidu Talent Intelligence Center, Baidu Inc, Beijing 100000, China. E-mail: zhuhengshu@baidu.com.*
- *Yanjie Fu is with the Missouri University of Science and Technology, Rolla, MO 65401 USA. E-mail: fuyan@mst.edu.*
- *Yuanchun Zhou is with the Computer Network and Information Center, Chinese Academy of Sciences, Beijing 100864, China. E-mail: zyc@cnic.cn.*
- *Hui Xiong is with the Artificial Intelligence Thrust, The Hong Kong University of Science and Technology, Guangzhou, China. E-mail: hxiong@rutgers.edu.*

Manuscript received 8 September 2020; revised 16 June 2021; accepted 24 August 2021. Date of publication 2 September 2021; date of current version 3 February 2023.
This research was supported in part by NSFC under Grants 62006118, 61836013, and 91746301, in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20200460, in part by CCF-Baidu Open Fund under Grant CCF-BAIDU OF2020011, in part by Baidu Fund under Grant 20121202OT03645, and in part by CAAI-Huawei MindSpore Open Fund under Grant CAAIXSJLJJ2021-014B.
(Corresponding author: Yang Yang.)
Recommended for acceptance by Q. He.
Digital Object Identifier no. 10.1109/TKDE.2021.3109131

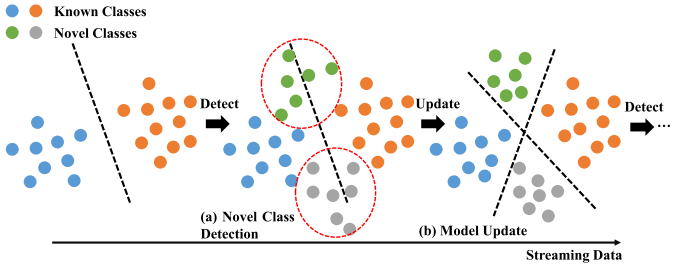


Fig. 1. (Best view in color) Schematic of class-incremental learning. Unknown categories occur with the streaming data, (a) the model first detects novel class with pre-trained model; (b) the model is then updated with newly labeled instances from unknown classes, without or with limited examples from known classes.

recent NCD approaches attempt to detect unknown classes by designing robust models, which can mainly be divided into two aspects: discriminative and generative models. Discriminative models mainly utilize the powerful feature learning and prediction capabilities of deep models to design corresponding distance or prediction confidence measures [11], [12]. On the other hand, Generative models mainly employ the adversarial learning to generate instances that can fool the discriminative model, thereby to detect the novel classes [13]. However, as shown in Fig. 2, it is notable that known and unknown classes have strong separability on the trained model in simple visual datasets such as MNIST (i.e., Figs. 2a and 2c), thereby the discriminative and generative models are effective, yet in more complex visual datasets such as CIFAR-10 (i.e., Figs. 2b and 2d), unknown and known categories have embedding confusion in feature space, i.e, instances of unknown and known classes are mixed, which will greatly weaken the performance of discriminative and generative methods. Besides, most existing detection methods are limited to detect single novel class, which is contrary to real applications.

Furthermore, we need incremental model update (IMU) with the newly labeled instances of novel classes after detecting. Different from re-training with all previous known data, IMU aims to re-train the model referring none or only limited known data, which can ensure the efficiency of incremental update. Therefore, a big challenge for IMU is the catastrophic forgetting phenomenon [14], i.e., it is obvious that the knowledge learned from the previous task (known classes classification) will be lost when information relevant to the current task (novel class classification) is incorporated. To mitigate the catastrophic forgetting, there are many attempts, including replay-based methods that explicitly re-train on stored examples while training on new tasks [15], [16], and regularization-based methods that utilize extra regularization term on output or parameters to consolidate previous knowledge [17], [18], [19].

To consider the NCD and IMU tasks simultaneously, we propose a Class-Incremental Learning without Forgetting (CILF) method, which aims to process novel class detection and model update iteratively. In detail, CILF initially develops a novel decoupled prototype based network to train the known classes with supervised data, which employs the constrained clustering loss to regularize the inter-class and intra-class structure. In testing, considering emergence of unsupervised single or multiple novel classes, we develop the curriculum operator for learning adaptive embedding, which aims to conduct learnable clustering to overcome the embedding confusion problem. Then, with the limited memory data of supervised known classes and newly labeled data of unknown classes, CILF updates the network with robust regularization to mitigate the catastrophic forgetting. In summary, the main contributions are summarized as follows:

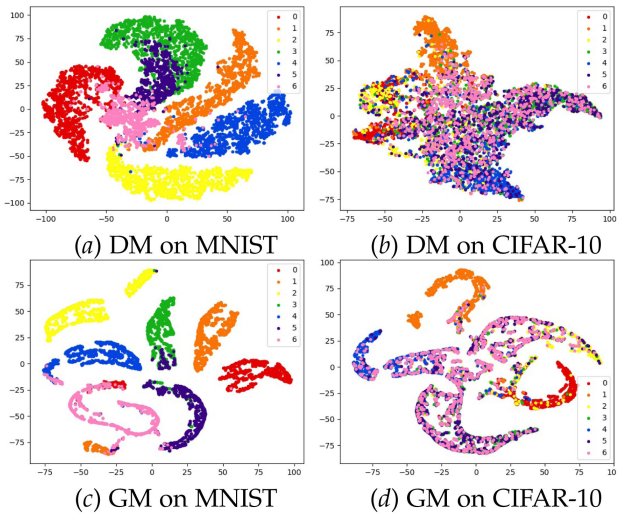


Fig. 2. (Best view in color.) T-SNE of discriminative model (DM) [18] and generative model (GM) [17] on simple (MNIST) and complex (CIFAR-10) datasets. $0-6$ in legend denotes different classes in two datasets. In detail, we train the two models with five classes (i.e., 0-4) in the training stage, then utilize the pre-trained model to achieve the feature embeddings of data from two unknown classes (i.e., 5, 6) and all known classes (i.e., 0-4) appearing in the testing stage, and give the T-SNE in (a)-(d).

- Propose the " Class-Incremental Learning without Forgetting" (CILF) framework, which considers both NCD and IMU tasks;
- Propose a novel decoupled prototype based network, which can conduct novel class detection and model update effectively;
- Propose the curriculum clustering operator for better multiple novel classes detection and robust regularization to mitigate catastrophic forgetting.

2 Related Work

Our work aims to detect novel classes in streaming data, and update the model with limited known data without forgetting. Therefore, our work is related to: novel class detection and incremental model update.

Traditional novel class detection approaches mainly restrict intra-class and inter-class distance property in training data, then detect novel class by identifying outliers. For example, Da *et al.* developed a SVM-based method, which learned the concept of known classes while incorporating the structure presented in the unlabeled data collected from open set [20]; Mu *et al.* proposed to dynamically maintain two low-dimensional matrix sketches for detecting novel classes [21]. However, these approaches are difficult to process high dimensional space considering complex matrix operations. Recently, with the development of deep learning techniques, several studies have applied convolutional neural network (CNN) on the detection scenario. Hendrycks and Gimpel verified that CNN trained on the MNIST images can predict high confidence (90%) on gaussian noise instances, thus the

softmax output probabilities can be used to distinguish known/unknown class [22]. Furthermore, Liang *et al*. directly utilized temperature scaling or added small perturbations to separate the softmax score distributions between in- and out-of-distribution images [23]; Neal *et al.* introduced a novel augmentation technique, which adopted an encoder-decoder GAN architecture to generate the synthetic instances similar to known class [17]; Wang *et al.* proposed a CNN-based prototype ensemble method, which adaptively update the prototype for robust detection [18]. Nevertheless, these methods are always limited to detect single novel class at one time. Therefore, Han *et al.* proposed an extended deep transfer embedded clustering method for multiple novel class detection [24]. However, existing NCD methods usually have superior detection performance on simple datasets, but are easily interfered by embedding confusion on complex datasets.

Incremental learning is always applied for streaming data. In most situations, only a few examples from known classes/features/distributions are available in the beginning, and data with new classes/features/distributions emerge thereafter. Incremental learning methods aim to update the models from streaming data sequentially only with newly coming data and limited previous data, without re-training with all previous data [25]. As a matter of fact, incremental deep learning can directly apply with online backpropagation, yet with one important drawback: catastrophic forgetting, which is the tendency for losing the learned knowledge of previous distribution (previously known classes/features/distributions). To solve this problem, there are many attempts, for example, Rebuffi *et al.* stored a subset of examples per class, which are selected to best approximate the mean of each class in the feature space [15], Lopez-Paz and Ranzato projected the estimated gradient direction on the feasible region outlined by previous tasks through a first order Taylor series approximation [26]; Li and Hoiem utilized the output of previous model as soft labels for previous tasks [27]; Kirkpatrick *et al.* proposed the elastic weight consolidation to reduce catastrophic forgetting [28]; Lee *et al.* proposed to incrementally match the moment of posterior distribution of the neural network [29].

3 Proposed Method

In this section, we formalize the problem of class-incremental learning with streaming data, and give the details of proposed framework.

