Information Sciences 744 (2026) 123355

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Portray learning: A novel learning paradigm for streaming emerging class detection

Ziming Zhao [a], Zhaoxuan Li [b], Xiaofei Yue [c,*], Tingting Li [a], Fan Zhang [a,*]

[a] Zhejiang University, Hangzhou, 310027, Zhejiang, China
[b] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100093, Beijing, China
[c] Beijing Institute of Technology, Beijing, 100081, Beijing, China

HIGHLIGHTS

- Portray Learning reframes emerging class detection as class-wise tests.
- Independent portrayers learn known classes in parallel.
- EVT-based calibration sets outlier thresholds automatically.
- Portrayers are updated as new classes emerge in streams.
- Consistently outperforms baselines on benchmarks and real-world data.

ARTICLE INFO

Keywords:
Emerging class detection
Class-incremental learning
Extreme value theory
Streaming data
Portray learning

ABSTRACT

In real-world classification tasks, the classes of data are not immutable, especially as new classes appear over time. Enabling the model to detect emerging classes in streaming data is an important research hotspot. It requires that the model can recognize new/known classes and be updated in an incremental manner. In this paper, we propose a novel learning paradigm Portray Learning to cope with this problem. So-called Portray Learning refers to maintaining a series of portrayers (*e.g.,* AutoEncoder) where each portrayer fits a category of data. Our scheme is different from typical class-incremental learning and semi-supervised learning. The core idea of Portray Learning is transforming known/new class identification issues into multiple independent anomaly detection problems. Each portrayer is responsible for detecting whether the instance belongs to its own class and these portrayers are independent of each other, so they can run in parallel and incrementally update portrayers of new classes. Moreover, we adopt the Extreme Value Theory to automatically determine outlier thresholds for each portrayer. The empirical evaluations involving several datasets and real-world streams show that our proposal achieves better outcomes than existing methods, even with little prior knowledge.

1. Introduction

With the widespread applications of machine learning in the real world, practitioners are increasingly aware that streaming data distributions in practice are often dynamic and evolving [1]. Instead of traditional supervised learning methods and category-invariant assumptions, the academic and industrial communities have invested a lot of research to advance streaming emerging class detection [2]. The latter refers to new classes that may emerge as the environment changes, while these novel classes are not included

* Corresponding authors.
Email addresses: zhaoziming@zju.edu.cn (Z. Zhao), xfyue@bit.edu.cn (X. Yue), fanzhang@zju.edu.cn (F. Zhang).

https://doi.org/10.1016/j.ins.2026.123355
Received 28 December 2025; Received in revised form 9 March 2026; Accepted 9 March 2026
Available online 12 March 2026

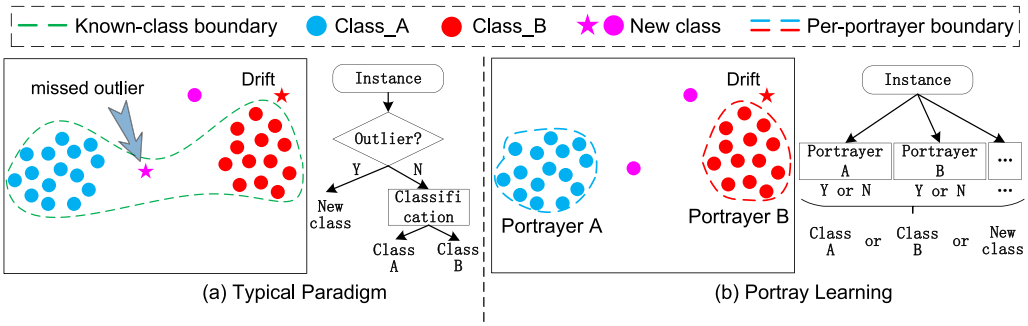

Fig. 1. Overview illustration of the typical paradigm for SENC and Portray Learning.

in prior knowledge when building the model [3]. This important problem is also known as classification under Streaming Emerging New Class (SENC) [4].

Although a series of methods are proposed to tackle this task, the detection performance is still limited, especially since the effect reaches a bottleneck as the number of categories increases [5]. We investigate the existing schemes [6] and outline them into the unified paradigm, as shown in Fig. 1(a). They usually address the SENC problem by following a general pipeline referring to *new class detection → known class classification → model update*. Notably, *model update* mainly requires that the model can be incrementally updated. When the number of known classes increases, it exhibits a certain accuracy loss due to missing outliers in the typical paradigm, as marked by the pink pentagram in Fig. 1(a). In other words, the new class detection becomes less sensitive when there are more known-class samples.

We rethink the SENC problem and suggest shifting the paradigm to decouple the need for model capabilities. In this paper, we present Portray Learning, a novel learning framework, to handle the above problems. At a high level, the proposed paradigm consists of a series of independent portrayers that correspond to data classes one by one, as depicted in Fig. 1(b). Essentially, each portrayer is responsible for detecting whether the test sample is its own class. Therefore, the *new class detection* and *known class classification* as a whole are converted into multiple independent anomaly detection problems. That is to say, the instance that cannot be covered[1] by any portrayer is considered the new class, otherwise, it will be assigned as the most similar class among the portrayers within the acceptable range. Furthermore, the per-portrayer possesses the corresponding outlier threshold. We employ Extreme Value Theory (EVT) [7] to automatically determine thresholds as boundaries of anomalies. The benefit is that it does not require manual threshold settings and does not rely on assumptions about the data distribution, which is closer to the practices.

In summary, this paper makes three key contributions.

- We propose Portray Learning, a novel learning paradigm to cope with the problem of streaming emerging class detection. It integrally transforms the novel class detection and known class classification problems into multiple independent outlier identification tasks. Meanwhile, our scheme supports class-incremental model updates.
- For each portrayer, we employ EVT to automatically determine the anomaly threshold. The proposal does not rely on data distribution assumptions that facilitate improving applicability in practice. Furthermore, we consider a two-step strategy to adapt to the concept drift.
- We conduct extensive experiments involving simulated streams from benchmark datasets and the real-world stream, and the results demonstrate the effectiveness of the proposed scheme. We also develop evaluations in terms of update efficiency and parameter analysis.

2. Background and related work

SENC problem requires the model to detect emerging classes, classify known classes, and update incrementally. These lines of related study can be categorized as follows.

2.1. Class-incremental learning

Class-incremental learning (C-IL) [8] is a significant branch of incremental learning which aims to enable the trained classifier to deal with emerging new classes [9]. It caters to the requirements of real-world applications and has drawn a lot of attention, *e.g.,* bot detection [10]. Learning with Augmented Class (LAC) [11] is proposed to cope with C-IL by the framework of LACU-SVM. It works as an SVM regularization and is on the batch setting. Moreover, realizing classification under Streaming Emerging New Class (SENC) [4], as a C-IL problem in the data stream context, is focused on by researchers. Some leading methods include clustering-based [12], matrix sketch [4], and tree-based [6]. ECSMiner [2] introduces time constraints for delayed classification and it assumes the ground-truth labels of new classes can be obtained after some time delay, which may lack applicability in practice. SAND [13] analyzes the confidence changes and suggests selecting a few samples based on the estimated confidence scores. SENC-MaS [4]

[1] The instance that cannot be covered refers to being identified as an outlier by the portrayer.

leverages matrix sketch to approximate original information based on the technique Frequent-Directions [14], thereby detecting the emerging classes and classifying known classes by maintaining two low-dimensional matrix sketches [15]. SENCForest [6] combines isolation-based anomaly detection and tree classification capabilities (called completely-random trees) to tackle the SENC problem, achieving the state-of-the-art (SOTA) performance.

