Detecting Sequentially Novel Classes with Stable Generalization Ability

Da-Wei Zhou[1], Yang Yang[2], and De-Chuan Zhan[1(✉)]

[1] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
zhoudw@lamda.nju.edu.cn, zhandc@nju.edu.cn
[2] Nanjing University of Science and Technology, Nanjing 210023, China
yyang@njust.edu.cn

Abstract. Recognizing object of unseen novel classes is of great importance in real-world incremental data scenarios. Additionally, novel classes arise frequently in data stream mining, *e.g.,* new topics in opinion monitoring, and novel protein families in protein sequence classification. Conducting streaming novel class detection is a complex problem composed of detection and model update. However, when updating the model solely with detected novel instances, it concentrates more on the novel patterns than known ones; thus the detection ability of the model may degrade consequently. This would exert harmful affections to further classification as the data evolving. To this end, in this paper, we consider the accuracy of the detection along with the robustness of model updating, and propose DEtecting Sequentially Novel clAsses with Stable generalization Ability (DesNasa). Specifically, DesNasa utilizes a prototypical network to reflect the structure information between scattered prototypes for novel class detection. Furthermore, the well-designed data augmentation method can help the model learning novel patterns robustly without degrading detection ability. Experiments on various datasets successfully validate the effectiveness of our proposed method.

Keywords: Novel class detection · Data stream classification · Open world learning

1 Introduction

In our dynamically evolving world, data is often with stream format, which may only be available temporarily for storage constraints or privacy issues. To tackle this, many incremental algorithms are proposed, *e.g.,* open-ended object detection [16], incremental image classification [17] and online video classification [21]. While these methods ignore an essential issue of streaming data, namely the emergence of unseen category [25], *e.g.,* in online opinion monitoring, new topics often emerge as news happens [13]; in protein sequence classification, new types of proteins would arise as nature evolves [23]. Under such circumstances, the problem can be decomposed into three sub-problems: detecting novel classes,

K. Karlapalem et al. (Eds.): PAKDD 2021, LNAI 12712, pp. 371–382, 2021.
https://doi.org/10.1007/978-3-030-75762-5_30

classifying known instances, and updating model in the data stream [14]. As a result, traditional anomaly detection methods cannot handle this situation. They only stress one of the three tasks as novelty detection, thus triggering the development in streaming sequentially novel class detection.

There is much research regarding efficiently detecting novel classes with model updating. [6] detects outliers having strong cohesion among themselves, and uses ensemble of clustering models for classification. [3] utilizes nearest neighbor ensemble to deal with problems of different geometric distances. [13] assumes the label of stream data is available after some time delay and adopts clustering method for novelty detection and known classification. [14] utilizes completely-random trees with growing mechanism for model update, and [15] approximates original information by a dynamic low-dimensional structure via matrix sketching. However, great hidden trouble often occurs when deploying these methods with sequentially new classes, namely *detection ability degrading* [6]. Since these methods are designed for detecting and model update in a few periods, they ignore the ability of forecasting in the long future. That is, solely updating the model with limited novel instances incorporates imbalanced confusion into the current model. The performance of the model shall suffer a decline in the following periods, which harms both the accuracy of classifying known classes and the ability to detect the unknown novel classes. As a result, after several times of updating, error accumulates and the models will fail to predict. These two problems are concurrent, and impose the challenge of detection efficiency and detection stability to exploit these methods for developing algorithms.

To solve the co-occurred problems in class incremental learning, we propose detecting sequentially novel classes with stable generalization ability (DesNasa), which can identify the emergency new classes and learn novel patterns simultaneously without degrading detection ability. Specifically, DesNasa utilizes a prototypical network with intra-class compactness, aiming to acquire the ability of learning distance function over input space and the task space. Thus, the framework can consider information from the scattered prototypes and incoming instance for new class detection. Moreover, the prototypes can help in maintaining forecasting ability when learning novel patterns by mixed replay. Consequently, we can detect and learn novel patterns more efficiently and meanwhile maintain stable performance during the data stream evolution.

