METER: A Dynamic Concept Adaptation Framework for Online Anomaly Detection

Jiaqi Zhu[†]
Beijing Institute of Technology
jiaqi_zhu@bit.edu.cn

Shaofeng Cai[*]
National University of Singapore
shaofeng@comp.nus.edu.sg

Fang Deng
Beijing Institute of Technology
dengfang@bit.edu.cn

Beng Chin Ooi
National University of Singapore
ooibc@comp.nus.edu.sg

Wenqiao Zhang[†]
Zhejiang University
wenqiaozhang@zju.edu.cn

ABSTRACT

Real-time analytics and decision-making require online anomaly detection (OAD) to handle drifts in data streams efficiently and effectively. Unfortunately, existing approaches are often constrained by their limited detection capacity and slow adaptation to evolving data streams, inhibiting their efficacy and efficiency in handling *concept drift*, which is a major challenge in evolving data streams. In this paper, we introduce METER, a novel dynamic concept adaptation framework that introduces a new paradigm for OAD. METER addresses concept drift by first training a base detection model on historical data to capture recurring *central concepts*, and then learning to dynamically adapt to *new concepts* in data streams upon detecting concept drift. Particularly, METER employs a novel *dynamic concept adaptation* technique that leverages a hypernetwork to dynamically generate the parameter shift of the base detection model, providing a more effective and efficient solution than conventional retraining or fine-tuning approaches. Further, METER incorporates a lightweight drift detection controller, underpinned by evidential deep learning, to support robust and interpretable concept drift detection. We conduct an extensive experimental evaluation, and the results show that METER significantly outperforms existing OAD approaches in various application scenarios.

PVLDB Reference Format:
Jiaqi Zhu, Shaofeng Cai, Fang Deng, Beng Chin Ooi, Wenqiao Zhang. METER: A Dynamic Concept Adaptation Framework for Online Anomaly Detection. PVLDB, 17(4): 794-807, 2023.
doi:10.14778/3636218.3636233

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/zjiaqi725/METER.

1 INTRODUCTION

Anomaly detection (AD), the process of identifying data samples that significantly deviate from the majority, plays a critical role in various systems by facilitating a deeper understanding of data, uncovering hidden anomalies, and enabling the implementation of appropriate measures to address associated concerns [10, 16, 46, 71, 79, 89]. While many methods have been developed for detecting anomalies in static data [38, 44, 53, 87, 94], the challenge of identifying anomalies in evolving data streaming applications has not been adequately addressed.

In real-world scenarios, data is often subject to constant updates and changes, primarily due to the dynamic nature of data sources (e.g., stock markets, traffic flow, social media), underlying infrastructures (e.g., IoT devices, cloud-based equipment, edge servers), or the influence of many other factors [17, 35, 50, 58, 59, 64]. Such dynamic nature of data requires real-time processing and analytics to effectively manage and respond to these changes. In particular, online processing and analysis of anomalies in evolving data streams have become increasingly essential for maintaining data quality and security across various domains. Notably, online anomaly detection (OAD) on evolving data streams enables real-time and better-informed decision-making, thereby ensuring data integrity and security while supporting the daily operations across business sectors [42, 43, 60, 70].

Evolving data streams are characterized by high-dimensional and heterogeneous data arriving continuously. Such ever-changing data streams make models trained on historical data outdated quickly. This phenomenon, commonly referred to as *concept drift*, as illustrated in Figure 1 (a), presents great challenges for the conventional AD approaches [20, 24, 38, 44], which train their detection models once on static data and assume the models keep functioning after deployment. This is not practical as they are built on the unrealistic assumption that all normal patterns are known prior to the model deployment. For instance, in the transaction monitoring, normal patterns evolve due to factors like seasonal variations, market trends, and new strategies. Traditional static AD approaches, trained on historical data stored in a database system, excel within specific time periods. However, once *concept drift* occurs, these static detection models could no longer effectively detect new abnormal behaviors. Therefore, AD approaches that can respond timely to drifts in data distributions become imperative.

Recently, several preliminary attempts have been made to support OAD in evolving data streams [7, 8, 11, 26, 37, 49, 53, 81, 86–88]. One popular approach for handling streaming data is based on incremental learning [7, 8, 11, 26, 86, 88]. These methods typically construct an initial model, which is then incrementally updated as new data arrives. However, as shown in Figure 1 (b), incremental

Proceedings of the VLDB Endowment, Vol. 17, No. 4 ISSN 2150-8097.
doi:10.14778/3636218.3636233
†Jiaqi Zhu and Wenqiao Zhang's work was done at NUS.
[*]Shaofeng Cai is the corresponding author.