2.2. Semi-supervised learning

Semi-supervised learning (SSL) [16] aims to exploit unlabeled data for training, usually using a small set of labeled data together with a large collection of unlabeled data. Typical SSL methods do not consider that previously unseen labels may exist in unlabeled data [17], so SEEN [1] strengthens iForest [18] and integrates it with a robust classifier into a semi-supervised framework.

2.3. Novel class detection and anomaly detection

Novel class detection and anomaly detection [19] focus on the identification of data that have not been seen during the training process, which can be regarded as the solution for the first subproblem (emerging class detection). For example, an isolation-based process (iForest [18]) is proposed to generate trees and forests. This technique uses a random selection of feature dimensions when splitting nodes, which is very different from typical decision trees. Thus, instances isolated by fewer paths are considered outliers, *i.e.,* new classes. Another study [20] investigates a robust random cut data structure that serves as a sketch (or synopsis) of the input stream, namely Robust Random Cut Forest (RRCF), for anomaly detection in dynamic data streams. However, these schemes ignore the problem of classification (*i.e., one-class learning*, only distinguishing normal/abnormal and treating all known classes as normal) and model updates. Previous work would combine them with additional classifiers (*e.g.,* SVM) as baselines.

2.4. Prototype learning

Prototype learning could provide a concise representation for the entire category of instances, where the prototype indicates an average or best exemplar of a category [21]. Some schemes leverage prototype learning to advance novel class detection. GCPL [22] designs a framework for convolutional prototype learning to cope with the open set problem [23], while GCPL only considers one unknown class and does not support incremental updates [24]. CPE [25] uses a prototype ensemble loss (PE) to improve the convolutional neural network (CNN) model to detect novel classes.

Recent advances in continual and prototype-based learning have further explored representation stability under evolving class distributions [26]. For example, PODNet [27] introduces a distillation-based prototype learning framework that preserves class prototypes across incremental learning stages, achieving strong performance in class-incremental benchmarks. Despite their effectiveness, these methods primarily focus on maintaining discriminative representations for known classes and typically assume that the class identity of incoming samples is available at update time. As a result, they are not directly designed for streaming emerging new class (SENC) settings, where instances arrive unlabeled and the model must simultaneously perform novel class detection, known class classification, and incremental updates. In contrast, our work tackles the SENC problem from a different perspective by transforming it into multiple independent class-wise anomaly detection tasks. By leveraging per-class portrayers and EVT-based adaptive thresholds, the proposed Portray Learning paradigm naturally supports unlabeled streaming data, emerging class discovery, and parallelizable incremental updates, which are not explicitly addressed in existing prototype-based continual learning methods.

3. Portray learning

In this section, we propose Portray Learning, a novel paradigm to deal with the emerging class problem in streaming data.

3.1. Overview

In Fig. 2, we depict the overall workflow of Portray Learning at a high level. Consider a data stream, the labeled samples could be used to construct models that include multiple portrayers. Among them, each portrayer is built upon per-class instances to compress and decompress data (*e.g.,* using encoder + decoder). Therefore, samples of the same class as the portrayer tend to obtain smaller reconstruction loss, whereas samples different from the portrayer class correspond to larger loss. When the unlabeled instance arrives, it will traverse through existing portrayers and calculate the corresponding reconstruction loss. If the sample does not fit any of the portrayers (*i.e.,* the reconstruction loss for each portrayer is greater than the corresponding acceptable threshold), it will be considered as an emerging class and enter the buffer for the subsequent model updates. On the contrary, if the instance matches at least one portrayer, its prediction result will be assigned as the class with the smallest reconstruction loss.

Algorithm 1 describes the procedure of Portray Learning. As shown in lines 2-7, we establish a class-wise portrayer for each known class and automatically compute its outlier threshold using EVT (introduced subsequently), which profiles the reconstruction-loss boundary of the corresponding class. Given an incoming unlabeled instance, lines 9–15 compute its reconstruction loss w.r.t. each existing portrayer. Line 16 then collects candidate classes whose portrayers accept the instance, *i.e.,* $L_r[j] < L_t[j]$. If no candidate exists, the instance is regarded as an emerging-class sample and buffered for subsequent updates (lines 17-23). Otherwise, we predict the label of the candidate portrayer that yields the minimum reconstruction loss, *i.e.,* $j^\star = \arg\min_{j \in M_c} L_r[j]$ (lines 24-27).

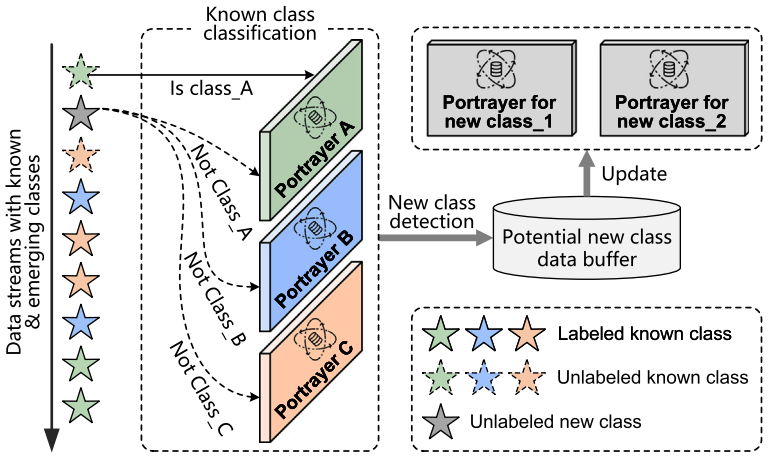

Fig. 2. The framework of Portray Learning.

Algorithm 1 Portray learning.

Require: The dataset $\mathcal{D}$ of known classes $\{k_1, k_2, \ldots, k_n\}$, the unlabeled data stream $\mathcal{S}$, buffer $\mathcal{B}$, maximum buffer size s
Ensure: $\hat{y}$ - predicted class label for each $x \in \mathcal{S}$

Initialize class-portrayer list $L_p = [\,]$; Initialize threshold list $L_t = [\,]$
for $i \in \{k_1, k_2, \ldots, k_n\}$ **do**
  Select subset $d_i \triangleq \{(x, y) \in \mathcal{D} \mid y = i\}$
  Construct portrayer p_i based on d_i
  $t_i \leftarrow$ *Outlier Threshold*(p_i, d_i)
  Append (i, p_i) to L_p; **Append** t_i to L_t
end for
Initialize prediction list $Result = [\,]$
for $x \in \mathcal{S}$ **do**
  Initialize reconstruction loss list $L_r = [\,]$
  for $j = 1$ to $|L_p|$ **do**
    $(c_j, p_j) \leftarrow L_p[j]$
    Calculate reconstruction loss $l_j \leftarrow p_j(x)$
    Append l_j to L_r
  end for
  $M_c \leftarrow \{\, j \mid L_r[j] < L_t[j] \,\}$ ▷ candidate categories
  if $|M_c| = 0$ **then**
    Append emerging-class label to *Result*
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{x\}$
    if $|\mathcal{B}| \geq s$ **then**
      $L_p, L_t \leftarrow$ *Update*$(L_p, L_t, \mathcal{B})$
      $\mathcal{B} \leftarrow \emptyset$
    end if
  else
    $j^\star \leftarrow \arg\min_{j \in M_c} L_r[j]$
    $(\hat{y}, \cdot) \leftarrow L_p[j^\star]$
    Append $\hat{y}$ to *Result*
  end if
end for
return *Result*

