S2OSC: A Holistic Semi-Supervised Approach for Open Set Classification

YANG YANG and HONGCHEN WEI, Nanjing University of Science and Technology
ZHEN-QIANG SUN, Nanjing Normal University
GUANG-YU LI, Nanjing University of Science and Technology
YUANCHUN ZHOU, Computer Network and Information Center, Chinese Academy of Sciences
HUI XIONG, Rutgers University
JIAN YANG, Nanjing University of Science and Technology

Open set classification (OSC) tackles the problem of determining whether the data are in-class or out-of-class during inference, when only provided with a set of in-class examples at training time. Traditional OSC methods usually train discriminative or generative models with the owned in-class data, and then utilize the pre-trained models to classify test data directly. However, these methods always suffer from the embedding confusion problem, i.e., partial out-of-class instances are mixed with in-class ones of similar semantics, making it difficult to classify. To solve this problem, we unify semi-supervised learning to develop a novel OSC algorithm, S2OSC, which incorporates out-of-class instances filtering and model re-training in a transductive manner. In detail, given a pool of newly coming test data, S2OSC firstly filters the mostly distinct out-of-class instances using the pre-trained model, and annotates super-class for them. Then, S2OSC trains a holistic classification model by combing in-class and out-of-class labeled data with the remaining unlabeled test data in a semi-supervised paradigm. Furthermore, considering that data are usually in the streaming form in real applications, we extend S2OSC into an incremental update framework (I-S2OSC), and adopt a knowledge memory regularization to mitigate the catastrophic forgetting problem in incremental update. Despite the simplicity of proposed models, the experimental results show that S2OSC achieves state-of-the-art performance across a variety of OSC tasks, including 85.4% of F1 on CIFAR-10 with only 300 pseudo-labels. We also demonstrate how S2OSC can be expanded to incremental OSC setting effectively with streaming data.

CCS Concepts: • **Information System** → **Data Mining**; • **Computing methodologies** → Machine learning algorithms;

Additional Key Words and Phrases: Open set classification, semi-supervised learning, embedding confusion, incremental learning

This work is supported by the National Natural Science Foundation of China under Grant (62006118, 62006119, 61836013, 91746301), Natural Science Foundation of Jiangsu Province of China under Grant (BK20200460, BK20190444). CCF-Baidu Open Fund (CCF-BAIDU OF2020011), Baidu TIC Open Fund.
Authors' addresses: Y. Yang (corresponding author), H. Wei, G.-Y. Li, and J. Yang, Nanjing University of Science and Technology, 200 Xiaolingwei, Nanjing, Jiangsu, 210094, China, emails: {yyang, weihc, guangyu.li2017, csjyang}@njust.edu.cn; Z.-Q. Sun, Nanjing Normal University, 1 Wenyuanlu, Nanjing, Jiangsu, 210023, China; email: enderman19980125@outlook.com; Y. Zhou, Computer Network and Information Center, Chinese Academy of Sciences, 2 Dongshengnanlu, Beijing, 100083, China; email: zyc@cnic.cn; H. Xiong, Rutgers University, 1 Washington Park, Newark, New Jersey, 07102; email: hxiong@rutgers.edu.

1556-4681/2021/08-ART34 $15.00
https://doi.org/10.1145/3468675

ACM Reference format:
Yang Yang, Hongchen Wei, Zhen-Qiang Sun, Guang-Yu Li, Yuanchun Zhou, Hui Xiong, and Jian Yang. 2021. S2OSC: A Holistic Semi-Supervised Approach for Open Set Classification. *ACM Trans. Knowl. Discov. Data.* 16, 2, Article 34 (August 2021), 27 pages.
https://doi.org/10.1145/3468675

1 INTRODUCTION

As the real-world is changing dynamically, many applications are non-stationary, and always receive data containing out-of-class (also called unknown class) instances, for example, self-driving cars need to identify unknown objects, face recognition system needs to distinguish unseen personal pictures, and image retrieval often emerges new categories. This problem is defined as "**Open Set Classification** (**OSC**)" in literature [Geng et al. 2018]. Different from traditional **Closed Set Classification** (**CSC**) which assumes training and testing data are drawn from same spaces, i.e., the label and feature spaces, OSC aims at not only accurating classify in-class (also called known class) instances, but also effectively detecting out-of-class instances (also called unknown class). Moreover, a generalized situation is that out-of-class instances will arise continuously with the streaming data, i.e., unknown classes appear incrementally, and this is defined as incremental OSC. Thereby, we focus on the OSC problem, and aim to extend the designed model to the incremental scenario.

Both anomaly detection [Liu et al. 2008; Xia et al. 2015] and **zero-shot learning (ZSL)** [Changpinyo et al. 2016; Li et al. 2019] are related to OSC. They have similar objectives to detect anomaly/out-of-class instances given a set of in-class examples. In contrast, anomaly detection (also called outlier detection) is an unsupervised learning task [Xia et al. 2015]. The goal is to separate abnormal in-class instances from normal ones, where the distinction from OSC is that differences between unknown and known classes are larger than that between anomalies and known classes [Cai et al. 2019]. Unlike anomaly detection, ZSL focuses on constructing related OSC models to address out-of-class detection issue, which merely utilize in-class examples and semantic information about unknown classes. Whereas the standard ZSL methods only test out-of-class instances, rather than test both known and unknown classes. Therefore, **generalized ZSL** (**GZSL**) is proposed, which automatically detect known and unknown classes simultaneously. For example, [Changpinyo et al. 2016; Li et al. 2019] learned more reliable classification models by measuring the distance between examples and corresponding in-class/out-of-class semantic embeddings. However, both ZSL and GZSL assume that semantic information (for example, attributes or descriptions) of the out-of-class is given, which is limited to classify with prior knowledge, either labeled examples or semantic side-information during training.

Therefore, a more practical classification should be able to detect out-of-class without any information of unknown classes. With the advent of deep learning, recent OSC approaches can mainly be divided into two aspects: discriminative and generative deep models. **Discriminative models** (**DMs**) mainly utilize the powerful feature learning and prediction capability of deep models to design corresponding distance or prediction confidence measures [Hendrycks and Gimpel 2017; Wang et al. 2019]. In contrast, **generative models** (**GMs**) mainly employ the adversarial learning to generate out-of-class instances near the decision margin that can fool the DM [Ge et al. 2017; Jo et al. 2018]. In summary, existing OSC approaches focus on learning a representative latent space for in-class examples that preserves details of the given classes. In this case, it is assumed that, when presented out-of-class instances to the pre-trained deep models, it will generate a poor embedding that reports a relatively higher classification error. However, this assumption does not hold for all situations. For example, as shown in Figure 1, experiments on MNIST suggest that networks (DM [Wang et al. 2019] and GM [Neal et al. 2018]) trained with simple content have high novelty

detection accuracy, i.e., the embeddings of out-of-class digits 5 and 6 are well separated from in-class examples. In contrast, instances with complex content, such as CIFAR-10, have much lower novelty detection accuracy. This is because latent embeddings learned for in-class examples can also inherently apply to represent some out-of-class instances, for example, the latent embeddings learned for cat (number 3 in Figure 1(b) and (d)) are also able to represent some instances of other out-of-class animal such as dog (number 5 in Figure 1(b) and (d)), considering similar appearance, color, and other information. This phenomenon is defined as *Embedding Confusion* in this article.

Facing the embedding confusion challenge, we note that out-of-class instances always include confused and distinct ones, i.e., distinct unknown class instances are far away from the examples of known classes, whereas the confused ones are mixed with the examples of known classes. Inspired by this phenomenon, we can firstly select the distinct out-of-class instances, then re-train a new detector by combing them with stored in-class examples in a semi-supervised paradigm. Meanwhile, the pre-trained model can be employed as a teacher model, which not only ensures that in-class instances are well represented, but also guarantees that out-of-class instances are poorly represented. In result, the learned detector can obtain well separated embeddings for in-class and out-of-class instances, and significantly improve the detection performance in return. Motivated by this intuition, we propose **Semi-Supervised OSC (S2OSC)** algorithm, a transductive detector learning process, to mitigate embedding confusion. At a high-level, S2OSC can also be adapted to **incremental S2OSC** (**I-S2OSC**) conveniently, by combing the sophisticated model update manner. I-S2OSC continues to accept test batches containing out-of-class data, and perform novelty detection and incremental model update interactively.

2 RELATED WORK

To begin the S2OSC, we first introduce existing methods for OSC, i.e., DM and GM, which are related to our S2OSC. Then, we present traditional anomaly detection and ZSL methods.

2.1 Discriminative OSC Models

These approaches mainly restrict intra-class and inter-class distance property on training data, then detect unknown classes by identifying outliers. For example, Da et al. [2014] developed the SVM-based method, which learned the concept of known classes while incorporating the structure presented in unlabeled data from open set; Mu et al. [2017] proposed to dynamically maintain two low-dimensional matrix sketches to detect emerging new classes. However, these linear approaches are difficult to process high dimensional space. Recently, several studies have applied deep learning techniques to OSC scenario. For example, Hendrycks and Gimpel [2017] distinguished known/unknown class with softmax output probabilities; Liang et al. [2018] directly utilized temperature scaling to separating the softmax score between in-distribution and out-of-distribution images; Wang et al. [2019] proposed a **Convolutional Neural Network** (**CNN**)-based prototype ensemble method, which adaptively updated prototype for robust detection. However, these methods can hardly consider the out-of-class instances in training phase.

2.2 Generative OSC Models

The key component of generation-based OSC models is to generate effective out-of-class instances. For example, Ge et al. [2017] proposed the **generative OpenMax** (**G-OpenMax**) algorithm, which provided probability estimation over generated out-of-class instances, that enabled the classifier to locate the decision margin according to both in-class and out-of-class knowledge; Jo et al. [2018] adopted the GAN technique to generate fake data considering representativeness as the out-of-class data, which can further enhance the robustness of a classifier for detection; Neal et al. [2018] introduced an augmentation technique, which adopted an encoder-decoder GAN

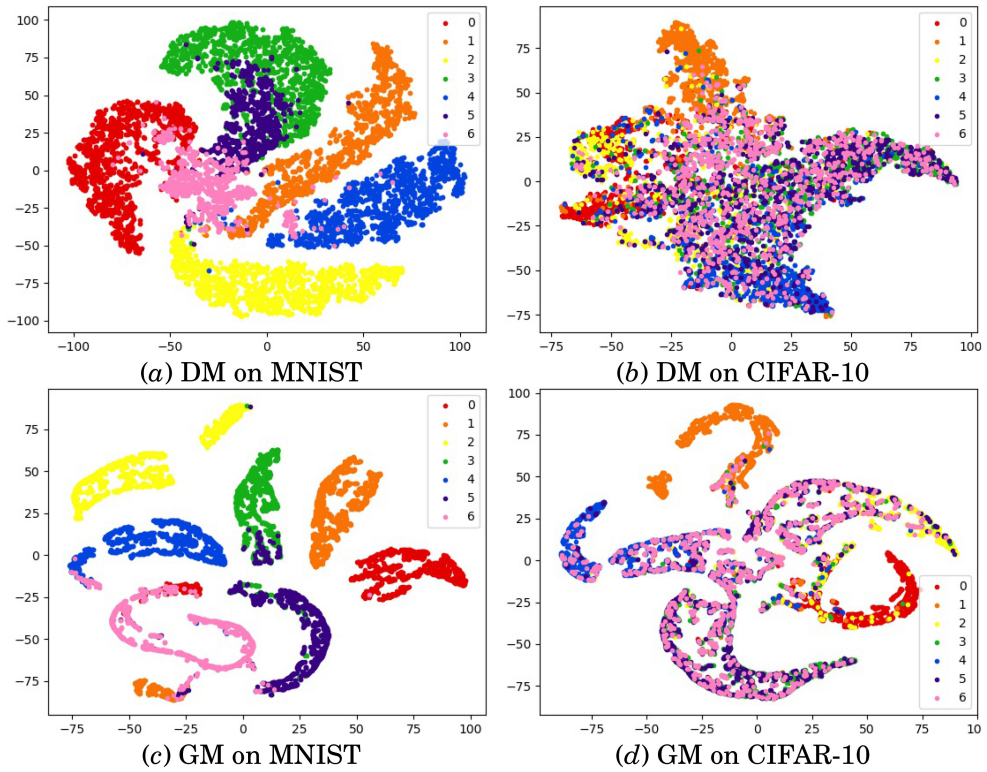


Fig. 1. T-SNE [Maaten and Hinton 2008] of DM [Wang et al. 2019] and GM [Neal et al. 2018] on simple (MNIST) and complex (CIFAR-10) datasets. We develop these two models with five known classes (i.e., 0-4) in training stage according to the raw article, then utilize the pre-trained models to achieve the embeddings of known classes (i.e., 0-4) and unknown classes (i.e., 5 and 6) appearing in testing stage. Note that traditional OSC methods usually classify instances based on the learned embeddings.

architecture to generate synthetic instances similar to known classes. Though these methods have achieved some improvements, generating more effective out-of-class instances with complex content still needs further research [Neal et al. 2018].