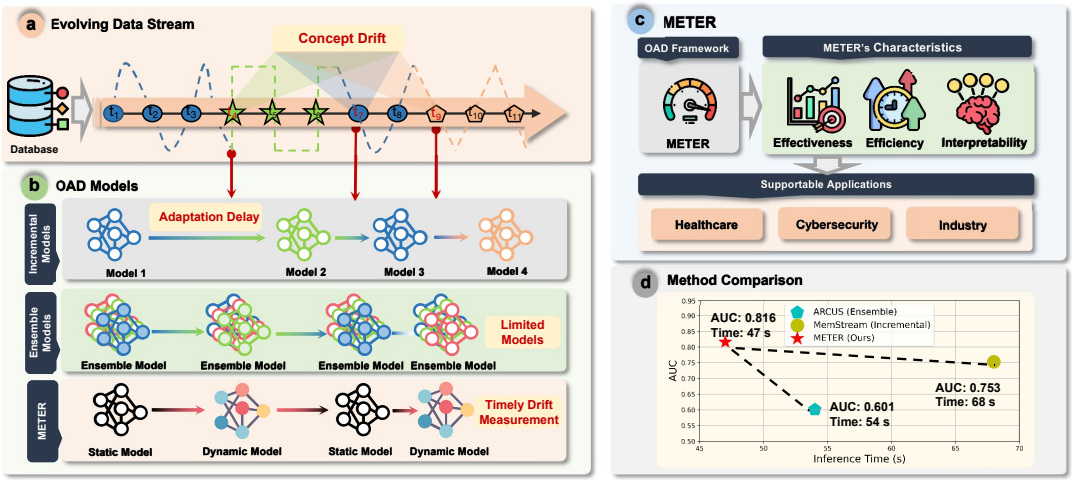

Figure 1: (a) The illustration of the evolving data stream with concept drifts in online anomaly detection (OAD). (b) The three main types of OAD approaches. (c) The functionalities of METER supporting various applications. (d) The comparison of our METER to state-of-the-art OAD approaches on the real-world dataset INSECTS [74], where Memstream [8] and ARCUS [87] are representative incremental and ensemble approaches: `AUC: 0.816 (METER) > 0.753 (Memstream) > 0.601 (ARCUS)`, Inference Time: 47 s (METER) < 54 s(Memstream) < 68 s (ARCUS).

approaches often require a considerable amount of time to adapt to new *concepts* [87] and do not provide performance guarantees for handling arbitrary concept drifts. Moreover, incremental learning methods typically suffer from inefficiency issues, as they need to update models to accommodate the changing concepts, either by retraining, fine-tuning models, or updating certain model statuses [8, 86, 88]. Another line of research in OAD focuses on the ensemble-based approaches [37, 53, 87], which involve training a combination of models to account for respective changing concepts. However, as shown in Figure 1 (b), the detection capacity of these model ensembles is constrained by the number of pre-trained models, and the model ensemble needs to adapt to drifted concepts frequently as new data arrives. Additionally, ensemble-based approaches require substantially more computation resources to maintain multiple models, further exacerbating the inefficiency issue. These limitations hinder existing approaches from meeting the requirements of supporting OAD in evolving data streams effectively and efficiently.

In particular, the deployment of an OAD framework in real-world applications necessitates not only detection effectiveness, but also efficiency and interpretability [7, 87, 88]. In terms of efficiency, the detection framework must provide accurate and timely decisions, and meanwhile, maintain a time complexity that does not increase linearly with the growing volume of data. As for interpretability, the ability of the detection framework to provide reliable uncertainty estimates for the detection results is necessary for users in critical applications such as healthcare and finance, which improves human understanding and engenders user trust in the framework [13, 18, 93]. Therefore, an ideal OAD framework should address all three crucial criteria, delivering effective, efficient and more interpretable detection in evolving data streams.

In this paper, we introduce a novel dyna**M**ic conc**E**pt adap**T**ation fram**E**wo**R**k (METER) for online anomaly detection in evolving data streams. Different from the conventional incremental learning and ensemble-based approaches, METER presents a new OAD paradigm that effectively, efficiently, and more interpretably addresses the concept drift challenge inherent in evolving data streams. METER is built upon the key observation that static historical data typically encompass the majority of anomaly patterns and thus comprises the recurring *central concepts*, whereas new arriving data streams, although may deviate from the main patterns occasionally, usually only drift slightly away from central concepts. Taking into account this observation in the framework design, METER first trains a base detection model to capture central concepts, and then, learns to adapt to *new concepts* once detecting concept drift without further training, fine-tuning, or updating model statuses as required in conventional approaches [8, 37, 53, 86, 88].

To improve efficiency and effectiveness, we further introduce a novel lightweight controller to detect whether concept drift occurs in the current data stream on a per-input basis, and design a novel *dynamic concept adaptation* technique that dynamically generates the corresponding *parameter shift* of the base detection model for the current input via a hypernetwork [28]. After training, METER can dynamically generate the *parameter shift* for the current data stream to handle the concept drifts on the fly and enhance the predictive performance of the base detection model. To support interpretability, this controller is derived from the *evidential deep learning* (EDL) theory [56, 72], which enables efficient and high-quality uncertainty modeling for the detection of concept drift, thereby supporting interpretable anomaly detection. Meanwhile, dynamic concept adaptation via the hypernetwork is more effective in handling concept drift, as the detection model can adapt to new concepts in an input-aware and timely manner, and is more efficient, since this novel approach requires no frequent model retraining or fine-tuning for the concept adaptation as in approaches based on incremental or ensemble techniques [8, 37, 53, 86, 88].

Our METER comprises the following main components: **(i)** Static Concept-aware Detector (SCD), an unsupervised deep autoencoder (AE) [40] to minimize the reconstruction error via compressed representations. SCD is pretrained on historical data to model the central concepts. **(ii)** Intelligent Evolution Controller (IEC), a concept detection controller that dynamically models *concept uncertainty* via *evidential deep learning* [72]. IEC detects concept drift for the current data stream. **(iii)** Dynamic Shift-aware Detector (DSD), a hypernetwork that dynamically updates SCD in an input-aware manner with the parameter shift upon the detection by concept drift by the IEC. DSD streamlines dynamic concept adaptation, as the concept drift can now be handled efficiently and more effectively by learning only the parameter shift, in a way reminiscent of residual learning [31, 84]. **(iv)** Offline Updating Strategy (OUS), a strategy introduced to enhance SCD with new central concepts. To improve efficiency, OUS only updates METER with new concepts when the recent data streams contain markedly different concepts from the existing central concepts. To achieve this, OUS employs a sliding window to aggregate the statistics of concept uncertainty to determine whether an update is needed. With these modules, METER delivers a more effective, efficient and interpretable OAD framework. We summarize our main contributions as follows:

