# Unsupervised Deep Embedding for Novel Class Detection over Data Stream

Ahmad M. Mustafa, Gbadebo Ayoade, Khaled Al-Naami,
Latifur Khan, Kevin W. Hamlen, Bhavani Thuraisingham
*Department of Computer Science*
*The University of Texas at Dallas*
*Richardson, Texas 75080*
{*ahmad.mustafa, gbadebo.ayoade, khaled.al-naami,*
*lkhan, hamlen, bhavani.thuraisingham*}*@utdallas.edu*

Frederico Araujo
*IBM Research*
*Thomas J. Watson Research Center*
*Yorktown Heights, NY 10598*
*frederico.araujo@ibm.com*

***Abstract*—Data streams are continuous flows of data points. Novel class detection is an important part of data stream mining. A novel class is a newly emerged class that has not previously been modeled by the classifier over the input stream. This paper proposes deep embedding for novel class detection—a novel approach that combines feature learning using denoising autoencoding with novel class detection. A *denoising autoencoder* is a neural network with hidden layers aiming to reconstruct the input vector from a corrupted version. A nonparametric multidimensional change point detection approach is also proposed, to detect concept-drift (the change of data feature values over time). Experiments on several real datasets show that the approach significantly improves the performance of novel class detection.**

***Keywords*-Stream Mining; Novel Class Detection; Concept-drift; Deep Learning**

## I. Introduction

Data stream classification has been a major research thrust for the past several years because of increasing demand in many business and security applications, such as credit card transaction monitoring, online blog or micro-blog (e.g., twitter message) categorization, and evolving malicious code detection. Traditional batch classification techniques are not applicable to data stream classification problems because of the evolving nature of the data. Data streams are continuous flows of data. Typical examples of data streams include network traffic, sensor data, and call center records, among others. Their sheer volume and throughput speed pose a great challenge for the data mining community to extract useful knowledge from such streams.

Data streams induce several unique properties when compared with traditional datasets, such as infinite length, concept-drift, and concept-evolution. Concept-drift occurs in data streams when the underlying concept of data changes over time [1], [2]. Concept-evolution occurs when new classes emerge in the data stream [3]. Multi-step techniques and multi-scan algorithms that are suitable for typical knowledge discovery and data mining cannot be readily applied to data streams due to well-known limitations [4], such as unbounded memory to handle infinite length, online data processing to handle concept-drift, and the need for one-pass techniques (i.e., forgotten raw data).

Stream classification falls into three categories: single model, ensemble classification, and hybrid. Single model classification techniques maintain and incrementally update the single classification model and effectively respond to concept-drift [5]. In ensemble based techniques, a collection of classification models are maintained, and over time some outdated classification models are replaced by new models [3]. Hybrid methods combine the strength of single and ensemble models [6]. Current state-of-the-art techniques suffer from a high number of misclassifications, such as missing novel class instances (false negatives) or incorrectly identifying existing classses as novel (false positives).

Concept-drift refers to the change of data feature values over time. To prevent significant performance degradation due to concept-drift, the classification model needs continual updating and maintenance via retraining. However, given that model updating is time consuming and requires labeled data, which is usually scarce, determining appropriate retraining frequencies becomes critical. A naïve approach is to retrain the classifier periodically [4]. However, this strategy results in unnecessary updates and missed drifts, which can degrade performance. Such limitations can be avoided by retraining only when the distribution of the data changes [7]. Toward this end, *change point detection* (CPD) techniques can be applied to observe and detect distributional changes.

CPD can also be applied to partition the data stream. The state-of-the-art stream classification techniques divide data streams in equal sizes (i.e., fixed size) [8], [1]. These approaches fail to capture concept-drift and concept-evolution immediately. Moreover, if the data chunk is too small, such techniques can engender models of poor quality (few training data points) and/or additional computational overhead to update the ensemble. Conversely, for large chunk sizes, these approaches must wait much longer to build the next classifier. As a result, the ensemble is updated less frequently than desired, meaning the ensemble remains outdated for a longer period of time. This ultimately causes increased error rates.

To overcome these disadvantages, our approach monitors data streams to determine the chunk size dynamically by promptly tracking distributional changes. Most of the CPD algorithms assume that distributions of data before and after the change are known [9], [10]. However, in practice, this assumption may not always hold. If the distributions of data before and after are unknown entirely, these approaches are not applicable. For this specific case, methods have been developed for detecting change points nonparametrically over single-dimensional data [11], [12]. If data is multidimensional, these approaches detect changes after reducing the number dimensions (e.g., using PCA in [13]). In contrast, our proposed CPD approach can be scaled over multidimensional data without reducing dimensionality. Moreover, since it is a nonparametric approach, we do not assume a particular data distribution—it can detect change points regardless of the distribution of the data. In addition, current state-of-the-art CPD techniques [7], [14] require the availability of class labels for some or all data points. Our proposed CPD technique is unsupervised and does not require class labels.

In addition to CPD, both clustering and outlier detection are essential parts of our novel class detection. Studies have shown that using deep learning techniques in classification, clustering and outlier detection have significantly outperformed conventional methods [15], [16], [17]. Motivated by such successes, we investigate the application of deep learning methods to novel class detection. In particular, we employ autoencoders for feature learning. Several papers have proposed outlier detection methods using autoencoders [17], [18]. Prior work has also shown that autoencoders can obtain stable and effective clustering [19], [16]. The aforementioned approaches have proved the effectiveness of autoencoders in both clustering and outlier detection. However, previous approaches have not used deep learning techniques in the context of novel class detection over stream data.

Denoising Autoencoders (DAEs) have been used in several studies for their ability to extract abstract features that can outperform the original input feature representation when used in classification or other tasks [20], [15]. A DAE is a type of neural network with hidden layers aiming to reconstruct the input vector from a corrupted version with minimum error. It is an unsupervised learning method. When designed with multiple hidden layers, DAEs can learn *deep* abstract features. In this paper, we propose to combine learning deep features using DAEs with novel class detection enhanced with nonparametric, multidimensional CPD.

Our contributions can be summarized as follows: First, to the best of our knowledge, this is the first study to combine deep learning with *novel class detection in stream data*. Second, we introduce a nonparametric multidimensional change point detection to detect concept-drift. Third, our proposed approach enriches traditional classification models with a novel class detection mechanism and unsupervised deep learning method, combining the strength of these techniques. Finally, we apply our technique on both synthetic and real-world data and obtain much better results than state-of-the-art stream classification algorithms.

The rest of the paper is organized as follows. Section II gives an overview on DAEs. Sections III and IV provide the details of our approach and Section V discusses baseline approaches. Section VI then describes the datasets and experimental evaluation of our technique. Discussions about our approach are provided in Section VII. Section VIII presents related work. Finally, Section IX concludes with directions to future works.