2.3 Traditional Detection Models

Anomaly detection and GZSL are also related to OSC task. The goal of anomaly detection is to separate outlier instances, for example, Liu et al. [2008] proposed a non-parametric method IForest, which detected outliers with ensemble trees. However, anomaly detection follows different protocols from OSC methods, and unable to subdivide known classes. GZSL aims at classifying known and unknown classes with side information. For example, Changpinyo et al. [2016] employed manifold learning to align semantic space with visual features; Li et al. [2019] introduced the feature confusion GAN, which adopted a boundary loss to maximize the margin of known and unknown classes. However, they assume that semantic information of unknown classes is already in existence, which is incomparable with OSC methods.

3 THE ALGORITHM PIPELINE

Considering that S2OSC can handle OSC and can effectively deal with incremental OSC issues, we firstly provide the pipeline of S2OSC and the I-S2OSC in Figure 2, where the dotted frame part

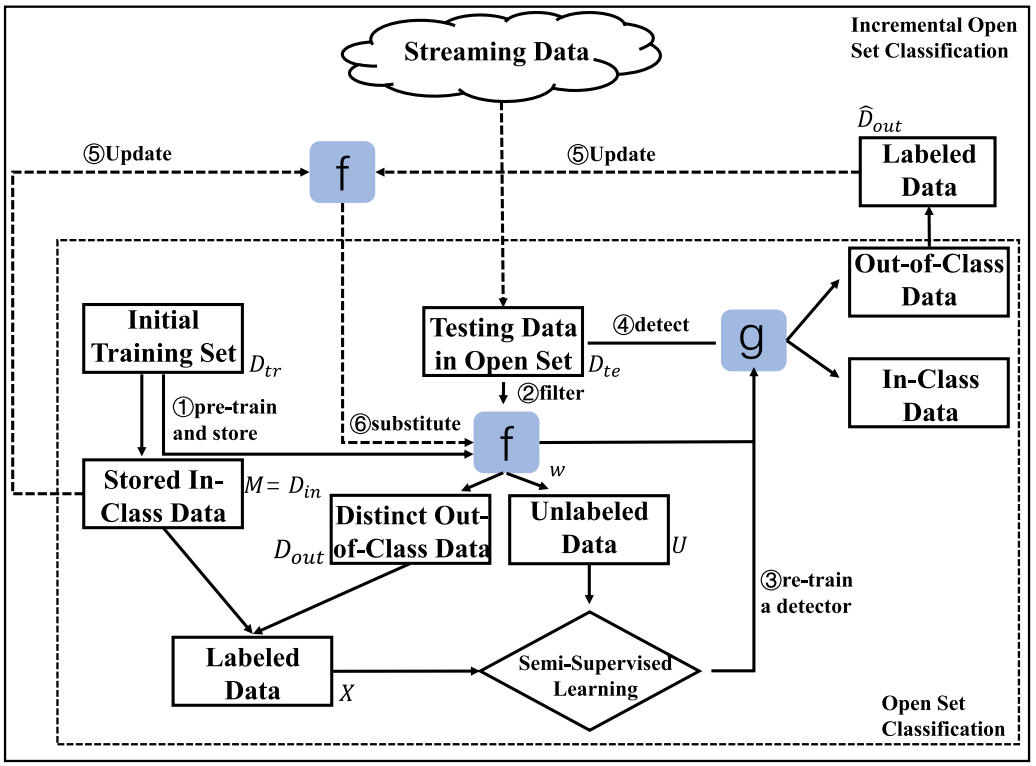


Fig. 2. Pipeline of the S2OSC and I-S2OSC.

is the S2OSC pipeline, and the solid line part is the extended I-S2OSC. Specifically, look from top to bottom inside the dotted frame. The framework includes several steps: (1) Given the initially in-class training set $\mathcal{D}_{tr}$, we carry on with two jobs: (a) pre-train an in-class classification model f; and (b) store limited in-class examples $\mathcal{D}_{in}$. (2) We receive the testing data $\mathcal{D}_{te}$ from open set, and $\mathcal{D}_{te}$ can be divided into two parts using the trained f: (a) distinct out-of-class instances $\mathcal{D}_{out}$; and (b) unlabeled data $\mathcal{U} = \{\mathcal{D}_{te} \backslash \mathcal{D}_{out}\}$. Here all instances in $\mathcal{D}_{out}$ are reduced to a unified super-class (i.e., one unknown class) as Da et al. [2014]; Liang et al. [2018]; Neal et al. [2018]; Wang et al. [2019]. (3) We possess in-class and out-of-class labeled data $\mathcal{X} = \{\mathcal{D}_{in}, \mathcal{D}_{out}\}$ and unlabeled data $\mathcal{U} = \{\mathcal{D}_{te} \backslash \mathcal{D}_{out}\}$, and develop a new detector g in a semi-supervised paradigm by considering f as teacher model simultaneously. (4) We can acquire the classification results of $\mathcal{D}_{te}$ using learned g in a transductive manner. Derived to incremental OSC scenario, we receive the testing data D^t of tth time window from the streaming data, and utilize S2OSC for OSC. (5) We query the ground-truths of potential unknown class data, and combine stored in-class data to incrementally update f^t. (6) We substitute last f^{t-1}. It is notable that g is re-trained from scratch for every time window. Following we will explain the details of S2OSC and I-S2OSC.

4 SEMI-SUPERVISED OPEN SET CLASSIFICATION (S2OSC)

In this section, we formalize the problem of OSC, and give the details of proposed S2OSC, i.e., our holistic S2OSC method, which incorporates the dominant components of OSC discussed in Section 1.

4.1 Problem Definition

Without any loss of generality, suppose we have a supervised training set $\mathcal{D}_{tr} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N_{tr}}$ at initial time, where $\mathbf{x}_i \in \mathcal{R}^d$ denotes the ith instance, and $\mathbf{y}_i \in Y = \{1, 2, \ldots, C\}$ denotes the corresponding label. Then, we receive a pool of unlabeled testing data $\mathcal{D}_{te} = \{(\mathbf{x}_j)\}_{j=1}^{N_{te}}$, where $\mathbf{x}_j \in \mathcal{R}^d$ denotes the jth instance, and label $\mathbf{y}_j \in \hat{Y} = \{1, 2, \ldots, C, C+1, \ldots, C+B\}$ is unknown. $\{1, 2, \ldots, C\}$ denotes in-class set and $\{C+1, \ldots, C+B\}$ represents out-of-class set. Therefore, OSC can be defined as follows:

Definition 1. **Open Set Classification (OSC)** With the initial training set $\mathcal{D}_{tr} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N_{tr}}$, we aim to construct a model $f : X \to Y$. Then with the pre-trained model f, OSC classifies the in-class and out-of-class instances in testing set, i.e., $\mathcal{D}_{te}$, accurately.

Following most OSC approaches Da et al. [2014]; Geng et al. [2018]; Liang et al. [2018]; Mu et al. [2017]; Neal et al. [2018]; Wang et al. [2019], we, first consider all unknown classes as a super-class for detection, then employ unsupervised clustering techniques such as k-means for subdividing (*out-of-class specifically refers to super-class in following*). Therefore, given the f and $\mathcal{D}_{te}$, we turn to build a new detector g in transductive manner for operating OSC on $\mathcal{D}_{te}$. In detail, S2OSC pre-trains a classification model f with $\mathcal{D}_{tr}$ and stores limited in-class examples $\mathcal{D}_{in}$ from $\mathcal{D}_{tr}$. f is then used for filtering distinct out-of-class instances $\mathcal{D}_{out}$ in $\mathcal{D}_{te}$. After this, we possess in-class and potential out-of-class labeled data $\mathcal{X} = \{\mathcal{D}_{in}, \mathcal{D}_{out}\}$, and unlabeled data $\mathcal{U} = \{\mathcal{D}_{te} \setminus \mathcal{D}_{out}\}$, thereby, we can develop a new detector g in a semi-supervised paradigm. Note that there are two ways to train g: (1) fine-tuning based on f directly; and (2) retraining from scratch while using f as a teacher for knowledge distillation. We select the second way considering the efficiency and effectiveness, and comparison results of the two training manners are shown in experiments. Consequently, we acquire the classification results of $\mathcal{D}_{te}$ using learned g in a transductive manner. In fact, S2OSC comprehensively considers the ideas of both discriminant and generative methods, i.e., trying to separate known classes as far as possible, while taking the potential information of unknown classes into account. Next, we will describe each part of S2OSC in specific.

4.2 Data Filtering

With the initial in-class training data $\mathcal{D}_{tr}$, we firstly develop a deep classification model f similar to many typical supervised methods:

$$\arg\min_{f} \sum_{i=1}^{N_{tr}} \ell(\mathbf{y}_i, f(\mathbf{x}_i)), \tag{1}$$

where ℓ can be any convex loss function, and we define it as cross-entropy loss for simplicity here. Meanwhile, we randomly select K examples from each class to constitute $\mathcal{D}_{in}$. f represents the deep model with fully connected prediction layers, for example, ResNet34 [He et al. 2016]. Then, we evaluate the weight of each instance in $\mathcal{D}_{te}$ by self-taught weighting function. In detail, we compute confidence score for each instance $\mathbf{x}_j$ in $\mathcal{D}_{te}$ using pre-trained model f:

$$w_j = u_j + \lambda d_j, \tag{2}$$

where λ is a fixed hyperparameter. u_j denotes statistic prediction confidence, which is done explicitly with the entropy: $u_j = -\sum_c f_c^*(\mathbf{x}_j) \log f_c^*(\mathbf{x}_j)$. d_j represents statistic distance to each in-class center, i.e., $d_j = \min(\|e_j - \mu_c\|_2^2)$, where e_j represents embeddings extracted from feature output layer of f, and $\mu_c = \frac{1}{|\mathcal{D}_{tr}^c|} \sum_{\mathbf{x} \in \mathcal{D}_{tr}^c} e_{\mathbf{x}}$ represents cth in-class center, in which $\mathcal{D}_{tr}^c$ denotes the cth class set. It is notable that highly certain out-of-class instances have larger weights, while in-class and confused instances have lower weights. In result, we can sort $\mathcal{D}_{te}$ according to w, and acquire

filtered instances set $\mathcal{D}_{out}$ with the same number K as in-class set, the corresponding super-class is C'. Therefore, we have owned in-class and out-of-class labeled data $\mathcal{X} = \{\mathcal{D}_{in}, \mathcal{D}_{out}\}$, unlabeled data $\mathcal{U} = \{\mathcal{D}_{te} \setminus \mathcal{D}_{out}\}$, and aim to develop the new detector g.

4.3 Objective Function

Inspired by Sohn et al. [2020], we adopt two semi-supervised techniques to learn g: consistency regularization and pseudo-labeling, which aim to effectively utilize unlabeled data by ensuring the consistency among different data-augmented forms. S2OSC has two contributions: (1) Pseudo-labeling threshold. For a given unlabeled instance, the pseudo-label is only retained if g produces a high confident prediction. (2) Pre-trained model teaching. For a given instance, we use the pre-trained model f for knowledge distillation by considering the predictions from known classes. Therefore, we can further separate the confused out-of-class instances with in-class instances.

Specifically, the loss function of g exclusively includes two terms: a supervised loss L_s applied to labeled data and an unsupervised loss L_u. L_s can be represented as

$$\begin{aligned} L_s &= \frac{1}{2|\mathcal{X}|}\sum_{l=1}^{|\mathcal{X}|}(\ell_{s1}(\mathbf{x}_l, \mathbf{y}_l) + \alpha\ell_{s2}(\mathbf{x}_l, f(\mathbf{x}_l))), \\ \ell_{s1} &= H_{in}(\mathbf{y}_l, g(\mathbf{x}_l)) + \mathbf{1}_{max(g(\mathbf{x}_l))\geq\tau} H_{out}(\mathbf{y}_l, g(\mathbf{x}_l)), \\ \ell_{s2} &= KL(f(\mathbf{x}_l) \| g_{\setminus C'}(\mathbf{x}_l)), \end{aligned} \tag{3}$$

where $H_{(\cdot)}(p, q) = -\sum_c p_c \log q_c$ is standard cross-entropy loss, and $KL(p\|q) = \sum_c p_c \log \frac{p_c}{q_c}$ denotes KL-divergence. α is a hyperparameter, and τ is a scalar parameter denoting the threshold. $g_{\setminus C'}(\mathbf{x}_l)$, is the prediction distribution with re-softmax except out-of-class C'. ℓ_{s1} adopts the standard cross-entropy loss, note that there may still have embedding confused known class data in $\mathcal{D}_{out}$, thus we utilize $\mathbf{1}_{max(g(\mathbf{x}_l))\geq\tau}$ term to produce a valid "one-hot" probability distribution. Meanwhile, ideally, for labeled known class data in $\mathcal{X}$, f can also produce confident probability distribution, otherwise f tends to predict uniform distribution. Thereby, ℓ_{s2} receives the soft targets from f for in-class and out-of-class examples, which aim to proceed knowledge distillation by restraining two prediction distributions. $f(\mathbf{x}_l)$ and $g_{\setminus C'}(\mathbf{x}_l)$ are with Softmax-T that sharpens distribution by adjusting its temperature T following [Hinton et al. 2015], i.e., raising all probabilities to a power of $\frac{1}{T}$ and re-normalizing.