3.2. Portrayer construction

The portrayer is responsible for per-class data reconstruction. A vanilla design is directly using the AutoEncoder as a portrayer. However, deep layers in this typical architecture may lose some feature information. To this end, we add the tensor from the encoder to the corresponding layer in the decoder, so as to preserve more semantic information in different tiers. The model architecture is shown in Fig. 3, which is somewhat similar to U-Net [28] (a convolutional network). It contains an encoder and a decoder, which are symmetrical to each other. For the encoder, the d-dimensional input is concatenated with four fully connected layers

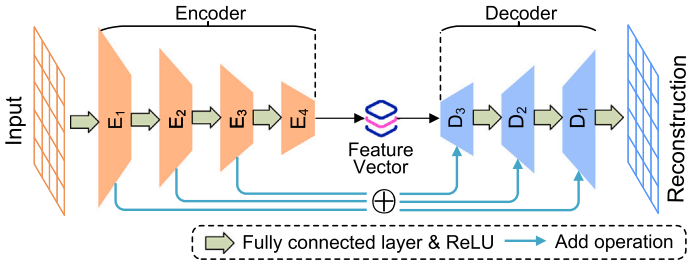

Fig. 3. The AutoEncoder architecture as the portrayer.

(with the ReLU activation), in which the corresponding feature vectors are $F_v \rightarrow E_1 \rightarrow E_2 \rightarrow E_3 \rightarrow E_4$ and the dimensions are $d \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32$. For the decoder, it will perform tensor reconstruction via linear layers (with the ReLU activation) symmetric to the encoder. The output vectors are $E_4 \rightarrow D_3 \rightarrow D_2 \rightarrow D_1 \rightarrow F_r$ and the dimensions are $32 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow d$. Notably, the decoder part will cascade the intermediate output of the encoder for an addition operation (the blue arrow in Fig. 3). For instance, D_2 can be calculated by $D_2 = ReLU(Linear(D_3 + E_3))$. Such an addition operation preserves richer features at each level to alleviate information loss during reconstruction.

Meanwhile, we employ mean squared error (MSE) as the loss function since the per-portrayer is essentially performing the anomaly detection task. Empirically, a well-trained portrayer will output a smaller MSE loss for samples of the same class, while data of different classes will lead to a larger loss. Next, we elucidate how the threshold of MSE loss is calculated automatically to determine whether the test instance falls to the class of the portrayer.

3.3. Portrayer outlier threshold

The core idea of Portray Learning is that *each class corresponds to a portrayer, and the test sample that cannot be covered by any portrayer is assigned to a new class; otherwise, this instance belongs to the class with the smallest reconstruction loss*. Therefore, a key issue is to determine the threshold indicating whether a portrayer can cover samples of its own class. Considering that a manually set threshold is not universal and often needs re-tuning across datasets, we design an automated threshold determination scheme. To solve this problem, we leverage the Extreme Value Theory (EVT) [7] to estimate the outlier bounds. The original intention of this theory in statistics is to study the occurrence of extreme events. Specifically, the form of Extreme Value Distributions (EVD) is given as follows:

$$G_\gamma(x) = \exp\left(-(1+\gamma x)^{-\frac{1}{\gamma}}\right), \quad \text{for } \gamma \in \mathbb{R} \text{ and } 1+\gamma x > 0. \tag{1}$$

Here, γ denotes the extreme value index. An elegant property of EVT is that the distribution of extreme values does not depend on the original data distribution. In other words, these extreme events converge to the same family of distributions, regardless of whether the original data follow, for example, Fréchet, Gamma, or Uniform distributions [29].

Given a trained portrayer, we compute a reconstruction-loss list $L = \{l_m\}_{m=1}^{n}$ on its corresponding in-class subset. We set the initialization threshold t as the upper quantile of L, *i.e.*, $t = Q_u(L)$, where we use $u = 0.98$ in all experiments (unless otherwise stated). We will report the sensitivity to u in the parameter study (Section 4.3). Then, according to the Pickands-Balkema-de Haan theorem (also called the second theorem in EVT) [30], the extrema of the cumulative distribution function F converge in distribution to G_γ if and only if a function σ exists, *i.e.,*

$$\bar{F}_t(x) = \mathbb{P}(X - t > x \mid X > t) \underset{t \rightarrow \tau}{\sim} \left(1 + \frac{\gamma x}{\sigma(t)}\right)^{-\frac{1}{\gamma}} \tag{2}$$

It means that $X - t$ (excess over threshold) tends to follow a Generalized Pareto Distribution (GPD)[2] with parameters γ and σ [31]. The POT/GPD approximation is derived under standard EVT conditions that the threshold exceedances (excess losses) are approximately independent and identically distributed (i.i.d.) within a (locally) stationary regime. In our setting, L is computed on in-class samples; when training/validating a portrayer, samples are typically drawn i.i.d. from the in-class subset (or shuffled mini-batches), which makes the i.i.d. assumption a reasonable approximation. In streaming scenarios where temporal dependence or concept drift may appear, we re-estimate thresholds on recent buffered samples (Section 3.4), which is intended to preserve local stationarity; nevertheless, residual dependence may still violate the i.i.d. assumption and we acknowledge this as a limitation.

Once we obtain estimates $\hat{\gamma}$ and $\hat{\sigma}$, the Peaks-Over-Threshold (POT) approach can be used to calculate the outlier threshold as

$$\mathcal{T} \simeq t + \frac{\hat{\sigma}}{\hat{\gamma}}\left(\left(\frac{qn}{N_p}\right)^{-\hat{\gamma}} - 1\right) \tag{3}$$

[2] The location μ, the third parameter of GPD, is null in our case.

where q denotes the risk parameter, n refers to the data size, and N_p represents the number of peaks. Concretely, in our POT implementation, "peaks" are exactly the threshold exceedances in the loss list: we define the excess set $\mathcal{P} = \{\mathcal{P}_i\}_{i=1}^{N_p} = \{ l - t \mid l > t, l \in L \}$, and thus $N_p = |\{ l \in L \mid l > t \}|$.

In order to estimate $\hat{\gamma}$ and $\hat{\sigma}$, maximum likelihood estimation is an efficient method, and it aims to maximize (after logarithmic transformation):

$$\log \mathcal{L}(\gamma, \sigma) = -N_p \log \sigma - \left(1 + \frac{1}{\gamma}\right) \sum_{i=1}^{N_p} \log\left(1 + \frac{\gamma}{\sigma}\mathcal{P}_i\right) \tag{4}$$

where $\mathcal{P} = \{l - t \text{ s.t. } l > t \text{ and } l \in L\}$ and L denotes the loss list. According to [32], Grimshaw's proposal can reduce the two-variable optimization problem to a single-variable equation. Specifically, if we get a solution (γ^*, σ^*), the variable $x^* = \gamma^*/\sigma^*$ is a solution of the scalar equation $u(x)v(x) = 1$ where:

$$u(x) = \frac{1}{N_p} \sum_{i=1}^{N_p} \frac{1}{1 + x\mathcal{P}_i} \tag{5}$$

$$v(x) = 1 + \frac{1}{N_p} \sum_{i=1}^{N_p} \log(1 + x\mathcal{P}_i) \tag{6}$$

That is to say, by finding a solution x^* of this equation, we can retrieve $\gamma^* = v(x^*) - 1$ and $\sigma^* = \gamma^*/x^*$. Finally, the estimates γ^* and σ^* can be used to calculate the threshold by Eq. (3).