2 Related Work

Three key sub-problems, *i.e.,* detecting emerging new classes, classifying known instances, and updating models with detected novel patterns form the current problem. Traditional anomaly detection methods [12,22] mainly focus on identifying anomaly or outlier under static environment. However, these methods need the whole dataset to process, which is not suitable for streaming format data in the dynamic environment. Consequently, several online novelty detection methods have been proposed. [2] extends novelty detection to dynamic streams, while the detection component and the update component differs in these methods, which results in accumulating error and hard to tune parameters. On the

other hand, these methods are designed for novelty detection, which is a binary classification problem. While in streaming novel class detection, a significant difference lies in that anomaly instances and known instances should be assigned with certain class labels [14]. The model should have the ability to classify known instances and update with data stream, thus making the above methods incapable of stream novel class detection scenarios. To solve the three sub-problems in a unified framework, many works try to introduce additional information in stream mining. [13] used a clustering approach ECSMiner which tackles the novel class detection and classification problems by introducing time constraints for delayed classification. LACU [4] needs an unlabeled dataset to help classification. [14] utilizes completely-random trees with growing mechanism, and [3] studied nearest neighbor ensemble for different geometric distances. These methods need additional information such as saving relatively large subsets of the original data or obtaining an extra unlabeled dataset. However, heuristically saving subsets consumes huge memory and complex select cost.

Other lines of study involve open-set recognition [5] and active learning [8]. Open-set recognition aims to conduct classification on known class instances and reject unknown ones for *pre-collected* datasets. However, it differs with the current scenario that the open-set model cannot update incrementally, thus these methods are incapable of handling data streams. Active learning is designed for achieving greater performance with fewer chosen labeled training instances. In the current problem, only novel instances are chosen to update the model, which can be seen as a particular case of active learning. Nevertheless, our target is not only learning novel patterns but also maintaining the knowledge of former classes, which is outside the scope of active learning.

3 Preliminaries

Notations. Given the initial model $\mathcal{M}$ pretrained on previous data, which contains k classes, we have the streaming data as: $S = \{(\mathbf{x}_t, y_t)\}_{t=1}^{\infty}$, where $\mathbf{x}_t \in R^d$ and $y_t \in Y = \{1, 2, \ldots, k, k+1, \ldots, K\}$ with $K \gg k$, t is the timestamp. The goal is to utilize $\mathcal{M}$ as a detector for emerging new class and a classifier for known class [3,14]. Then $\mathcal{M}$ is updated timely such that it maintains accurate predictions for known and emerging new classes on streaming data S.

Problem Overview. The goal is to detect novel candidates and classify known instances in the data stream. Stream data are fed into the model, and it scores each instance with the probability of being a novel class. Instances predicted as known classes would be passed to the classifier for prediction, while those predicted as novel instances are saved in the buffer $\mathcal{B}$. After the prediction of every period, the model will request the true label of instances in the buffer to reduce query cost, and remove instances belonging to existing classes. *Only true-novel instances in the buffer will be used to update the model* [14]. The left part of Fig. 1 shows the framework. More novel classes will arise, and the model can learn to classify with time.

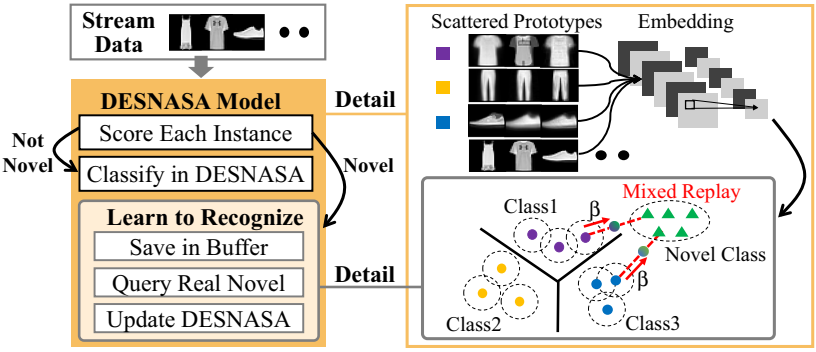


Fig. 1. Illustration of the framework. Stream data are fed into the model, and DesNasa scores each instance of being novel. Instances predicted as novel are saved in the buffer. Only real-novel instances in the buffer will be used to update the model. Specifically, it assigns probability by the normalized distance between instance and prototypes in embedding space. In the update stage, prototypes and novel instances are mixed for alleviating detection ability decay.