- We propose a novel unsupervised OAD framework METER for evolving data streams, which offers a new OAD paradigm that effectively, efficiently, and interpretably addresses the concept drift challenge inherent in evolving data streams.
- We incorporate into our METER a lightweight Intelligent Evolution Controller (IEC) for detecting concept drift in evolving data streams on a per-input basis, which enables efficient and high-quality uncertainty modeling for the detection of concept drift, thereby supporting interpretable anomaly detection.
- We develop a Dynamic Shift-aware Detector (DSD) that dynamically updates the base detection model with the parameter shift via a hypernetwork in an input-aware manner. DSD streamlines dynamic concept adaptation and handles concept drift efficiently and more effectively.
- Through extensive experiments, we demonstrate that our METER framework efficiently detects various types of anomalies with high accuracy and interpretability, and meanwhile, outperforms existing incremental and ensemble learning methods for online anomaly detection.

2 PRELIMINARIES

In this section, we outline the fundamental concepts and techniques related to anomaly detection, concept drift, hypernetwork, and evidential deep learning.

Anomaly Detection and Concept Drift. The objective of *Anomaly detection* is to identify data samples that deviate from the majority or exhibit unusual patterns. In particular, the focus of this paper is on *online anomaly detection* (OAD), which is to detect anomalies in data streams in real-time, as formulated in Definition 2.1 below. Further, our framework focuses on unsupervised anomaly detection in data streams, where no labeling information is available.

In real-world scenarios, data is often dynamic and subject to constant changes [48], a phenomenon known as *concept drift*, where the statistical or distributional properties of the data within a certain domain change over time, as formally defined in Definition 2.2. This change in distribution can lead to a significant drift in the patterns and relationships of the data, presenting challenges in detecting anomalous events.

Definition 2.1 (Online anomaly detection). Considering an incoming data stream $\mathcal{X} = \{\vec{x}_1, \ldots, \vec{x}_t, \ldots\}$, where each entry $\vec{x}_t = (x_{t1}, \ldots, x_{td})$ comprises d attribute fields that can be either categorical or numerical features, *online anomaly detection* aims to predict whether a data sample $\vec{x}_t$ in the incoming data stream is anomalous or not at each time step t.

Definition 2.2 (Concept drift). A concept drift occurs at time step t if the underlying joint probability $P(\vec{x}, y)$ of input data $\vec{x}$ and the corresponding label y changes at time t, that is, $P(\vec{x}_t, y_t) \neq P(\vec{x}, y)$.

To address this challenge, the model needs to capture the dynamic behavior of the data stream and effectively adapt to drifted concepts over time. Formally, OAD can be achieved by dynamically computing an anomaly score $\mathcal{M}(\vec{x}_t; \Theta_m)$ for each time step t, using a detection model $\mathcal{M}(\cdot)$ parameterized by Θ_m.

Hypernetwork. A *hypernetwork* is a neural network that generates the weights for another neural network, known as the *primary network*. In our METER framework, we leverage a hypernetwork to learn the parameter shift for the Static Concept-aware Detector (SCD), enabling it to adapt to the evolving data stream with new concepts in an instance-aware manner. The basic architecture of SCD is a multi-layer perception (MLP) based autoencoder [40]. To measure the parameter shift of the SCD with N_l layers, the hypernetwork models each MLP layer as a matrix $K^{(n)} \in \mathbb{R}^{N_{in} \times N_{out}}$, where N_{in} and N_{out} are the number of input and output neurons of the n-th layer of the MLP respectively. The generation process of matrix $K^{(n)}$ can then be regarded as matrix factorization as below:

$$K^{(n)} = \xi(\vec{r}^{(n)}; \Theta_h), \forall n = 1, \cdots, N_l \tag{1}$$

where $\vec{r}^{(n)}$ is a vector, $\xi(\cdot)$ is a randomly initialized MLP, and Θ_h is the parameters of $\xi(\cdot)$. Such a generation process enables the gradients to backpropagate to $\vec{r}^{(n)}$ and $\xi(\cdot)$ for effective end-to-end training. In this way, the parameter shift of the SCD can be adaptively measured by $\vec{r}^{(n)}$ and $\xi(\cdot)$ instead of $K^{(n)}$ directly.

Evidential Deep Learning. *Evidential Deep Learning* (EDL) [72] is a probabilistic deep learning approach that interprets the categorical predictions of a neural network as a *distribution* over class probabilities by placing a Dirichlet prior upon the class probabilities. This allows the network to provide not only point estimates for the detection but also *uncertainty estimates* for each prediction. In METER, we utilize EDL to measure concept uncertainty, enabling the Intelligent Evolution Controller (IEC) to dynamically evolve the SCD to the Dynamic Shift-aware Detector (DSD) based on the concept uncertainty of the input data.