## II. Background: Denoising autoencoder (DAE)

An autoencoder consists of two functions: encoder $f$ and decoder $g$. The encoding function $f(x) = \sigma(Wx+b)$ encodes input $x \in \mathbb{R}^d$ to a hidden representation $z \in \mathbb{R}^p$. Function $\sigma(s) = (1 + exp(s))^{-1}$ is the sigmoid, $W$ is the weight matrix, and $b$ is the bias. Multiple hidden layers can be added to incorporate deeper features. Let $d$ be the number of dimensions of the input vector $x$, and let $p < d$ be the number of dimensions of the deepest hidden layer. The decoding function $g(z) = \sigma(W^T z + b)$ decodes $z$ to $x'$. Autoencoders aim to minimize the reconstruction error. We use cross entropy as the cost function of the reconstruction error:

$$\mathcal{L}(x, x') = -\sum_{k=1}^{d} x_k \ \log(x'_k) + (1 - x_k) \ \log(1 - x'_k)$$

which can be minimized by gradient descent.

DAEs are autoencoders trained to reconstruct a clean input from a corrupted version [15]. To create a corrupted version $\widetilde{x}$ of $x$, we use the **additive Gaussian noise** corrupting method [21]. This method adds a random value $v$ to each feature in $x$, in which $\widetilde{x}_k = x_k + v_k$, where $k = [1 \ldots d]$ and $v_k \sim \mathcal{N}(0, \sigma^2)$. Encoder $f$ encodes input $\widetilde{x}$ to $z$. The decoder function $g$ decodes $z$ to $x'$ while minimizing the error of reconstructing the original (clean) input $x$.

## III. Novel class detection

This section describes our approach to **detect** novel class data points and **classify** points that belong to existing classes.

A novel class is a new class that has not been modeled by the classifier. Before describing our novel class detector, we give an informal definition of the data stream classification problem. We assume that a data stream is a continuous flow of data $D = x_1, \ldots, x_n$, where $x_i$ is the $i$th instance (i.e., data point) in the stream. Assuming that the class labels of all the instances in $D$ are unknown, the problem is to predict their class labels. Let $y_i$ and $\hat{y}_i$ be the actual and predicted class labels of $x_i$, respectively. If $\hat{y}_i = y_i$, then the prediction is correct; otherwise it is incorrect. The goal is to minimize the prediction error. The predicted class labels are then used for various purposes depending on the application.

We maintain an ensemble of multiple classification models. An unlabeled instance is classified by taking a majority vote among the classifiers in the ensemble. Concept-drift is handled

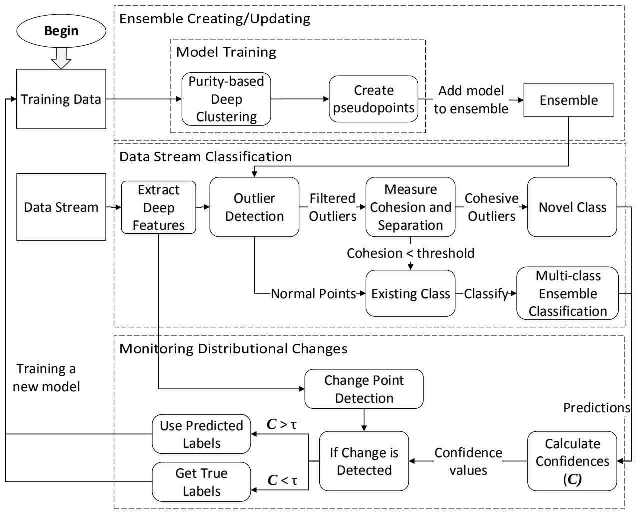

Figure 1. Our Approach for Novel Class Detection

by continuously updating the ensemble so that it represents the most recent concept in the stream. The update is performed as follows: As soon as a distributional change is detected, a new model is trained and it replaces the oldest existing model in the ensemble.

Our approach provides a solution to the concept-evolution problem by enriching each classifier in the ensemble with a novel class detector. If all of the classifiers discover a novel class, then arrival of a novel class is declared, and potential novel class instances are separated and classified as members of the novel classes. Thus, a novel class can be automatically identified without manual intervention.

The main concern with data stream classification is building the classification model and keeping it up-to-date by frequently updating the model with the most recent labeled data. Several ensemble-based methods address this concern [4], [7]. We present our ensemble-based novel class detection method below, in which each classifier in the ensemble integrates multi-class classification with outlier detection.

**Overview**. Our approach combines several techniques to detect novel class instances, as illustrated in Figure 1. The first $n_{init}$ instances of the stream are used to build an initial ensemble and to train a DAE to extract deep abstract features. These instances are labeled. We apply *purity-based (or semi-supervised) deep clustering* to cluster these instances (see Figure 1). This clustering method uses deep abstract features extracted using the DAE. It aims to minimize both intra-cluster dispersion and cluster impurity. We keep a summary of the boundaries of each resultant cluster, called a *pseudopoint*. It consists of centroid, radius, and class frequencies. We use pseudopoints as models for both classification and outlier detection. When an instance is outside the boundaries of all pseudopoints it is labeled as an outlier.

Pseudopoints can be used for classification of existing classes. When an instance $x$ is inside a pseudopoint, $x$ takes the label of the most frequent class in the pseudopoint. Multiple chunks of instances create multiple models. An *ensemble* combines the results of these models. These models are replaced with new ones once changes in data distribution are detected.

During the classification of new unlabeled instances (the data stream in Figure 1), we use the ensemble to *detect outliers*. If all models in the ensemble declare an instance $x$ an outlier, we call $x$ a *filtered outlier*. Next, a silhouette coefficient measures the *cohesion* and *separation* of the filtered outliers. Instances that are not detected as outliers or have a silhouette coefficient less than a specified threshold are considered members of an *existing class* and classified using the ensemble (i.e., using the most frequent class of the nearest pseudopoint). We then calculate the confidence $\mathcal{C}$ of each classification.

While we classify new data points, a nonparametric multidimensional change point detection procedure keeps monitoring the incoming data. Once a change in the distribution of the data is detected, a set of new training data is formed using the recently classified instances to build a new model. This set of new training instances is labeled using two ways. The instances with low confidence values are labeled using their true labels. High confidence instances, on the other hand, are labeled using predicted labels. This labeling process reduces the dependency on human annotation because the use of the predicted labels minimizes the amount of true labeling, which requires human effort. The newly built model replaces the oldest one in the ensemble.

The rest of this section details the approach.

**Ensemble Creation**. We build an ensemble of classifiers with the first $n_{init}$ labeled data points, and we keep updating the classifiers with the new data. Each classifier in the ensemble uses a $k$-NN type classification model. A naïve approach to build such a model is to store all the training data in memory. However, this is inefficient and does not scale to real operating environments. To optimize space utilization and time performance, our approach uses a semi-supervised (purity-based) clustering technique based on Expectation Maximization (E-M) [4], which minimizes both intra-cluster dispersion and cluster impurity, and caches a summary of each cluster (centroid and frequencies of data points belonging to each class), discarding the raw data points. We call a cluster's summary a *pseudopoint*.