3.4. Model update

Incrementally Add Portrayer. Since the portrayers are independent of each other, the update operation can be conveniently performed. Similar to previous work [6], we assume that instances stored in the buffer $\mathcal{B}$ are eventually assigned with true class labels. This assumption corresponds to a delayed-labeling setting, which is common in streaming scenarios where annotations are obtained periodically or after a verification stage (*e.g.,* human inspection or offline labeling). Subsequently, instances belonging to each newly discovered class are used to construct the corresponding portrayer and update the portrayer list and threshold list, as shown in lines 1–7 of Algorithm 2. After the update, the newly discovered class is treated as a known class in subsequent streaming data. Overall, since portrayers correspond to classes in a one-to-one manner, adding new classes naturally follows an incremental update strategy. Moreover, the independence among portrayers enables parallel construction and deployment, facilitating horizontal scalability and reducing system overhead, *e.g.,* distributing different portrayers across multiple computing nodes.

Concept Drift Adaptation. Due to the dynamism and evolution of streaming data, we need to consider the problem of concept drift [33]. *Concept drift* refers to the data representation of the same category changing over time. To cope with this problem, we propose a two-step solution, *i.e.,* recalculating the threshold with new samples to adapt to slight drifts, otherwise retraining the portrayer. The former refers to updating the portrayer p_i for class i by computing reconstruction losses on recent class-i samples from the incoming stream (under concept drift). Then append them to the reconstruction loss list L to get a new outlier threshold $\mathcal{T}'$. Subsequently, for portrayer p_i, if it can cover more than the e ratio of other class instances (where e denotes the error coefficient, set as 0.05), we consider p_i to be outdated. In this situation, we tend to retrain the portrayer p' based on class-i samples from the original data set and the new data stream.

3.5. Deep insights of Portray learning

In this section, we clarify why our approach is justifiable and how it overcomes the drawbacks of the typical pipeline.

(i) As shown in Fig. 1(a), the typical zero-positive learning produces broader classification boundaries (green dotted line) compared to per-class samples, forming a generalized sample center. It corresponds to the first step of the typical pipeline (new class detection). Therefore, there are some samples that are incorrectly detected in the first step of the classical paradigm. *For the divide-and-conquer*

Algorithm 2 Incrementally add portrayer.

Require: The portrayer list L_p, the threshold list L_t, buffer $\mathcal{B}$
Ensure: The updated portrayer list and the updated threshold list
1: **for** $\mathcal{Y} \in$ new classes of $\mathcal{B}$ **do**
2: **Select** subset $b_i = \{(x, y) \in \mathcal{B} | y = \mathcal{Y}\}$
3: Construct portrayer p_i based on b_i
4: $t_i \leftarrow$ *Outlier Threshold*(p_i, b_i)
5: Append t_i to L_t
6: Append $(\mathcal{Y}, p_i)$ to L_p
7: **end for**
8: **return** L_p, L_t

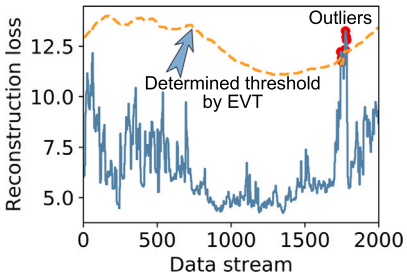

Fig. 4. Dynamically update thresholds based on EVT in streaming data.

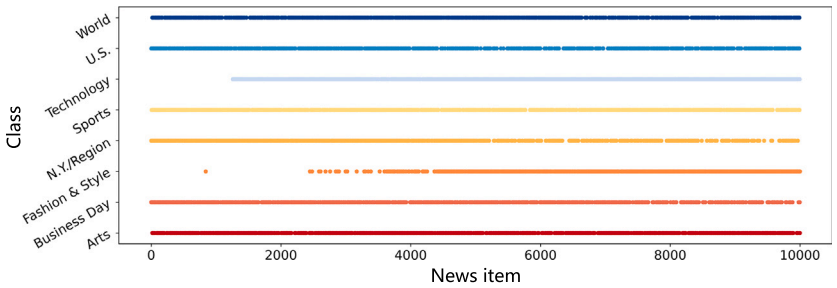

Fig. 5. New York Times news stream. The X-axis is data streaming and the Y-axis is class information of each item.

idea in Portray Learning, Fig. 1(b), *per-portrayer only corresponds to one-class samples, making each portrayer tend to produce tighter boundaries.*

(ii) For the noise instance (also called drift), it is difficult to distinguish it from the new class (real anomalies) in the typical paradigm, as Fig. 1(a). Based on the assumption of previous work [6] (that new-classes are more anomalous than known-class anomalies), *our scheme is more promising in distinguishing between noise instances and new-class samples*, as shown in Fig. 1(b). Compared with the two algorithms *CPE [25] and SENCForest [6] that can handle concept drift*, our proposal shows the advantages.

Meanwhile, the portrayer can adapt data streams by *dynamically adjusting EVT thresholds*, in Fig. 4. In the data stream, we can use the sliding window (data buffer) to maintain the threshold (consistent with EVT paper [7]). Even if there is data drift, the threshold determined by EVT can be updated dynamically, as shown in Fig. 4. Moreover, the model training for each portrayer is independent and decoupled, causing no additional overhead for other models.

4. Experiments

In this section, we extensively evaluate Portray Learning on both simulated and real-world streams. The online code repository is available.[3]

4.1. Experimental setup

Datasets. For the evaluation, we use six public datasets including KDDCup 99 [34], Forest Cover, HAR [35], Satimage, USPS, and Pendigits [36]. The four largest classes in KDDCup 99 are used for evaluation, *i.e.,* normal, neptune, smurf and back. Also, the 10 attributes (except binary attributes) of Forest Cover are used as the feature vectors. The settings for KDDCup 99 and Forest Cover are the same as those for SENCForest [6]. The HAR dataset refers to human activity recognition using smartphones such as walking, laying, etc. The Satimage dataset consists of the multi-spectral values of pixels in 3×3 neighborhoods in a satellite image, and it aims to predict the classification associated with the central pixel in each neighborhood. The USPS and Pendigits datasets are both digit samples. The USPS dataset is obtained by scanning envelopes by the U.S. Postal Service and they have been normalized to 16×16 grayscale images. Also, Pendigits is pen-based recognition of the handwritten digits dataset which has 16 integer attributes. For a given dataset, the data stream is simulated as follows. In the beginning, only one-class samples are known, and a new class will appear in each subsequent period. This setting is consistent with SEEN [1]. The experiments on each dataset are repeated 10 times with different simulated data streams and both the mean and standard deviation of the performance are reported.