For unlabeled data, S2OSC first obtains the pseudo-label by computing the prediction for a given unlabeled instance: $q_u = g(\mathbf{x}_u)$, and $\hat{q}_u = \arg\max(q_u)$ is the pseudo-label, which is then used to enforce the loss against model's output for an augmented version of $\mathbf{x}_u$:

$$\begin{aligned} L_u &= \frac{1}{2|\mathcal{U}|}\sum_{u=1}^{|\mathcal{U}|}\mathbf{1}_{max(q_u)\geq\tau}(\ell_{u1}(\mathbf{x}_u, \hat{q}_u) + \alpha\ell_{u2}(\mathbf{x}_u, f(\mathbf{x}_u))), \\ \ell_{u1} &= H(\hat{q}_u, g(\Phi(\mathbf{x}_u))), \\ \ell_{u2} &= KL(f(\mathbf{x}_u) \| g_{\setminus C'}(\Phi(\mathbf{x}_u))), \end{aligned} \tag{4}$$

where τ denotes threshold similar to Equation (3). Φ represents weak augmentation using a standard flip-and-shift strategy or strong augmentation using CTAugment [Berthelot et al. 2020] with Cutout technique [Devries and Taylor 2017]. In detail, we leverage the weak and strong augmentation, i.e., $\Phi(\mathbf{x})$, as follows [Sohn et al. 2020]:

— **Weak augmentation.** It adopts a standard flip-and-shift strategy. On all datasets, Images are randomly flipped horizontally with a probability of 50%, and randomly translated by up to 12.5% vertically and horizontally;

— **Strong augmentation.** It adopts CTAugment [Berthelot et al. 2020] with Cutout technique based on AutoAugment [Cubuk et al. 2019]. AutoAugment learns an augmentation strategy with reinforcement learning technique, which refers to image transformations from the Python Imaging Library,[1] and requires labeled data. CTAugment is a variant of AutoAugment, which requires no labeled data. Further details on CTAugment can be found in Berthelot et al. [2020]. Cutout is a simple regularization technique that randomly masks out square regions of input image.

Therefore, weak augmentation produces a slightly distorted version of a given image, while strong augmentation produces heavily distorted version of a given image. We employ the weak augmentation strategy considering efficiency, and compare the effectiveness of these two strategies in experiments. ℓ_{u2} in Equation (4) employs similar knowledge distillation function on unlabeled data as Equation (3).

In summary, L_u encourages the model's predictions to be low-entropy (i.e., high-confidence) on unlabeled data combining hard-label and soft-label. The loss minimized by S2OSC is: $L = L_s + \lambda_u L_u$, where λ_u is a fixed scalar hyperparameter denoting the relative weight. Consequently, the detector g can classify in-class or out-of-class instances in D_{te} by using the self-taught manner gradually, and then employ clustering for sub-dividing, i.e., our method classifies novel classes as one superclass, then adopts K-means to group the novel instances into sub-classes. The overall procedure can refer to the Algorithm 1.

ALGORITHM 1: The pseudo-code of S2OSC

Input:
Data: Initially in-class training set $\mathcal{D}_{tr}$, Open set testing data $\mathcal{D}_{te}$
Parameters: λ, λ_u, τ, α
Output:
Detector: g

Receive $\mathcal{D}_{tr}$ do:
Pre-train an in-class classification model f according to Eq. 1;
Store limited in-class examples $\mathcal{D}_{in}$ according to Eq. 2;
Receive $\mathcal{D}_{te}$ do:
Acquire distinct out-of-class instances $\mathcal{D}_{out}$ using f;
Acquire unlabeled data $\mathcal{U} = \{\mathcal{D}_{te} \setminus \mathcal{D}_{out}\}$;
Constitute the labeled data $\mathcal{X} = \{\mathcal{D}_{in}, \mathcal{D}_{out}\}$;
while stop condition is not triggered **do**
  for mini-batch **do**
    Calculate L according to Equation 3;
    Update model parameters of g using SGD;
  end for
end while

5 INCREMENTAL S2OSC (I-S2OSC)

In this section, we aim to demonstrate that S2OSC can be extended into **incremental OSC** (**IOSC**) scenario conveniently.

[1] https://www.pythonware.com/products/pil/.

5.1 Problem Definition

In real applications, we always receive the data in stream form, in which unknown classes also emerge incrementally. Thereby, IOSC is a more generalized setting, which has two characteristics: (1) Data pool. At time window t, we only get the data of current time window, i.e., $\mathcal{D}_{te}^t$, not the full amount of previous data; and (2) Unknown class continuity. At time window t, unknown classes appear partially, thereby, we need to incrementally conduct OSC, i.e., OSC needs to be performed every time after receiving the data of time window t. Specifically, the streaming data $\mathcal{D}$ can be divided into $\mathcal{D} = \{\mathcal{D}^t\}_{t=0}^T$, where $\mathcal{D}^0 = \mathcal{D}_{tr}$ is the initial training set. $\mathcal{D}^t = \{\mathbf{x}_j^t\}_{j=1}^{N_t}, t \geq 1$ is with N_t unlabeled instances, and the underlying label $\mathbf{y}_j^t \in \hat{Y}^t$ is unknown, with $\hat{Y}^t = \hat{Y}^{t-1} \cup Y^t$, where $\hat{Y}^{t-1}$ is the cumulative known classes until $(t-1)$th time window and Y^t is the unknown class set in tth time window. Therefore, we provide the definition of IOSC:

Definition 2. **Incremental Open Set Classification (IOSC)** At time $t \in \{1, 2, \ldots, T\}$, we have pre-trained model f^{t-1} and limited stored in-class examples M^{t-1} until $(t-1)$th time, then receive newly coming data pool $\mathcal{D}^t$. First, we aim to classify known and unknown classes in $\mathcal{D}^t$ as Definition 1. Then, with the labeled data from novel classes and stored data M^{t-1}, we update the model while mitigating forgetting to acquire f^t. Cycle this process until terminated.

S2OSC can be applied directly for OSC of $\mathcal{D}^t$ at tth time window, then the extra challenge is to update the model while mitigating forgetting [Ratcliff 1990] of previous in-class knowledge.

5.2 Model Update

There exist two labeling cases after OSC, i.e., manually labeling and self-taught labeling [Mu et al. 2017]. We consider first setting following most approaches [Geng et al. 2018; Mu et al. 2017; Wang et al. 2019] to avoid label noise accumulation. In detail, after known/unknown classification operator, we can achieve potential out-of-class instances to query their true labels. However, there exist catastrophic forgetting (i.e., it is obvious that the knowledge learned from the known classes will be lost when information relevant to the current novel class is incorporated) if we only use the new data to update the model.

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

$$\begin{aligned} L^t &= \sum_l \ell(\mathbf{y}_l, f^t(\mathbf{x}_l)), \\ s.t. \quad & \ell(M^{t-1}, f^t) \leq \ell(M^{t-1}, f^{t-1}). \end{aligned} \tag{5}$$

The loss term encourages the labeled unknown class examples to fine-tune f^{t-1} for better performance, while the constraint term imposes M^{t-1} for less forgetting of old in-class knowledge. We utilize directly joint optimization on M^{t-1} to optimize the variant of Equation (5). In detail, after S2OSC operator, we can achieve potential out-of-class instances for querying their true labels. Suppose at time window t, we acquire Q examples with ground-truths for novel classes, note that $Q \gg K$, where K is the number of each class instances in data filtering. And these newly labeled examples constitute $\hat{\mathcal{D}}_{out}$. Therefore, the overall loss for fine-tuning f can be relaxed as: $L = \ell(\hat{\mathcal{D}}_{out}, f^t) + \ell(M^{t-1}, f^t)$ according to the Equation (3) in Song and Tan [2019], which rephrases the constraint term as the task of better performance on M^{t-1}. Equation (5) can be done

with commonly used incremental update approach iCaLR [Rebuffi et al. 2017]:

$$L = -\left(\sum_{\mathbf{x}_i \in \hat{\mathcal{D}}_{out} \bigcup M^{t-1}} \mathbf{y}_i \log f^t(\mathbf{x}_i) + \sum_{\mathbf{x}_j \in M^{t-1}} \mathbf{q}_j \log f^t(\mathbf{x}_j) \right),$$
$$\mathbf{q}_j = f^{t-1}(\mathbf{x}_j), \tag{6}$$

where $\mathbf{q}_j$ denotes the scores calculated in the previous step. Consequently, the loss function encourages the network to output the correct class indicator (classification loss) for all labeled examples, and reproduces the scores calculated in the previous step (distillation loss) for stored in-class examples. Besides, in the memory update phase, we need to update the M^{t-1} to store key points of unknown classes. If M^{t-1} is not full, we can fill selected instances from unknown class directly. Otherwise, we remove equal instances for each known class, i.e., $\frac{|Y^t||M|}{|\hat{Y}^{t-1}||\hat{Y}^t|}$, and fill instances for each unknown class, i.e., $\frac{|M|}{|\hat{Y}^t|}$. The details of model update are shown in Algorithm 2.

ALGORITHM 2: Model Update

Input:
Data: $\hat{\mathcal{D}}_{out}$ //labeled examples of novel classes at time window t
Memory: M^{t-1} // stored examples of known classes at time window t
Model: f^{t-1} // last time model

```
for x_j ∈ M^{t-1} do
    q_j ← f^{t-1}(x_j) // store network output with pre-trained model
end for
while stop condition is not triggered do
    for mini-batch do
        Calculate L according to Equation 6;
        Update model parameters of f^t using SGD;
    end for
end while
```

Here, we adopt the replay-based methods as Rebuffi et al. [2017], with extra regularization term on parameters to consolidate previous knowledge, rather than use regularization-based methods such as **Elastic Weight Consolidation (EWC)** [Kirkpatrick et al. 2016], and **Incremental Moment Matching (IMM)** [Lee et al. 2017]. The reason is that regularization-based methods always calculate fisher information matrix [Kirkpatrick et al. 2016] for all parameters in deep network, which is hard to accomplish for CNNs. It is notable that EWC and IMM mainly experiment with shallow fully connected networks.

5.3 Time Complexity

We present the time complexity of each component for *S2OSC* and *I-S2OSC*. The common components of *S2OSC* and *I-S2OSC* are the multi-class classifier f and the detector g. For I-S2OSC, there is an additional component, i.e., model update of f. The training of f and g both involve multi-class optimization, which has time complexity $O(|v|n)$, where $|v|$ is the size of class set, n is the number of instances. Besides, the construction of f/g and the update of f all refer to the deep networks, which has time complexity $O(\sum_l F_l * C_{l-1} * C_l)$, where F represents the product of feature map and convolution kernel area, C represents the number of input/output channels, l represents the number of layers.

6 EXPERIMENTS

We validate the effectiveness of S2OSC and I-S2OSC on common OSC benchmarks, including, image and text domains (Sections 6.3, 6.9, and 6.12). Our ablation study test the contribution of each component (Section 6.4).

6.1 Datasets and Baselines

Considering that IOSC is an extension of OSC, the IOSC methods can also be applied to the setting of OSC. Therefore, we adopt commonly used OSC and IOSC datasets for validation here. In detail, we utilize three visual datasets in this article following Wang et al. [2019], i.e., CIFAR-10 [Krizhevsky et al. 2009], SVHN [Netzer et al. 2011], **Modified National Institute of Standards and Technology database (MNIST)** [LeCun et al. 1998], and two textual datasets, i.e., OTTO and SNSR datasets [Kim et al. 2020]. To validate the effectiveness of proposed approach, we compared it with existing state-of-the-art OSC and IOSC methods. First, we compare it with traditional anomaly detection and linear OSC/IOSC methods: Iforest [Liu et al. 2008], One-Class **Support Vector Machines (SVM)** (One-SVM) [Scholkopf et al. 2001], LACU-SVM (LACU) [Da et al. 2014], and SENC-MAS (SENC) [Mu et al. 2017]. Second, we compare it with recent deep methods: ODIN-CNN (**Out-of-Distribution Images in Neural Networks (ODIN)**) [Liang et al. 2018], CFO [Neal et al. 2018], **CNN-based Prototype Ensemble (CPE)** [Wang et al. 2019], and **Deep Transfer Clustering (DTC)** [Han et al. 2019]. Abbreviations in parentheses. DTC is a clustering based method for multiple unknown classes detection. One-Class-SVM is applied as an ensemble method for multi-class classification in sklearn. Note that Iforest, One-SVM, **Learning with Augmented Class with Unlabeled data (LACU)**, ODIN, **Counterfactual Open Set Learnings (CFO)**, and DTC are OSC methods, SENC and CPE are IOSC methods. All OSC baselines except Iforest can be updated incrementally using newly labeled unknown class data and memory data. The results report averaged performance and std over five random class partitions as Geng et al. [2018].