3.1 Problem Definition

Without any loss of generality, at initial time, we have a supervised training set $D^0 = \{(\mathbf{x}_i^0, \mathbf{y}_i^0)\}_{i=1}^{N}$, where $\mathbf{x}_i^0 \in \mathbb{R}^d$ denotes the $i-$th instance, and $\mathbf{y}_i^0 \in Y^0 = \{1, 2, \ldots, C\}$ denotes the corresponding label, 0 represents initial time. Then, we receive a non-stationary unlabeled testing data $D^1 = \{(\mathbf{x}_j)\}_{j=1}^{N_1}$, where $\mathbf{x}_j \in \mathbb{R}^d$ denotes the $j-$th instance, and label $\mathbf{y}_j \in \hat{Y} = \{1, 2, \ldots, C, C+1, \ldots, C+K^1\}$ is unknown, K^1 is the number of unknown classes. Thus, novel class detection can be defined as:

Definition 1 Novel Class Detection (NCD). *With the initial training set D^0, we aim to construct a model, i.e., $f^0 : X^0 \rightarrow Y^0$. Then NCD aims to classify the known and unknown classes in D^1 accurately with the pre-trained model f^0.*

On the other hand, it is notable that streaming data with novel classes has two characteristics: (1) Data window. At time window t, we only get the data of current time window, not the full amount of streaming data for detection; and (2) Novel class continuity. At time window t, an indeterminate number of novel classes will appear, or even no novel class. Therefore, we need to incrementally detect novel classes, i.e., with the streaming data, every time after receiving the data of time window t, NCD is performed [30]. Specifically, the streaming test data D can be denoted as $D = \{D^t\}_{t=1}^{T}$, where $D^t = \{\mathbf{x}_j^t\}_{j=1}^{N_t}$ has N_t unlabeled instances, and the underlying label $\mathbf{y}_j^t \in \hat{Y}^t$ is unknown, $\hat{Y}^t = \hat{Y}^{t-1} \cup Y^t$, where $\hat{Y}^{t-1}$ is the cumulative known classes until $(t-1)-$th time window and Y^t is the new classes in $t-$th window. $\hat{Y}^T = \hat{Y} = \{1, 2, \ldots, C, C+1, \ldots, C+K\}$. Therefore, we can give the definition of class-incremental learning:

Definition 2 Class-Incremental Learning (CIL). *At time $t \in \{1, 2, \ldots, T\}$, we have pre-trained model f^{t-1}, finite stored instance set M^{t-1} from known classes and received streaming data D^t. First, CIL aims to classify known and unknown classes in D^t as Definition 1. Then, CIL updates the model while mitigating forgetting to acquire f^t with the newly labeled data from novel classes and stored data M^{t-1}. Cycle this process until terminated.*

Note that there exist two labeling cases after novel class detection, i.e., manual labeling and self-taught labeling [21]. Actually, manual labeling reduces the generation of label noise compared to self-taught labeling, so it is more conducive to the incremental update of the model. However, considering the cost and time overhead of manual labeling, labeling a large number of instances will affect the efficiency of the model. Considering the fairness of comparison methods, we adopt the setting of manual labeling following most approaches [18], [21], [30]. In addition, in view of the problems of manual labeling, techniques such as random sampling can be used to reduce labor and time costs. In the experiment Section 4.6, we also verify the performance of the model under different number of instance queries.

3.2 CILF Framework

The main idea of CILF is to learn the feature embeddings such that instances exhibit distinguishing characteristics for label prediction, novel class detection, and subsequent model update over the non-stationary streaming data. Therefore, the most critical parts of CILF are: (1) feature embedding network, (2) novel class detection operator, and (3) model update mechanism.

- *Feature Embedding Network:* With the initial training data, i.e., the blue and orange dots shown in Fig. 3, the decoupled neural network model is trained using the labeled initial data with the prototype based loss, which concerns the intra-class/inter-class structure and can be easily transformed for NCD;
- *Novel Class Detection Operator:* At time t, we receive a set of unlabeled examples from the streaming data source, i.e., the gray dots shown in Fig. 3, which includes known (blue and orange dots) and unknown (green dots) classes. The observed instances set D^t are transformed through learned network, and achieve

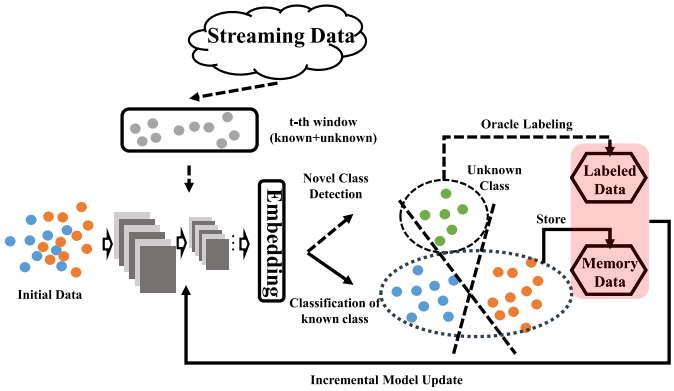


Fig. 3. Overview of the CILF framework. The blue and orange dots are initial training data for developing the deep network. While the gray dots are unlabeled testing data of $t-$th time window, which are received from the streaming data. With the trained deep network, CILF aims to classify the known and novel classes, then query the ground-truths of novel class instances for updating the network continuously.

feature representations. Then we employ the pre-trained f^{t-1} for curriculum clustering, which can detect multiple unknown classes from easy to difficult in self-taught form;

- *Model Update Mechanism:* After NCD, true labels of instances from novel classes are queried (partially or fully), then IMU is performed on f^{t-1} with the newly labeled data, while regularizing the performance of stored data from known classes to mitigate forgetting. The updated model is then used to further classify incoming instances along the stream.

This process is repeated until the end of streaming data. Fig. 3 illustrates the overall streaming data classification process performed by CILF framework. And Table 1 provides the definition of symbols used in this paper.

3.3 Feature Embedding Network

Given the initial training data D^0, our primary objective is to build an effective model f^0 for subsequent classification. Recent researches have demonstrated the effectiveness of deep model on feature embedding and subsequent tasks, thereby we employ the deep models for building f^0, for example, convolution neural networks for images. Importantly, the built deep model needs to consider two aspects [31]: (1) Distance measure. The model needs to emphasize the exploitation of feature embeddings considering intra-class compactness and inter-class separability, thus leaves larger space for novel class detection; (2) Model scalability. The model needs to effectively learn novel class knowledge and incrementally update model with the emergent novel class data. Nevertheless, traditional cross entropy based deep models rarely consider the distance measurement, and are difficult to conduct the model update considering that the prediction layer is coupled to the fully connected layer [18]. Consequently, we develop a decoupled deep embedding network with prototype based loss to improve the inter-class and intra-class structure.

Particularly, for a given input $\mathbf{x}_i^0$, the output feature representations are denoted as $f^0(\mathbf{x}_i^0, \theta)$, θ is the correspond network parameters, and we utilize notation $f^0(\mathbf{x}_i^0)$ for brevity. Inspired from the topic of metric learning [32], the loss can be defined as:

TABLE 1
Description of Symbols

Sym.	Definition
$D^0 = \{(\mathbf{x}_i^0, \mathbf{y}_i^0)\}_{i=1}^N$	initial supervised training data
$D^t = \{\mathbf{x}_j^t\}_{j=1}^{N_t}$	set of unlabeled data at $t-$th time window
D_{new}^t	set of labeled new class data at time t
$\hat{Y}^t$	label set at $t-$th time window
f^t	trained model $t-$th time window
Y^t	the new class set at $t-$th time window
K^t	the size of new class set at $t-$th time window
M^t	stored memory data of known classes until $t-$th time window
$f^t(\mathbf{x})$	feature embedding of instance $\mathbf{x}$
p_i^t	prediction of $i-$th instance at time t
μ_c^t	prototype of $c-$th class at time t
w_j^t	weight of each instance at time t
$g(l)$	pacing function to determine the number of selected instances in each mini-batch

$$L = L_{intra} + \lambda L_{inter}, \tag{1}$$

where L_{intra} aims to pull data towards their same-class neighbors, and L_{inter} aims to push data from different classes away with each other. λ is the hyperparameter.

3.3.1 Intra-Class Compactness

L_{intra} can be obtained by calculating the distance between each instance and corresponding prototype, here we utilize the class center. Similar to cross entropy loss [33], i.e., $\sum_{i=1}^N -\mathbf{y}_i log(g(f(\mathbf{x}_i, \theta)))$, where $\mathbf{y}_i$ is the ground truth of $i-$th instance, $g(\cdot)$ denotes the fully connected layer with softmax function. L_{intra} is developed on the feature output $f^0(\mathbf{x}^0)$. Consequently, we define the prototype-based cross entropy loss as following:

$$L_{intra} = \sum_{i=1}^{N}\sum_{c=1}^{C} -y_{i_c}^0 log(p_{i_c}^0), \tag{2}$$

where $p_{i_c}^0$ is the probability of $\mathbf{x}_i^0$ being classified as y_c^0, which is negatively related to the distance between instance and prototype of $c-$th class, i.e., the probability is larger if the distance is closer, vice versa. Therefore, $p_{i_c}^0 \propto -\|\mathbf{x}_i^0 - \mu_c^0\|_2^2$, where μ_c^0 is the representations of $c-$th class prototype, can be defined as following:

$$p_{i_c}^0 = \frac{exp(-\alpha\|f^0(\mathbf{x}_i^0) - \mu_c^0\|_2^2)}{\sum_{m=1}^{C} exp(-\alpha\|f^0(\mathbf{x}_i^0) - \mu_m^0\|_2^2)}, \tag{3}$$

where C is the class number, and α is a hyperparameter that controls the strength of distance similar to large margin cross-entropy [34]. Note that Eq. (3) minimizes loss via maximizing the probability of $\mathbf{x}_i^0$ being associated with the prototype $\mu_{y_i}^0$. Moreover, it is crucial to initialize and update each class prototype effectively. The labels of initial training data $\mathcal{D}^0$ are provided, thereby we use the output representation $f^0(\mathbf{x}^0)$, for each prototype initialization:

$$\mu_c^0 = \frac{1}{|\pi_c|}\sum_{\mathbf{x}^0 \in \pi_c} f^0(\mathbf{x}^0),$$

where $|\pi_c|$ is the size of $c-$th class. On the other hand, the key idea of prototype update is to anneal clusters slowly, so that eliminates the biased instances in each mini-batch. Thus we propose to smoothen the annealing process via temporal ensemble [35]:

$$\mu_c^{0_e} = \beta\mu_c^{0_{e-1}} + (1-\beta)\mu_c^{0_e} \tag{4}$$

where β is a momentum term controlling the ensemble, and 0_e indicates the $e-$th epoch for the initial training.