4 Proposed Method

4.1 The Framework of DesNasa

Since the stream data is dynamic and ever-changing, a proper way to model each class is to pick prototypical samples. To this end, we design a prototypical deep model, which embeds scattered class centers and arrange probability of belonging to a known class from distance in the embedding space. In detail, DesNasa utilizes a deep network as embedding module $\mathcal{E}(\cdot)$. Besides, the class centers in the data steam can be maintained to represent every class, leading to the class center set C. Then for class k, the mean of all instances belong to this class can be calculated as: $\mathbf{c}_k = \frac{1}{|\Omega_k|}\sum_{\mathbf{x}\in\Omega_k}\mathbf{x}$. In consideration of the property of data stream, we can maintain class centers through an incremental way:

$$\hat{\mathbf{c}}_k = \mathbf{c}_k + \frac{\mathbf{x} - \mathbf{c}_k}{|\Omega_k| + 1}. \tag{1}$$

In order to obtain more diverse class centers, we introduce scattered class centers with instance picking probability p:

$$\hat{\mathbf{c}}_{k,p} = \mathbf{c}_{k,p} + \mathbb{I}(\gamma < p)\frac{\mathbf{x} - \mathbf{c}_{k,p}}{|\Omega_k| + 1}, \tag{2}$$

where $\mathbb{I}(\cdot)$ is the indicator function, and γ is randomly sampled from $\mathcal{U}(0, 1)$. We define the picking probability of each prototype by dividing $0 \sim 1$ into P equal parts, where P is the number of prototypes maintained for each class. That is, prototypes with higher p will be calculated with more instances from this class, thus different prototypes of same class would reveal diversity from each other.

We denote scattered prototypes as C_{ij}, where $i \in \{1, 2, \cdots, k\}$ represents class index and $j \in \{1, 2, \cdots, P\}$ stands for prototype index.

After calculating the scattered prototypes, the embedding module $\mathcal{E}(\cdot)$ will then produce feature maps of $\mathcal{E}(C_{ij})$ and $\mathcal{E}(\mathbf{x})$ in the embedding space. We assume that the distance in the embedding space can be used to measure the similarity between samples and prototypes, *i.e.*, $p\left(\mathbf{x} \in C_{ij} \mid \mathbf{x}\right) \propto -\left\|\mathcal{E}(\mathbf{x}) - \mathcal{E}(C_{ij})\right\|_2^2$. Thus, by normalizing the distance of all known prototypes, we are able to acquire the probability of $\mathbf{x}$ belonging to prototype C_{ij} by:

$$p\left(\mathbf{x} \in C_{ij} \mid \mathbf{x}\right) = \frac{e^{-d(\mathcal{E}(\mathbf{x}), \mathcal{E}(C_{ij}))/\tau}}{\sum_{m=1}^{k} \sum_{n=1}^{P} e^{-d(\mathcal{E}(\mathbf{x}), \mathcal{E}(C_{mn}))/\tau}}, \tag{3}$$

where $d\left(\mathcal{E}(\mathbf{x}), \mathcal{E}(C_{ij})\right) = \left\|\mathcal{E}(\mathbf{x}) - \mathcal{E}(C_{ij})\right\|_2^2$ represents the distance between $\mathcal{E}(\mathbf{x})$ and $\mathcal{E}(C_{ij})$, τ corresponds to the temperature which controls the smoothness of probability assignment. We further get the probability of $p(y \mid \mathbf{x})$ by summing the probability belonging to prototypes of same class up:

$$p(y \mid \mathbf{x}) = \sum_{j=1}^{P} p\left(\mathbf{x} \in C_{yj} \mid \mathbf{x}\right). \tag{4}$$

Considering the probability calculated from embedding space, we then optimize the embedding module by defining distance-based cross entropy:

$$Loss = -\log p(y \mid \mathbf{x}) = -\log\left(\sum_{j=1}^{P} p\left(\mathbf{x} \in C_{yj} \mid \mathbf{x}\right)\right). \tag{5}$$

During the training process, we optimize the model by minimizing loss function Eq. 5. Then for instances from known class, the model maximizes the probability of $\mathbf{x}$ being associated with a prototype among prototype set C. Thus it decreases the distance between incoming instances with the centroids from the genuine class of the samples. We then introduce the novel class detection algorithm.

4.2 Novel Class Detection

Classical novel class detection methods base on predict uncertainty or structure information [3,14], and score each instance with the probability of being novel. These methods will then manually set fixed threshold to determine the decision boundary between unknown and known. However, the fixed threshold differs in different tasks with different input distribution and thus hard to tune. In this section, we develop a suitable strategy to set a dynamic threshold, which can utilize the predicted uncertainty with structure information for better separation.