Considering a general C-class classification task, given an instance $\vec{x}$, a standard DNN with the softmax operator is usually adopted after processing features of $\vec{x}$ to convert the predicted logit vector into the class probability vector $\vec{p}$. When using EDL for such a DNN, a Dirichlet distribution is placed over the categorical likelihood $\vec{p}$ to model the probability density of each possible $\vec{p}$. The

probability density function of $\vec{p}$ for $\vec{x}$ is obtained by:

$$P(\vec{p}|\vec{x};\Theta_e) = Dir(\vec{p}|\vec{\alpha}) = \begin{cases} \frac{\Gamma(\sum_{c=1}^{C}\alpha_c)}{\prod_{c=1}^{C}\Gamma(\alpha_c)}\prod_{c=1}^{C} p_c^{\alpha_c-1}, & \text{if } \vec{p}\in\Delta^C \\ 0 \qquad\qquad\qquad , & \text{otherwise} \end{cases} \tag{2}$$

where $\vec{\alpha}$ is the parameters of the Dirichlet distribution $Dir(\vec{p}|\vec{\alpha})$ for the sample $\vec{x}$, $\Gamma(\cdot)$ is the Gamma function, and Δ^C is the C-dimensional unit simplex: Δ^C=$\{\sum_{c=1}^{C} p_c = 1 \text{ and } 0 \leq p_c \leq 1\}$. Particularly, $\vec{\alpha}$ can be modeled as $\vec{\alpha} = g(f(\vec{x},\Theta_f))$, where $f(\cdot)$ is another DNN model, and $g(\cdot)$ is the exponential function to keep $\vec{\alpha}$ positive. In this way, the prediction of the sample $\vec{x}$ can be interpreted as a distribution over the probability, *i.e.*, the concept uncertainty modeled using IEC, rather than the simple and unreliable predictive uncertainty [56, 72].

3 METHODOLOGY

This section first presents the overview of METER, which is designed to adaptively detect the online anomaly data under concept drift. We then elaborate on each module and introduce the optimization scheme. We further discuss the effectiveness, efficiency and interpretability of our METER.

3.1 Overview

The overview of METER is illustrated in Figure 2, which depicts the entire proposed pipeline. The main intuition is that the evolving data stream with different concepts should be identified and measured for dynamic model evolution. To achieve this goal, we propose four modules, namely Static Concept-aware Detector (Sec. 3.2.1), Intelligent Evolution Controller (Sec. 3.2.2), Dynamic Shift-aware Detector (Sec. 3.2.3) and Offline Updating Strategy (Sec. 3.2.4). We will elaborate on these modules in the following subsections.

3.2 Architecture

3.2.1 Static Concept-aware Detector. The Static Concept-aware Detector (SCD) aims to detect the anomaly data from the static data stream with the central concepts, *i.e.*, measuring the overall distribution of the historical data stream. In anomaly detection tasks, the superior performance and unsupervised learning nature of autoencoders make them widely applicable involving unlabeled data [2, 6]. Thus, we employ an autoencoder as the static base model that is trained on the historical data stream with central concepts to detect the central types of anomalies.

Static Autoencoder. An autoencoder is a deep neural network that learns to reconstruct its inputs. Concretely, given the input instance $\vec{x}$ in the data stream $\mathcal{X}$, a static autoencoder with parameters $\Theta_s = (\Theta_s^E, \Theta_s^D)$ learns to reconstruct $\vec{x}$ by:

$$\vec{x} \Rightarrow \mathcal{E}_s(\vec{x};\Theta_s^E) = \vec{z} \Rightarrow \mathcal{D}_s(\vec{z};\Theta_s^D) = \vec{y}, \quad s.t. \quad \vec{x} \approx \vec{y} \tag{3}$$

where the static autoencoder comprises two main components, an encoder $\mathcal{E}_s(\cdot)$ and a decoder $\mathcal{D}_s(\cdot)$ with parameters Θ_s^E and Θ_s^D, respectively. The encoder $\mathcal{E}_s(\cdot)$ compresses the representation of the input $\vec{x}$ into a latent representation vector $\vec{z}$, and then the decoder $\mathcal{D}_s(\cdot)$ reconstructs the original input into $\vec{y}$ using $\vec{z}$.

Essentially, an autoencoder attempts to learn the identity function of the original data distribution. Therefore, certain constraints are placed on the neural network, forcing it to learn meaningful concepts and relationships among features of $\vec{x}$. As such, the static autoencoder gains the capability to reconstruct unseen inputs during the reconstruction training from the same data distribution $\mathcal{X}$. For anomaly detection, in particular, if the input does not belong to the central concepts learned from $\mathcal{X}$, then we expect the reconstruction to have a larger error.

3.2.2 Intelligent Evolution Controller. While the small/large reconstruction error of the static autoencoder indicates whether the input data is normal/abnormal, this result becomes unreliable when the data is out-of-distribution [30], especially in the presence of concept drift. To determine whether the concept of a given input $\vec{x}$ belongs to central concepts from historical data or new concepts from evolving data streams, we introduce an Intelligent Evolution Controller (IEC). IEC can adaptively and timely examine the necessity of evolving the static detector by detecting the concept drift in the current data stream.

Pseudo Labeling Strategy. IEC is a lightweight evidential classifier with parameters Θ_c trained with the high-confidence pseudo labels from the static autoencoder. The rationale behind introducing pseudo labels in METER is to treat the input instance that the static autoencoder cannot reconstruct well as negative with label 1, while treating the well-reconstructed input as positive with label 0, so that IEC can learn to detect whether the current input belongs to the central concepts already captured in SCD or new concepts:

$$\tilde{y}(\vec{x}) = \begin{cases} 1(\texttt{Positive}), & \text{if } L_2(\vec{x},\vec{y};\Theta_s) > \mu_p \text{ and } \mathcal{U}_{ctr}(\vec{x}) \leqslant \mu_e \\ 0(\texttt{Negative}), & \text{if } L_2(\vec{x},\vec{y};\Theta_s) \leqslant \mu_p \text{ and } \mathcal{U}_{ctr}(\vec{x}) \leqslant \mu_e \\ \text{-}(\texttt{Unknown}), & \mathcal{U}_{ctr}(\vec{x}) > \mu_e \end{cases} \tag{4}$$