6.2 Implementation

We develop f based on convolutional network structure ResNet34 [He et al. 2016], and g based on ResNet18 [He et al. 2016]. Note that we use an identical set of hyperparameters ($\lambda = 1$, $\alpha = 0.3$, $\lambda_u = 0.2$, $\tau = 0.85$, $T(softmax-T) = 3$, $M = 2000$). In all of our models and experiments, we adopt standard SGD with Nesterov momentum [Sutskever et al. 2013], where the momentum $\beta = 0.9$. We train the initial model f as following: The number of epochs is 20, the batch size is 64, the learning rate is 0.01, and weight decay is 0.001, while train g as following: The number of epochs is 30, the batch size is 64, the learning rate is 0.005, and weight decay is 0.0005. We implement all baselines and perform all experiments based on code released by corresponding authors, and tune the parameters according to the original article to obtain the best results. For CNN based methods, we use the same network architecture and parameters during training, such as optimizer, learning rate schedule, and data pre-processing. For non-deep methods, we adopt the pre-trained Resnet34 to extract feature embeddings as the input. Our method is implemented on a Nvidia TITAN X.

6.3 Open Set Classification

To rearrange each dataset for emulating the OSC form, we randomly hold out 50% classes as initial training set, and select one class from remaining categories for testing as [Kim et al. 2020]. Moreover, we extracted 33% of the known class data into test set, so that the test set is a mixture of known and unknown classes. Here, we utilize four commonly used criteria, i.e., Accuracy, Precision, Recall, and F1 (Weighted F1), to measure the classification performance, which considers all known

Table 1. Comparison Performances of OSC

Methods	Accuracy			F1		
	CIFAR-10	SVHN	MNIST	CIFAR-10	SVHN	MNIST
Iforest	.243 ± .082	.198 ± .053	.632 ± .065	.243 ± .081	.197 ± .078	.625 ± .088
One-SVM	.260 ± .016	.195 ± .044	.537 ± .023	.223 ± .068	.102 ± .032	.520 ± .038
LACU	.325 ± .017	.193 ± .038	.695 ± .039	.326 ± .021	.091 ± .015	.681 ± .076
SENC	.215 ± .027	.184 ± .068	.358 ± .022	.171 ± .030	.124 ± .042	.302 ± .060
ODIN	.426 ± .010	.601 ± .075	.778 ± .074	.380 ± .099	.584 ± .019	.767 ± .023
CFO	.502 ± .029	.663 ± .087	.514 ± .058	.514 ± .072	.656 ± .057	.513 ± .051
CPE	.438 ± .080	.645 ± .034	.961 ± .012	.353 ± .057	.791 ± .037	.960 ± .042
DTC	.363 ± .032	.534 ± .067	.741 ± .070	.495 ± .017	.606 ± .018	.717 ± .034
S2OSC	**.847 ± .050**	**.898 ± .028**	**.985 ± .025**	**.854 ± .024**	**.901 ± .054**	**.985 ± .041**
Methods	Precision			Recall		
	CIFAR-10	SVHN	MNIST	CIFAR-10	SVHN	MNIST
Iforest	.554 ± .071	.252 ± .090	.243 ± .026	.657 ± .011	.632 ± .008	.245 ± .063
One-SVM	.474 ± .023	.286 ± .046	.260 ± .045	.616 ± .060	.537 ± .041	.274 ± .033
LACU	.394 ± .044	.331 ± .073	.325 ± .029	.676 ± .030	.695 ± .018	.363 ± .021
SENC	.420 ± .016	.253 ± .072	.215 ± .015	.448 ± .031	.358 ± .053	.211 ± .050
ODIN	.563 ± .053	.520 ± .039	.426 ± .028	.878 ± .049	.778 ± .022	.554 ± .017
CFO	.639 ± .026	.579 ± .032	.502 ± .046	.598 ± .058	.514 ± .068	.436 ± .051
CPE	.698 ± .088	.336 ± .046	.408 ± .063	.955 ± .048	.961 ± .022	.302 ± .036
DTC	.576 ± .024	.435 ± .057	.463 ± .042	.699 ± .034	.681 ± .030	.428 ± .016
S2OSC	**.972 ± .076**	**.888 ± 0.23**	**.847 ± .016**	**.986 ± .017**	**.985 ± .022**	**.799 ± .011**

and unknown classes. For example, accuracy $A = \frac{\sum_{i=1}^{|\hat{Y}|}(TP_i+TN_i)}{\sum_{i=1}^{|\hat{Y}|} TP_i+TN_i+FP_i+FN_i}$, where TP, FP, FN, TN denote the true positives, false positive, false negatives and true negatives.

Table 1 compares the classification performances of S2OSC with all baselines. We observe the following: (1) Outlier detection and linear methods perform poorly on most complex datasets, i.e., CIFAR-10, SVHN, and CINIC, which indicates that they are difficult to process high dimensional data with complex content. (2) CNN-based methods perform better than traditional OSC approaches, i.e., One-SVM, LACU, and SENC. This indicates that neural networks can provide better feature embeddings for prediction. (3) The generative method, i.e., ODIN-CNN, performs worse in our setting. Because it conducts experiments under the scenario with obvious class distribution drift, e.g., it originally trains with CIFAR-10 and tests with Image-Net to detect out-of-class instances, which is easier than our setup. Besides, ODIN-CNN performs better on simple datasets, i.e., MNIST and FASHION-MNIST, whereas performs worse on more complex datasets. (4) S2OSC consistently outperforms all baselines over various criteria by a significant margin. For example, in all datasets, S2OSC provides at least 20% improvements than baselines. This indicates the effectiveness of semi-supervised operation for mitigating embedding confusion.

Figure 3 shows feature embedding results using T-SNE with the similar setting as Figure 1. Clearly, the Figure 3(b) shows that the output embeddings of S2OSC have learned distinct groups, which are much better than original embeddings and corresponding embeddings of other deep methods in Figure 1. This validates that instances of unknown classes are well separated from other known clusters, which can benefit for unknown class detection in result.

6.4 Ablation Study

S2OSC has several variant designs. Therefore, in this subsection, we aim to analyze following questions: (1) Why choose the cross-entropy loss function for training g, without considering the

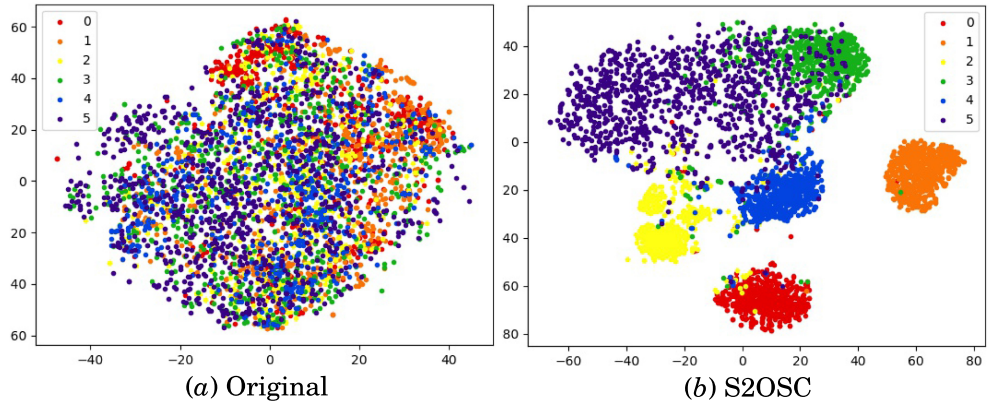


Fig. 3. T-SNE Visualization on CIFAR-10 dataset. (a) original feature space; (b) learned embeddings by proposed S2OSC.

large margin based loss function in traditional OSC methods [Da et al. 2014; Geng et al. 2018; Wang et al. 2019], such as triplet loss [Kulis 2013], largin margin softmax [Liu et al. 2016]? (2) Why not directly perform semi-supervised learning based on the pre-trained model f to obtain g, i.e., fine-tune the f to acquire g? (3) Why not utilize only labeled data to calculate supervised loss for training g?

Therefore, we conduct extra baselines and ablation experiments with the same setting as S2OSC. In detail, the baselines are as follows:

- **S2OSC-S** calculates loss L_u including, strongly-augmented instances instead of weakly-augmented version.
- **S2OSC-LM** utilizes the common triplet loss to train g for S2OSC.
- **S2OSC-FN** indicates that S2OSC directly fine-tunes based on f to obtain g.
- **S2OSC-U** represents that S2OSC with the unsupervised term removed.
- **S2OSC-Random** randomly samples the same number (i.e., 300) of novel class data as S2OSC.
- **S2OSC-All** gives all the test data pseudo labels.
- **S2OSC-KD** removes the knowledge distillation term in S2OSC.
- **CPE-SSL** studies other SOTA open set learning method (i.e., CPE) in conjunction with semi-supervised learning.

Table 2 records the results, with the best results in bold. We observe that S2OSC outperforms all baselines and variant methods on three datasets with different criteria. This validates the following: (1) The performance of strong augmentation is worse than weak augmentation, which indicates that the strong augmentation may introduce more noises that are difficult to train. (2) Large-margin loss is not suitable for training g, even only with one unknown class. Traditional OSC methods adopt large margin loss to restrict intra-class and inter-class distance property, and then detect novel class by identifying outliers. All of these approaches have a strong assumption that the embeddings or predictions learned by pre-trained model for out-of-class instances are apart from in-class ones. However, this assumption will fail on the data with complex content. Therefore, there is no need to use large margin loss here, and a unified classifier f is more convenient for knowledge distillation in training g. More importantly, in our semi-supervised paradigm, for training detector g, we first unify all unknown classes in $\mathcal{D}_{te}$ as a super-class. If we utilize large margin loss function here, it will also reduce the intra-class distance of super-class, and affect g's training considering the semantic confusion and subsequent clustering operation. (3) The model f