During the stream data deployment, for every instance in the stream S, we should determine if it belongs to a known class or a novel class. Considering the probability arranged by embedding distance in Eq. 3. Instances with higher probability are those closer to corresponding prototypes in the embedding

Algorithm 1. Novel Class Detection

Input: Data stream : $S = \{\mathbf{x}_t\}_{t=1}^{\infty}$
Output: Class label of each $\mathbf{x}_t$

Empty buffer $\mathcal{B} \leftarrow \varnothing$;
repeat
  Get a test instance $\mathbf{x}_t$;
  Calculate the embedding of instance $\mathcal{E}(\mathbf{x}_t)$;
  Calculate the embeddings of prototypes $\mathcal{E}(C_{ij}), \forall i = 1, 2, \cdots, k; \forall j = 1, 2, \cdots, P$;
  Calculate normalized probability $p(y \mid \mathbf{x}_t) \leftarrow$ Eq.4 ;
  Calculate the confidence of predicted instance $Conf(\mathbf{x}_t) = \max p(y \mid \mathbf{x}_t)$;
  if $Conf(\mathbf{x}_t) <$ dynamic threshold dt **then**
    Save $\mathbf{x}_t$ in the buffer; $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{x}_t\}$;
    $y_t = k + 1$, Predict as a novel class item;
    Update dummy class center $C_{novel} \leftarrow$ Eq.1;
    Update dynamic threshold $dt \leftarrow$ Eq.7;
  else
    $y_t = \underset{y=1,\ldots,k}{\operatorname{argmax}}\, p(y \mid \mathbf{x})$, predict as a known class;
  end if
until Stream is empty

space. This indicates that the confidence of the model is equivalent to the output highest probability $\max p(y \mid \mathbf{x})$ of Eq. 4. For a known instance, we have the scattered prototypes maintained in the training process, and the corresponding confidence should be relatively high. But for an unseen novel instance, we have no prior about what this class is like, and the arranged probability shall separate between all known prototypes; thus the model would not indicate relatively high confidence than a known instance. We can design the detection formula of the model:

$$f(\mathbf{x}) = \begin{cases} \underset{y=1,\ldots,k}{\operatorname{argmax}}\, p(y \mid \mathbf{x}) & \text{if } \max p(y \mid \mathbf{x}) > \text{ threshold} \\ k+1 & \text{otherwise} \end{cases} . \tag{6}$$

Note that the *threshold* is designed to tell apart known instances from novel classes, which is hard to tune between different tasks. We then design a *dummy class* method for convenient threshold implementation. In detail, once we detect novel-like candidates in the test stream, we then save them in the novelty buffer $\mathcal{B}$. Buffered instances will form the dummy class to estimate novel class distribution by forming novel prototypes with Eq. 1. The dynamic threshold dt is updated by:

$$\hat{dt} = \lambda * dt + (1 - \lambda) * (\max p(y \mid C_{dummy}) + m) , \tag{7}$$

where λ is a trade-off parameter close to 1, and m is a positive margin, which aims at making the threshold converge to the confidence of novel prototype and keep it slightly higher. And m can be estimated with $\eta std(\max p(y \mid C_{dummy}))$. Considering when initially define the threshold, it may not well stand for the cutoff between known classes and novel class. Each time the model detects a novel candidate, the dynamic threshold will be updated through Eq. 7. Thus it

Algorithm 2. Learn to Recognize

Input: Predicted novel instance buffer: $\mathcal{B} = \{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^{m}$
Output: Updated model : $\mathcal{M}$

Request true labels from $\mathcal{B}$, and remove instances belonging to known classes.
for each novel instance $(\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{B}$ **do**
  Update novel class center $\leftarrow$ Eq.2
  Sample a known class prototype $(\mathbf{x}_i, \mathbf{y}_i)$
  Mix known class center and novel instance $\leftarrow$ Eq.8
  Calculate the embedding of the mixed instance $\mathcal{E}(\tilde{\mathbf{x}})$
  Calculate the embeddings of prototypes $\mathcal{E}(C_{ij}), \forall i = 1, 2, \cdots, k; \forall j = 1, 2, \cdots, P$;
  Calculate the normalized probability $p(y \mid \tilde{\mathbf{x}}) \leftarrow$ Eq.4
  Calculate distance-based cross-entropy $L_j \leftarrow$ Eq. 5
  Obtain the derivative $\frac{\partial L_j}{\partial \Theta}$, update model;
end for

steps closer towards the ideal cutoff between novel dummy prototypes and known prototypes. The algorithm for novel class detection is shown in Algorithm 1.

4.3 Learn to Recognize

Another essential part of DesNasa is to learn the pattern of novel class without degrading detection ability. That is, how to update the model with only true-novel instances in the buffer. Traditional methods based on structure information may fail for unable to depict the novel class distribution. In contrast, DNN-based methods may suffer severe imbalanced learning due to pure novel instances. While in our model, we solve the problem by learning *mixed replay*.