In addition, a real news data stream is used to evaluate performance. We crawled over a period of time by the New York Times API [37], there are 10k news items (across 1981/01/02~2023/01/11) categorized into 8 classes, *i.e.,* "Arts", "Business Day", "Fashion & Style", "N.Y./Region", "Sports", "Technology", "U.S.", "World". Each item is preprocessed into a 1000-dimensional feature vector with the "word2vec" technique [38]. To clearly illustrate the data distribution, we depict the news item stream in Fig. 5. The first 1000 samples are used as the initial set to train the model. We summarize the dataset information in Table 1, *the number of classes*

[3] Online repository https://github.com/Secbrain/PortrayL

Table 1
A summary of datasets used for evaluation.

Dataset	#classes	#attributes
KDDCup 99	4	41
Forest Cover	7	10
HAR	6	561
Satimage	6	32
USPS	10	256
Pendigits	10	16
New York Times	8	1000

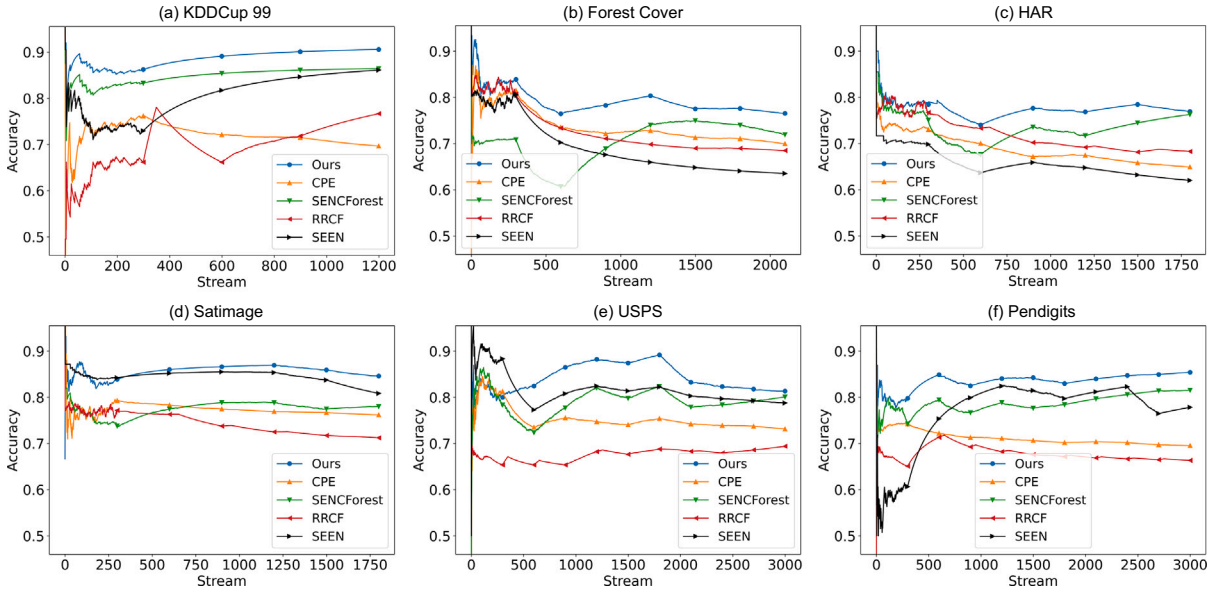

Fig. 6. Accuracy results on the data streams of six datasets.

is [4, 10]*, and sample attributes are* [10, 1000]. This is consistent with previous work, *note that the data stream arrives sample-wise, rather than class-wise.*

Competing Algorithms. We compare with: *CPE* [25]: it uses a prototype ensemble loss to improve the convolutional neural network (CNN) model to detect novel classes. *RRCF* [20]: this method investigates a random cut data structure that can be used as a sketch of the dynamic input stream. Since RRCF does not make classifications, we combine it with SVM as a classifier. *SENCForest* [6]: it leverages the isolation-based idea [18] and builds completely-random trees for emerging class detection. *SEEN* [1]: it is a semi-supervised technique to deal with dynamically evolving data streams.

Algorithm Settings. For CPE, the maximum number of per-class prototypes K is 10, and $M = 100$ is the maximum amount of exemplars. For RRCF, the shingle size is 4, and the SVM classifier is set to RBF kernel. For SENCForest, the number of trees is set to 100, the minimum internal node size $MinSize = 10$, and the distance coefficient $\alpha = 1$ in the mass estimation. For SEEN, the maximum tree height $h_m = 7$, and the weight matrix uses standard RBF similarity for initialization. The subset size for training is set to 100 in SENCForest and SEEN. In our scheme, the risk parameter $q = 5e-2$, and we also evaluate other values for q in the evaluation section. For all the above methods, the maximum buffer size is set to 50 and the initialization is realized by the data of randomly selected one known class (for simulated streams). Unless otherwise stated, the hyperparameters of all baselines follow the settings recommended in their original papers (or publicly released implementations), and we did not apply extensive per-dataset tuning to any method. In particular, SENCForest is configured with the commonly used setting in [6], and all methods share the same stream simulation, initialization protocol, and buffer size to ensure a fair comparison.

4.2. Results

Simulated Stream. The results of simulated stream are summarized in Table 2 and Fig. 6. Table 2 reports the accuracy and F1 scores on six datasets, including the mean and standard deviation over 10 independent runs. Portray Learning outperforms other approaches and maintains good performance in data streams with continually emerging new classes. The closest contenders are SENCForest and SEEN, both of them employ an ensemble framework combining supervised learning and unsupervised learning. Compared with SENCForest (the best baseline), our scheme improves accuracy and F1 by an average of 0.035 and 0.031 respectively.

Fig. 6 further shows the change curve of average accuracy in the whole data stream with the six datasets. The overall identification performance is Portray Learning > SENCForest > SEEN > CPE > RRCF. Specifically, SENCForest achieves comparable accuracy to

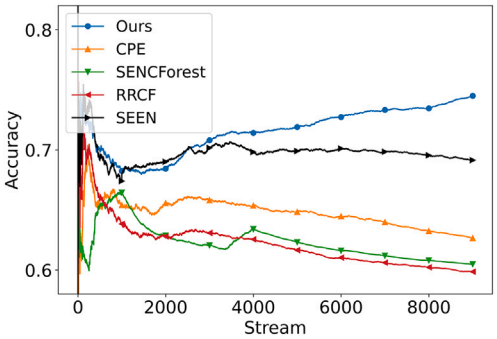

Fig. 7. Accuracy results on New York Times stream.

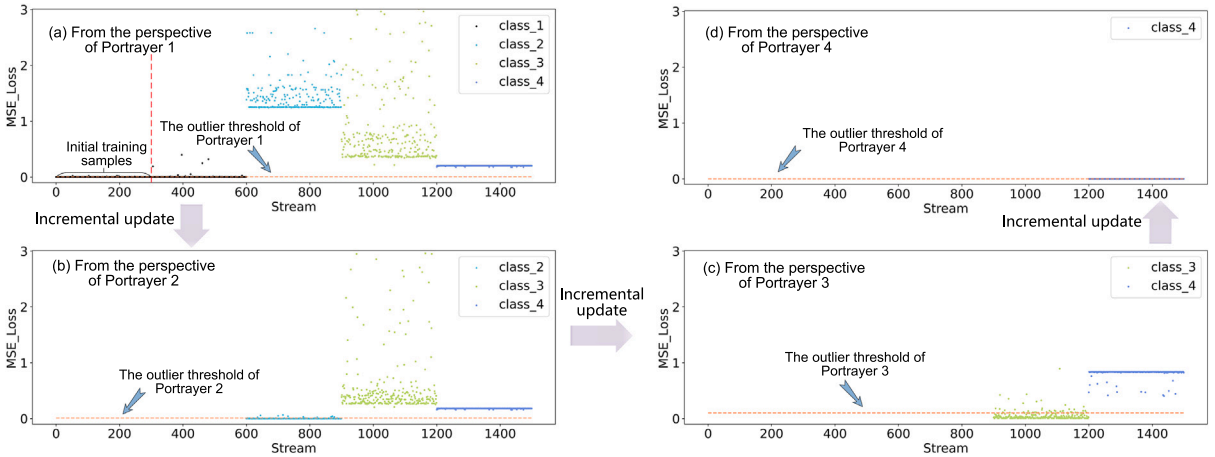

Fig. 8. The deep insights into Portray Learning on the KDDCup 99 dataset.