Table 2. Ablation Study about Variants of S2OSC

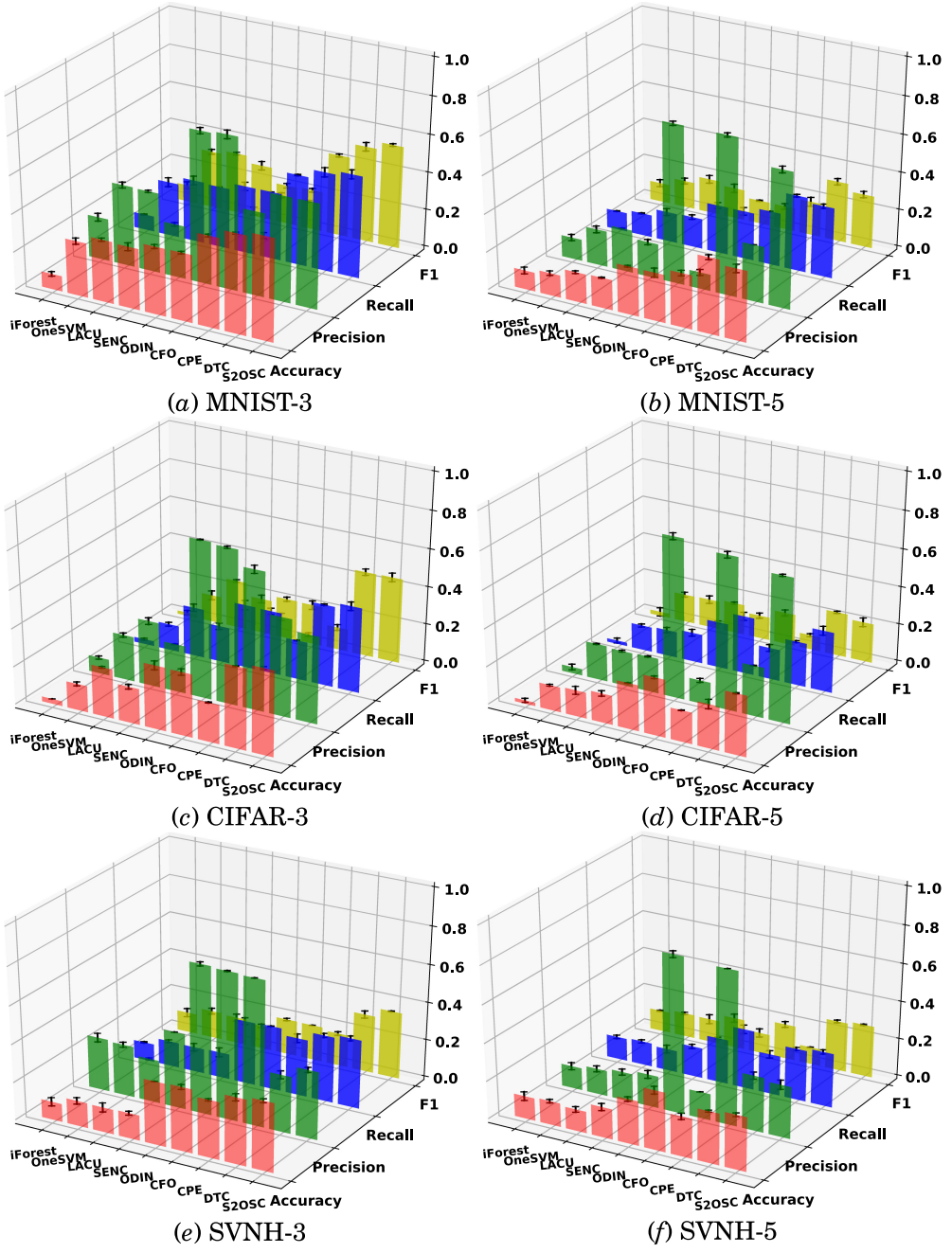
Methods	Accuracy			F1		
	MNIST	CIFAR-10	SVHN	MNIST	CIFAR-10	SVHN
CPE-SSL	.980 ± .059	.469 ± .048	.598 ± .037	.981 ± .048	.398 ± .025	.832 ± .040
S2OSC-KD	.963 ± .054	.783 ± .044	.817 ± .068	.967 ± .018	.844 ± .031	.852 ± .027
S2OSC-Random	.374 ± .033	.799 ± .025	.583 ± .026	.140 ± .019	.829 ± .029	.469 ± .013
S2OSC-ALL	.368 ± .054	.377 ± .019	.354 ± .052	.135 ± .024	.149 ± .021	.117 ± .006
S2OSC-S	.834 ± .034	.812 ± .016	.748 ± .013	.792 ± .040	.771 ± .042	.666 ± .031
S2OSC-LM	.793 ± .23	.675 ± .014	.589 ± .039	.754 ± .017	.608 ± .035	.505 ± .043
S2OSC-FN	.891 ± .037	.817 ± .029	.880 ± .044	.846 ± .024	.777 ± .033	.834 ± .064
S2OSC-U	.886 ± .014	.786 ± .022	.812 ± .015	.853 ± .037	.813 ± .012	.833 ± .038
S2OSC	**.985 ± .045**	**.847 ± .017**	**.898 ± .011**	**.985 ± .048**	**.854 ± .029**	**.901 ± .037**
Methods	Precision			Recall		
	MNIST	CIFAR-10	SVHN	MNIST	CIFAR-10	SVHN
CPE-SSL	.426 ± .012	.737 ± .033	.389 ± .016	.338 ± .015	.983 ± .019	.975 ± .048
S2OSC-KD	.821 ± .038	.783 ± .047	.769 ± .029	.724 ± .029	.789 ± .038	.837 ± .036
S2OSC-Random	.372 ± .021	.800 ± .025	.397 ± .020	.204 ± .017	.804 ± .021	.454 ± .043
S2OSC-ALL	.368 ± .014	.357 ± .016	.159 ± .013	.198 ± .024	.206 ± .017	.102 ± .007
S2OSC-S	.758 ± .010	.736 ± .019	.602 ± .036	.834 ± .036	.812 ± .042	.748 ± .054
S2OSC-LM	.727 ± .013	.561 ± .026	.465 ± .013	.793 ± .022	.675 ± .025	.589 ± .014
S2OSC-FN	.807 ± .012	.746 ± .017	.795 ± .027	.891 ± .037	.817 ± .036	.880 ± .046
S2OSC-U	.765 ± .025	.758 ± .032	.876 ± .042	.786 ± .056	.886 ± .054	.812 ± .013
S2OSC	**.847 ± .028**	**.972 ± .029**	**.888 ± .030**	**.799 ± .048**	**.986 ± .028**	**.985 ± .026**

is pre-trained with in-class data, performing semi-supervised re-training based on the pre-trained model f has two limitations: (a) the number K (examples in each class) of $\mathcal{X}$ is much smaller than that of f. Therefore, it is more inclined to classify the known classes and ignore the unknown classes if we fine-tune based on f and (b) the model f cannot be regarded as teacher network for knowledge distillation any more. (4) SSL can effectively improve the novel class detection, i.e., CPE-SSL/S2OSC performs better than CPE/S2OSC-U on all datasets, and our method is superior to other baselines, which validate the effectiveness of unlabeled data. The reasonable explanation is that the number K of examples in each class is limited, thus supervised training may lead to overfitting, while unlabeled data can enlarge the training data and contribute to the learning process. (5) S2OSC performs better than the S2OSC-Random, which validates the effectiveness of our detector g for detecting novel class. (6) S2OSC is superior to the S2OSC-All, for the reason that the embedding confused instances will introduce additional noise labels if we give all the test data pseudo labels for training g. (7) S2OSC-KD performs worse, which indicates that the effectiveness of knowledge distillation using pre-trained model f. (8) Moreover, we also adopt the confidence criteria to select the in-class examples for constructing D_{in}, i.e., we select the examples by prediction confidence, which aims at validating different selections criteria. The F1 result of selecting instances by confidence is 85.4% on CIFAR-10, which is similar to random selection.

6.5 Influence of Unknown Class Number

To explore the influence of unknown class number, we conduct more experiments. In detail, we randomly hold out 50% classes as initial training data, and tune the unknown class ratio in {60%, 100%} of remaining classes (Section 6.3 has already given the results of 20%). Besides, we extract 33% of the known class data into test set, so that the test set is a mixture of known and unknown classes.

Figure 4 records the experiment results (mean and std) of three typical datasets, which reveal the following: (1) With the number of unknown classes increases, performances of all approaches decrease. This indicates that, once multiple unknown classes emerge in testing phase, the problem of embedding confusion will exacerbate, making OSC more complicated. (2) The precision of S2OSC is not high, whereas the recall of high precision model (e.g., CPE) is not very good, this indicates



(a) MNIST-3 (b) MNIST-5

(c) CIFAR-3 (d) CIFAR-5

(e) SVNH-3 (f) SVNH-5

Fig. 4. Classification performance of S2OSC and baseline methods on MNIST, CIFAR-10, and SVNH. Dataset-p denotes with p unknown classes. The X-axis represents different OSC methods, the Y-axis denotes the metrics, and the Z-axis gives the performance.

Table 3. Comparison of Movel Class Detection (F_{out}) with Different Number of Unknown Classes (SVM/Our Denotes One-SVM/S2OSC)

	One Class			Three Classes			Five Classes		
	MNIST	CIFAR	SVHN	MNIST	CIFAR	SVHN	MNIST	CIFAR	SVHN
Iforest	.065 ± .010	.251 ± .032	.230 ± .021	.292 ± .047	.139 ± .041	.224 ± .037	.407 ± .060	.224 ± .031	.219 ± .014
SVM	.461 ± .035	.279 ± .016	.300 ± .028	.494 ± .093	.525 ± .057	.519 ± .060	.488 ± .057	.479 ± .067	.464 ± .044
LACU	.223 ± .026	.190 ± .009	.251 ± .018	.381 ± .058	.449 ± .026	.465 ± .102	.514 ± .067	.604 ± .036	.594 ± .019
SENC	.005 ± .001	.110 ± .003	.007 ± 0	.530 ± .002	.554 ± .040	.472 ± .075	.690 ± .004	.689 ± .003	.706 ± .018
ODIN	.390 ± .085	.031 ± .003	.219 ± .010	.006 ± .001	.006 ± .001	.006 ± .001	.130 ± .002	.130 ± .001	.130 ± .002
CFO	.278 ± .013	.275 ± .017	.285 ± .012	.296 ± .110	.442 ± .036	.452 ± .096	.543 ± .002	.556 ± .010	.596 ± .072
CPE	.923 ± .015	.383 ± .022	.563 ± .053	.424 ± .055	.385 ± 0.026	.338 ± .094	.619 ± .040	.557 ± .050	.577 ± .119
DTC	.249 ± .041	.252 ± .028	.142 ± .016	.074 ± .009	.241 ± 0.176	.309 ± .084	.553 ± .131	.593 ± .002	.756 ± .055
S2OSC	**.987 ± .019**	**.853 ± .083**	**.932 ± .032**	**.629 ± 013**	**.632 ± 0.024**	**.666 ± .117**	**.974 ± .001**	**.858 ± .028**	**.830 ± .029**

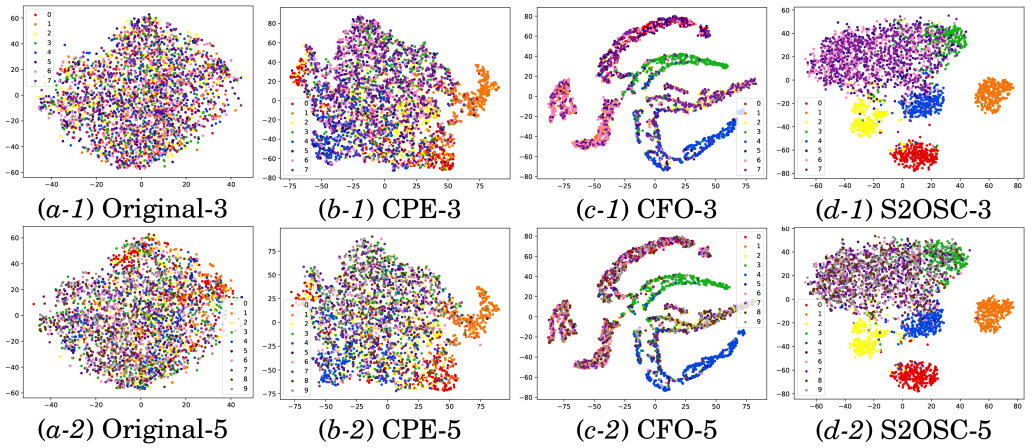


Fig. 5. T-SNE Visualization on CIFAR-10 dataset. Method-p denotes with p unknown classes. (a) original feature space; (b) learned representations by discriminative method CPE; (c) learned representations by generative detection method CFO; and (d) learned representations by proposed S2OSC.

that most unknown classes are divided into known classes. The F_{out} of novel class detection in Table 3, also validates this phenomenon. (3) Under three unknown classes scenario, S2OSC still outperforms all baselines on various criteria except recall. Yet the recall of S2OSC is competitive with other baselines under five unknown classes, i.e., S2OSC is lower than several baselines on MNIST dataset, and lower than CPE of precision on other two datasets.

Figure 5 shows feature embedding results using T-SNE. The figures in upper row are T-SNE results with three unknown classes (i.e., 5, 6, and 7), and figures in bottom row are with five unknown classes (i.e., 5, 6, 7, 8, 9, and 10). Clearly, S2OSC can obviously distinguish between known and unknown classes, i.e., instances from unknown classes are well separated from known clusters comparing with other baselines, which benefits unknown class detection in result.

For further measuring the discrimination of known and unknown classes, we utilize another criterion in [Wang et al. 2019], which treats OSC as a binary classification problem, placing emphasis on novel class detection. In detail, we consider all known classes as negative and all unknown classes as positive. $F_{out} = \frac{2TP}{2TP+FP+FN}$ is F1 of unknown classes, TP, FP, FN, TN denote the true positives, false positive, false negatives, and true negatives. Table 3 represents the results, with the best results in bold. We can observe that novel class detection of S2OSC is significantly higher than other methods on various settings and that ODIN is difficult to handle multiple unknown classes. An interesting phenomenon is that the F1 of three unknown classes is lower than that of one and

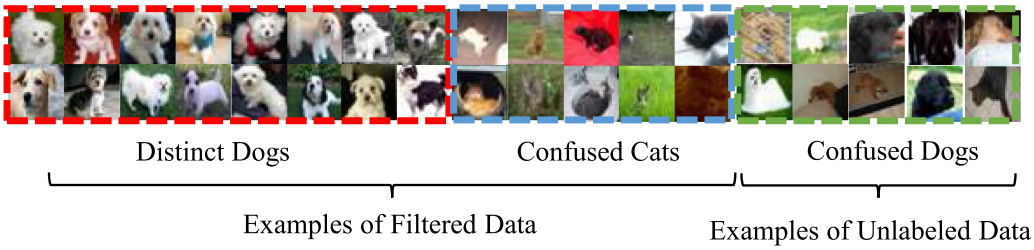


Fig. 6. Examples of filtered instances with large weights w and unselected out-of-class instances by proposed S2OSC.

five unknown classes. This is because one unknown class does not have the problem of intra-class confusion existing in multiple unknown classes, thus performs better. On the other hand, with the number of unknown classes increases, the number of embedding confused instances in the filtered set decrease, thus it is more conducive to the training of g and improves performance.

6.6 Case Study

We also exhibit some examples of distinct and confused instances for display. Here, we consider one unknown class case on CIFAR-10 dataset, in which the known class set includes: "airplane, cat, deer, automobile, truck", and the unknown class is "dog". Then we sort $\mathcal{D}_{te}$ according to the weights w, and select instances according to the ranking. As shown in Figure 6, we observe the following: (1) most of the distinct instances are dogs, but still include few known class data, for example, instances from cat. It can be seen from the examples that many confused cats are outliers, which are difficult to distinguish; (2) most distinct dogs have more diagnostic characteristics, for example, the images with full-body shot; and (3) unselected dogs are ambiguous, for example, dogs with only head or unclear dog images. Thus, combining labeled data with unlabeled loss is helpful for training detector g.