3.3.2 Inter-Class Separability

The prototype-based cross entropy loss guarantees the local intra-class compactness, while neglects the inter-class separability. To make the projection of instances robust in distance measure, L_{inter} focuses on improving global separation between different classes. Particularly, L_{inter} aims to transform instances from the same classes to be closer than those from different classes, i.e., $d(f^0(\mathbf{x}_i^0), f^0(\mathbf{x}_p^0)) < d(f^0(\mathbf{x}_i^0), f^0(\mathbf{x}_n^0))$, where $\mathbf{x}_i^0, \mathbf{x}_p^0$ share the same class and $\mathbf{x}_n^0$ is from a different class, $d(f^0(\mathbf{x}_i^0), f^0(\mathbf{x}_j^0))$ is a metric function measuring distance in the embedding space, and we use notation $d_{i,j}$ for clarity. The distance is known as the triplet loss with a pre-specified margin value m, i.e., $\left[m + d_{i,p} - d_{i,n}\right]_+ = \max\{0, m + d_{i,p} - d_{i,n}\}$. It is notable that triplet loss always suffers from slow convergence, thus triplet construction is central for improving the performance [36], [37]. Inspired by [38], [39], we consider the hard triplet to fully explore multiple negative examples from different classes in each mini-batch, which can further improve the inter-class distances. In result, hard triplet of $i-$th instance is denoted as:

$$\mathbf{\Omega}_i = (\mathbf{x}_i^0, \mathbf{x}_p^0, \mathbf{x}_{n_1}^0, \ldots, \mathbf{x}_{n_{C-1}}^0), \tag{5}$$

where negative examples $\mathbf{x}_{n_c}$ are randomly sampled from $C-1$ different classes, and positive example $\mathbf{x}_p$ is randomly sampled from the same category. Specifically, the relationship between hard triplet and triplet can be described as follows: each hard triplet $\mathbf{\Omega}_i$ contains $C-1$ triplets, sharing the same positive pair $(\mathbf{x}_i^0, \mathbf{x}_p^0)$. Thereby, the triplet constraint can be generalized to the hard triplet as follows: $\forall c = 1, 2, \ldots, C-1$, the $d(f^0(\mathbf{x}_i^0), f^0(\mathbf{x}_p^0)) < d(f^0(\mathbf{x}_i^0), f^0(\mathbf{x}_{n_c}^0))$, which can better consider the global inter-class distances. Thereby the L_{inter} can be defined as:

$$L_{inter} = \sum_{i=1}^{N}\left[m + d_{i,p} - \min_{\mathbf{x}_{n_c}^0 \in \mathbf{\Omega}_i} d_{i,n_c}\right]_+, \tag{6}$$

where $\mathbf{\Omega}_i$ denotes the hard triplet set of $i-$th instance. Here we utilize euclidean distance to evaluate the distance between two examples:

$$d_{i,j} = \|f^0(\mathbf{x}_i^0) - f^0(\mathbf{x}_j^0)\|_2^2. \tag{7}$$

Consequently, we can learn discriminative feature embedding, and boost the performance of classification and detection via optimizing Eq. (1) from two perspectives: (1) Prototype-based loss highlights the compactness of representation, i.e., the intra-class would be more compact and inter-class would be more distant. This property is suited for distinguishing the known and unknown classes. (2) Prototype-based loss is based on the feature output embedding, which is independent of the prediction layer. Therefore, it is easy to update the model and learn novel classes, without the expansion of model structure (prediction layer). The details are shown in Algorithm 1.

Algorithm 1. Feature Embedding Network

- **Input**:
- Data set: $D^0 = \{(\mathbf{x}_i^0, \mathbf{y}_i^0)\}_{i=1}^N$
- Parameter: λ, α, Learning rate parameter: η

- **Output**:
- Decoupled deep clustering network: f^0

Initialize model parameters θ;
Initialize the prototype μ for each class;
while stop condition is not triggered **do**
  for instance mini-batch **do**
    Calculate L_{intra} according to Eq. (2);
    Calculate L_{inter} according to Eq. (6);
    Calculate loss $L = L_{intra} + \lambda L_{inter}$ according to Eq. (1);
    Update model parameters using gradient descent;
  end for
  Update the prototype μ according to Eq. (4);
end while

3.4 Novel Class Detection

Traditional closed-set methods predict the known classes of training phase, in which the number of possible labels at testing is known and fixed. However, in class-incremental setting, instances belonging to unknown classes may appear with the streaming test data [30]. Therefore, we need to distinguish known and unknown classes. Specifically, we receive a set of unlabeled data D^t at $t-$th time, and there may occur K^t novel classes, where $K^t \geq 0$. However, most current detection methods either assume that only one novel class appears per time [21], [40], i.e., $K^t = 1$, or classify multiple novel classes into a super-class [18], which is impractical and difficult to operate efficiently. To solve this problem, we aim to fine-tune the deep clustering network f^{t-1} of last time for multiple novel class detection. As shown in Fig. 2, a key challenge is that adversarial instances of novel classes are mixed with known classes in a complex scenario, leading to embedding confusion and greatly affecting the clustering effect, i.e., biased prototypes for known and unknown classes. To solve this problem, we employ a learnable curriculum clustering operator, which aims to conduct clustering from easy (distinguishable) to difficult (confused) instances via curriculum learning [41], [42], [43]. Curriculum learning is related to self-paced learning [44], but relies on the ranking of training points by their difficulty with respect to target hypothesis rather than current hypothesis, and is more beneficial [45]. Consequently, we can acquire more reliable prototype and novel class detection results.

In detail, considering that the model training in Algorithm 1 is entirely supervised, whereas D^t is unsupervised, we aim to discover novel classes in D^t by unsupervised clustering, which fine-tunes the f^{t-1} trained during $(t-1)$th phase with easy instances first, and then cluster the mixed ones. We address this challenge by decomposing the learnable curriculum clustering into two closely related sub-tasks in

curriculum learning [45]: 1) *weighting function* to calculate the weight of each instance, and initialize the prototype with weighted k-means; and 2) *pacing function* to determine the pace for which data are presented to fine-tune the model, thus conduct curriculum clustering.

3.4.1 Weighting Function

Inspired by [45], we evaluate the weight of each instance by self-taught weighting function. In detail, we compute confidence score for each instance $\mathbf{x}_j^t$ in D^t using existing model f^{t-1}. We first obtain the statistic confidence by applying intra-class distance using Eq. (2), i.e., $u_j^t = \sum_{c=1}^{\hat{Y}^{t-1}} -y_{j_c}^{t-1} log(p_{j_c}^t)$. It is notable that u_j^t of the instances near prototype are smaller, and u_j^t of the instances away from all class prototypes are larger. Therefore, the weight of each instance can be denoted as $w_j^t = (u_j^t - \gamma)^2$, where γ is the threshold parameter. Thereby the highly confident instances of known and unknown classes have larger weights (i.e., the instances near prototype or away from all class prototypes), and confusing ones have lower weights.

On the other hand, Algorithm 1 requires initial setting for prototypes $\mu_c^t, c \in \hat{Y}^t$. Thus we initialize prototypes by running semi-supervised weighted k-means algorithm [46] combing the unlabeled set D^t and pre-trained μ_c^{t-1}. In result, we can obtain more robust initial prototypes:

$$\mu_c^t = \begin{cases} \beta\mu_c^{t-1} + (1-\beta)\sum_{\mathbf{x}_j^t \in \pi_c} \frac{w_j^t f^{t-1}(\mathbf{x}_j^t)}{\sum_{\mathbf{x}_j^t \in \pi_c} w_j^t}, & when \quad c \in \hat{Y}^{t-1}, \\ \sum_{\mathbf{x}_j^t \in \pi_c} \frac{w_j^t f^{t-1}(\mathbf{x}_j^t)}{\sum_{\mathbf{x}_j^t \in \pi_c} w_j^t}, & when \quad c \in Y^t, \end{cases} \tag{8}$$

where π_c denotes the data set predicted to be class c in D^t, in which the pseudo-label $\arg\max\{p_{j_c}^t\}$ of each instance in D^t can be calculated by Eq. (3). Note that the instances' labels of the unknown classes cannot be obtained in prior, so we randomly initialize the class prototypes of the unknown classes according to the set K as traditional k-means algorithm, and train for several epochs of preliminary update.

3.4.2 Pacing Function

A direct way for classifying known and unknown classes in D^t is to fine-tune f^{t-1} using all the instances. However, considering the embedding confusion, the initialized prototypes are biased because pseudo-labels exist noises. If we randomly sample batches from the full amount of data to fine-tune model, the embedding confusion will further affect the update of prototypes and pseudo-labels. Therefore, inspired by [45], we turn to sort the instances according to the difficulty, then present instances from easy to hard for fine-tuning according to the increase of the model capability.