For our model, a straightforward way to learn the novel patterns is to conduct stochastic gradient descent on these pure-novel instances. While the performance of the model will degrade when novel instances incorporating. This is caused by the imbalanced learning phase in model updating, which harms the prior knowledge, and thus makes the model incapable of forecasting as data evolves. The model should concentrate equally on both former and current distributions.

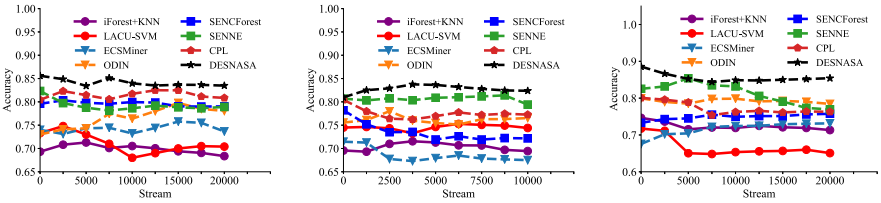
Considering DesNasa maintains the prototypes in the data stream, which stand for the data from the former distribution. A proper way is to retrain them when updating the model with novel instances. That is, to train with the prototype along with novel instances. Thus the model can optimize for novel patterns and meanwhile preserving former classes knowledge. However, a fatal problem of overfitting often occurs when conducting many times of retraining on limited prototypes [7]. To solve this problem, we design mixed replay which combines scattered class centers along with novel instances to form mixed instance:

$$\tilde{\mathbf{x}} = \beta \mathbf{x}_i + (1 - \beta)\mathbf{x}_j; \quad \tilde{\mathbf{y}} = \beta \mathbf{y}_i + (1 - \beta)\mathbf{y}_j \ , \tag{8}$$

where $(\mathbf{x}_i, \mathbf{y}_i)$ is a known prototype, and $(\mathbf{x}_j, \mathbf{y}_j)$ is one novel class instance, $\mathbf{y}_i, \mathbf{y}_j$ are corresponding one-hot label encodings, and $\beta \in [0, 1]$, we utilize the interpolation of one known class center and one novel instance to force the model concentrate on knowledge learned before. [24] proved Eq. 8 establishes a linear

Table 1. Prediction results (mean ± standard deviation) of different compared methods on simulated streams. The best performance is bolded.

Methods	MNIST		Fashion-MNIST		HAR		CIFAR10	
	F-Measure	Accuracy	F-Measure	Accuracy	F-Measure	Accuracy	F-Measure	Accuracy
iForest+KNN	0.697 ± 0.04	0.687 ± 0.05	0.623 ± 0.03	0.679 ± 0.06	0.678 ± 0.02	0.696 ± 0.02	0.627 ± 0.04	0.666 ± 0.03
ODIN	0.752 ± 0.05	0.787 ± 0.05	0.744 ± 0.04	0.729 ± 0.03	0.776 ± 0.02	0.783 ± 0.01	0.801 ± 0.03	0.796 ± 0.03
LACU-SVM	0.699 ± 0.06	0.705 ± 0.08	0.633 ± 0.06	0.677 ± 0.04	0.691 ± 0.05	0.752 ± 0.03	0.644 ± 0.02	0.652 ± 0.04
ECSMiner	0.714 ± 0.06	0.737 ± 0.09	0.700 ± 0.02	0.746 ± 0.03	0.660 ± 0.04	0.682 ± 0.06	0.727 ± 0.05	0.731 ± 0.05
SENCForest	0.747 ± 0.04	0.763 ± 0.03	0.779 ± 0.02	0.781 ± 0.06	0.762 ± 0.06	0.782 ± 0.04	0.766 ± 0.03	0.717 ± 0.03
SENNE	0.807 ± 0.04	0.789 ± 0.04	0.794 ± 0.06	0.783 ± 0.05	0.811 ± 0.02	0.808 ± 0.03	0.798 ± 0.03	0.770 ± 0.02
CPL	0.747 ± 0.06	0.755 ± 0.06	0.766 ± 0.08	0.750 ± 0.07	0.744 ± 0.04	0.785 ± 0.06	0.764 ± 0.03	0.768 ± 0.03
DesNasa	**0.830 ± 0.03**	**0.829 ± 0.03**	**0.816 ± 0.03**	**0.812 ± 0.02**	**0.819 ± 0.03**	**0.824 ± 0.02**	**0.862 ± 0.03**	**0.860 ± 0.03**



(a) MNIST (b) HAR (c) CIFAR10

Fig. 2. Accuracy curve of different methods over sequential stages.

relationship between data augmentation and the supervision signal, and thus leads to a strong regularizer that improves generalization. Considering the phenomenon that solely replays over instances from former distribution shall cause overfitting, Mixed replay offers a simple way to overcome this drawback.