Portray Learning in HAR and USPS datasets, as shown in subfigures (c) and (e). The outcome of SEEN is not robust enough given that it performs badly on the Forest Cover and HAR datasets, *i.e.,* subfigures (b)-(c). CPE and RRCF perform more misclassifications as classes increase. In most cases, CPE has a slight advantage compared to RRCF, even when RRCF is combined with the SVM classifier.

Real-World Stream. We also evaluate the real-world stream which would contain potential concept drifts, as stated in the experimental setup section. In Fig. 7, it is clear that Portray Learning outperforms other methods. For baseline models, SEEN stands out as being very suitable for such semi-supervised scenarios, while Portray Learning possesses ~5.35% accuracy advantage over SEEN. CPE and RRCF show significant accuracy drops in the later stage. And the performance of SENCForest is unstable in the real stream as its accuracy curve presents obvious fluctuations. This instability may be attributed to the fact that SENCForest is more suitable for data streams with a certain distribution mode, *e.g.,* simulated streams. As stated in the introduction section, existing methods exhibit a performance bottleneck, when dealing with more data categories and instances. Overall, Portray Learning maintains superior performance in both the simulated and real-world data streams.

Table 2
Comparisons of different methods on simulated streams.

Dataset	Metric	RRCF	CPE	SENCForest	SEEN	Ours
KDDCup 99	Accuracy	0.767 ± 0.008	0.697 ± 0.032	0.864 ± 0.015	0.861 ± 0.018	**0.906 ± 0.009**
	F1	0.742 ± 0.007	0.646 ± 0.017	0.874 ± 0.013	0.882 ± 0.009	**0.919 ± 0.005**
Forest Cover	Accuracy	0.685 ± 0.015	0.700 ± 0.019	0.719 ± 0.027	0.635 ± 0.011	**0.765 ± 0.012**
	F1	0.619 ± 0.013	0.708 ± 0.015	0.711 ± 0.030	0.546 ± 0.007	**0.764 ± 0.010**
HAR	Accuracy	0.683 ± 0.011	0.649 ± 0.034	**0.763 ± 0.043**	0.620 ± 0.020	**0.769 ± 0.019**
	F1	0.659 ± 0.010	0.665 ± 0.029	**0.746 ± 0.037**	0.553 ± 0.021	**0.750 ± 0.018**
Satimage	Accuracy	0.712 ± 0.016	0.762 ± 0.038	0.780 ± 0.024	0.808 ± 0.017	**0.846 ± 0.021**
	F1	0.695 ± 0.009	0.772 ± 0.032	0.790 ± 0.021	0.753 ± 0.014	**0.844 ± 0.016**
USPS	Accuracy	0.694 ± 0.006	0.731 ± 0.026	**0.801 ± 0.028**	0.788 ± 0.012	**0.813 ± 0.025**
	F1	0.660 ± 0.008	0.736 ± 0.028	**0.829 ± 0.019**	0.770 ± 0.016	**0.831 ± 0.018**
Pendigits	Accuracy	0.663 ± 0.016	0.695 ± 0.012	0.815 ± 0.033	0.778 ± 0.007	**0.854 ± 0.014**
	F1	0.670 ± 0.011	0.708 ± 0.014	0.826 ± 0.031	0.783 ± 0.009	**0.852 ± 0.011**

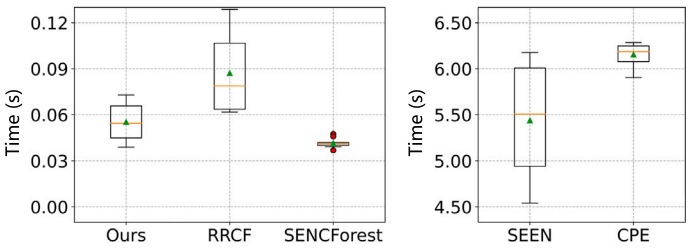

Fig. 9. Updating time analysis for baselines and our scheme.

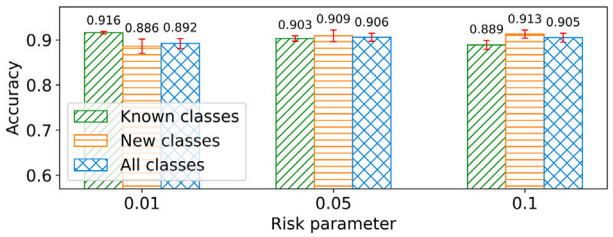

Fig. 10. The influence of the risk parameter q.

4.3. Performance study

Subsequently, we further explore Portray Learning for deep insights and evaluate its update efficiency, as well as conduct experiments for parameter analysis. The following experiments are carried out based on the KDDCup 99, and similar results can be concluded on other datasets.

(i) **Deep Insights.** To understand how Portray Learning detects new classes and performs classification, we visualize the reconstruction loss of the samples from each portrayer's perspective. Fig. 8 depicts the evolving process of our model over the data stream. At the beginning, *portrayer 1* is built upon the initial training set which solely includes instances of *class 1*. During the test, it contains 4 stages corresponding to different types of data, *i.e., class 1* → *class 2* → *class 3* → *class 4*. Stage 1 can be regarded as the known class classification. For the following three stages, each stage corresponds to updating one portrayer in an incremental manner. Finally, four portrayers are generated, corresponding to the data classes one by one.

From the perspective of *portrayer 1*, we find that it outputs a small reconstruction loss for most instances of *class 1* but a large reconstruction loss for the other three classes. Those samples of *class 1* covered by the threshold (below the orange line) will be marked as class 1, while samples of other classes are considered outliers by *portrayer 1*, *i.e.,* not *class 1*. This process also applies to *portrayers 2-4*. That is to say, each portrayer tends to generate a small reconstruction loss for its own class instances, and other-class samples will be considered outliers. In this way, the tasks for known class classification and novel class detection are solved together based on the results of these portrayers. It echoes back to our original intention for designing Portray Learning.

(ii) **Update Efficiency.** We measure the update time overhead based on the KDDCup 99 for four baseline algorithms and our scheme, and the results are summarized in Fig. 9. All models run on an Ubuntu 18.04.2 server with an Intel i7-12700K CPU and 64 GB of memory. Overall, RRCF, SENCForest, and ours are on one level ($\sim 0.1s$), while SEEN and CPE are on the other level ($\sim 6s$). Therefore, the most time-consuming methods are CPE and SEEN. Portray Learning and SENCForest take the least time, and their difference is only about $0.01s$. Particularly, we observe that the standard deviation of SEEN is relatively large, which could be attributed to diverse situations of semi-supervised streaming data (with the increase in samples and classes). In contrast, Portray Learning can adapt better since its portrayers for each class are independent. Furthermore, the above results are based on CPU, and our model architecture is easily GPU-accelerated.