In detail, the pacing function $h : \mathcal{B} \rightarrow D^t$ is used to determine a sequence of subsets $\mathcal{B} = \{B_1, B_2, \ldots, B_L\} \in D^t$, with size $|B_l| = h(B_l)$. L is the number of batches. The lth subset B_l includes the first $h(B_l)$ elements of the instances, which are sorted by the scoring function in ascending order. Here, we utilize the fixed exponential pacing, which has a fixed step length, and exponentially increasing size in each batch. Formally, it is given by:

$$h(B_l) = min(\upsilon \cdot \delta^{\lfloor \frac{l}{\phi} \rfloor}, 1) \cdot N_t. \tag{9}$$

Where υ denotes the fraction of data in the initial step, δ is the exponential factor for increasing the size of sampled mini-batches in each step, ϕ is the number of iterations in each step, $\lfloor \cdot \rfloor$ denotes round down, l is the index of batches, N_t is the number of instances. Consequently, in each mini-batch, we select episodic data with variable length for reliable fine-tuning.

3.4.3 Fine-tune Clustering

With the sampled mini-batches $\{B_1, B_2, \ldots, B_L\}$ in each epoch, we aim to fine-tune the f^{t-1} from easy to hard, and Eq. (2) can be reformulated as:

$$\begin{aligned} L^t &= L_{intra}^t + \lambda_1 L_{inter}^t + \lambda_2 R^t \\ L_{intra}^t &= \sum_{l=1}^{L} \sum_{j=1}^{|B_l|} \sum_{c=1}^{|\hat{Y}^t|} -\bar{y}_{l_{j_c}}^t log(p_{l_{j_c}}^t) \\ L_{inter}^t &= \sum_{l=1}^{L} \sum_{j=1}^{N_t} \left[m + d_{l_{j,p}} - \min_{\mathbf{x}_{n_c}^t \in \mathbf{\Omega}_j} d_{l_{j,n_c}} \right]_+ \\ R^t &= \sum_{c=1}^{|\hat{Y}^{t-1}|} \|\mu_c^t - \mu_c^{t-1}\|_2^2, \end{aligned} \tag{10}$$

where R^t aims to constraint the updated prototypes of known classes approaching the pre-trained ones, which can regularize the embeddings of known classes. The pseudo-labels $\bar{\mathbf{y}}_j^t = \arg\max\{p_{j_c}^t\}$ for each instance can be calculated by Eq. (3). So far, we assume that the number of classes K^t is known, which is impractical in real applications. Thus, we aim to estimate the number of classes in the unlabeled data. Specifically, we fine-tune clustering using D^t by varying the number of unknown classes. The resulting clusters are then examined by computing cluster validity index (CVI), which concerns the intra-cluster cohesion versus inter-cluster separation. And we select the generally used Silhouette index [47]:

$$CVI = \sum_{\mathbf{x} \in D^t} \frac{b(\mathbf{x}) - a(\mathbf{x})}{\max\{a(\mathbf{x}), b(\mathbf{x})\}}, \tag{11}$$

where $a(\mathbf{x})$ is the average distance between $\mathbf{x}$ and all other data instances within the same cluster, and $b(\mathbf{x})$ is the smallest average distance of $\mathbf{x}$ to all instances in any other different cluster. The optimal number of categories is the inflection point of CVI with maximum curvature. The details are shown in Algorithm 2.

3.5 Incremental Model Update

Ideally, the initial model training and novel class detection processes can identify the known and unknown classes. However, considering streaming data with unceasing novel classes, we need reliable training data of novel classes to create new prototypes and update the model parameters in incremental fashion. Thus, we need to collect novel class data for labeling, which can be used to re-train f^{t-1}. Similar to previous studies [21], [48], after curriculum clustering operator for detection, we can achieve potential novel class instances D_{new}^t for querying their true labels. Note that we

can query full or only partial data of the novel class. However, only using the new data to update the model will lead the catastrophic forgetting of known classes.

Algorithm 2. Novel Class Detection

- **Input**:
- Data set: $D^t = \{\mathbf{x}_j^t\}_{j=1}^{N_t}$
- Parameter: β, γ, υ, δ, ϕ

- **Output**:
- Novel Class Detection Network: $\hat{f}^t$

for $0 \leq K \leq K_{max}^t$ **do**
  Initialize prototypes μ_c^t according to Eq. (8);
  while stop condition is not triggered **do**
    Generate mini-batches $\{B_1, B_2, \ldots, B_L\}$ according to Eq. (9);
    for instance mini-batch B_l **do**
      Calculate L^t using Eq. (10) similar to Algorithm 1;
      Fine-tune model parameters using gradient descent;
    end for
    Update the prototype μ_c^t according to Eq. (4);
    Update the pseudo-labels $\bar{\mathbf{y}}$ according to Eq. 3;
  end while
  Computer CVI for D^t according to Eq. (11);
end for
Let $\hat{f}^t$ as the K^* with optimal CVI value.

To solve this problem, we develop a mechanism to incorporate the stored memory and novel class information incrementally, which can mitigate the forgetting of discriminatory characteristics about known classes. In detail, we utilize the exemplary data M^{t-1} for regularization in re-training:

$$\begin{aligned} L(D_{new}^t, M^{t-1}) &= \hat{L}_{intra}^t + \lambda_1 \hat{L}_{inter}^t + \lambda_2 R^t \\ \hat{L}_{intra}^t &= \sum_{\mathbf{x}_i \in D_{new}^t \cup M^{t-1}} -y_i^t log(p_i^t) - \\ &\quad \sum_{\mathbf{x}_i \in M^{t-1}} f^{t-1}(\mathbf{x}_i) \log f^t(\mathbf{x}_i) \\ \hat{L}_{inter}^t &= \sum_{j=1}^{N_t} \left[m + d_{l_{j,p}} - \min_{\mathbf{x}_{n_c}^t \in \Omega_i} d_{l_{j,n_c}} \right]_+ \\ R^t &= \sum_{c=1}^{|\hat{Y}^{t-1}|} \|\mu_c^t - \mu_c^{t-1}\|_2^2. \end{aligned} \tag{12}$$

The first term encourages the network to output the correct class indicator (classification loss) for all labeled examples, i.e., D_{new}^t and M^{t-1}. In detail, M^{t-1} represents the stored in-class examples, the second term of $\hat{L}_{intra}^t$ aims to reproduce the scores calculated in the previous step (distillation loss) for stored in-class examples, i.e., ensuring that the stored in-class examples in M^{t-1} can also output the consistent prediction distributions $f^{t-1}(\mathbf{x})$ predicted by historical network through the current network $f^t(\mathbf{x})$, thereby alleviating the forgetting problem of historical knowledge. After retraining, we need to update the M^{t-1} to store key points of novel classes. Therefore, we randomly remove $\frac{|Y^t||M^{t-1}|}{|\hat{Y}^{t-1}||\hat{Y}^t|}$ instances from each known class, and sample $\frac{|M^{t-1}|}{|\hat{Y}^t|}$ instances from each novel class. The details are shown in Algorithm 3.

It is notable that CILF only needs to conduct the NCD task, i.e., novel class detection, according to the Algorithm 2 under the setting that without streaming data, and does not require manual labeling and model update operations.

Algorithm 3. Class-Incremental Learning

- **Input**:
- Data set: memory data M^{t-1}, labeled novel class data D_{new}^t
- Learning rate parameter: η

- **Output**:
- Re-trained deep clustering Network: f^t

Calculate the $f^{t-1}(\mathbf{x}_j)$ of the examples from M^{t-1} and D_{new}^t;
while stop condition is not triggered **do**
  for instance mini-batch **do**
    Calculate $L(D_{new}^t, M^{t-1}, f^t)$ according to Eq. (12);
    Re-train model parameters using gradient descent;
  end for
  Update the prototype μ according to Eq. (4);
end while

4 Experiments

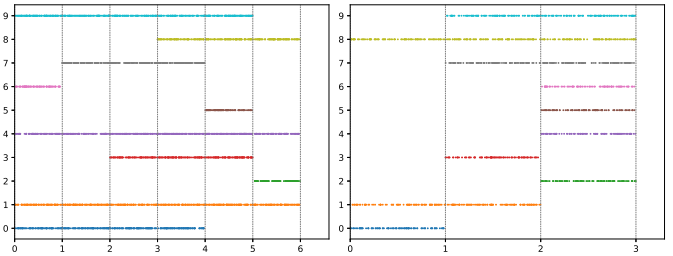
In this section, we mainly verify the proposed CILF from two aspects: (1) classification of known and novel classes; and (2) forgetting of known classes. Considering that most large-scale datasets are concentrated on images, we empirically evaluate CILF by comparing it with the state-of-the-art approaches on four simulated stream image datasets.

4.1 Datasets

There are many streaming image or text data in real applications, such as road condition images collected by unmanned vehicles during driving, sequential remote sensing images collected by satellites and incremental tweet data collected by social media platforms. Considering data privacy and versatility, we utilize three commonly used visual datasets for class-incremental scenario following most NCD methods [17], [18], [30] (more experimental results of complex image data and real time sequential text data can refer to the supplementary), i.e., MNIST [49], CIFAR-10 [50], and CIFAR-100.[1] In detail, MNIST dataset contains labeled handwritten digits images from 10 categories, where each class contains between 6313 and 7877 monochrome images; CIFAR-10 dataset has a total of 60000 color images of 32x32 pixels from 10 natural image classes; CIFAR-100 dataset is enlarged CIFAR-10, and we structure CIFAR-100 into 2 datasets: CIFAR-50 and CIFAR-100 according to [17].

Inspired from [17], [18], [30], [51], [52], we utilize the given testing data from the raw dataset as a holdout set to evaluate forgetting, and use the given training data to generate the streaming data. Specifically, we rearrange instances in each dataset to emulate a streaming form with novel classes considering two conditions: (1) single novel class for each time window; (2) multiple novel classes for each time window. For the single novel class case, we randomly choose C initial classes, and only 1 novel class may start in each time window.

1. http://www.cs.toronto.edu/kriz/cifar.html



(a) Single Novel Class (b) Multiple Novel Classes

Fig. 4. The class distribution of simulated stream on CIFAR-10 dataset as an example. (a) represents the single novel class case, and (b) denotes the multiple novel classes case. The X-axis denotes the streaming data and the Y-axis is the class information.