Indeed, after many rounds of training with mixed replay $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, the model acquires the pattern of novel classes and meanwhile consolidates prior knowledge. The guideline for learning to recognize novel class is shown in Algorithm 2.

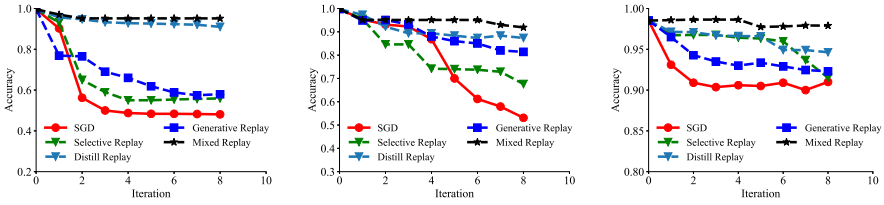
5 Experiments

Datasets and Configurations. We provide the empirical results and performance comparison of DesNasa. In particular, we test with 4 benchmark datasets, *i.e.,* MNIST [10], Fashion-MNIST [19], CIFAR10 [9], and HAR [1], and 1 real-world incremental dataset, *i.e.,* NYTimes [14], which consists of 35,000 latest news crawled with the New York Times API. Each news item is classified into six categories, namely, 'Arts', 'Business Day', 'Sports', 'U.S.', 'Technology' and 'World'. Each instance is converted into a 100 dimension vector with word2vec.

Following the typical setting as [3,14], to simulate emerging new classes in the data stream, we assume that training set D with two known classes is available at the initial stage. Instances of these two known classes and an emerging new class appear in the first period of the data stream with uniform distribution. In the second period, instances of these three classes seen in the first period and

Table 2. Prediction results (mean ± standard deviation) of different compared methods on novel class forecasting, which stands for performance of the model during next period when updating with different compared methods.

Methods	MNIST		Fashion-MNIST		HAR		CIFAR10	
	F-Measure	Accuracy	F-Measure	Accuracy	F-Measure	Accuracy	F-Measure	Accuracy
SGD	0.310 ± 0.06	0.386 ± 0.05	0.619 ± 0.05	0.656 ± 0.05	0.626 ± 0.09	0.676 ± 0.07	0.671 ± 0.05	0.705 ± 0.04
Selective Replay	0.410 ± 0.04	0.463 ± 0.03	0.604 ± 0.07	0.648 ± 0.08	0.640 ± 0.07	0.680 ± 0.04	0.681 ± 0.06	0.707 ± 0.05
Distill Replay	0.496 ± 0.08	0.552 ± 0.06	0.666 ± 0.09	0.693 ± 0.08	0.656 ± 0.05	0.691 ± 0.04	0.772 ± 0.02	0.768 ± 0.02
Generative Replay	0.401 ± 0.09	0.468 ± 0.08	0.498 ± 0.06	0.557 ± 0.07	0.581 ± 0.07	0.609 ± 0.09	0.680 ± 0.02	0.729 ± 0.02
Mixed Replay	**0.832 ± 0.03**	**0.831 ± 0.03**	**0.834 ± 0.07**	**0.835 ± 0.05**	**0.792 ± 0.06**	**0.797 ± 0.05**	**0.863 ± 0.02**	**0.862 ± 0.02**



(a) MNIST (b) HAR (c) CIFAR10

Fig. 3. The test accuracy on initial classes when model update with different methods.

another emerging new class appear with uniform distribution. Instances appear one at a time, and the model should make a prediction for every instance before processing the next. Each dataset is used to simulate a data stream over five trails, and both mean and standard variance of the performance are reported. Considering novel classes may arise simultaneously in a single period, while some compared methods cannot handle it. We simulate one novel class to arrive in a single period for a fair comparison.