(iii) **Parameter Analysis.** The most significant parameter that impacts the predictions of each portrayer is the risk parameter q. Based on the KDDCup 99, we study the influence of the risk parameter by setting $q = 0.01, 0.05, 0.1$, respectively. As Fig. 10 shows, the trend presents that with larger q (means smaller outlier threshold), the performance of known class classification drops while it improves for new class detection. Therefore, the risk parameter can be tuned to trade off between known class classification and unknown class detection performance in practice.

(iv) **Sensitivity to the Upper Quantile u.** In the EVT-based threshold initialization, the upper quantile u determines the initial threshold $t = Q_u(L)$ used for the Peaks-Over-Threshold procedure. To examine the robustness of Portray Learning with respect to this choice, we conduct a sensitivity analysis by varying u over $\{0.95, 0.97, 0.98, 0.99\}$ on the KDDCup 99 dataset, while keeping other parameters fixed. As shown in Table 3, Portray Learning exhibits stable performance across a wide range of upper quantiles. A smaller u leads to a lower initial threshold, which slightly favors new-class detection at the expense of known-class classification. Overall, the performance variation is limited, indicating that Portray Learning is not overly sensitive to the choice of u.

(v) **Sensitivity to the Drift Error Coefficient e.** In the concept drift adaptation mechanism, the error coefficient e controls the tolerance for a portrayer covering samples from other classes before being considered outdated and retrained. We evaluate Portray

Table 3
Sensitivity analysis of the upper quantile u on KDDCup 99.

Upper quantile u	Known-class Acc.	New-class Acc.	Overall Acc.
0.95	0.891	0.915	0.901
0.97	0.899	0.912	0.904
0.98	0.903	0.909	0.906
0.99	0.907	0.904	0.905

Table 4
Sensitivity analysis of the drift error coefficient e on KDDCup 99.

Error coefficient e	Known-class Acc.	New-class Acc.	Overall Acc.
0.01	0.893	0.916	0.899
0.03	0.901	0.912	0.903
0.05	0.903	0.909	0.906
0.10	0.905	0.904	0.902

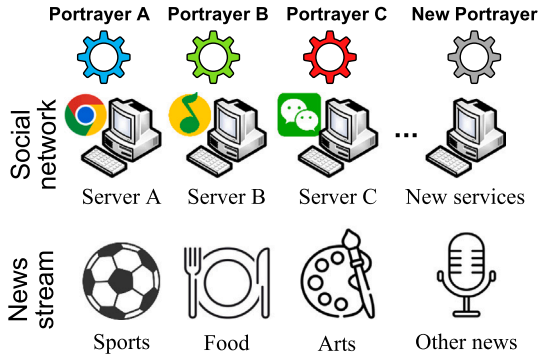

Fig. 11. Potential application scenarios in the real world.

Learning under different values of $e \in \{0.01, 0.03, 0.05, 0.10\}$ on the KDDCup 99 stream. From Table 4, we observe that a smaller e makes the model more sensitive to concept drift, which may improve adaptability but can also cause frequent retraining and reduce stability. Overall, the performance of Portray Learning remains stable across a reasonable range of e, demonstrating robustness of the proposed drift-handling strategy.

5. Conclusion, limitations, and future work

In this paper, we propose Portray Learning, a novel paradigm to tackle the SENC problem. Our proposal transforms known/new class identification issues into multiple independent outlier detection tasks. Meanwhile, we leverage the Extreme Value Theory to automatically determine outlier thresholds for each portrayer, and it supports concept drift adaptation. Empirical evaluation shows that Portray Learning clearly outperforms existing methods in both the simulated and real-world data streams. Moreover, Portray Learning is computationally efficient, requiring only tens of milliseconds for incremental updates.

The scenario of our proposal is widely applicable. In Fig. 11, each portrayer can analyze the traffic data from a unique social service, and each kind of news clicked by the user can be used for customized recommendations. Overall, prior SENC methods typically follow a three-stage pipeline: *new-class detection*, *known-class classification*, and *model update*. Our proposal aims to transform new class detection and known class classification as a whole into multiple outlier detection tasks, as described in Fig. 1. By constructing a series of independent portrayers corresponding to data classes one-to-one, the incoming instance will be identified as new classes or known ones. This manner of *each portrayer performing its own duty* offers a novel approach to addressing the SENC problem.

The performance of the portrayer could be limited when facing data poisoning, *e.g.,* some samples have inaccurate ground-truth labels. These low-quality datasets also affect other baselines. Our two-step design for concept drift adaptation could mitigate the impact on portrayers, and some forgetting mechanisms [39] may be combined. Meanwhile, sliding windows can be introduced to adapt to long-term data streams and unlearn outdated samples. Portray Learning brings new technology routes for class-incremental learning. In the future, we will explore more possible application scenarios (*e.g., zero-shot learning* [40]) of our scheme [41].

Currently, we mainly adopt the mean squared error (MSE) as the reconstruction loss for each portrayer. This choice is primarily motivated by its simplicity, numerical stability, and widespread adoption in reconstruction-based anomaly detection methods. In our setting, the reconstruction loss is not only used for anomaly detection within each class, but also serves as a comparable score across different class-wise portrayers, which favors a smooth and scale-consistent loss function such as MSE. We acknowledge that alternative loss functions, such as mean absolute error (MAE) [42], Huber loss [43], or perceptual similarity metrics [44], could also be employed in the portrayer framework and may exhibit different robustness properties under noise or heavy-tailed feature distributions. A systematic ablation study over different reconstruction losses may further improve performance in specific scenarios;

however, this exploration is beyond the scope of the current work. Importantly, the proposed Portray Learning paradigm is agnostic to the specific form of reconstruction loss, and replacing MSE with other suitable loss functions does not require any modification to the overall framework. We leave a comprehensive investigation of loss-function choices as future work.

CRediT authorship contribution statement

Ziming Zhao: Writing – review & editing, Writing – original draft, Validation, Methodology, Conceptualization. **Zhaoxuan Li:** Writing – original draft, Methodology. **Xiaofei Yue:** Writing – review & editing, Writing – original draft, Methodology. **Tingting Li:** Visualization, Validation. **Fan Zhang:** Writing – original draft, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (62227805, 62572437).

Data availability

Data will be made available on request.