In order to be more in line with real-world applications, each known class may disappear randomly at the end of current time window. Specifically, we set $C = 5$ for MNIST and CIFAR-10, $C = 30$ for CIFAR-50, and $C = 50$ for CIFAR-100. Fig. 4a presents a simulated example of the CIFAR-10, i.e., we randomly choose 5 initial classes, and there are 5 time windows with 1 novel class appearing in for each time window. For the multiple novel class case, we randomly choose C initial classes, and K^t novel classes (i.e., $K^t \in [2, K]$ novel classes) may randomly appear from each time window. Similar to single class setting, each class may disappear randomly. Specifically, we set $C = 3$ for MNIST and CIFAR-10, $C = 30$ for CIFAR-50, and $C = 50$ for CIFAR-100. Fig. 4b presents a simulated example of the CIFAR-10, i.e., we choose 3 initial classes and there are 3 time windows, 2 novel classes emerges in the first time window, 3 novel classes in the second time window, 2 novel classes in the last time window. In detail, for each dataset, we randomly choose pre-set number of categories from the total classes as known class set, the rests are regarded as unknown class set. The data of known class set can be divided into two parts: 1) 50% of data are regarded as initial training data; and 2) the remaining data are used to constitute a streaming data. We simulate a streaming data as shown in Fig. 4, the data before time step 1 are training data. Then each class simulates an independent streaming data by shuffling instances randomly and arranging the data according to the index. A new class of streaming data appends every fixed time interval. Therefore, the instances that occurred in constitute a time window data mixed with known and unknown instances.

4.2 Compared Methods

To validate the effectiveness of proposed CILF, we compared it with existing state-of-the-art novel class detection approaches and incremental learning methods.

First, we compared CILF with existing NCD and incremental NCD methods. Including traditional anomaly detection and linear methods: Iforest [53], One-Class SVM (One-SVM) [40], LACU-SVM (LACU) [20], SENC-MAS (SENC) [21]; as well as deep methods: ODIN-CNN (ODIN) [23], CFO [17], CPE [18] and DTC [24]. Abbreviations in parentheses. DTC is a clustering based method for multiple unknown classes detection. Note that Iforest, One-SVM, LACU, ODIN, CFO, and DTC are NCD methods, SENC and CPE are incremental NCD methods. All NCD baselines except Iforest can be updated incrementally using newly labeled unknown class data and memory data. Specifically, 1) Iforest, ODIN, and CFO can only perform binary classifications, i.e., whether the instance is an unknown class or not. Thus we further conduct unsupervised clustering on both known and unknown class data for subdividing. 2) all baselines are one-class methods except DTC, i.e., they perform NCD in two steps: first detect the super-class that includes all unknown classes, and second perform unsupervised clustering. 3) all of baselines are NCD methods except LACU, SENC and CPE, but they can be applied in incremental NCD by combing memory data to update following [18].

To validate the incremental model update, we also compare our method with state-of-the-art forgetting methods: DNN-Base, DNN-L2, DNN-EWC [28], IMM [29], DEN [54], and each time window is regarded as a task for these methods.

4.3 Evaluation Metrics

CILF can distinguish the known and unknown classes, while mitigating the forgetting. Therefore, we measure the proposed method from two aspects: (1) NCD performance; (2) Forgetting performance.

Following [30], we adopt the commonly used evaluation metrics for novel class detection: 1) Normalized Accuracy (NA), which weights the accuracy for known and novel classes [55]; 2) Macro-F-measure and Micro-F-measure; and 3) AUROC, which considers the NCD task as a combination of novelty detection and multi-class recognition [17]. Moreover, to validate the effect of overcoming catastrophic forgetting, we calculate the performance on the forgetting profile of different learning algorithms as [52], i.e., let $acc_{m,n}$ be the accuracy evaluated on the hold-out sets, i.e. the novel classes emerge on $n-$th time window ($n \leq m$), after training the network incrementally from stage 1 to m, the average accuracy at time m is defined as: $A_m = \frac{1}{m}\sum_{n=1}^{m} acc_{m,n}$ [52]. higher A_m represents for better classifier. $Forgetting = \frac{A^* - mean(A)}{A^*}$, where A^* is the optimal accuracy with the entire data. We repeat all experiments 5 times, and record the mean and std.

4.4 Implementation

We develop CILF based on convolutional network structure as ResNet18 [56]. Note that we use an identical set of hyper-parameters ($\lambda_1 = 1$, $\lambda_2 = 1$, $\alpha = 0.3$, $\beta = 0.8$, $\upsilon = 0.2$, $\delta = 3$, $\phi = 10$). In all of our models and experiments, we adopt standard SGD with Nesterov momentum [57], where the momentum is 0.9. We train the initial model f as following: the number of epochs is 20, the batch size is 128, the learning rate is 0.01, and weight decay is 0.001. We implement all baselines and perform all experiments based on code released by corresponding authors. For CNN based methods, we use the same network architecture and parameters during training, such as optimizer, learning rate schedule, and data pre-processing. Our method is implemented on a RTX 2080TI GPU.

4.5 Single Novel Class Detection

Table 2 compares the detection performance of CILF with all baseline methods on each streaming data under the single novel class case. We observe that: (1) CNN-based methods perform better than traditional detection approaches, i.e.,

TABLE 2
Classification of Known Classes and Novel Class Detection Performance Over Streaming Data in Single Novel Class Case

Methods	Average NA ↑				Average Macro-F-Measure ↑			
	MNIST	CIFAR-10	CIFAR-50	CIFAR-100	MNIST	CIFAR-10	CIFAR-50	CIFAR-100
Iforest	.189±.021	.131±.023	.045±.006	.040±.004	.156±.023	.096±.011	.035±.005	.027±.004
One-SVM	.211±.032	.136±.031	.043±.007	.039±.000	.135±.021	.090±.014	.032±.003	.024±.005
LACU-SVM	.222±.034	.141±.020	.045±.005	.039±.004	.170±.018	.096±.010	.034±.004	.026±.006
SENC-MAS	.203±.030	.134±.033	.043±.006	.038±.001	.149±.031	.082±.013	.031±.003	.022±.005
ODIN-CNN	.293±.011	.194±.020	.068±.001	.034±.027	.323±.029	.156±.033	.031±.002	.055±.045
CFO	.276±.008	.202±.015	.037±.001	.022±.008	.292±.022	.167±.025	.045±.002	.033±.013
CPE	.298±.013	.250±.020	.055±.001	.047±.017	.337±.034	.281±.033	.109±.002	.087±.031
DEC	.289±.058	.245±.152	.035±.001	.027±.001	.357±.081	.261±.344	.048±.001	.018±.001
CILF	**.371±.022**	**.288±.034**	**.092±.008**	**.055±.004**	**.365±.012**	**.302±.033**	**.158±.005**	**.092±.008**
Methods	Average Micro-F-Measure ↑				Average AUROC ↑			
	MNIST	CIFAR-10	CIFAR-50	CIFAR-100	MNIST	CIFAR-10	CIFAR-50	CIFAR-100
Iforest	.234±.036	.142±.021	.066±.008	.052±.008	.083±.014	.140±.343	.077±.014	.105±.009
One-SVM	.214±.074	.147±.031	.064±.008	.046±.007	.117±.009	.101±.021	.096±.011	.074±.020
LACU-SVM	.258±.042	.148±.022	.064±.007	.050±.008	.123±.013	.068±.012	.089±.016	.117±.081
SENC-MAS	.216±.065	.140±.036	.062±.008	.044±.006	.104±.036	.082±.026	.043±.010	.079±.096
ODIN-CNN	.285±.021	.188±.040	.035±.001	.058±.041	.259±.189	.185±.063	.102±.045	.123±.096
CFO	.351±.016	.304±.030	.054±.001	.047±.017	.208±.180	.162±.073	.076±.030	.102±.092
CPE	.336±.026	.299±.040	.110±.001	.092±.033	.270±.184	.193±.060	.114±.037	.119±.100
DEC	.323±.073	.289±.305	.064±.001	.025±.001	.264±.187	.190±.058	.108±.036	.114±.097
CILF	**.355±.025**	**.310±.025**	**.122±.008**	**.103±.008**	**.317±.018**	**.255±.039**	**.125±.012**	**.130±.027**

The best results are highlighted in bold.

One-SVM, LACU-SVM, SENC-MAS. This indicates that neural network provides superior feature embeddings for prediction and detection over high dimensional streaming data; (2) CILF consistently outperforms all compared CNN-based methods over all the criteria. For example, in CIFAR-10, CILF provides at least 2% improvements than other methods. This indicates the effectiveness of both prototype based loss for feature embedding and curriculum clustering operator for detection; and (3) The detection performance for large-scale datasets, (e.g., CIFAR-100) still needs to be improved, as the results of all methods are unsatisfactory. This is a roadmap for future work. A possible solution is to use external corpus to build a larger-scale deep pre-train model or consider transfer learning technology to transfer extra source domain corpus knowledge.