where $L_2(\vec{x},\vec{y};\Theta_s)$ is the reconstruction error [66] of the input instance given the static autoencoder computed by taking the root mean squared error between $\vec{x}$ and the reconstructed output $\vec{y}$; μ_p is a predefined pseudo labeling threshold determined by setting a proportion of the sorted reconstruction error over all training samples; $\tilde{y}(\vec{x})$ denotes the pseudo label of $\vec{x}$; $\mathcal{U}_{ctr}(\vec{x})$ is the concept uncertainty that determines the necessity of model evolution, which will be introduced in detail next; μ_e is a predefined threshold to ensure that the static autoencoder is only trained with high-confidence samples with a low concept uncertainty $\mathcal{U}_{ctr}(\vec{x})$, *i.e.*, $\vec{x}$ is not involved in training if the $\mathcal{U}_{ctr}(\vec{x}) > \mu_e$ ($\tilde{y}(\vec{x})$=Unknown).

Concept Uncertainty Estimation. We first give the definition of the predicted probability of the evidential learning-based model. Considering the instance $\vec{x}$, the predicted probability $\hat{P}(y=c|\vec{x};\Theta_c)$ of the class c (0 or 1) following Eq. (2) by marginalizing over $\vec{p}$ is:

$$\begin{aligned} \hat{P}(y=c|\vec{x};\Theta_c) &= \int P(y=c|\vec{p};\Theta_c)P(\vec{p}|\vec{x};\Theta_c)d\vec{p} \\ &= \frac{\alpha_c}{\sum_{k=1}^{C}\alpha_k} = \frac{g(f_c(\vec{x}))}{\sum_{k=1}^{C} g(f_k(\vec{x}))} = \mathbb{E}[Dir(\vec{p}_c|\vec{\alpha})]\,, \end{aligned} \tag{5}$$

where $g(\cdot)$ adopts the exponential function so that the softmax-based prediction can be interpreted as the expectation of the Dirichlet distribution.

For the evidential IEC trained using the supervision of pseudo labels from central concepts, when the concepts of current samples are obviously distinct from central concepts, *i.e.*, the data is out-of-distribution, the evidence collected for these samples will be insufficient, as the controller now lacks the knowledge of such new

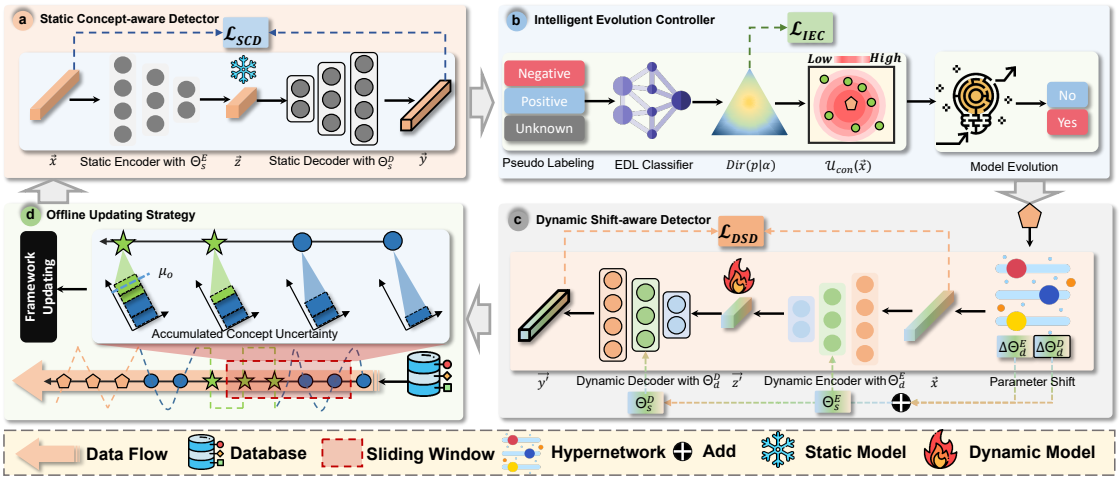

Figure 2: Overview of the proposed METER. (a) Static Concept-aware Detector (SCD) is first trained on historical data to model the central concepts. (b) Intelligent Evolution Controller (IEC) timely measures the concept uncertainty to determine the necessity of dynamic model evolution. (c) Dynamic Shift-aware Detector (DSD) dynamically updates SCD with the instance-aware parameter shift by considering the concept drift. (d) Offline Updating Strategy (OUS) introduces an effective framework updating strategy according to the accumulated concept uncertainty in a given sliding window.

concepts [56, 72]. Built upon this, we introduce the uncertainty resulting from the lack of evidence, called *concept uncertainty*, to measure the extent to which the current concept drifts. Formally, concept uncertainty $\mathcal{U}_{ctr}$ of the instance $\vec{x}$ is defined as:

$$\mathcal{U}_{ctr}(\vec{x};\Theta_c) = \sum_{c=1}^{C} \hat{P}(y=c|\vec{x};\Theta_c)\,(\Phi(\alpha_c+1) - \Phi(\sum_{k=1}^{C}\alpha_k+1)) - \sum_{c=1}^{C}\hat{P}(y=c|\vec{x};\Theta_c)\log\hat{P}(y=c|\vec{x})\,, \tag{6}$$

where $\Phi(\cdot)$ is the digamma function. Here, we use mutual information to measure the spread of Dirichlet distribution on the simplex following [72]. A higher $\mathcal{U}_{ctr}(\vec{x})$ indicates a larger concept uncertainty, *i.e.*, the Dirichlet distribution is widely dispersed across the probability simplex.