We cluster data using their deep abstract features learned from the original data features using the DAE.

**Deep Feature Extraction using the DAE**. To extract deep features, we compute DAE weights ($W$ and $b$) from the original input features by training the DAE with the instances of the first $n_{init}$ data. We keep the learned $W$ and $b$ to transform the feature values of the rest of stream instances, and denote the transformation of the features of instance $x \in \mathbb{R}^d$ to $z \in \mathbb{R}^p$ as $z = \sigma(Wx + b)$.

Note that we learn the weights ($W$ and $b$) using the first few chunks and we do not update them during the stream, because updating DAE weights requires performing more time-consuming back propagation. This makes our approach faster, especially since the number of transformed features is significantly less than the original. The ensemble models and all our techniques use data points in deep feature dimensional space.

**Outlier Detection using Pseudopoints**. Figure 2 illustrates pseudopoints of one model in the ensemble. The axes represent features. There are 4 groups of pseudopoints (A, B, C, and D). Each pseudopoint is labeled based on the most frequent class label. The pseudopoints of groups A and B are labeled *negative* because most of the labeled instances located inside its boundaries (i.e., during clustering) are negative. Similarly, the pseudopoints of groups C and D are labeled *positive*.

When a new unlabeled test instance emerges, the ensemble is used to classify the instance. A classifier labels the instance based on whether it is inside or outside the decision boundaries of the pseudopoints. For example, in Figure 2, $X^i$ is located inside the boundaries of a pseudopoint, unlike $X^j$. We therefore call $X^j$ a raw outlier (or *Routlier*). If all classifiers of the ensemble label an instance a Routlier, we identify the instance as a filtered outlier (or *Foutlier*) (see Figure 3). If the test instance is identified as a *Foutlier*, it is temporarily stored in a buffer *buf* for further inspection. Otherwise, if it is not *Foutlier*, it is classified as one of the existing classes using $k$-NN (i.e., the most frequent class of the nearest pseudopoint).

**Detecting a Novel Class**. The buffer *buf* is periodically checked to see whether a novel class has appeared. The central concept of our novel class detection technique is that the data points belonging to a common class should be closer to each other (cohesion) and should be far apart from the data points belonging to other classes (separation). When *buf* is examined for novel classes, we look for strong cohesion among the outliers in *buf*, and large separation between the outliers and ensemble's pseudopoints. If such strong cohesion and separation is found, we declare a novel class. We estimate the $q$-Neighborhood Silhouette Coefficient, or $q$-NSC [3]. This is defined based on the $q, c$-neighborhood of a *Foutlier* $x$ ($q, c(x)$ in short), which is the set of $q$ instances from class $c$ that are nearest to $x$. Parameter $q$ is user defined.

Let $\bar{D}_{c_{out},q}(x)$ be the mean distance of a *Foutlier* $x$ to its $q$-nearest *Foutlier* neighbors. Also, let $\bar{D}_{c,q}(x)$ be the mean distance from $x$ to $q, c(x)$, and let $\bar{D}_{c_{min},q}(x)$ be the minimum among all $\bar{D}_{c,q}(x)$ for existing classes $c$. In other words, $q, c_{min}$ is the nearest existing class neighborhood of $x$. Then $q$-NSC [8] of $x$ is given by:

$$q\text{-NSC}(x) = \frac{\bar{D}_{c_{min},q}(x) - \bar{D}_{c_{out},q}(x)}{\max(\bar{D}_{c_{min},q}(x), \bar{D}_{c_{out},q}(x))} \quad (1)$$

$q$-NSC considers both cohesion and separation, and yields a value between $-1$ and $+1$. A positive value of $q$-NSC indicates that the *Foutliers* are closer to other *Foutlier* instances stored in the buffer (more cohesion), and farther away from the instances from existing classes (more separation). The $q$-NSC($x$) value of an *Foutlier* $x$ must be computed separately for each classifier $M_i$ in ensemble $\mathcal{M}$. we declare emergence of a novel class if we find at least $q' > q$ *Foutlier*s having a positive $q$-NSC score for all the classifiers $M_i \in \mathcal{M}$.

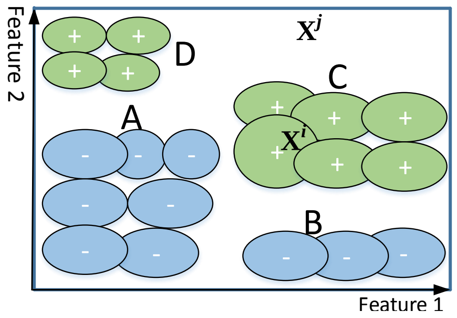

Figure 2. An example of a classifier in the ensemble. The circles represent the pseudopoints.

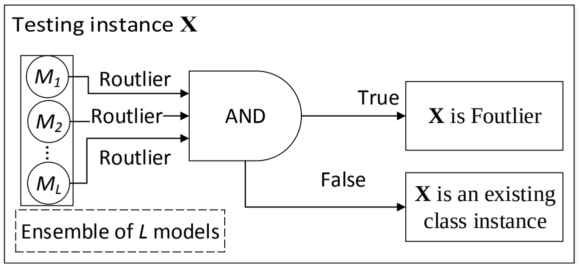

Figure 3. Using the ensemble to detect and filter outliers

**Ensemble Updating**. We update the ensemble models when a change in the distribution of the data points is detected. During the classification of any instance $x$, we estimate the confidence of each individual model $M_i \in \mathcal{M}$ by calculating two measurements: *association* $\mathcal{A}_i$ and *purity* $\mathcal{P}_i$.

- $\mathcal{A}_i = \mathcal{R}(h_{ip}) - \mathcal{D}_{ip}(x)$, where $\mathcal{R}(h_{ip})$ is the radius of the $p$th pseudopoint ($h_{ip}$) in model $M_i$, assuming $h_{ip}$ is the nearest pseudopoint to instance $x$, and $\mathcal{D}_{ip}(x)$ is the distance between $x$ and the center of $h_{ip}$.
- $\mathcal{P}_i = \frac{|\mathcal{L}_{ip}(c_m)|}{|\mathcal{L}_{ip}|}$, where $|\mathcal{L}_{ip}|$ is the sum of all frequencies in $h_{ip}$, and $|\mathcal{L}_{ip}(c_m)|$ is the frequency of the most frequent class ($c_m$) in $h_{ip}$.