6.7 Large-Scale OSC

To validate the effectiveness of our proposed method on large-scale OSC setting, i.e., dataset with large-scale classes. We conduct more experiments on CIFAR-50 following Geng et al. [2018]. In detail, we adopt the split class case following Geng et al. [2018], which holds out 10 classes as out-of-class for testing, and leaves the remaining classes as the initial training set. The experimental setup is same as the case of multiple unknown classes. Table 4 records the results, with the best results in bold. We can observe the following: (1) the performances of all methods decrease rapidly facing large-scale OSC and (2) we acquire similar results to other setups that S2OSC consistently outperforms all baselines on various criteria except F1, which ranks runner-up. For example, S2OSC provides at least 10% improvements of accuracy than other baselines. This shows that S2OSC can well perform OSC on different class scales.

6.8 Influence of Filter Size

Figure 7 indicates the influence of important parameter K (filtering size), i.e., we tune the size of $K = \{50, 300, 1,000, 2,000\}$. The results reveal that, at first, different criteria improve with the increase of filtered instances, whereas, after filtering size exceeds a threshold (i.e., around 300), the performance starts to decrease. For example, on CIFAR-10, the accuracy on 300 filtering is about 84.7%, yet the performance decreases after 300 filterings, which could attribute to the introduction of embedding confused instances with the increase of K.

Table 4. Comparison of Large-Scale OSC with Multiple Unknown Classes

Methods	Iforest	One-SVM	LACU	SENC	S2OSC
Accuracy	.006 ± 0	.009 ± .001	.008 ± .001	.007 ± .001	**.302 ± .006**
Precision	.709 ± 0	.703 ± .003	.195 ± .006	.179 ± .003	**.791 ± .002**
Recall	.006 ± 0	.009 ± .001	.008 ± .001	.007 ± .002	**.302 ± .003**
F1	.006 ± 0	.008 ± .002	.008 ± .002	.007 ± .003	.165 ± .004
F_{out}	.649 ± 0	.614 ± .007	.691 ± .007	.706 ± .005	**.940 ± .008**
Methods	ODIN	CFO	CPE	DTC	S2OSC
Accuracy	.163 ± .007	.147 ± .002	.176 ± .003	.181 ± .003	**.302 ± .006**
Precision	.340 ± .006	.436 ± .004	.245 ± .007	.248 ± .003	**.791 ± .002**
Recall	.163 ± .004	.147 ± .003	.176 ± .006	.181 ± .004	**.302 ± .003**
F1	.147 ± .003	.125 ± .006	.165 ± .004	**.168 ± .006**	.165 ± .004
F_{out}	.917 ± .007	.898 ± .005	.917 ± .006	.917 ± .005	**.940 ± .008**

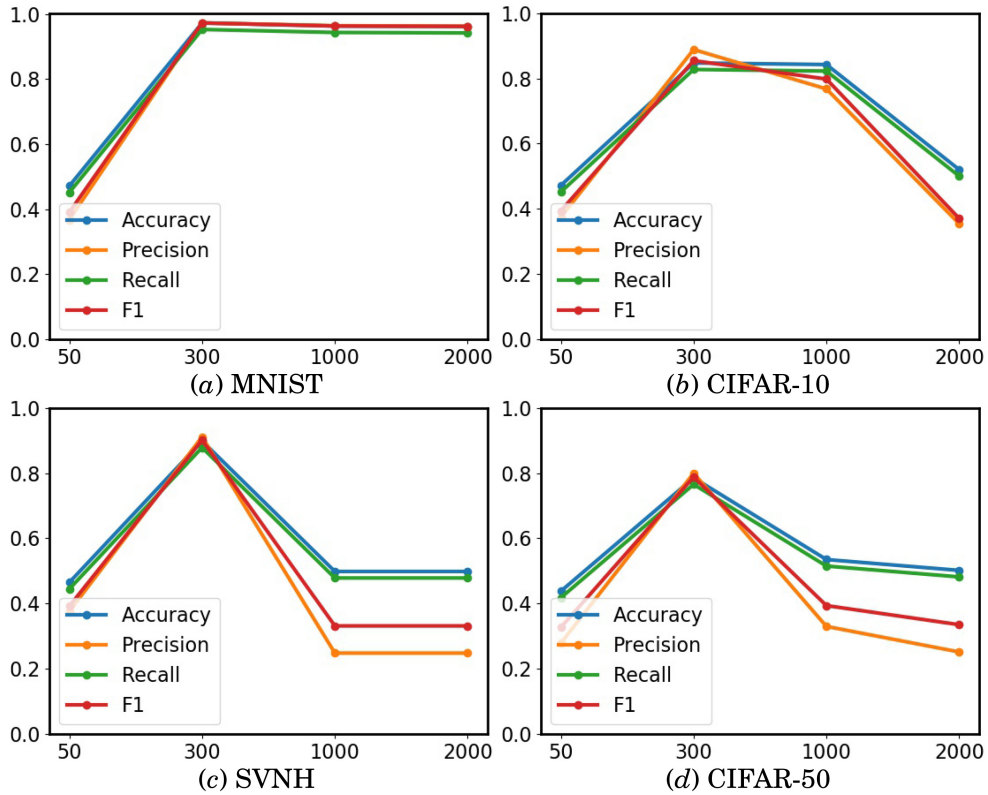


Fig. 7. Classification performance with various number of filtered out-of-class instances.

6.9 Incremental Open Set Classification

Furthermore, we rearrange instances in each dataset to emulate a streaming form with incremental unknown classes as Wang et al. [2019]. We utilize the same four criteria, i.e., average Accuracy, average Precision, average Recall, and average F1, over various data pools to measure the

Table 5. Comparison Performance of IOSC

Methods	Average Accuracy			Average F1		
	CIFAR	SVHN	MNIST	CIFAR	SVHN	MNIST
Iforest	.221 ± .032	.170 ± .016	.606 ± .023	.217 ± .013	.162 ± .025	.595 ± .034
One-SVM	.218 ± .038	.167 ± .013	.471 ± .031	.169 ± .040	.085 ± .021	.476 ± .027
LACU	.199 ± .023	.149 ± .028	.171 ± .026	.137 ± .014	.052 ± .012	.088 ± .037
SENC	.197 ± .027	.156 ± .020	.296 ± .018	.145 ± .012	.100 ± .013	.251 ± .015
ODIN	.334 ± .030	.625 ± .018	.853 ± .033	.284 ± .035	.593 ± .024	.850 ± .056
CFO	.306 ± .015	.485 ± .014	.745 ± .012	.304 ± .027	.472 ± .019	.722 ± .051
CPE	.368 ± .009	.695 ± .019	**.961 ± .043**	.343 ± .019	.701 ± .028	**.960 ± .073**
DTC	.393 ± .007	.514 ± .015	.711 ± .022	.445 ± .018	.566 ± .032	.717 ± .014
I-S2OSC	**.660 ± .013**	**.771 ± .029**	.926 ± .027	**.609 ± .055**	**.732 ± .017**	.913 ± .030
Methods	Average Precision			Average Recall		
	CIFAR	SVHN	MNIST	CIFAR	SVHN	MNIST
Iforest	.503 ± .055	.226 ± .013	.221 ± .009	.621 ± .035	.607 ± .060	.225 ± .016
One-SVM	.405 ± .032	.216 ± .037	.219 ± .023	.672 ± .050	.472 ± .011	.210 ± .014
LACU	.332 ± .019	.133 ± .060	.200 ± .013	.266 ± .031	.172 ± .008	.143 ± .023
SENC	.342 ± .022	.207 ± .027	.198 ± .016	.390 ± .016	.297 ± .014	.197 ± .024
ODIN	.781 ± .074	.456 ± .033	.334 ± .009	.920 ± .027	.853 ± .036	.372 ± .023
CFO	.659 ± .048	.307 ± .010	.306 ± .015	.803 ± .024	.745 ± .028	.285 ± .012
CPE	.619 ± .063	.427 ± .062	.368 ± .036	**.965 ± .010**	**.961 ± .054**	.332 ± .016
DTC	.586 ± .010	.536 ± .049	.394 ± .018	.800 ± .060	.711 ± .041	.469 ± .022
I-S2OSC	**.818 ± .037**	**.597 ± .034**	**.661 ± .021**	.893 ± .020	.922 ± .023	**.509 ± .028**

performance following Wang et al. [2019], which aims at calculating the overall performance for streaming data.

Moreover, in IOSC, we need to update model with the labeled instances of novel classes after detecting. Different from re-training with the entire previous in-class data, incremental model update aims to fine-tune the model only referring limited data from known classes. Therefore, the catastrophic forgetting phenomenon becomes an obstacle [Ratcliff 1990]. To validate the catastrophic forgetting of f, we calculate the performance on forgetting profile of different learning algorithms as Chaudhry et al. [2018], which defines the difference between maximum knowledge gained of emerging classes on a particular window throughout the learning process and what we currently have about it. The lower forgetting the better.

Table 5 compares the classification performance of I-S2OSC with all baselines on streaming data. Table 6 compares the forgetting performance. "N/A" denotes no result considering that Iforest has no update process. We observe the following: (1) Comparing with results in Table 1, most average classification metrics of deep methods have improved while metrics of linear methods declined, which indicates that deep models can still effectively distinguish known classes for streaming data, that can further benefit OSC. (2) I-S2OSC is superior to other baselines over accuracy and F1 metrics except for the MNIST dataset, and performs well on other two metrics. But I-S2OSC is not as obvious as the effect under OSC setting. These phenomenons are generated since we uniformly set K to 300, with the increase of emerging classes, the number of inclusive in-class in filtering data also increases, which will affect the training of g. Thereby the value of K needs to be tuned carefully. (3) I-S2OSC has the smallest forgetting except MMIST dataset by considering exemplary regularization, which benefits to preserve known class knowledge.

Table 6. Forgetting Measure over Streaming Data

Methods	Forgetting				
	Iforest	One-SVM	LACU	SENC	I-S2OSC
CIFAR-10	N/A	.202 ± .005	.127 ± .007	.172 ± .003	**.117 ± .005**
SVHN	N/A	.243 ± .009	.330 ± .012	.249 ± .009	**.123 ± .001**
MNIST	N/A	.141 ± .008	.080 ± .002	.061 ± .004	.037 ± 0
Methods	Forgetting				
	ODIN	CFO	CPE	DTC	I-S2OSC
CIFAR-10	.132 ± .002	.128 ± .001	.118 ± .002	.120 ± .003	**.117 ± .005**
SVHN	.168 ± .005	.130 ± .005	.124 ± .001	.159 ± .007	**.123 ± .001**
MNIST	.049 ± .002	.040 ± .003	**.033 ± .004**	.044 ± .006	.037 ± 0

Table 7. Execution Time Comparison of OSC

Methods	Training (hours)			Testing (hours)		
	CIFAR-10	SVHN	MNIST	CIFAR	SVHN	MNIST
ODIN	1.210 ± .030	1.190 ± .020	1.200 ± .030	.050 ± .005	.033 ± .003	.034 ± .005
CFO	1.680 ± .070	1.650 ± .030	1.680 ± .040	.056 ± .003	.054 ± .002	.054 ± .005
CPE	1.300 ± .050	1.210 ± .030	1.380 ± .050	.042 ± .005	.040 ± .004	.042 ± .002
DTC	.840 ± .040	.740 ± .060	.760 ± .040	.033 ± .002	.030 ± .002	.034 ± .004
S2OSC	**.510 ± .030**	**.450 ± .050**	**.500 ± .030**	**.017 ± .004**	**.012 ± .005**	**.012 ± .006**

Table 8. Execution Time Comparison of IOSC

Methods	Training (hours)			Testing (hours)		
	CIFAR-10	SVHN	MNIST	CIFAR	SVHN	MNIST
ODIN	9.790 ± .480	9.840 ± .226	10.320 ± .120	.061 ± .005	.060 ± .006	.054 ± .006
CFO	13.610 ± .670	13.380 ± .262	13.590 ± .520	.065 ± .003	.062 ± .004	.060 ± .006
CPE	10.020 ± .554	9.950 ± .411	10.320 ± .351	.062 ± .005	.058 ± .003	.058 ± .004
DTC	7.580 ± .310	7.020 ± .584	6.980 ± .390	.036 ± .003	.032 ± .003	.030 ± .004
I-S2OSC	**1.050 ± .440**	**1.010 ± .221**	**1.020 ± .254**	**.017 ± .002**	**.012 ± .006**	**.013 ± .005**

6.10 Execution Time

We conduct more experiments to explore the running time (training and test) in selected experiments (with the same data setting as Section 6.3). Note that the base models used by traditional linear approaches (i.e., Iforest, One-SVM, LACU, and SENC) are non-deep structures, so we only compared our methods with deep approaches. Table 7 records the results of OSC, and Table 8 records the results of IOSC. The unit is hour. The results reveal that our methods use less time than the comparison methods for both training and testing time, under OSC and IOSC settings. The reason is that ODIN adopts the modified softmax operator, which is with similar deep structure to our method, however our methods converge faster. Besides, CFO is a generative method, which needs to iteratively generate adversarial examples during the training phase. CPE is a discriminant method, employing deep embedding output to calculate the classification loss, which is costly. DTC is a clustering approach, which requires multiple clustering operations. Therefore, these three methods are with higher computational complexity.