Table 3 compared the forgetting performance of CILF with all baseline methods, which defines the forgetting of emerge class on a particular window, i.e., the difference between maximum knowledge gained about that window throughout learning process and the knowledge we currently have about it, and the lower difference the better. The results show that CILF has the least forgetting, which validates that the memory distillation and prototype regularization can mitigate the forgetting of known class data. Moreover, Fig. 5 exhibits the

TABLE 3
Forgetting Measure of Known Classes Over Streaming Data in Single Novel Class Case

Methods	Forgetting ↓			
	MNIST	CIFAR-10	CIFAR-50	CIFAR-100
Iforest	.027±.006	.021±.002	.043±.001	.067±.001
One-SVM	.028±.007	.019±.004	.042±.002	.068±.002
LACU-SVM	.032±.005	.029±.002	.038±.003	.061±.001
SENC-MAS	.028±.005	.020±.001	.036±.002	.060±.001
ODIN-CNN	.040±.004	.012±.006	.023±.002	.018±.005
CFO	.023±.012	.010±.003	.014±.001	.016±.005
CPE	.022±.001	.007±.001	.017±.001	.013±.003
DEC	.026±.005	.009±.001	.015±.002	.014±.001
DNN-Base	.042±.005	.012±.007	.015±.003	.024±.006
DNN-L2	.032±.010	.011±.004	.012±.007	.029±.003
DNN-EWC	.023±.014	.016±.010	.014±.009	.016±.007
IMM	.030±.012	.009±.003	.016±.004	.025±.004
DEN	.024±.003	.007±.004	.011±.009	.017±.001
CILF	**.020±.016**	**.006±.001**	**.010±.001**	**.013±.003**

The best results are highlighted in bold.

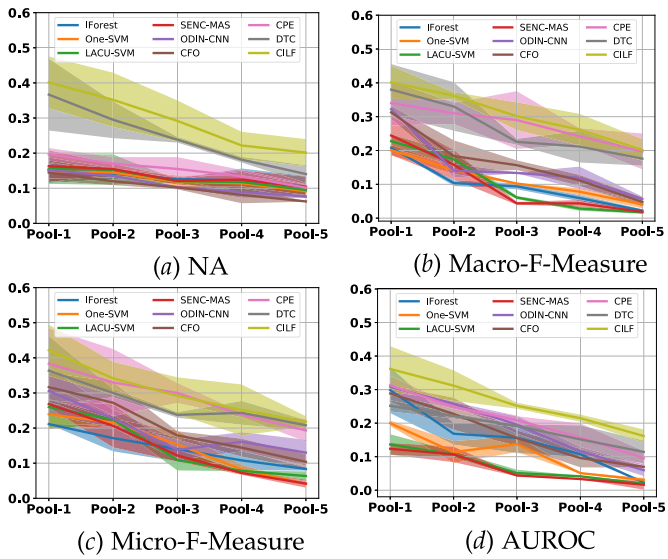


Fig. 5. (Best view in color.) Performance of known classes on different time windows of CIFAR-10.

TABLE 4
Classification of Known Classes and Novel Class Detection Performance Over Streaming Data in Multiple Novel Class Case

Methods	Average NA ↑				Average Macro-F-Measure ↑			
	MNIST	CIFAR-10	CIFAR-50	CIFAR-100	MNIST	CIFAR-10	CIFAR-50	CIFAR-100
Iforest	.348±.069	.277±.062	.133±.021	.095±.014	.181±.062	.103±.004	.040±.004	.028±.002
One-SVM	.361±.085	.277±.049	.136±.017	.089±.014	.188±.085	.107±.008	.038±.005	.026±.003
LACU-SVM	.357±.054	.274±.062	.133±.012	.090±.017	.191±.054	.098±.008	.032±.005	.023±.002
SENC-MAS	.336±.075	.280±.050	.139±.021	.091±.013	.163±.071	.106±.002	.035±.004	.024±.002
ODIN-CNN	.396±.021	.304±.039	.181±.008	.071±.007	.268±.032	.243±.005	.160±.006	.054±.004
CFO	.353±.012	.287±.027	.212±.009	.104±.008	.380±.021	.363±.001	.288±.007	.183±.005
CPE	.413±.030	.401±.061	.240±.020	.132±.020	.236±.047	.339±.037	.249±.022	.244±.021
DEC	.350±.142	.398±.084	.206±.012	.089±.016	.327±.315	.302±.217	.230±.017	.110±.011
CILF	**.493±.051**	**.422±.054**	**.258±.040**	**.179±.031**	**.392±.045**	**.407±.033**	**.332±.035**	**.259±.041**
Methods	Average Micro-F-Measure ↑				Average AUROC ↑			
	MNIST	CIFAR-10	CIFAR-50	CIFAR-100	MNIST	CIFAR-10	CIFAR-50	CIFAR-100
Iforest	.272±.079	.153±.009	.067±.002	.049±.003	.140±.015	.126±.019	.113±.023	.140±.009
One-SVM	.293±.122	.161±.018	.068±.008	.045±.003	.158±.016	.153±.011	.146±.008	.092±.010
LACU-SVM	.294±.060	.144±.006	.062±.006	.043±.002	.149±.002	.158±.016	.145±.005	.152±.024
SENC-MAS	.250±.082	.165±.012	.063±.008	.045±.004	.107±.092	.123±.006	.086±.051	.043±.032
ODIN-CNN	.242±.024	.193±.006	.072±.009	.067±.006	.231±.052	.145±.245	.193±.140	.133±.250
CFO	.245±.016	.212±.004	.108±.010	.104±.008	.195±.050	.214±.257	.115±.145	.159±.245
CPE	.207±.037	.367±.035	.200±.023	.146±.022	.241±.056	.255±.243	**.201±.140**	.183±.254
DEC	.294±.292	.231±.204	.141±.016	.116±.011	.234±.061	.217±.076	.197±.137	.171±.253
CILF	**.422±.043**	**.442±.043**	**.212±.022**	**.152±.016**	**.286±.028**	**.261±.022**	.189±.034	**.192±.016**

The best results are highlighted in bold.

results more directly. Due to page limitation, we only report the results of CIFAR-10. The results indicate that the performance of known classes falls slower at different windows, which states that CILF can mitigate forgetting efficiently.

4.6 Multiple Novel Class Detection

Table 4 compares the detection performance of CILF with all baseline methods under the multiple novel classes case. We observe that: (1) multiple novel class detection method, i.e., DEC, does not outperform single novel class detection methods with subsequent clustering operator. This indicates that direct clustering method may be influenced by the embedding confusion; and (2) CILF consistently outperforms all compared CNN-based methods in all the criteria except AUROC on CIFAR-50, for the reason that there exist class-imbalance phenomenon in CIFAR-50, i.e., the number of novel class instances are fewer. CILF sacrifices precision in order to improve the high recall rate of the new class. This further indicates the effectiveness of curriculum clustering operator for detection. Table 5 and Fig. 6 compared the forgetting performance of CILF with baseline methods. Identically, the results show that CILF has the least forgetting, and the performance of known classes fall slower, which shows

TABLE 5
Forgetting Measure of Known Classes Over Streaming Data in Multiple Novel Class Case

Methods	Forgetting ↓			
	MNIST	CIFAR-10	CIFAR-50	CIFAR-100
Iforest	.014±.015	.006±.004	.035±.002	.048±.005
One-SVM	.018±.010	.005±.003	.033±.004	.049±.003
LACU-SVM	.017±.010	.004±.002	.029±.003	.042±.004
SENC-MAS	.011±.011	.006±.002	.032±.002	.041±.004
ODIN-CNN	.011±.008	.013±.011	.015±.006	.013±.010
CFO	.019±.005	.008±.006	.009±.008	.023±.003
CPE	.007±.001	.011±.003	.009±.002	.005±.002
DEC	.002±.003	.009±.001	.006±.002	.023±.001
DNN-Base	.022±.002	.023±.009	.032±.007	.039±.006
DNN-L2	.021±.002	.026±.001	.035±.005	.041±.002
DNN-EWC	.016±.010	.017±.010	.020±.005	.021±.009
IMM	.022±.030	.023±.010	.024±.012	.030±.019
DEN	.010±.009	.013±.005	.013±.002	.021±.011
CILF	**.001±.001**	**.005±.001**	**.001±.001**	**.008±.001**

The best results are highlighted in bold.

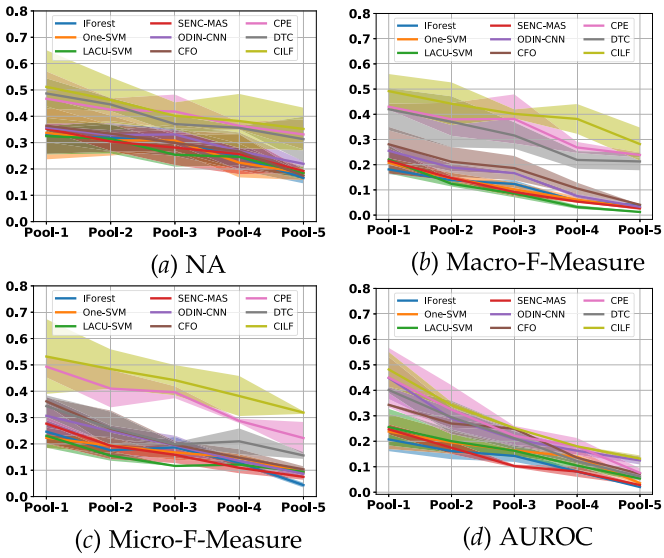


Fig. 6. Performance criteria on different time window on CIFAR-10 in multiple novel class case.