References

[1] Y. Zhu, Y. Li, Semi-supervised streaming learning with emerging new labels, in: AAAI, AAAI Press, 2020, pp. 7015–7022.
[2] M.M. Masud, J. Gao, L. Khan, J. Han, B.M. Thuraisingham, Classification and novel class detection in concept-drifting data streams under time constraints, IEEE Trans. Knowl. Data Eng. 23 (6) (2011) 859–874.
[3] Z. Wu, H. Wang, J. Guo, Q. Yang, J. Shao, Learning evolving prototypes for imbalanced data stream classification with limited labels, Inf. Sci. 679 (2024) 120979.
[4] X. Mu, F. Zhu, J. Du, E. Lim, Z. Zhou, Streaming classification with emerging new class by class matrix sketching, in: AAAI, AAAI Press, 2017, pp. 2373–2379.
[5] E.R. de Faria, A.C.P. de Leon Ferreira de Carvalho, J. Gama, MINAS: multiclass learning algorithm for novelty detection in data streams, Data Min. Knowl. Discov. 30 (3) (2016) 640–680.
[6] X. Mu, K.M. Ting, Z. Zhou, Classification under streaming emerging new classes: a solution using completely-random trees, IEEE Trans. Knowl. Data Eng. 29 (8) (2017) 1605–1618.
[7] J. Beirlant, Y. Goegebeur, J. Segers, J.L. Teugels, Statistics of Extremes: Theory and Applications, John Wiley & Sons, 2006.
[8] M. Fink, S. Shalev-Shwartz, Y. Singer, S. Ullman, Online multiclass learning by interclass hypothesis sharing, in: ICML, Vol. 148 of ACM International Conference Proceeding Series, ACM, 2006, pp. 313–320.
[9] Y. Sun, K. Tang, L.L. Minku, S. Wang, X. Yao, Online ensemble learning of data streams with gradually evolved classes, IEEE Trans. Knowl. Data Eng. 28 (6) (2016) 1532–1545.
[10] F. Chen, S. Ranjan, P. Tan, Detecting bots via incremental LS-SVM learning with dynamic feature adaptation, in: KDD, ACM, 2011, pp. 386–394.
[11] Q. Da, Y. Yu, Z. Zhou, Learning with augmented class by exploiting unlabeled data, in: AAAI, AAAI Press, 2014, pp. 1760–1766.
[12] X. Cai, P. Zhao, K. Ting, X. Mu, Y. Jiang, Nearest neighbor ensembles: an effective method for difficult problems in streaming classification with emerging new classes, in: ICDM, IEEE, 2019, pp. 970–975.
[13] A. Haque, L. Khan, M. Baron, SAND: semi-supervised adaptive novel class detection and classification over data stream, in: AAAI, AAAI Press, 2016, pp. 1652–1658.
[14] E. Liberty, Simple and deterministic matrix sketching, in: KDD, ACM, 2013, pp. 581–588.
[15] W.J. Scheirer, A. de Rezende Rocha, A. Sapkota, T.E. Boult, Toward open set recognition, IEEE Trans. Pattern Anal. Mach. Intell. 35 (7) (2013) 1757–1772.
[16] Y. Li, L. Guo, Z. Zhou, Towards safe weakly supervised learning, IEEE Trans. Pattern Anal. Mach. Intell. 43 (1) (2021) 334–346.
[17] T. Wagner, S. Guha, S.P. Kasiviswanathan, N. Mishra, Semi-supervised learning on data streams via temporal label propagation, in: ICML, Vol. 80 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 5082–5091.
[18] F.T. Liu, K.M. Ting, Z. Zhou, Isolation forest, in: ICDM, IEEE Computer Society, 2008, pp. 413–422.
[19] S. Han, X. Hu, H. Huang, M. Jiang, Y. Zhao, Adbench: anomaly detection benchmark, in: NeurIPS, 2022.
[20] S. Guha, N. Mishra, G. Roy, O. Schrijvers, Robust random CUT forest based anomaly detection on streams, in: ICML, Vol. 48 of JMLR Workshop and Conference Proceedings, JMLR.org, 2016, pp. 2712–2721.
[21] L.I. Kuncheva, J.C. Bezdek, Nearest prototype classification: clustering, genetic algorithms, or random search? IEEE Trans. Syst. Man Cybern. Part C 28 (1) (1998) 160–164.
[22] H. Yang, X. Zhang, F. Yin, C. Liu, Robust classification with convolutional prototype learning, in: CVPR, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 3474–3482.
[23] S. Goel, P. Lymperopoulos, R. Thielstrom, E. Krause, P. Feeney, P. Lorang, S. Schneider, Y. Wei, E. Kildebeck, S. Goss, et al., A neurosymbolic cognitive architecture framework for handling novelties in open worlds, Artif. Intell. 331 (2024) 104111.
[24] G. Kim, C. Xiao, T. Konishi, Z. Ke, B. Liu, Open-world continual learning: unifying novelty detection and continual learning, Artif. Intell. 338 (2025) 104237.
[25] Z. Wang, Z. Kong, S. Chandra, H. Tao, L. Khan, Robust high dimensional stream classification with novel class detection, in: ICDE, IEEE, 2019, pp. 1418–1429.
[26] C. Simon, P. Koniusz, M. Harandi, On learning the geodesic path for incremental learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 1591–1600.
[27] A. Douillard, M. Cord, C. Ollion, T. Robert, E. Valle, Podnet: pooled outputs distillation for small-tasks incremental learning, in: ECCV (20), Vol. 12365 of Lecture Notes in Computer Science, Springer, 2020, pp. 86–102.
[28] O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation, in: MICCAI (3), Vol. 9351 of Lecture Notes in Computer Science, Springer, 2015, pp. 234–241.
[29] A. Siffer, P. Fouque, A. Termier, C. Largouët, Anomaly detection in streams with extreme value theory, in: KDD, ACM, 2017, pp. 1067–1075.
[30] J.P. III, Statistical inference using extreme order statistics, Ann. Stat. (1975) 119–131.
[31] A.A. Balkema, L.D. Haan, Residual life time at great age, Ann. Probab. (1974) 792–804.
[32] S.D. Grimshaw, Computing maximum likelihood estimates for the generalized pareto distribution, Technometrics (1993) 185–191.
[33] A.H. Madkour, H.M. Abdelkader, A.M. Mohammed, Dynamic classification ensembles for handling imbalanced multiclass drifted data streams, Inf. Sci. 670 (2024) 120555.

[34] T.U.K. Archive, KDD CUP 1999 data, [EB/OL], 1999, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (Accessed 25 December 2022).
[35] UCI machine learning repository, [EB/OL], 2022, https://archive.ics.uci.edu (Accessed 29 December 2022).
[36] C.-C. Chang, C.-J. Lin, Libsvm data: classification, regression, and multi-label, [EB/OL], 2011, https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/ (Accessed 27 December 2022).
[37] Dev portal, [EB/OL], 2011, http://developer.nytimes.com/ (Accessed 29 December 2022).
[38] Gensim, Topic modelling for humans, [EB/OL], 2022, https://radimrehurek.com/gensim/index.html (Accessed 3 February 2023).
[39] M. Du, Z. Chen, C. Liu, R. Oak, D. Song, Lifelong anomaly detection through unlearning, in: CCS, ACM, 2019, pp. 1283–1297.
[40] M. Palatucci, D. Pomerleau, G.E. Hinton, T.M. Mitchell, Zero-shot learning with semantic output codes, in: NIPS, Curran Associates, Inc., 2009, pp. 1410–1418.
[41] H. Larochelle, D. Erhan, Y. Bengio, Zero-data learning of new tasks, in: AAAI, AAAI Press, 2008, pp. 646–651.
[42] J. Qi, J. Du, S.M. Siniscalchi, X. Ma, C. Lee, On mean absolute error for deep neural network based vector-to-vector regression, IEEE Signal Process. Lett. 27 (2020) 1485–1489.
[43] P.J. Huber, Robust estimation of a location parameter, in: Breakthroughs in Statistics: Methodology and Distribution, Springer, 1992, pp. 492–518.
[44] R. Zhang, P. Isola, A.A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: CVPR, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 586–595.