Once the true labels of instances are available, we create a vector $v_i$ for each model $M_i$. If $M_i$ classifies instance $x$ correctly, then $v_i^x = 1$; otherwise $v_i^x = 0$. After $v_i$, $\mathcal{A}_i$, and $\mathcal{P}_i$ are calculated for model $M_i$ over multiple instances, we calculate Pearson's correlation coefficients between $v_i$ and both $\mathcal{A}_i$ and $\mathcal{P}_i$, resulting in values $r_A$ and $r_P$, respectively. During classification of any instance $x$, we compute the confidence $\mathcal{C}_i^x$ for model $M_i$, $\mathcal{C}_i^x = \mathcal{A}_i^x * r_A + \mathcal{B}_i^x * r_P$. Similarly, we calculate $\mathcal{C}_i^x$ for other models $M_i$ in the ensemble. Then the scores are normalized and the average confidence of models is taken ($\mathcal{C}^x$).

As the data stream instances are being classified, we keep monitoring the distributional changes of the stream data. Nonparametric Change point detection is used (Section IV). Once a significant change is detected, we update the ensemble by replacing the oldest model with a new one. The new model is trained using the recently classified instances. We require the true labels of the instances that have confidence values less than a threshold $\tau$. On the other hand, for instances with confidence greater than $\tau$ the predicted labels are used. Both sets are then used to train a new classifier.

## IV. Change Point Detection Algorithm

Change point detection (CPD) is used to detect abrupt changes in characteristics of data at unknown time instants. Abrupt changes arise when changes in characteristics occur very fast with respect to the sampling period of the measurements. Change point detection is particularly useful for quality control, system monitoring, and fault detection. In this paper, we use this technique to detect changes in distribution parameters of the data points. Identifying a change point triggers training a new model to replace the oldest one in the ensemble.

Let $\{x_j\}$ be a sequence of data points, where $x_j$ has a distribution $F$ for $j \leq \nu$ and a distribution $G$ otherwise. The problem of change point detection is to discover a change and to estimate the change point parameter $\nu$ from these data. CPD algorithms in general can be divided into two categories: parametric and nonparametric algorithms. Parametric CPD algorithms assume that the distribution families before and after the change point are known beforehand. However, in many applications, although one may know the distribution before the change, when the process is "in control," it is usually impossible to know the distribution after the change, when the process goes "out of control." It may also be the case that distribution families both before and after the change point are unknown. In those cases, nonparametric change point detection algorithms are used to detect the change point in data.

Several nonparametric change point estimation approaches have been proposed in the literature [11]. The pre- and post-change empirical distributions are usually compared for each $k = 1, \ldots, n$, where $n$ is the number of data points. Then the point $\nu$ that maximizes some predefined metric or *measure of diversity* between these distributions is used as an estimator of the actual change point $\nu$.

Prior work has established an efficient nonparametric change point detection method based on log-likelihood ratio random walk, which is mathematically proven to achieve high accuracy [12]. It can be summarized as follows.

Let $X_1 \ldots X_k$ and $X_{k+1} \ldots X_n$ be the pre and post-change sequence of data points, where $X_i$ is the $i$th element in the data sequence. Let $\chi_{1:k}$ and $\chi_{(k+1):n}$ be the histograms (of $r$ number of bins) modeling $X_1 \ldots X_k$ and $X_{k+1} \ldots X_n$, respectively. For any potential change point $k$, let $p^{(m)}_{1:k}$ be the ratio of the number of points in bin $m$ of $\chi_{1:k}$ to the total number of points $k$. Similarly, $p^{(m)}_{(k+1):n}$ is the ratio in bin $m$ of $\chi_{(k+1):n}$. Note that $\sum_m p^{(m)}_{1:k} = \sum_m p^{(m)}_{(k+1):n} = 1$. The log-likelihood ratio is calculated by:

$$S_{k,n} = (n-k) \sum_{m=1}^{r} p^{(m)}_{(k+1):n} \log \frac{p^{(m)}_{(k+1):n}}{p^{(m)}_{1:k}} \tag{2}$$

We monitor changes in multidimensional data points ($X_i \in \mathbb{R}^D$) and estimate $S_{k,n}$ for each dimension separately. In other words, for each dimension $d$, the data values $X^{(d)}_{1:n}$ of dimension $d$ are used to estimate $S^{(d)}_{k,n}$. We refer to the dimensional $S_{k,n}$ as $S^{(d)}_{k,n}$. The maximum $S^{(d)}_{k,n}$ in each dimension $d$ is estimated using the following formula:

$$W^d_n = \max_{\gamma \leq k < n-\gamma} S^{(d)}_{k,n}, \forall d \in D \tag{3}$$

where $\gamma = n^\epsilon$ is a cushion distance and $0 < \epsilon < 1$. The maximal $W^d_n$ across all dimensions is calculated ($W_n$):

$$W_n = \max_{1 \leq d \leq D} W^d_n \tag{4}$$

We record the dimension of the maximum change:

$$maxd = \arg\max_{1 \leq d \leq D} W^d_n \tag{5}$$

If $W_n$ exceeds threshold $h$, defined by

$$h = -\log(\alpha) \tag{6}$$

then change point $\nu$ is declared, where $\alpha$ is the probability of false alarm. A probability of $\alpha = 0.05$ is commonly used in the literature.

$$\nu = \arg\max_{\gamma \leq k < n-\gamma} S^{(maxd)}_{k,n} \tag{7}$$

The time complexity of detecting change is $O(\mathrm{D}\times\mathrm{n}\times\mathrm{r})$, where $D$ is the number dimensions, $n$ is the number of examined data points, and $r$ is the number of bins of estimated histogram.

This change point detection technique is suitable for detecting distributional changes in the incoming data points. As the incoming data points keep emerging, we transform the data points into deep abstract feature space and store them in a queue-type sliding window $S$. We invoke our CPD algorithm to monitor instances in $S$. Once a change point is detected, we train a new model with the data points in $S$. However, instead of requesting true labels of all the instances, we get true labels only for the instances with low confidence value. Instances with high confidence are labeled using the predicted class.

## V. Baseline Approaches

To evaluate the performance of our approach we compare it to other approaches. In this section we present these baseline approaches. The first two are considered the current state-of-the-art.

## A. ECSMiner

ECSMiner [4] is an ensemble-based novel class detection approach that divides the data stream into equal-size chunks. The ensemble classifies each chunk and detects novel classes. True labels are revealed periodically. Once they are revealed, a new model is trained using true labels. Then the new model replaces the model with the lowest accuracy. Unlike our approach, ECS-Miner does not exploit deep features, and does not detect changes.

## B. ECHO

ECHO [7] utilizes change point detection to detect concept-drift. However, this change detection technique monitors the classifier's confidence estimates instead of the original data stream. The assumption is that a change in the classifier's confidence indicates occurrence of a concept-drift. Therefore, if there is a significant change in confidence estimates, a concept-drift is detected and the ensemble is updated. Using this approach requires prior knowledge of the distribution family of monitored data (i.e., confidence values). In other words, the user must identify the data distribution. This work proposes a nonparametric change detection technique, which operates on any data distribution family without the need to specify a particular distribution.