Table 9. T-test Performances (i.e., p-value) of OSC and IOSC

Methods	Open set classification			Incremental open set classification		
	CIFAR-10	SVHN	MNIST	CIFAR	SVHN	MNIST
ODIN	1.58 ± .100	2.23 ± .174	1.51 ± .016	1.58 ± .090	2.14 ± .177	1.49 ± .170
CFO	1.13 ± .140	2.42 ± .135	1.62 ± .140	1.11 ± .280	2.37 ± .123	1.58 ± .140
CPE	1.46 ± .350	2.86 ± .314	1.56 ± .282	1.64 ± .140	2.32 ± .102	1.41 ± .210
DTC	1.29 ± .410	2.71 ± .052	1.38 ± .130	1.24 ± .390	2.62 ± .069	1.29 ± .140

Table 10. T-test Performances of OSC and IOSC

Methods	Open set classification											
	Accuracy			Precision			Recall			F1		
	CIFAR-10	SVHN	MNIST	CIFAR	SVHN	MNIST	CIFAR-10	SVHN	MNIST	CIFAR	SVHN	MNIST
ODIN	.003	.003	.010	.002	.000	.000	.023	.000	.000	.001	.001	.001
CFO	.000	.011	.000	.002	.000	.000	.000	.000	.000	.001	.006	.000
CPE	.002	.001	.008	.015	.000	.000	.050	.025	.000	.000	.044	.005
DTC	.000	.001	.005	.001	.000	.000	.000	.000	.000	.000	.001	.001
Methods	Incremental Open set classification											
	Accuracy			Precision			Recall			F1		
	CIFAR-10	SVHN	MNIST	CIFAR	SVHN	MNIST	CIFAR-10	SVHN	MNIST	CIFAR	SVHN	MNIST
ODIN	.000	.002	.041	.008	.007	.000	.036	.049	.003	.001	.001	.061
CFO	.000	.000	.000	.010	.000	.000	.008	.001	.000	.001	.000	.005
CPE	.000	.019	.298	.009	.014	.000	.005	.314	.001	.001	.017	.361
DTC	.000	.000	.000	.000	.015	.000	.014	.001	.024	.008	.001	.001

6.11 Significance Analysis

To verify the significance of the results, we conduct the t-test [Vovk and Wang 2012], i.e., p-values, comparing our methods with the deep approaches (with the same data setting as Section 6.3). Table 9 records the results, and we find that our proposed S2OSC and I-S2OSC indeed outperform other algorithms significantly on both OSC and IOSC, e.g., the p-value of S2OSC/I-S2OSC are mostly lower than 0.05 comparing with other deep models, except MNIST under IOSC scenario.

6.12 Experiments on Text Datasets

We add more experiments on other domain datasets, i.e., OTTO and SNSR, for verifying the effectiveness of our methods. Specifically, the OTTO dataset from Kaggle and the SNSR dataset from UCI repository are commonly chosen for novelty detection [Kim et al. 2020]. OTTO dataset contains 61,878 examples, and 93 classes, and belongs to the E-commerce domain. SNSR dataset contains 58,509 examples and 48 classes, and belongs to the Electric Currents domain. The streaming data simulation and other process follow [Kim et al. 2020; Wang et al. 2019]. Tables 10 and 11 compare the detection performance of S2OSC/I-S2OSC with all baseline methods under the open set scenarios and incremental open set scenarios, respectively. Table 12 compares the forgetting performance of I-S2OSC with all baseline methods. The results validate that S2OSC/I-S2OSC can still perform superior to comparing methods on novelty detection task with less forgetting, on other domain datasets.

6.13 Parameter Sensitivity

The important parameters in S2OSC include λ, α, λ_u, τ, $T(softmax - T)$, which are introduced in implements. To explore the parameter sensitivity, we tune λ in $\{0.2, 0.4, 0.6, 0.8, 1\}$, α in $\{0.2, 0.4, 0.5, 0.8, 1\}$, λ_u in $\{0.2, 0.4, 0.6, 0.8, 1\}$, τ in $\{0.2, 0.4, 0.6, 0.8, 1\}$, $T(softmax - T)$ in $\{2, 3, 4\}$,

Table 11. Comparison Performance of IOSC on Text Datasets

Methods	Accuracy		F1		Precision		Recall	
	OTTO	SNSR	OTTO	SNSR	OTTO	SNSR	OTTO	SNSR
Iforest	.110 ± .036	.153 ± .059	.214 ± .066	.186 ± .043	.146 ± .036	.114 ± .032	.063 ± .017	.105 ± .032
One-SVM	.191 ± .051	.182 ± .058	.237 ± .066	.139 ± .032	.136 ± .007	.109 ± .034	.127 ± .041	.113 ± .027
LACU	.147 ± .051	.102 ± .029	.106 ± .031	.074 ± .013	.137 ± .042	.108 ± .018	.109 ± .025	.076 ± .019
SENC	.097 ± .017	.117 ± .018	.103 ± .018	.103 ± .023	.097 ± .019	.112 ± .017	.080 ± .020	.063 ± .030
ODIN	.173 ± .019	.215 ± .059	.264 ± .066	.218 ± .068	.179 ± .046	.109 ± .009	.166 ± .039	.133 ± .029
CFO	.285 ± .057	.179 ± .028	.172 ± .028	.279 ± .081	.227 ± .051	.233 ± .019	.182 ± .031	.185 ± .027
CPE	.285 ± .038	.337 ± .031	.306 ± .017	.319 ± .063	.332 ± .040	.214 ± .051	.227 ± .049	.179 ± .022
DTC	.365 ± .053	.352 ± .032	.421 ± .069	.333 ± .063	.356 ± .048	.171 ± .009	.249 ± .011	.294 ± .028
I-S2OSC	**.529 ± .047**	**.453 ± .019**	**.471 ± .008**	**.442 ± .073**	**.438 ± .047**	**.411 ± .042**	**.364 ± .071**	**.351 ± .032**

Table 12. Forgetting Measure over Streaming Data on Text Datasets

Methods	Forgetting				
	Iforest	One-SVM	LACU	SENC	I-S2OSC
OTTO	N/A	.054 ± .002	.098 ± .002	.078 ± .001	**.028 ± .001**
SNSR	N/A	.049 ± .002	.065 ± .001	.061 ± .002	**.009 ± .001**
Methods	Forgetting				
	ODIN	CFO	CPE	DTC	I-S2OSC
OTTO	N/A	.045 ± .002	.040 ± .003	.055 ± .003	**.028 ± .001**
SNSR	N/A	.028 ± .003	.017 ± .003	.027 ± .002	**.009 ± .001**

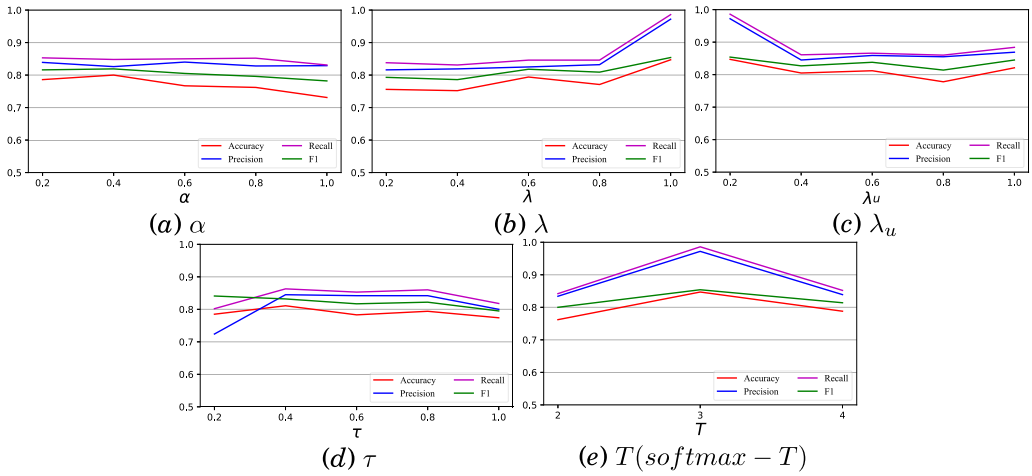


Fig. 8. Influence of the parameters λ, α, λ_u, τ and $T(softmax - T)$ on the CIFAR-10 datasets.

respectively. Note that we adjust each parameter while fixing other parameters as implements. We experiment on the challenge CIFAR-10 dataset and record the results in Figure 8. The experiment found that our parameter settings in implements are all optimal.

6.14 Convergence Analysis

To investigate the convergence of S2OSC and I-S2ISC empirically. We record the objective function value and the classification performance in each iteration (i.e., each batch). The Figure 9(a), (b), (c), and (d) record the results of S2OSC on CIFAR-10 dataset, and Figure 9(e), (f), (g), and (h)

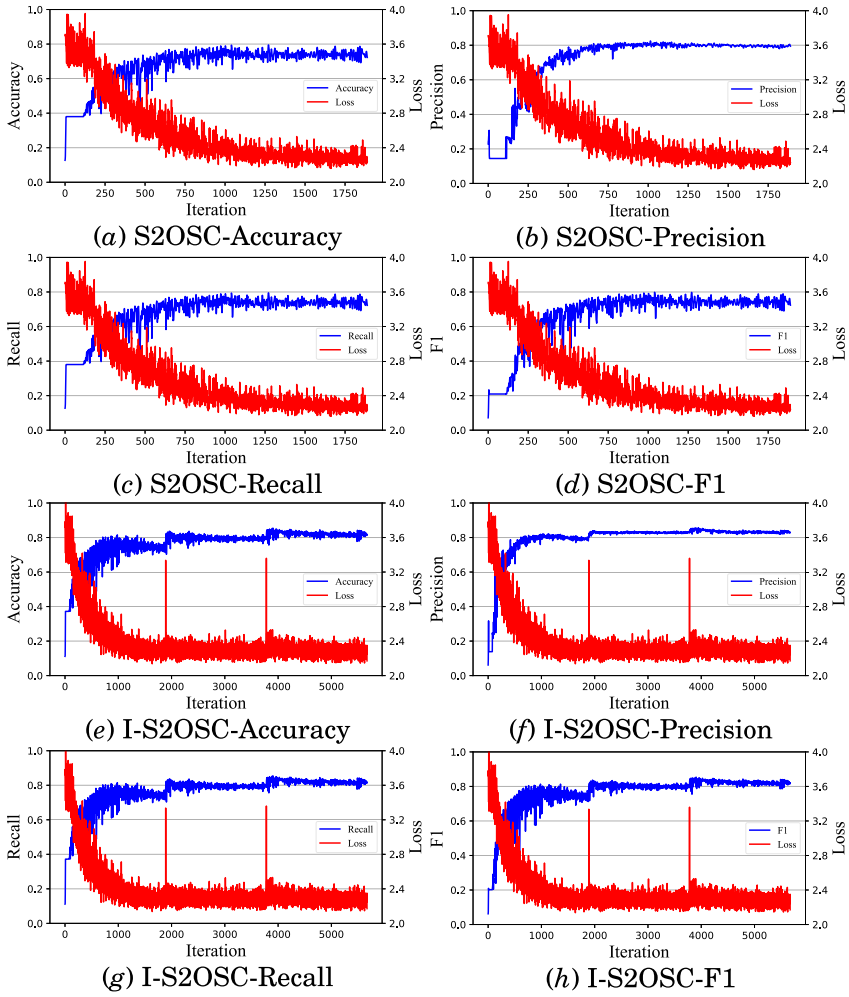


Fig. 9. Objective function value vs. number of iterations on CIFAR-10 dataset.

record the results of I-S2OSC on CIFAR-10 dataset. It clearly reveals that the objective function value decreases as the iterations increase, and the classification performance is stable after several iterations. Moreover, these additional experiment results indicate that our S2OSC/I-S2OSC can converge fast, i.e., S2OSC converges after 750 batches. Meanwhile, I-S2OSC has the similar phenomenon, it is notable that the decline of classification performance and the increase of loss in each time window are caused by the addition of new class examples in the training process.

7 CONCLUSION

Real-word applications always receive the data with unknown classes, thus it is necessary to promote the OSC. The key challenge in OSC is to overcome the embedding confusion caused by out-of-class instances. To this end, we propose a holistic semi-supervised OSC algorithm, i.e., S2OSC. S2OSC incorporated out-of-class instances filtering and semi-supervised model training in a transductive manner, and integrated in-class pre-trained model for teaching. Moreover, S2OSC can be

adapted to incremental OSC setting efficiently. Experiments showed the superior performances of S2OSC and I-S2OSC.

A APPENDIX

A.1 Datasets and Baselines

In this subsection, we will provide the details of datasets and baseline methods.