When deriving a higher concept uncertainty $\mathcal{U}_{ctr}(\vec{x}) > \mu_e$, the OAD prediction from the static autoencoder is interpreted as unreliable. In this way, IEC measures instance-aware concept shift and enables the dynamic shift-aware detector to further process the current input instance.

3.2.3 Dynamic Shift-aware Detector. As discussed above, the SCD module can effectively measure the central concepts derived from the historical data stream. However, SCD may fail to support accurate detection when the statistical or distributional properties of the data stream shift in new data streams over time. In this sense, modeling new concepts when concept drift occurs is critical, which enhances the detector for accurate detection across time. To handle evolving data streams, we introduce a Dynamic Shift-aware Detector (DSD) that effectively enhances the static autoencoder with the new concepts in the current data stream.

Parameter Shift Measurement. To measure concept drift, we leverage a hypernetwork to learn the parameters shift $\Delta\Theta_d$ for SCD with the same set of parameters, namely the encoder parameter shift $\Delta\Theta_d^E$ and the decoder parameter shift $\Delta\Theta_d^D$. However, as shown in Eq.(1), the original hypernetwork only utilizes a randomly initialized $\vec{r}^{(n)}$ to generate the model parameters, which lacks the interaction between the parameter generation process and the current input instance. To measure the instance-aware parameter shift, we propose to model the parameter shift by replacing the $\vec{r}^{(n)}$ with representations of the current input instance. Specifically, given the input $\vec{x}$, the hypernetwork first employs a layer-specific subnetwork $E^{(n)}(\cdot)$ that captures $\vec{x}$ related features as $\vec{e}^{(n)}$ for the parameter shift generation of the n-th layer parameters. To reuse features and reduce model parameters, different layers share one encoder network $E_{\text{share}}(\cdot)$ while employing different linear layers to get the layer-specific representation vector $e^{(n)}$ in the hypernetwork:

$$\vec{e}^{(n)} = E^{(n)}(\vec{x}) = L^{(n)}_{\text{layer}}(E_{\text{share}}(\vec{x})), \text{n} = 1, \cdots, \mathcal{N}_{\text{d}}, \tag{7}$$

where N_d is the number of the encoder and decoder layers, and $L^{(n)}_{\text{layer}}(\cdot)$ is the linear layer to transform the output of $E_{\text{share}}(\cdot)$ to the n-th layer features.

The parameter shift of the n-th layer of the SCD encoder/decoder can be formatted as a matrix $K^{(n)} \in \mathbb{R}^{N_{in}\times N_{out}}$, where N_{in} and N_{out} are the number of input neurons and output neurons respectively. Then, we transform the input-aware feature vector $\vec{e}^{(n)}$ into the parameter shift of the corresponding layer. Specifically, the hypernetwork further employs the following two MLP layers to generate the parameter shift of the n-th layer:

$$W^{(n)} = (W_1\vec{e}^{(n)} + \vec{b}_1)W_2 + \vec{b}_2, \qquad K^{(n)} = W^{(n)} + \vec{b}^{(n)}, \tag{8}$$

where W_1 and W_2 are weights of the two MLP layers of the hypernetwork respectively, $\vec{b}_1$, $\vec{b}_2$ and $\vec{b}$ are the biases. Altogether, parameter

shifts of the encoder and decoder for the static autoencoder can be denoted as $\Delta\Theta_d^E$ and $\Delta\Theta_d^D$.

Dynamic Autoencoder. After obtaining the parameter shift of the static autoencoder, the parameters for the encoder and decoder of the dynamic autoencoder are as follows:

$$\Theta_d = \begin{cases} \Theta_d^E = \Theta_s^E + \Delta\Theta_d^E \\ \Theta_d^D = \Theta_s^D + \Delta\Theta_d^D \end{cases} \tag{9}$$

where Θ_d^E and Θ_d^D are the parameters of the dynamic autoencoder, which can adapt the weight of SCD by measuring the parameter shift, thereby improving the OAD accuracy by dynamically modeling the new concepts in the evolving data stream.

Based on the learned parameters of the dynamic autoencoder, the reconstructing procedure is similar to the static autoencoder:

$$\vec{x} \Rightarrow \mathcal{E}_d(\vec{x};\Theta_d^E) = \vec{z'} \Rightarrow \mathcal{D}_d(\vec{z'};\Theta_d^D) = \vec{y'}, \quad s.t. \quad \vec{x} \approx \vec{y'} \tag{10}$$

where $\vec{z'}$ and $\vec{y'}$ are the latent vector and the reconstructed representation using the dynamic autoencoder respectively.

3.2.4 Offline Updating Strategy. The entire framework automatically updates its modules based on the accumulated concept uncertainty within a sliding window over the evolving data stream. Specifically, METER keeps monitoring if the concept uncertainty accumulated within this window surpasses a predetermined threshold, and frequent occurrence of such events indicates that the SCD trained on the historical data streams is incapable of handling the current data stream due to increased concept drift over time. This suggests that the entire framework should be updated, namely the SCD, IEC and DSD module, so as to better adapt to the current data stream. The Offline Updating Strategy (OUS) is as follows:

$$\mathcal{UP} = \begin{cases} 1, & \text{if } \sum_{i=t}^{t+\Delta L}(\mathbb{1}_{(\mathcal{U}_{ctr}(\vec{x}_i)>\mu_e)} \cdot \mathcal{U}_{ctr}(\vec{x}_i)) > \mu_o \text{ or } \Delta t > T_{max} \\ 0, & \text{else} \end{cases} \tag{11}$$

where $\mathcal{UP}$=1 indicates the framework should be updated, ΔL is the interval of the sliding window, Δt is the time since the last framework update, and μ_o and T_{max} are the threshold of the framework update and maximum interval since the last update respectively. $\sum_{i=t}^{t+\Delta L}(\mathbb{1}_{(\mathcal{U}_{ctr}(\vec{x}_i)>\mu_e)} \cdot \mathcal{U}_{ctr}(\vec{x}_i)$ aggregates the concept uncertainty greater than the threshold μ_e within the current sliding window. We set the threshold μ_e to the largest value of the concept uncertainty during training and update it using an exponential moving average (EMA) strategy [41] during updating. Specifically, we fine-tune the model using data within the current sliding window. Also, we adopt a parallel training strategy that performs online OAD inference using the latest fine-tuned modules and meanwhile, fine-tunes key modules offline, which decouples the training and inference for efficiency and modularity, and allows for executing OUS without affecting the online OAD inference service.