TABLE 6
Classification of Known Classes and Novel Class Detection Performance Over Streaming Data

Methods	Average NA ↑				Average Macro-F-Measure ↑			
	CIFAR-100 (sin)	OTTO (sin)	CIFAR-100 (mul)	OTTO (mul)	CIFAR-100 (sin)	OTTO (sin)	CIFAR-100 (mul)	OTTO (mul)
w/o FEN	.301±.032	.208±.022	.053±.005	.012±.002	.253±.008	.232±.023	.056±.013	.032±.013
w/o NCD	.273±.021	.185±.028	.033±.003	.009±.001	.204±.013	.201±.029	.043±.009	.022±.005
w/o IMU	.325±.017	.221±.032	.050±.006	.008±.001	.289±.019	.198±.027	.051±.003	.043±.012
CILF	**.371±.022**	**.288±.034**	**.092±.008**	**.055±.004**	**.365±.012**	**.302±.033**	**.158±.005**	**.092±.008**
Methods	Average Micro-F-Measure ↑				Average AUROC ↑			
	CIFAR-100 (sin)	OTTO (sin)	CIFAR-100 (mul)	OTTO (mul)	CIFAR-100 (sin)	OTTO (sin)	CIFAR-100 (mul)	OTTO (mul)
w/o FEN	.274±.026	.223±.019	.054±.005	.043±.006	.208±.028	.185±.032	.044±.014	.075±.009
w/o NCD	.256±.021	.183±.029	.044±.003	.055±.006	.164±.013	.143±.022	.053±.008	.065±.014
w/o IMU	.304±.031	.219±.021	.041±.006	.067±.003	.161±.014	.144±.031	.061±.009	.045±.013
CILF	**.355±.025**	**.310±.025**	**.122±.008**	**.103±.008**	**.317±.018**	**.255±.039**	**.125±.012**	**.130±.027**

** (sin) denotes the single novel class case and * (mul) represents the multiple novel class case. The best results are highlighted in bold.*

that CILF can mitigate forgetting under multiple novel classes scenario.

4.7 Ablation Study

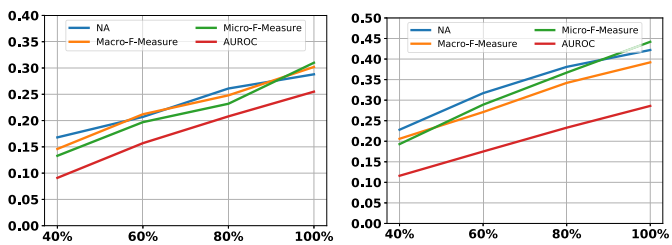
We conduct more ablation studies to verify the effectiveness of the proposed sampling method: 1) w/o FEN, we replace the feature embedding network with the classification network considering prediction layer; 2) w/o NCD, we replace the learnable curriculum clustering with the traditional uncertainty prediction for novel class detection; 3) w/o IMU, we directly update the model without considering the forgetting mechanism. Table 6 records the forgetting performance on each streaming data with single novel class. Table 7 records the detection performance on each streaming data with single novel class. The results validate that: 1) the detection performance of w/o NCD, w/o FEN, and w/o IMU declines, which indicates that each component contributes to the detection. In detail, the feature embedding network is more efficient to learn the inter-class and intra-class structure than classification network, and is more convenient for model expansion. The learnable curriculum clustering performs better than traditional uncertainty prediction, which indicates the effectiveness of curriculum learning. And the forgetting regularization is benefit for classifying known class in the unsupervised streaming data. 2) the degradation of w/o NCD is more obvious, for the reason that learnable curriculum clustering is more convenient for multiple novel class detection, while comparing methods need conduct second clustering after detection. Thereby, all three components contribute to the class-incremental learning, and the learnable curriculum clustering contributes more under the multiple novel class case.

4.8 Influence of Query Size

Fig. 7 shows the influence of querying number about potential novel class instances, and we only give the results on CIFAR-10 considering page limitation. Here, we randomly query a subset, i.e., a certain percentage of potential instances from the current window. From the Figure, we observe that the prediction performance is improved with the increase of labeled data, which verifies the importance of ground-truths for model update.

4.9 Parameter Sensitivity

The main parameters in novel class detection and model update are the λ_1 and λ_2 in Eq. (8). We vary these parame-



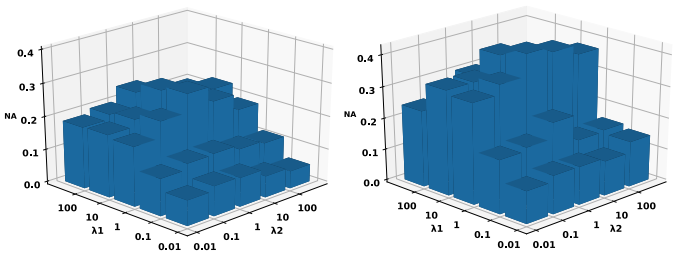
(*a*) Single CIFAR-10 (*d*) Multiple CIFAR-10

Fig. 7. Relationship between detection performance during stream with label request percentage for every time window.

TABLE 7
Forgetting Measure of Known Classes Over Streaming Data in Single Novel Class Case

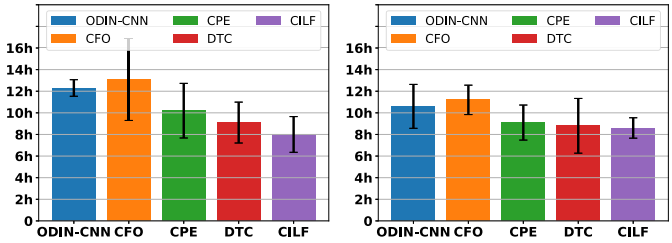
Methods	Forgetting ↓			
	CIFAR-100 (sin)	OTTO (sin)	CIFAR-100 (mul)	OTTO (mul)
w/o FEN	.046±.019	.026±.008	.027±.008	.032±.012
w/o NCD	.029±.011	.012±.005	.019±.007	.018±.009
w/o IMU	.037±.023	.019±.003	.024±.003	.025±.005
CILF	**.020±.016**	**.006±.001**	**.010±.001**	**.013±.003**

The best results are highlighted in bold.



(*a*) Single CIFAR-10 (*b*) Multiple CIFAR-10

Fig. 8. Parameter sensitivity of λ_1 and λ_2 for the CIFAR-10 in novel detection. (a) is single novel class case, (c) is multiple novel class case.



(*c*) Multiple CIFAR-10 (*d*) Multiple CIFAR-50

Fig. 9. Execution time analysis.

ters in $\{0.01, 0.1, 1, 10, 100\}$ to study their sensitivity to classification performance and record the AUROC results in Fig. 8. Both the single and multiple cases indicate that the performances are better when setting λ_1 with a larger value, i.e., larger than 1.

4.10 Execution Time for Model Update

Considering that our method focuses on multiple novel class detection, we analyze execution time for detecting and updating model under the multiple novel class case. In detail, we select five deep methods, i.e. ODIN-CNN, CFO, CPR, DTC and CILF, and record their execution time under the multiple novel class case in Fig. 9. CILF takes less time, for the following reasons: 1) Other methods require additional clustering operations; and 2) Embedding confusion will slow down the clustering convergence, which indicates that the curriculum clustering can accelerate detection.

5 Conclusion

Real-word application always receive streaming data, which emerges previously unknown classes sequentially. Class-Incremental Learning faces two main challenges: Novel class detection, streaming test data will accept unknown classes. Model expansion, the model needs to be effectively updated after the new class detection. However, traditional methods have not always fully considered these two challenges. To this end, we propose a Class-Incremental Learning without Forgetting (CILF) framework. CILF designed to regularize classification with decoupled prototype based loss, which can improve the intra-class and inter-class structure significantly, and acquire a compact embedding representation for novel class detection in result. Then, CILF employed a learnable curriculum clustering operator to estimate the number of semantic clusters via fine-tuning the learned network. Last, CILF updated the network effectively with robust regularization to mitigate the catastrophic forgetting. In result, CILF utilized the decoupled prototype network to effectively detect multiple novel classes and update the model in a unified framework. Empirical studies showed the superior performances of CILF. As verified by experiments, there is still room for improvement of NCD on large-scale datasets. Therefore, how to further improve the performance of detection in complex environment and further demonstrate interpretability are very interesting future research directions. We build the model with MindSpore tool [58].

References

[1] I. Masi, Y. Wu, T. Hassner, and P. Natarajan, "Deep face recognition: A survey," in *Proc. 31st SIBGRAPI Conf. Graph. Patterns Images*, Parana, Brazil, 2018, pp. 471–478.

[2] C. Qin *et al.*, "An enhanced neural network approach to person-job fit in talent recruitment," *ACM Trans. Inf. Syst.*, vol. 38, no. 2, pp. 1–33, 2020.

[3] L. Schmarje, M. Santarossa, S.-M. Schroder, and R. Koch, "A survey on semi-, self- and unsupervised techniques in image classification," 2020, *arXiv:2002.08721*.

[4] Y. Sun, F. Zhuang, H. Zhu, Q. Zhang, Q. He, and H. Xiong, "Market-oriented job skill valuation with cooperative composition neural network," *Nat. Commun.*, vol. 12, no. 1, pp. 1–12, 2021.

[5] H. Lin, H. Zhu, J. Wu, Y. Zuo, C. Zhu, and H. Xiong, "Enhancing employer brand evaluation with collaborative topic regression models," *ACM Trans. Inf. Syst.*, vol. 38, no. 4, pp. 1–33, 2020.

[6] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Proc. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2009, pp. 1410–1418.

[7] Y. Fu, T. Xiang, Y.-G. Jiang, X. Xue, L. Sigal, and S. Gong, "Recent advances in zero-shot recognition: Toward data-efficient understanding of visual content," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 112–125, Jan. 2018.

[8] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 3, pp. 453–465, Mar. 2014.

[9] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha, "Synthesized classifiers for zero-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 5327–5336.

[10] J. Li, M. Jing, K. Lu, L. Zhu, Y. Yang, and Z. Huang, "Alleviating feature confusion for generative zero-shot learning," in *Proc. ACM Int. Conf. Multimedia*, Nice, France, 2019, pp. 1587–1595.

[11] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 1563–1572.

[12] M. Hassen and P. K. Chan, "Learning a neural-network-based representation for open set recognition," 2018, *arXiv:1802.04365*.