Compared Methods. We first compare to a common-used novelty detection method iForest [12] and deep anomaly detection algorithm ODIN [11]. We also compare to the state-of-the-art methods which combine novel class detection with model update: LACU-SVM [4], ECSMiner [13], SENC-Forest [14], SENNE [3]. Convolutional prototype learning method CPL [20] is also compared. We also conduct experiments to evaluate the forecasting ability of the model. We adopt the *same* embedding module and detection rules for novel class detection, and 4 major methods, *i.e.,* SGD, Selective Replay [7], Generative Replay [18] and Distill Replay [17] are implemented for the model update process: **SGD**: update model with only novel instances; **Selective Replay**: save former instances with reservoir sampling, and retrain them with the novel instance at update stage; **Generative Replay**: training generative model for former data distribution, and rehearsal generated data with novel instance to update; **Distill Replay**: consider knowledge distillation loss of former instance when updating the model. Note that the latter three methods are initially proposed to overcoming forgetting of former knowledge in incremental learning [21],

which can be viewed as a similar problem as the detection ability decline in our paper.

F-Measure and Accuracy are recorded to evaluate the detection and classification performance, respectively [3,14]. F-measure evaluates the effect of novel class detection, which is calculated at several intervals and use the average as the final evaluation. Accuracy= $\frac{A_{\text{novel}} + A_{\text{known}}}{m}$, where m is the total number of stream instances, A_{novel} is the number of novel instances identified correctly, A_{known} is the number of known class instances classified correctly.

Real-World Stream Data Classification. The stream prediction results, which calculate the Accuracy and F-measure are reported in Table 1. Figure 2 shows the prediction accuracy curve on corresponding simulated streams. From Table 1, it reveals that for all datasets, DesNasa almost consistently achieves the significant superior performance comparing to other methods. LACU-SVM requires additional unlabelled instances in training and every model update, and ECSMiner needs all the true labels to train a new classifier, and they still performed worse than DesNasa in both measures. The performance of iForest is dissatisfactory on the dataset with high dimensions for it is tree-based, which only uses a few dimensions for tree building. SENNE needs to store huge amounts of data/ensemble models, but our DesNasa still outperforms it in all datasets. ODIN and CPL utilize the same DNN backbone as DesNasa , while they reveal unsatisfactory results for the fixed threshold is hard to tune, but DesNasa can dynamically adjusting threshold with data stream evolving.

Detection Ability Maintenance. We also conduct experiments about detection ability maintenance. Table 2 and Fig. 3 shows the result of *same base model* (embedding and detection rules) updating with different methods. In Fig. 3, after updating with other compared methods, prior knowledge of the model will be disturbed, which results in the degrading of detection ability. Additionally, Table 2 shows the corresponding prediction performance of the *next* period when updating with these methods, which reveals that the forecasting ability will be degraded with other methods. In comparison, the model updated with mixed replay suffers the least disturbance in model updating, and can achieve the best performance in the next-period prediction.

Stream News Identification. In this part, we conduct experiments in a real-world text stream, *i.e.,* NYTimes. News categorization for a news stream is an important issue, where a new topic of news may arise due to a newly occurred event. News data are categorized into six classes, and we treat two of them as known classes at the initial stage and simulate others to appear sequentially in the later stages. Figure 4 shows the experiment results. We can infer from Fig. 4(a) that in a real-world text stream, DesNasa can still achieve the best performance among all compared methods. Figure 4(b) indicates the mixed replay in DesNasa can help the model dealing with sequentially novel classes with the least detection ability degrading.

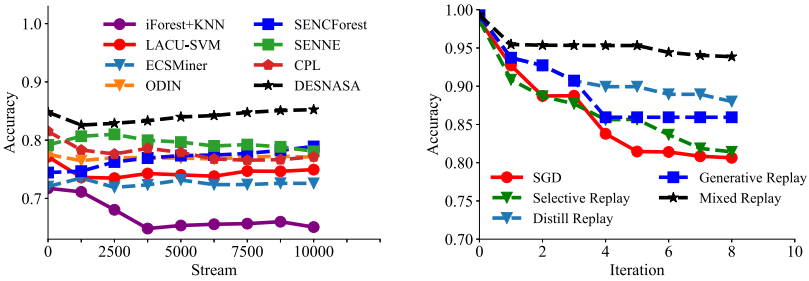


Fig. 4. Accuracy curve on NYTimes. Left: prediction accuracy with data evolves. Right: test accuracy on initial known classes when model update with different methods.

6 Conclusion

In many real-world incremental data scenarios, data arise with novel classes. To solve novel class detection and model extension problem, we propose a novel framework to detect sequentially novel classes with stable generalization ability (DesNasa). It can efficiently detect the novel class and extend the model with the least detection ability degrade. More specifically, DesNasa can detect novel class instances by utilizing the structure information of a prototypical network. Furthermore, it can learn to recognize through data augmentation between prototypes and novel instances without degrading detection ability. Experiments on image and text datasets successfully validate DesNasa stabilizes the performance of the regularly updated model.