3.3 Optimization

METER is trained in two stages: (i) In the first stage, we train the Static Concept-aware Detector (SCD) using the historical data stream $\mathcal{X}_h$. We note that the initial training of SCD only uses a small subset of the data stream. The reconstruction error can be computed by taking the $L_2(\cdot)$ loss, *i.e.*, the squared difference between the input and the reconstructed output. Given $\vec{x}$ in $\mathcal{X}_h$, the $\mathcal{L}_{SCD}$ is:

$$\mathcal{L}_{SCD}(\Theta_s) = L_2(\vec{x},\vec{y}) = \frac{\sum_{i=1}^{n}(\vec{x}_i - \vec{y}_i)^2}{n} \tag{12}$$

where n is the dimension of features of the input instance. (ii) In the second stage, the historical instances are first labeled following Eq.(4), then we follow [72] to train the Intelligent Evolution Controller (IEC). Specifically, we treat $Dir(\vec{p}|\alpha)$ as a prior on the likelihood and obtain the negated logarithm of the marginal likelihood $\mathcal{L}_{IEC}$ by integrating out the class probabilities:

$$\mathcal{L}_{IEC}(\Theta_c) = \sum_{c=1}^{C}(\log(\sum_{c=1}^{C}\alpha_c) - \log\alpha_c) \tag{13}$$

The training process of the Dynamic Shift-aware Detector (DSD) is similar to the SCD. Notably, the gradients backpropagated to the hypernetwork together with the SCD. $\mathcal{L}_{DSD}$ is defined as below:

$$\mathcal{L}_{DSD}(\Theta_d) = L_2(\vec{x},\vec{y'}) = \frac{\sum_{i=1}^{n}(\vec{x}_i - \vec{y'}_i)^2}{n} \tag{14}$$

After the two-stage training, METER can perform inference for the incoming data stream, which is summarized in Algorithm 1.

Algorithm 1: METER Inference

Input: Evolving stream data $\mathcal{X}$
Output: Anomaly scores of $\mathcal{X}$
Initialization: Trained SCD, IEC and DSD parametrized by Θ_s, Θ_c and Θ_d
for $\vec{x}$ *in* $\mathcal{X}$ **do**
  Computing the concept uncertainty $\mathcal{U}_{ctr}(\vec{x})$ using Eq.(6)
  if $\mathcal{U}_{ctr}(\vec{x}) > \mu_e$ **then**
    Update SCD to DSD using Eq.(9)
    return *Anomaly score* $L_2(\vec{x},\vec{y'})$
  else
    return *Anomaly score* $L_2(\vec{x},\vec{y})$
if $\sum_{i=t}^{t+\Delta L}(\mathbb{1}_{(\mathcal{U}_{ctr}(\vec{x}_i)>\mu_e)} \cdot \mathcal{U}_{ctr}(\vec{x}_i) > \mu_o$ or $\Delta t > T_{max}$ **then**
  $\mathcal{X}_h \Rightarrow \mathcal{X}_s$
  Update METER
  Break

3.4 Analysis and Discussion

Effectiveness. METER addresses the concept drift challenge by integrating two detectors and an IEC. The SCD detector leverages historical data and prior knowledge for the detection, while the DSD detector dynamically learns the parameter shift to enhance SCD and adapts to new concepts effectively in an instance-aware manner. Notably, IEC determines whether concept drift occurs, circumventing the risk of employing an ineffective detection model for anomaly detection. With IEC, the adaptability and generalizability of METER are substantially improved, leading to enhanced accuracy in anomaly detection. In addition, the offline update strategy provides an efficient way to keep up with new concepts in evolving data streams, thereby ensuring high-quality detection results.

Efficiency. Efficiency is an important consideration in OAD due to the need for timely responses to evolving data streams. METER introduces several strategies to achieve high detection efficiency. METER employs the uncertainty estimate derived from evidential

Table 1: Overall performance comparison of unknown drifts in a discrete setting. A larger score has better performance. Acronym notations of baselines can be found in Sec. 4.1.2. We mark best (bold and underline) and second best(bold) in each row.