We utilize five publicly visual datasets for evaluating: CIFAR-10 dataset includes 60,000 natural color images of 32x32 pixels from 10 different classes, SVHN dataset also includes 100,000 natural color images of 32x32 pixels about house numbers from 10 different classes, MNIST dataset contains 70,000 labeled handwritten digits images from 10 categories. To validate the effectiveness of our method on dataset with large classes, we further experiment on CIFAR-50 as Geng et al. [2018], which randomly select 50 classes from CIFAR-100.

For baseline methods, we compare nine state-of-the-art methods including: (1) traditional outlier detection method: Iforest; (2) linear one-class OSC methods: One-Class SVM (One-SVM), LACU-SVM (LACU), and SENC-MAS (SENC); (3) deep one-class OSC methods: ODIN-CNN (ODIN), CFO, and CPE; and (4) deep multiple-class OSC methods: DTC. Abbreviations in parentheses. Specifically,

- Iforest: an ensemble tree method to detect outliers;
- One-Class SVM (One-SVM): a baseline for out-of-class detection and classification;
- LACU-SVM (LACU): a SVM-based method that incorporates the unlabeled data from open set for unknown class detection;
- SENC-MAS (SENC): a matrix sketching method that approximates original information with a dynamic low-dimensional structure;
- ODIN-CNN (ODIN): a CNN-based method that distinguishes in-distribution and out-of-distribution over softmax score;
- CFO: a generative method that adopts an encoder-decoder **Generative Adversarial Networks (GAN)** to generate synthetic unknown instances;
- CPE: a CNN-based ensemble method, which adaptively updates the prototype for detection;
- DTC: an extended deep transfer clustering method for novel class detection.

There are several instructions for baselines: (1) Iforest, ODIN, and CFO can only perform binary classifications, i.e., whether the instance is an unknown class. Thus we further conduct unsupervised clustering on both know and unknown class data for subdividing; (2) all baselines are one-class methods except DTC, i.e., they also perform OSC in two steps: first detect the super-class of unknown classes, and second perform unsupervised clustering; (3) all of baselines are OSC methods except LACU, SENC, and CPE but they can be applied in incremental OSC by combining memory data to update following Wang et al. [2019], except Iforest which replies on the quality of clustering in current time window.

A.2 Streaming Dataset

In this subsection, we mainly provide the details of streaming data construction and measure criteria. In this article, we perform incremental OSC, rather than online OSC, thus we need to accept the testing data before performing unknown class detection. There exists many related scenarios, such as sequential task learning in lifelong learning.

We consider single novel class case here following Da et al. [2014]; Geng et al. [2018]; Wang et al. [2019]. In detail, for each dataset, we randomly choose 50% 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

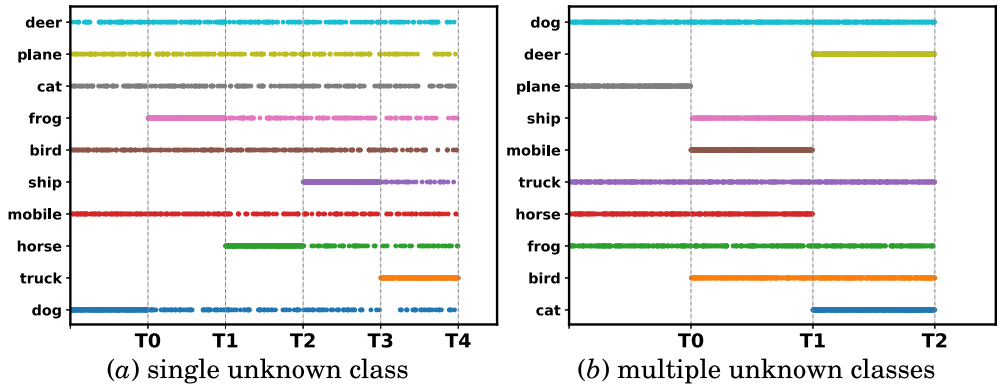


Fig. 10. The class distribution of simulated streams on CIFAR-10 dataset. The X-axis denotes time scale and Y-axis is class information. (a) is streaming data with single unknown class, (b) denotes streaming data with multiple unknown classes. Note that some known classes may disappear in time window t in Figure (b) in order to be more in line with real applications.

Table 13. Comparison of IOSC with Multiple unknown Classes

Methods	Iforest	One-SVM	LACU	SENC	I-S2OSC
Accuracy	.106 ± .003	.186 ± .005	.175 ± .006	.220 ± .001	**.444 ± .002**
Precision	.257 ± .005	.296 ± .002	.281 ± .007	.381 ± .002	**.606 ± .005**
Recall	.106 ± .002	.186 ± .007	.175 ± .004	.220 ± .008	**.442 ± .002**
F1	.132 ± .001	.205 ± .008	.194 ± .002	.134 ± .007	**.487 ± .006**
F_{out}	.230 ± .007	.406 ± .007	.404 ± .009	.685 ± .006	**.802 ± .007**
Forgetting	N/A	.625 ± .008	.633 ± .005	.678 ± .007	**.305 ± .009**
Methods	ODIN	CFO	CPE	DTC	I-S2OSC
Accuracy	.333 ± .003	.315 ± .005	.314 ± .004	.356 ± .007	**.444 ± .002**
Precision	.473 ± .004	.508 ± .004	.471 ± .005	.471 ± .004	**.606 ± .005**
Recall	.333 ± .007	.315 ± .001	.356 ± .009	.356 ± .001	**.442 ± .002**
F1	.271 ± .005	.259 ± .004	.399 ± .002	.399 ± .009	**.487 ± .006**
F_{out}	.796 ± .008	.736 ± .006	.233 ± .004	.233 ± .007	**.802 ± .007**
Forgetting	.606 ± .006	.390 ± .007	.374 ± .008	.374 ± .006	**.305 ± .009**

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 Figure 10(a). The data before time t_0 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 Δt. Thus, the instances that occurred in Δt constitute a time window data mixed with known and unknown instances.

For calculating Forgetting criterion, let $acc_{k,j}$ be the accuracy evaluated on the known class set, i.e., the data of classes emerging on jth time window ($j \leq k$), after training the network incrementally from stage 1 to k, the average accuracy at time k is defined as: $A_k = \frac{1}{k}\sum_{j=1}^{k} acc_{k,j}$ [Chaudhry et al. 2018]. Higher A_k represents better classifier. Thus, to validate the catastrophic forgetting, we calculate the performance on forgetting profile as Chaudhry et al. [2018], i.e., $Forgetting = \frac{A^* - mean(A)}{A^*}$, A^* is the optimal accuracy with entire data, and A is the set of average accuracy.

A.3 Incremental Multi-Class Case

To be more in line with the real scene, we consider an additional case of multiple unknown classes here. The main difference from single unknown class is that more than one unknown classes may appear in each time window. To solve this problem, since all datasets contain 10 classes, we firstly select 5 classes into known class set, the rest 5 classes are regarded as unknown class set. Then, we randomly select a number l from 1–5 as the number of time windows, and randomly divide the unknown class set into l parts. In result, we can obtain an incremental class order that appears in each time window. The generation of streaming data is the same as that of single unknown class setting. A case of generated streaming data of CIFAR-10 is shown in Figure 10(b).

We only give the results of CIFAR-10 which is with complex content here. Table 13 compares the classification and forgetting performance of I-S2OSC with all baselines over streaming data under multiple novel classes case. Best results are in bold, and N/A denotes no result. We can observe similar results as under the single novel class setting that I-S2OSC consistently outperforms all baselines on various criteria and has the least forgetting for update. We further compare the run time of I-S2OSC with other deep baselines. Considering the superior performance and model comparability, we only compared with CPE and DTC here. Specifically, the run time of I-S2OSC/CPE/DTC is 1.033/1.583/1.8 hours. The run time of I-S2OSC is significantly less than other methods, because CPE and DTC employ losses based on embeddings, which convergences slowly.

REFERENCES

David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. 2020. ReMixMatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *Proceedings of the International Conference on Learning Representations*.

Xin-Qiang Cai, Peng Zhao, Kai-Ming Ting, Xin Mu, and Yuan Jiang. 2019. Nearest neighbor ensembles: An effective method for difficult problems in streaming classification with emerging new classes. In *Proceedings of the 2019 IEEE International Conference on Data Mining*. 970–975.

Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. 2016. Synthesized classifiers for zero-shot learning. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*. 5327–5336.

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2018. Riemannian walk for incremental learning: understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision*. 556–572.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2019. AutoAugment: Learning augmentation strategies from data. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 113–123.

Qing Da, Yang Yu, and Zhi-Hua Zhou. 2014. Learning with augmented class by exploiting unlabeled data. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. 1760–1766.

Terrance Devries and Graham W. Taylor. 2017. Improved regularization of convolutional neural networks with cutout. arXiv:1708.04552. Retrieved from https://arxiv.org/abs/1708.04552.

Zongyuan Ge, Sergey Demyanov, and Rahil Garnavi. 2017. Generative openmax for multi-class open set classification. In *Proceedings of the British Machine Vision Conference*.

Chuanxing Geng, Sheng-Jun Huang, and Songcan Chen. 2020. Recent Advances in Open Set Recognition: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. early access. DOI: 10.1109/TPAMI.2020.2981604

Kai Han, Andrea Vedaldi, and Andrew Zisserman. 2019. Learning to discover novel visual categories via deep transfer clustering. In *Proceedings of the International Conference on Computer Vision*. 8400–8408.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of the International Conference on Learning Representations*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. arXiv:1503.02531. Retrieved from https://arxiv.org/abs/1503.02531.

Inhyuk Jo, Jungtaek Kim, Hyohyeong Kang, Yong-Deok Kim, and Seungjin Choi. 2018. Open set recognition by regularising classifier with fake data generated by generative adversarial networks. In *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2686–2690.

Ki Hyun Kim, Sangwoo Shim, Yongsub Lim, Jongseob Jeon, Jeongwoo Choi, Byungchan Kim, and Andre S. Yoon. 2020. RaPP: Novelty detection with reconstruction along projection pathway. In *Proceedings of the International Conference on Learning Representations*.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. In *Proceedings of the National Academy of Sciences*. 3521–3526.

Alex Krizhevsky, Geoffrey Hinton, and others. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report TR-2009, University of Toronto, Toronto.

Brian Kulis. 2013. Metric Learning: A Survey. *Foundations and Trends in Machine Learning* 5, 4 (2013), 287–364.

Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. 1998. The MNIST database of handwritten digits, 1998. Retrieved from http://yann.lecun.com/exdb/mnist. 10 (1998), 34.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. 2017. Overcoming catastrophic forgetting by incremental moment matching. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 4655–4665.

Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, Yang Yang, and Zi Huang. 2019. Alleviating feature confusion for generative zero-shot learning. In *Proceedings of the 27th ACM International Conference on Multimedia*. 1587–1595.

Shiyu Liang, Yixuan Li, and R. Srikant. 2018. Enhancing the reliability of out-of-distribution image detection in neural networks. In *Proceedings of the International Conference on Learning Representations*.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *Proceedings of the 80th IEEE International Conference on Data Mining*. 413–422.

Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. 2016. Large-margin softmax loss for convolutional neural networks. In *Proceedings of the 33rd International Conference on Machine Learning*. 507–516.

Laurens Van Der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605.

Xin Mu, Feida Zhu, Juan Du, Ee-Peng Lim, and Zhi-Hua Zhou. 2017. Streaming classification with emerging new class by class matrix sketching. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. 2373–2379.

Lawrence Neal, Matthew L. Olson, Xiaoli Z. Fern, Weng-Keen Wong, and Fuxin Li. 2018. Open set learning with counterfactual images. In *Proceedings of the European Conference on Computer Vision*. 620–635.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. 2011. Reading digits in natural images with unsupervised feature learning. In *Proceedings of the 25st International Conference on Neural Information Processing Systems*. 1–9.

Roger Ratcliff. 1990. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological Review* 97, 2 (1990), 285–308.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. 2017. iCaRL: Incremental classifier and representation learning. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*. 5533–5542.

Bernhard Scholkopf, John C. Platt, John Shawe-Taylor, Alexander J. Smola, and Robert C. Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural Computation* 13, 7 (2001), 1443–1471.

Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. 2020. FixMatch: Simplifying semi-supervised learning with consistency and confidence. In *Proceedings of the 33st International Conference on Neural Information Processing Systems*. 6–12.

Ge Song and Xiaoyang Tan. 2019. Sequential learning for cross-modal retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4531–4539.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning, PMLR*. 1139–1147.

V. Vovk and R. Wang. 2020. Combining p-values via averaging. *Biometrika* 107, 4 (2020), 791–808.

Zhuoyi Wang, Zelun Kong, Swarup Chandra, Hemeng Tao, and Latifur Khan. 2019. Robust high dimensional stream classification with novel class detection. In *Proceedings of the 2019 IEEE 35th International Conference on Data Engineering*. 1418–1429.

Yan Xia, Xudong Cao, Fang Wen, Gang Hua, and Jian Sun. 2015. Learning discriminative reconstructions for unsupervised outlier removal. In *Proceedings of the 2019 IEEE 35th International Conference on Data Engineering*. 1511–1519.

Received October 2020; revised April 2021; accepted May 2021