Acknowledgments. This research was supported by NSFC (61773198, 61632004, 61921006, 62006118), NSFC-NRF Joint Research Project under Grant 61861146001, Collaborative Innovation Center of Novel Software Technology and Industrialization, CCF- Baidu Open Fund (CCF-BAIDU OF2020011), Baidu TIC Open Fund, Natural Science Foundation of Jiangsu Province of China under Grant (BK20200460) and Nanjing University Innovation Program for PhD candidate (CXYJ21-53). De-Chuan Zhan is the corresponding author.

References

1. Aggarwal, C.C.: A survey of stream clustering algorithms. In: Data Clustering, pp. 231–258. Chapman and Hall/CRC (2018)
2. Ahmad, S., Lavin, A., Purdy, S., Agha, Z.: Unsupervised real-time anomaly detection for streaming data. Neurocomputing **262**, 134–147 (2017)
3. Cai, X.-Q., Zhao, P., Ting, K.-M., Mu, X., Jiang, Y.: Nearest neighbor ensembles: an effective method for difficult problems in streaming classification with emerging new classes. In: ICDM, pp. 970–975. IEEE (2019)
4. Da, Q., Yu, Y., Zhou, Z.-H.: Learning with augmented class by exploiting unlabeled data. In: AAAI, pp. 1760–1766 (2014)

5. Geng, C., Huang, S.-J., Chen, S.: Recent advances in open set recognition: a survey. TPAMI (2020)
6. Haque, A., Khan, L., Baron, M.: Sand: semi-supervised adaptive novel class detection and classification over data stream. In: AAAI, vol. 16, pp. 1652–1658 (2016)
7. Isele, D., Cosgun, A.: Selective experience replay for lifelong learning. In: AAAI, pp. 3302–3309 (2018)
8. Konyushkova, K., Sznitman, R., Fua, P.: Learning active learning from data. In: NIPS, pp. 4225–4235 (2017)
9. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Technical report, Citeseer (2009)
10. LeCun, Y., Cortes, C., Burges, C.J.: Mnist handwritten digit database. AT&T Labs, 2:18 (2010). http://yann.lecun.com/exdb/mnist
11. Liang, S., Li, Y., Srikant, R.: Enhancing the reliability of out-of-distribution image detection in neural networks. In: ICLR (2018)
12. Liu, F.T., Ting, K.M., Zhou, Z.-H.: Isolation forest. In: ICDM, pp. 413–422 (2008)
13. Masud, M., Gao, J., Khan, L., Han, J., Thuraisingham, B.M.: Classification and novel class detection in concept-drifting data streams under time constraints. TKDE **23**(6), 859–874 (2010)
14. Mu, X., Ting, K.M., Zhou, Z.-H.: Classification under streaming emerging new classes: a solution using completely-random trees. TKDE **29**(8), 1605–1618 (2017)
15. Mu, X., Zhu, F., Du, J., Lim, E.-P., Zhou, Z.-H.: Streaming classification with emerging new class by class matrix sketching. In: AAAI, pp. 2373–2379 (2017)
16. Perez-Rua, J.-M., Zhu, X., Hospedales, T.M., Xiang, T.: Incremental few-shot object detection. In: CVPR, pp. 13846–13855 (2020)
17. Rebuffi, S.-A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: incremental classifier and representation learning. In: CVPR, pp. 2001–2010 (2017)
18. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: NeurIPS, pp. 2990–2999 (2017)
19. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-Mnist: a novel image dataset for benchmarking machine learning algorithms. preprint arXiv:1708.07747 (2017)
20. Yang, H.-M., Zhang, X.-Y., Yin, F., Liu, C.-L.: Robust classification with convolutional prototype learning. In: CVPR, pp. 3474–3482 (2018)
21. Yang, Y., Zhou, D.-W., Zhan, D.-C., Xiong, H., Jiang, Y.: Adaptive deep models for incremental learning: considering capacity scalability and sustainability. In: SIGKDD, pp. 74–82 (2019)
22. Yang, Y., Zhang, J., Carbonell, J., Jin, C.: Topic-conditioned novelty detection. In: SIGKDD, pp. 688–693 (2002)
23. Zhang, D., Liu, Y., Si, L.: Serendipitous learning: learning beyond the predefined label space. In: SIGKDD, pp. 1343–1351 (2011)
24. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: beyond empirical risk minimization. In: ICLR (2018)
25. Zhou, Z.-H.: Learnware: on the future of machine learning. Front. Comput. Sci. **10**(4), 589–590 (2016)