Furthermore, ECHO detects novel classes and classifies the data points in the original input feature space. It does not extract deep features.

## C. One-class SVM Ensemble

In this approach, we build an ensemble of one-class SVM classifiers. One-class SVM classifier is an unsupervised learning method. It learns the decision boundary of training instances and is used to predict whether an instance is inside or outside the learned boundary [3]. We train one classifier for each class. For instance, if our training data consist of instances of $C$ classes, our ensemble must contain $C$ one-class SVM classifiers.

During classification, once a new unlabeled instance $x$ emerges, we classify it using all the one-class SVM classifiers in the ensemble, as follows:

1) If all classifiers predict $x$ as outside the boundary, we label it as a novel class instance.
2) If one classifier predicts $x$ as inside the boundary, we label $x$ as the class of that classifier.
3) If more than one one classifier predicts $x$ as inside the boundary, we label $x$ as the class of the classifier with highest confidence.

We build our ensemble using the first $n_{init}$ chunks. During the classification of the stream, once new labeled (novel class or existing) instances emerge, we train new classifiers (i.e., one classifier for each class of the emerged instances). Then we add them to the ensemble without removing the old classifiers.

## D. Denoising Autoencoder Ensemble

Several previous works have suggested using autoencoders for outlier detection [17], [18]. The main idea is as follows. By training an autoencoder with instances of a certain class (e.g., $b$), we expect that the reconstruction error ($\mathcal{L}$) for any instance of the class $b$ will be less than $\mathcal{L}$ for instances from any other classes (not $b$). We extend this idea to detect novel class instances in data streams by using an ensemble of DAEs. Each DAE detects outliers from its perspective. The results of the DAEs are combined to determine the class of the tested instance and whether it is a novel class. We present this technique and compare it with our approach.

We build an ensemble of DAEs using the instances of the first $n_{init}$ data points by training one DAE for each class of the $C$ classes in $n_{init}$ data. As a result, our initial ensemble contains $C$ DAEs, where each DAE$^b$ is trained with class $b$ instances. Training is done using gradient descent after applying data corruption as described in Section II. To predict whether an instance $x$ is an instance of class $b$, we feed forward $x$ in DAE$^b$ and calculate the resulting reconstruction error, which can be computed using cross-entropy ($\mathcal{L}$ in Section II). If $\mathcal{L}$ is less than a threshold $h$, we classify it as inside the boundary of DAE$^b$. Otherwise, it is outside (or *Routlier*). In our experiments, we assume that we know the best threshold value.

If all DAEs in the ensemble classify $x$ as Routlier, we classify $x$ as novel class instance. Otherwise, if at least one DAE classifies $x$ as inside, we classify $x$ as the class of the DAE with the minimum $\mathcal{L}$.

The ensemble is updated by adding new DAEs to the ensemble whenever the labels of new instances are revealed. In our experiments we assume the labels are revealed periodically.

Notice that in this approach we use DAEs as outlier detectors, not as feature transformers.

## VI. Results

We have conducted several experiments to evaluate our approach. Our novel class detection approach (here referred to as **NovelDetector**$^{\text{DAE}}$) is described in Section III. We claim that using deep abstract features extracted using DAEs boosts the performance of detecting novel classes. We tested NovelDetector$^{\text{DAE}}$ against the four other approaches described in Section V: **ECS-Miner**, **ECHO**, **OneSVM** Ensemble, and **DAE Ensemble**.

**Datasets**. Our experiments are applied to two synthetic datasets and five real datasets of several types. Table I summarizes the datasets. We have used MOA [14] to generate synthetic datasets with RandomRBFGeneratorDrift, 7 classes, 70 attributes, 100,000 instances, 12 centroids, and 0.002 drift value for dataset **Syn1** and 0.003 for **Syn2**. We show results on two security datasets: **Packets** and **SystemCalls** [22], [23]. These two datasets are from network attacks targeting a web server over a period of time. The goal is to detect the novel and existing attacks. Each one of the two datasets has about 6,000 numeric features and 28 classes (12 benign

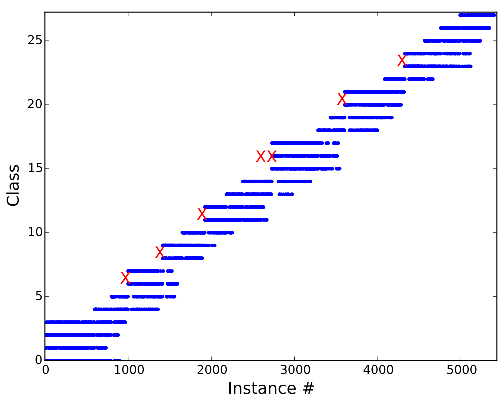

Figure 4. Distribution of class instances over time

and 16 attacks). The Packets dataset consists of packets sizes, time, and counts [24]; and the SystemCalls dataset is system call $N$-grams. We have additionally tested our approach on Forest CoverType (**Fcover**) and Physical Activity Monitoring (**PAMAP2**) datasets. The latter contains sensory data from 18 different physical activities. Finally, the Internet Movie database (**IMDB**), which consists of textual reviews, is used to predict the authors of the reviews (i.e., author attribution). For all datasets, we normalized features between 0 and 1 and treat 0s and 1s in binary features as numeric. We transformed network packets into feature vectors [25].

Table I
SUMMARY OF THE DATASETS

| Dataset | # Classes | # Dims | # Samples |
|---|---|---|---|
| Syn1 | 7 | 70 | 100,000 |
| Syn2 | 7 | 70 | 100,000 |
| Packets | 28 | 6,000 | 5,600 |
| SystemCalls | 28 | 6,000 | 5,600 |
| Fcover | 7 | 54 | 150,000 |
| IMDB | 15 | 1000 | 15,000 |
| PAMAP2 | 6 | 53 | 150,000 |

**Data Shuffling**. In our experiments, novel classes appear gradually. One or more concurrent novel classes may appear along with existing classes. Figure 4 shows the distribution of Packets dataset instances and their classes over time. The x-axis represents time (or instance number) and the y-axis records the classes of the instances. As time increases, new instances emerge. Each instance is represented as a point (blue). Each X indicates concurrent novel classes. For example, the first X shows the emergence of novel classes 6 and 7. At the same time, classes 4 and 5 are considered existing classes. At instance number 2,700 (indicated by XX), three concurrent novel classes emerge. Note that old classes in the Packets dataset disappear (Figure 4), but in other datasets like Fcover old classes do not disappear.

**The Metrics**. We measured the performance of the approaches by calculating False Positive Rate (FPR%), False Negative Rate (FNR%), and Error (ERR%). False positives are instances incorrectly classified as novel class instances. False Negatives are novel class instances misclassified as belonging to an existing class. ERR is the total classification error, including multi-class misclassifications (e.g., misclassifying instance $A$ as $B$, regardless of $A$ and $B$ being novel or existing).