[13] Z. Ge, S. Demyanov, and R. Garnavi, "Generative openmax for multi-class open set classification," in *Proc. Brit. Mach. Vis. Conf.*, London, U.K., 2017.

[14] R. Ratcliff, "Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions," *Psychol. Review*, vol. 97, no. 2, pp. 285–308, 1990.

[15] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "ICARL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* Honolulu, HI, USA, 2017, pp. 5533–5542.

[16] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proc. AAAI*, New Orleans, LA, USA, 2018, pp. 3302–3309.

[17] L. Neal, M. L. Olson, X. Z. Fern, W.-K. Wong, and F. Li, "Open set learning with counterfactual images," in *Proc. Eur. Conf. Comput. Vis.*, Munich, Germany, 2018, pp. 620–635.

[18] Z. Wang, Z. Kong, S. Chandra, H. Tao, and L. Khan, "Robust high dimensional stream classification with novel class detection," in *Proc. 35th Int. Conf. Data Eng.*, Macao, China, 2019, pp. 1418–1429.

[19] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," 2017, *arXiv:1710.10628*.

[20] Q. Da, Y. Yu, and Z.-H. Zhou, "Learning with augmented class by exploiting unlabeled data," in *Proc. AAAI*, QC, Canada, 2014, pp. 1760–1766.

[21] X. Mu, F. Zhu, J. Du, E.-P. Lim, and Z.-H. Zhou, "Streaming classification with emerging new class by class matrix sketching," in *Proc. AAAI*, San Francisco, CA, USA, 2017, pp. 2373–2379.

[22] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *Proc. Int. Conf. Learn. Representations*, Toulon, France, 2017.

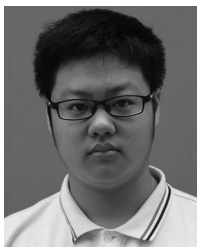
[23] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *Proc. Int. Conf. Learn. Representations*, Vancouver, BC, Canada, 2018.

[24] K. Han, A. Vedaldi, and A. Zisserman, "Learning to discover novel visual categories via deep transfer clustering," in *Proc. Int. Conf. Comput. Vis.*, Seoul, South Korea, 2019, pp. 8400–8408.

[25] L. J. Zhang, T. Yang, R. Jin, Y. Xiao, and Z.-H. Zhou, "Online stochastic linear optimization under one-bit feedback," in *Proc. Int. Conf. Mach. Learn.*, New York, NY, USA, 2016, pp. 392–401.

[26] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proc. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 6467–6476.
[27] Z. Li and D. Hoiem, "Learning without forgetting," in *Proc. Eur. Conf. Comput. Vis.*, Amsterdam, The Netherlands, 2016, pp. 614–629.
[28] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," 2016, *arXiv:1612.00796*.
[29] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," in *Proc. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 4655–4665.
[30] C. Geng, S.-J. Huang, and S. Chen, "Recent advances in open set recognition: A survey," 2018, *arXiv:1811.08581*.
[31] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[32] B. Kulis, "Metric learning: A survey," *Found. Trends Mach. Learn.*, vol. 5, no. 4, pp. 287–364, 2013.
[33] C. M. Bishop and N. M. Nasrabadi, "Pattern recognition and machine learning," *J. Electron. Imag.*, Berlin, Germany: Springer, 2007.
[34] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, New York, NY, USA, 2016, pp. 507–516.
[35] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," in *Proc. Int. Conf. Learn. Representations.*, Toulon, France, 2017.
[36] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, 2015, pp. 815–823.
[37] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," 2017, *arXiv:1703.07737*.
[38] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Proc. Neural Inf. Process. Syst.*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., Barcelona, Spain, 2016, pp. 1849–1857.
[39] B. Yu and D. Tao, "Deep metric learning with tuplet margin loss," in *Proc. Int. Conf. Comput. Vis.*, Seoul, South Korea, 2019, pp. 6489–6498.
[40] B. Scholkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.
[41] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. Int. Conf. Mach. Learn.*, Montreal, QC, Canada, 2009, pp. 41–48.
[42] A. Graves *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.
[43] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated curriculum learning for neural networks," in *Proc. Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, 2017, pp. 1311–1320.
[44] M. P. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *Proc. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2010, pp. 1189–1197.
[45] G. Hacohen and D. Weinshall, "On the power of curriculum learning in training deep networks," in *Proc. Int. Conf. Mach. Learn.*, Long Beach, CA, USA, 2019, pp. 2535–2544.
[46] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors A multilevel approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1944–1957, Nov. 2007.
[47] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Perez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognit.*, vol. 46, no. 1, pp. 243–256, 2013.
[48] A. Haque, L. Khan, and M. Baron, "SAND: Semi-supervised adaptive novel class detection and classification over data stream," in *Proc. AAAI*, Phoenix, AZ, USA, 2016, pp. 1652–1658.
[49] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist
[50] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[51] M. M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham, "Classification and novel class detection in concept-drifting data streams under time constraints," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 6, pp. 859–874, Jun. 2011.
[52] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," 2018, *arXiv:1801.10112*.
[53] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *Proc. Int. Conf. Data Mining*, Pisa, Italy, 2008, pp. 413–422.
[54] J. Lee, J. Yoon, E. Yang, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," 2017, *arXiv:1708.01547*.
[55] P. R. M. Junior *et al.*, "Nearest neighbors distance ratio open-set classifier," *Mach. Learn.*, vol. 106, no. 3, pp. 359–386, 2017.
[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 770–778.
[57] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.
[58] Mindspore. 2020. [Online]. Available: https://www.mindspore.cn/

Yang Yang received the PhD degree in computer science from Nanjing University, China, in 2019. In 2019, he became a faculty member with the Nanjing University of Science and Technology, China. He is currently a professor with the School of Computer Science and Engineering. His research interests include machine learning and data mining, including heterogeneous learning, model reuse, and incremental mining. He is currently the PC in leading conferences, IJCAI, AAAI, ICML, and NIPS.



Zhen-Qiang Sun is currently working toward the MSc degree with the School of Computer Science and Technology, Nanjing Normal University, China. His research interests include machine learning, data mining, and incremental learning.

Hengshu Zhu (Senior Member, IEEE) received the BE and PhD degrees in computer science from the University of Science and Technology of China, China, in 2009 and 2014, respectively. He is currently a principal data scientist and an architect with Baidu Inc. He has authored or coauthored prolifically in refereed journals and conference proceedings, including the *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Mobile Computing*, *ACM Transactions on Information Systems*, *ACM Transactions on Knowledge Discovery* from Data, ACM SIGKDD, ACM SIGIR, WWW, the IJCAI, and AAAI. His research interests include data mining and machine learning, with a focus on developing advanced data analysis techniques for innovative business applications. He was regularly on the organization and program committees of numerous conferences, including the program co-chair of the KDD Cup-2019 Regular ML Track and the founding co-chair of the first International Workshop on Organizational Behavior and Talent Analytics and the International Workshop on Talent and Management Computing, in conjunction with ACM SIGKDD. He was the recipient of the Distinguished Dissertation Award of CAS (2016), the Distinguished Dissertation Award of CAAI (2016), the Special Prize of President Scholarship for Postgraduate Students of CAS (2014), the Best Student Paper Award of KSEM-2011, WAIM-2013, CCDM-2014, and the Best Paper Nomination of ICDM-2014. He is the senior member of the ACM and CCF.

Yanjie Fu received the BE degree from the University of Science and Technology of China in 2008, the ME degree from the Chinese Academy of Sciences in 2011, and the PhD degree from Rutgers University in 2016. He is currently an assistant professor with the Missouri University of Science and Technology. He has authored or coauthored prolifically in refereed journals and conference proceedings, including the *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on Knowledge Discovery from Data*, *IEEE Transactions on Mobile Computing*, and ACM SIGKDD. His research interests include data mining and big data analytics.

Yuanchun Zhou received the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2006. He is currently a professor, PhD supervisor, and the associate director of Computer Network Information Center, Chinese Academy of Sciences. His research interests include data mining, big data processing, and knowledge graph.

Hui Xiong (Fellow, IEEE) received the PhD degree in computer science from the University of Minnesota, USA. He is currently a professor with Rutgers, the State University of New Jersey. His research interests include data mining, mobile computing, and their applications in business. He was regularly on the organization and program committees of numerous conferences as a program co-chair of the Industrial and Government Track for the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, the program co-chair for the IEEE 2013 International Conference on Data Mining, the general co-chair for the 2015 IEEE International Conference on Data Mining, and the program co-chair of the Research Track for the 2018 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. He was the recipient of the 2021 AAAI Best Paper Award and the 2011 IEEE ICDM Best Research Paper Award. For his outstanding contributions to data mining and mobile computing, he was elected an AAAS fellow and an IEEE fellow in 2020.

Jian Yang (Member, IEEE) received the PhD degree in pattern recognition and intelligence systems from the Nanjing University of Science and Technology (NUST), Nanjing, China, in 2002. In 2003, he was a postdoctoral researcher with the University of Zaragoza, Zaragoza, Spain. From 2004 to 2006, he was a postdoctoral fellow with Biometrics Centre, The Hong Kong Polytechnic University, Hong Kong. From 2006 to 2007, he was a postdoctoral fellow with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA. He is currently a Chang-Jiang professor with the School of Computer Science and Engineering, NUST. He has authored more than 200 scientific papers in pattern recognition and computer vision. His papers have been cited more than 6000 times in the Web of Science and 15,000 times in the Google Scholar. His current research interests include pattern recognition, computer vision, and machine learning. He is currently a fellow of the IAPR. He is currently an associate editor for *Pattern Recognition*, *Pattern Recognition Letters*, the *IEEE Transactions On Neural Networks And Learning Systems*, and *Neurocomputing*.