The metrics are measured as the average over all points. More specifically, after we classify a test instance, we estimate the accuracy. Later we use the instance in building new classification model during training and update the existing ensemble.

Note that the first $n_{init}$ data points are used to initialize the ensemble. These *labeled* points are not counted in the calculation of our metrics.

**NovelDetector$^{\text{DAE}}$ Parameters**. NovelDetector$^{\text{DAE}}$ was applied with deep abstract features extracted using a DAE with two hidden layers (where the number of features in the first hidden layer is 2/3 the original features, and in the second is 1/3 the first hidden layer) and using additive Gaussian noise where $\sigma = 1.1$. The results are shown in Table II.

**Observations**. NovelDetector$^{\text{DAE}}$ outperforms other approaches in all metrics. FPR, FNR, and the total error are less than 3% on all datasets. On the other hand, the performance degrades using ECS-Miner (e.g., 34.89% FPR, 23.64% FNR, and 49.98% ERR for SystemCalls), whereas ECHO scored 6.11% FPR, 27.5% FNR, and 30.38% ERR. Also, our approach shows significant improvement on results of the synthetic datasets generated with concept-drift. The performance of One-class SVM Ensemble fluctuates depending on the dataset, but is always less than NovelDetector$^{\text{DAE}}$. We tested One-class SVM Ensemble with and without DAE features.

**Impact of varying NovelDetector$^{\text{DAE}}$ parameters**. We have varied the number of DAE hidden layers, number of abstract features in the hidden layers, the cost function, and corruption amount $\sigma^2$. We have observed that the cost function significantly affects performance. For example, when using L2 norm instead of cross entropy as a cost function, FPR increases from 2% to 70.31%, FNR to 7.61%, and ERR to 70.59% for the Syn1 dataset. The number of layers also affects performance if we use less than two layers; using two or more layers yields significant improvement. On the other hand, the number of features has less performance impact for NovelDetector$^{\text{DAE}}$. We report the results of applying NovelDetector$^{\text{DAE}}$ with these variations in Table III. Note that $d$ in the table equals the number of original features.

The accuracy of NovelDetector$^{\text{DAE}}$ increases when we increase the amount of corruption in the DAE. We have tested the impact of varying the noise amount by changing the variance $\sigma^2$ of the additive Gaussian noise corrupting method. (See Section II for more details about the corrupting method.) We have evaluated the behavior by testing the performance of NovelDetector$^{\text{DAE}}$ with abstract features extracted using a DAE with one hidden layer on the Syn1 data. We set the number of abstract features to equal the original features. The results are reported in Table IV. They show that increasing the amount of noise leads to better performance of NovelDetector$^{\mathrm{DAE}}$. That is, the total error, FPR, and FNR significantly improve when using $\sigma > 1.0$. For example, using $\sigma = 1.9$ yields 0.01% FPR, 0.07% FNR, and 0.02% ERR.

## VII. Discussion

The new abstract deep features generated using the DAE approach significantly boost the performance of novel class detection in our experiments. Moreover, the experiments show that increasing the amount of noise significantly improves DAE accuracy. This noise is randomly sampled from $\mathcal{N}(0, \sigma^2)$. Such noise corrupts the original features and the decision boundaries of the pseoudopoints (clusters). The DAE produces the weights $W$ and biases $b$, which correct this corruption. These weights produce pseudopoints robust to noise and drift that may occur in the original input data (yielding better generalization). This explains the increase in the accuracy of NovelDetector$^{\mathrm{DAE}}$ on all the tested datasets, and mainly the synthetic data which are generated using a random radial basis function with drift.

Since the DAE reduces the number of dimensions significantly, novel class detection becomes much faster. Note that the DAE learns weights only one time (during the first $n_{init}$ chunks) without updating or retraining. In other words, DAE training can be done offline as a warm up phase while building the initial ensemble. So, DAE training time does not negatively affect the processing time after the first $n_{init}$ chunks.

In our experiments, the nonparametric change point detection technique monitors the data in deep abstract feature space. However, the user can choose to detect changes in the original input space or can monitor other variables, such as the distribution of the classifier's confidence values or class labels (if available).

The results of NovelDetector$^{\mathrm{DAE}}$ on the security datasets demonstrate that we can successfully detect novel types of attacks. Our work can be extended to address zero-day attacks. Such attacks have not been modeled by the classifier. However, our approach also identifies new benign classes. In future work, we want to enable NovelDetector$^{\mathrm{DAE}}$ to distinguish between novel attacks and novel benign classes.

## VIII. Related Work

Our novel class detection technique differs from traditional one-class novelty detection techniques [26] that can only distinguish between normal and anomalous data. Traditional novelty detection techniques assume that there is only one normal class, and any instance that does not belong to the normal class is an anomaly/novel class instance. Therefore, they are unable to distinguish among different types of anomaly. Our approach overcomes this limitation by employing a multi-class framework for the novelty detection problem, which can distinguish between different classes of normal and anomalous behavior, and discover new emerging classes. In addition, traditional novelty detection techniques identify data points as outliers that deviate from the normal class. Conversely, our approach discovers whether a group of such outliers constitutes a new class by displaying strong cohesion. Therefore, our approach synergizes with a multi-class classification model and a novel class detection model.

Unlike other novel class detection methods (cf., [4], [3]), our approach does not divide data steams into fixed-size chunks, but instead uses a sliding window. This allows our approach to capture concept-drift immediately. Many approaches that use a sliding window (e.g., [27]) detect changes in classification error, and hence require true labels of classified instances (i.e., supervised). To minimize dependency on true labeling, ECHO [7] estimates the classifier's confidence values in an unsupervised way without requiring true labels. Assuming that confidence values can capture concept-drift, ECHO monitors the change in these values. However, the user must have prior knowledge about the distribution family of the confidence values. Our change detection method monitors the distributional changes in the original data stream, so it requires no prior knowledge about the distribution, and does not require true labels.

Prior work [2] has proposed a method to classify multi-stream data points. The method applies supervised and unsupervised change point detection techniques. The supervised technique is used for labeled streams by monitoring changes in the labels. On the other hand, the unsupervised one is used for unlabeled streams by monitoring changes in the distribution of classifier confidence values (as in ECHO). Novel class detection is not addressed in this prior work.

None of previous approaches have considered deep learning in novel class detection over data streams. However, several papers have proposed feature learning with autoencoders to perform different tasks in various domains. In image processing, higher level representations learned by stacked DAEs help boost the performance of SVMs [15]. Adaptive DAEs for unsupervised domains have also been proposed [28]. Recursive autoencoders have been leveraged to generate vector space representations for variable-sized phrases [29]. Prior work [17], [18] has proposed outlier detection methods using autoencoders (but not for novel class detection). Combining autoencoders with clustering has been shown to outperform typical clustering methods [19], [16]. Marginalized DAEs [21] marginalize out corruption during training with fewer training epochs.

In our approach, we propose to use DAEs rather than other types of autoencoders because of their ability to generate robust features for clustering and novel class detection in data streams, where outliers, noise, and drifts are challenges. DAEs work well in such environments [30].

## IX. Conclusion

In this paper we presented a novel class detection approach that combines deep learning, outlier detection, and ensemble-based classification techniques. We also presented a multidimensional nonparametric change point detection. Our approach outperforms current state-of-the-art methods.

Table II
SUMMARY RESULT ON ALL DATASETS

| Metric | Novel Class Detector | Syn1 | Syn2 | Packets | SystemCalls | Fcover | PAMAP2 | IMDB |
|---|---|---|---|---|---|---|---|---|
| FPR % | NovelDetector$^{DAE}$ | **2.01** | **2.01** | **1.01** | **1.01** | **1.82** | **1.01** | **0.01** |
| | ECS-Miner | 71.04 | 71.63 | 26.66 | 34.89 | 4.64 | 18.01 | **0.01** |
| | ECHO | 14.90 | 15.21 | **0.01** | 6.11 | 2.83 | 4.97 | **0.01** |
| | OneSVM (DAE) | 5.17 | 4.09 | 85.61 | 71.50 | 7.16 | 88.50 | 79.99 |
| | OneSVM (Original) | 70.16 | 46.05 | 31.88 | 45.14 | 30.74 | 91.62 | 46.97 |
| | DAE Ensemble | 65.74 | 38.24 | 26.31 | 48.33 | 28.16 | 99.01 | 34.91 |
| FNR % | NovelDetector$^{DAE}$ | **2.01** | **2.34** | **1.52** | **1.30** | **1.05** | **0.02** | **1.01** |
| | ECS-Miner | 2.03 | 17.43 | 25.09 | 23.64 | 4.38 | 0.05 | 29.14 |
| | ECHO | 44.27 | 21.19 | 91.25 | 27.50 | 7.43 | 0.11 | 33.00 |
| | OneSVM (DAE) | 80.62 | 98.01 | 23.47 | 19.91 | 93.84 | 8.25 | 21.06 |
| | OneSVM (Original) | 39.91 | 61.87 | 55.94 | 45.75 | 28.86 | 0.01 | 35.12 |
| | DAE Ensemble | 66.31 | 57.14 | 63.29 | 52.13 | 35.12 | 0.64 | 42.26 |
| ERR % | NovelDetector$^{DAE}$ | **2.02** | **2.06** | **1.46** | **1.26** | **1.97** | **1.58** | **2.46** |
| | ECS-Miner | 89.64 | 86.37 | 42.53 | 49.98 | 6.99 | 17.38 | 12.93 |
| | ECHO | 31.19 | 27.49 | 75.52 | 30.38 | 5.09 | 5.37 | 9.78 |
| | OneSVM (DAE) | 97.52 | 97.28 | 78.59 | 63.08 | 87.18 | 94.83 | 71.05 |
| | OneSVM (Original) | 95.85 | 94.31 | 64.98 | 61.90 | 72.56 | 94.63 | 65.17 |
| | DAE Ensemble | 95.00 | 92.21 | 69.10 | 62.92 | 68.91 | 98.35 | 61.90 |

Table III
THE IMPACT OF VARYING DAE PARAMETERS ON NOVELDETECTOR$^{DAE}$ RESULTS

| Metric | Layers | Dimensions | Cost | Syn1 | Syn2 | Packets | SystemCalls | Fcover | PAMAP2 | IMDB62 |
|---|---|---|---|---|---|---|---|---|---|---|
| FPR % | 3 | $(1/13)d$ | entropy | 2.03 | 2.01 | 1.01 | 1.01 | 1.01 | 1.01 | 0.01 |
| | 2 | $(1/5)d$ | entropy | 2.01 | 2.01 | 1.01 | 1.01 | 1.82 | 1.01 | 0.01 |
| | 1 | $d$ | entropy | 5.10 | 2.82 | 1.01 | 20.05 | 5.37 | 1.01 | 0.04 |
| | 1 | $(2/3)d$ | entropy | 7.16 | 2.57 | 1.01 | 21.88 | 5.38 | 1.01 | 0.04 |
| | 1 | $(1/3)d$ | entropy | 3.23 | 2.51 | 1.01 | 19.59 | 3.30 | 1.01 | 0.01 |
| | 1 | $d$ | L2 | 70.31 | 43.25 | 34.13 | 26.09 | 6.54 | 5.88 | 0.13 |
| FNR % | 3 | $(1/13)d$ | entropy | 2.01 | 2.01 | 1.52 | 1.33 | 1.04 | 0.04 | 1.01 |
| | 2 | $(1/5)d$ | entropy | 2.01 | 2.34 | 1.52 | 1.30 | 1.05 | 0.02 | 1.01 |
| | 1 | $d$ | entropy | 2.07 | 20.17 | 16.77 | 8.11 | 1.61 | 0.04 | 1.25 |
| | 1 | $(2/3)d$ | entropy | 6.92 | 2.31 | 16.27 | 8.90 | 1.01 | 0.02 | 1.27 |
| | 1 | $(1/3)d$ | entropy | 2.03 | 2.03 | 17.01 | 4.48 | 1.02 | 0.01 | 1.10 |
| | 1 | $d$ | L2 | 7.61 | 52.54 | 17.48 | 22.69 | 3.16 | 0.05 | 32.14 |
| ERR % | 3 | $(1/13)d$ | entropy | 2.06 | 2.03 | 1.46 | 1.28 | 1.06 | 1.61 | 2.5 |
| | 2 | $(1/5)d$ | entropy | 2.02 | 2.06 | 1.46 | 1.26 | 1.97 | 1.58 | 2.46 |
| | 1 | $d$ | entropy | 6.01 | 4.17 | 5.23 | 22.72 | 5.40 | 1.12 | 1.17 |
| | 1 | $(2/3)d$ | entropy | 8.56 | 3.00 | 5.21 | 20.81 | 5.33 | 1.10 | 1.18 |
| | 1 | $(1/3)d$ | entropy | 3.92 | 2.73 | 5.32 | 16.11 | 3.13 | 1.01 | 1.04 |
| | 1 | $d$ | L2 | 70.59 | 48.04 | 41.02 | 44.74 | 7.45 | 5.87 | 14.93 |

Table IV
THE IMPACT OF VARIANCE $\sigma^2$ IN ADDITIVE GAUSSIAN NOISE WHEN APPLYING NOVELDETECTOR$^{DAE}$ ON SYN1 DATA

| $\sigma^2$ | FPR% | FNR% | ERR% |
|---|---|---|---|
| 0.05 | 92.30 | 2.13 | 93.15 |
| 0.1 | 86.58 | 8.80 | 90.30 |
| 0.3 | 70.13 | 40.66 | 71.71 |
| 0.5 | 18.51 | 41.09 | 22.70 |
| 0.7 | 9.75 | 41.62 | 12.29 |
| 0.9 | 6.63 | 41.78 | 9.42 |
| 1.1 | 5.1 | 2.07 | 6.01 |
| 1.3 | 2.5 | 2.03 | 3.01 |
| 1.9 | 0.01 | 0.07 | 0.02 |
| 2.9 | 0.01 | 0.07 | 0.01 |
| 4.9 | 0.01 | 0.07 | 0.01 |
| 5.9 | 0.01 | 0.07 | 0.01 |

In the future we would like to investigate efficient ways to update DAE weights incrementally and to extend our work to detect zero-day attacks.

## ACKNOWLEDGMENTS

Funding for this work came from the National Science Foundation under Grant Numbers SBE-SMA-1539302, DMS-1737978 and the Air Force Office of Scientific Research under Award No. FA9550-14-1-0173.

## REFERENCES

[1] C. C. Aggarwal and P. S. Yu, "On classification of high-cardinality data streams," in *Proc. SIAM Int. Conf. Data Mining (SDM)*, 2010, pp. 802–813.

[2] S. Chandra, A. Haque, L. Khan, and C. Aggarwal, "An adaptive framework for multistream classification," in *Proc. 25th ACM Int. Conf. Information and Knowledge Management (CIKM)*, 2016, pp. 1181–1190.

[3] T. Al-Khateeb, M. M. Masud, K. Al-Naami, S. E. Seker, A. M. Mustafa, L. Khan, Z. Trabelsi, C. C. Aggarwal, and J. Han, "Recurring and novel class detection using class-based ensemble for evolving data stream," *IEEE Trans. Knowledge and Data Engineering (TKDE)*, vol. 28, no. 10, pp. 2752–2764, 2016.

[4] M. M. Masud, T. M. Al-Khateeb, L. Khan, C. Aggarwal, J. Gao, J. Han, and B. Thuraisingham, "Detecting recurring and novel classes in concept-drifting data streams," in *Proc. 11th IEEE Int. Conf. Data Mining (ICDM)*, 2011, pp. 1176–1181.

[5] Y. Yang, X. Wu, and X. Zhu, "Combining proactive and reactive predictions for data streams," in *Proc. 11th ACM SIGKDD Int. Conf. Knowledge Discovery in Data Mining (KDD)*, 2005, pp. 710–715.

[6] D. Brzezinski and J. Stefanowski, "Combining block-based and online methods in learning ensembles from concept drifting data streams," *Information Sciences*, vol. 265, pp. 50–67, 2014.

[7] A. Haque, L. Khan, and M. Baron, "SAND: Semi-supervised adaptive novel class detection and classification over data stream," in *Proc. 13th AAAI Conf. Artificial Intelligence*, 2016, pp. 1652–1658.

[8] M. M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham, "Classification and novel class detection in concept-drifting data streams under time constraints," *IEEE Trans. Knowledge and Data Engineering (TKDE)*, vol. 23, no. 6, pp. 859–874, 2011.

[9] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes: Theory and Application*. Englewood Cliffs, NJ: PTR Prentice-Hall, Inc., 1993.

[10] J. Chen and A. K. Gupta, *Parametric Statistical Change Point Analysis: With Applications to Genetics, Medicine, and Finance*. Boston, MA: Birkhäuser, 2012.

[11] D. Ferger, "Nonparametric change-point detection based on U-statistics," Ph.D. dissertation, University of Giessen, Hesse, Germany, 1991.

[12] M. I. Baron, "Nonparametric adaptive change point estimation and on line detection," *Sequential Analysis*, vol. 19, no. 1–2, pp. 1–23, 2000.

[13] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang, "A PCA-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams," in *Proc. 21th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2015, pp. 935–944.

[14] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," *J. Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[15] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[16] J. Xie, R. B. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," *CoRR*, vol. abs/1511.06335, 2015.

[17] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proc. 2nd Work. Machine Learning for Sensory Data Analysis (MLSDA)*, 2014.

[18] O. Mazhelis, "One-class classifiers: A review and analysis of suitability in the context of mobile-masquerader detection," *South African Computer J. (SACJ)*, vol. 36, pp. 29–48, 2006.

[19] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *Proc. 18th Iberoamerican Congress Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications (CIARP)*, 2013, pp. 117–124.

[20] F. Weninger, S. Watanabe, Y. Tachioka, and B. Schuller, "Deep recurrent de-noising auto-encoder and blind de-reverberation for reverberated speech recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 4623–4627.

[21] M. Chen, K. Q. Weinberger, F. Sha, and Y. Bengio, "Marginalized denoising auto-encoders for nonlinear representations," in *Proc. 31st Int. Conf. Machine Learning (ICML)*, 2014, pp. 1476–1484.

[22] F. Araujo, "Engineering cyber-deceptive software," Ph.D. dissertation, The University of Texas at Dallas, Richardson, Texas, 2016.

[23] F. Araujo, K. W. Hamlen, S. Biedermann, and S. Katzenbeisser, "From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation," in *Proc. 21st ACM Conf. Computer and Communications Security (CCS)*, 2014, pp. 942–953.

[24] K. Alnaami, G. Ayoade, A. Siddiqui, N. Ruozzi, L. Khan, and B. Thuraisingham, "P2v: Effective website fingerprinting using vector space representations," in *2015 IEEE Symposium Series on Computational Intelligence*, Dec 2015, pp. 59–66.

[25] K. Al-Naami, S. Chandra, A. M. Mustafa, L. Khan, Z. Lin, K. W. Hamlen, and B. M. Thuraisingham, "Adaptive encrypted traffic fingerprinting with bi-directional dependence," in *Proc. 32nd Annual Computer Security Applications Conf. (ACSAC)*, 2016, pp. 177–188.

[26] M. Markou and S. Singh, "Novelty detection: A review," *Signal Processing*, vol. 83, no. 12, pp. 2481–2521, 2003.

[27] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *Proc. 8th Int. Sym. Intelligent Data Analysis: Advances in Intelligent Data Analysis (IDA)*, 2009, pp. 249–260.

[28] J. Deng, Z. Zhang, F. Eyben, and B. Schuller, "Autoencoder-based unsupervised domain adaptation for speech emotion recognition," *IEEE Signal Processing Letters*, vol. 21, no. 9, pp. 1068–1072, 2014.

[29] P. Li, Y. Liu, and M. Sun, "Recursive autoencoders for ITG-based translation," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2013, pp. 567–577.

[30] C. Xing, L. Ma, and X. Yang, "Stacked denoise autoencoder based feature extraction and classification for hyperspectral images," *J. Sensors*, vol. 2016, 2016.