Model Class	Model	Ion.		Pima		Satellite		Mamm.		BGL		NSL		KDD99		Average
		AUCROC	AUCPR	AUCROC	AUCPR	AUCROC	AUCPR	AUCROC	AUCPR	AUCROC	AUCPR	AUCROC	AUCPR	AUCROC	AUCPR	Rank
Traditional	LOF [12]	0.874	0.827	0.542	0.371	0.598	0.481	0.720	0.089	0.542	0.206	0.586	0.428	0.653	0.359	10.133
	IF [47]	0.860	0.817	0.677	0.502	0.676	0.375	0.867	0.211	**0.823**	0.295	0.530	0.577	0.784	0.406	7.333
	KNN	**0.929**	**0.932**	0.615	0.457	0.677	0.539	0.839	0.156	0.765	0.274	0.897	0.899	0.946	0.902	6.133
	STORM [5]	0.640	0.526	0.529	0.373	0.680	0.452	0.615	0.418	0.203	0.043	0.513	0.138	0.913	0.822	10.933
Incremental	RRCF [26]	0.586	0.411	0.575	0.393	0.553	0.356	0.713	0.524	0.540	0.076	0.604	0.534	0.773	0.347	11.067
	MStream [7]	0.681	0.486	0.524	0.440	0.647	0.457	0.798	0.076	0.531	0.105	0.759	0.716	0.958	**0.912**	9.333
	MemStream [8]	0.821	0.672	**0.703**	0.551	0.722	**0.682**	**0.902**	0.225	0.694	0.144	**0.988**	**0.967**	**0.979**	0.857	5.200
Ensemble	HS-Trees [75]	0.687	0.574	0.667	0.344	0.512	0.348	0.797	0.623	0.599	0.174	0.806	0.735	0.901	0.728	9.267
	iForestASD [21]	0.744	0.601	0.515	0.356	0.642	0.451	0.575	0.031	0.701	**0.382**	0.511	0.483	0.532	0.227	11.000
	RS-Hash [69]	0.743	0.502	0.518	0.372	0.640	0.586	0.776	0.622	0.436	0.245	0.684	0.524	0.783	0.707	10.000
	LODA [62]	0.514	0.373	0.501	0.347	0.500	0.316	0.500	0.023	0.523	0.074	0.504	0.535	0.507	0.197	14.000
	Kitsune [53]	0.920	0.896	0.590	0.451	0.732	0.673	0.603	0.202	0.514	0.074	0.947	0.918	**0.982**	**0.993**	6.533
	xStream [52]	0.773	0.591	0.656	**0.583**	0.659	0.533	0.847	**0.630**	0.623	0.356	0.540	0.327	0.954	0.881	7.067
	PIDForest [25]	0.821	0.718	0.669	0.474	0.718	0.543	0.847	0.202	0.791	0.300	0.503	0.561	0.864	0.772	7.467
	ARCUS [87]	0.919	0.894	0.607	0.420	**0.797**	0.560	0.812	**0.261**	0.768	0.185	0.262	0.365	0.972	0.807	7.533
Ours	METER	**0.950**	**0.956**	**0.733**	**0.654**	**0.796**	**0.777**	**0.913**	0.491	**0.895**	**0.369**	**0.982**	**0.963**	0.973	0.853	3.0001

deep learning to detect concept drift, and thus avoids the need for frequent model retraining. This considerably reduces computational costs and enhances the responsiveness and overall efficiency of the detection model. In addition, METER dynamically switches between the base detection model and a dynamic shift-aware detector (DSD) supported by the hypernetwork for detecting anomalies. DSD dynamically generates parameter shifts to account for the current concept, thereby handling concept drift on the fly and enhancing the predictive performance of the base detection model without further fine-tuning or training. The dynamic concept adaptation technique enables METER to harness the strengths of both models, which supports efficient AD in rapidly changing data streams.

Interpretability. METER supports interpretability for OAD in terms of providing reliable uncertainty estimates for detection results, which is important and necessary for users in high-stakes applications [18, 90, 91]. We note that softmax probabilities produced by detection models of existing OAD approaches are unreliable uncertainty estimates [56, 72]. As such, providing a reliable measure of prediction results is challenging, and is often neglected by existing OAD approaches [8, 37, 87, 88]. Inspired by recent research utilizing subjective logic (SL) theory [36] to improve the interpretability of decision-making processes, e.g., in domains such as multi-view classification [23, 29] and molecular property prediction [73], we derive high-quality uncertainty modeling via evidential deep learning, which utilizes SL theory to explicitly model the reliability of predictions generated by METER. Specifically, SL formalizes Dempster-Shafer Evidence Theory's notion of belief assignments over a frame of discernment as a Dirichlet distribution [72], forming opinions for anomaly detection. In practice, METER monitors whether concept drift occurs for the current input using concept uncertainty modeled by IEC, which can be visualized on a per-input basis for improving the user's understanding of the detection.

4 EXPERIMENTS

In this section, we conduct experiments to systematically evaluate the effectiveness, efficiency and interpretability of METER.

4.1 Experimental Setup

4.1.1 Datasets and Applications. We adopt 17 real-world benchmark datasets from various domains with different types of concept drift, dimensions, number of data points, and anomaly rates. These datasets are widely benchmarked in related studies and are representative of evaluating the effectiveness of different OAD approaches to detect anomalies and adapt to concept drift. Given the diverse types of data encountered in data streams for OAD, we categorize the datasets into two settings: discrete and continuous. In the continuous setting, data streams exhibit temporal dependencies between successive time steps, whereas in the discrete setting, such dependencies may either be absent or remain unknown. By conducting evaluations on both settings, we aim to comprehensively evaluate the performance of different OAD approaches.

We first adopt four commonly used anomaly detection datasets from the UCI repository and ODDS library [63], namely Ionosphere (Ion.), Pima, Satellite, Mammography (Mamm.). Secondly, we utilize the BGL [57] dataset, a large public dataset consisting of log messages collected from a BlueGene/L supercomputer system at Lawrence Livermore National Labs. To facilitate analysis, each log message is processed into the structured data format. The third category is popular multi-aspect datasets of intrusion detection, namely KDDCUP99 [1] (KDD99) and NSL-KDD [77] (NSL). The next category is time-series datasets procured from two benchmarks: Numenta anomaly detection benchmark [3] (NAB) and HexagonML [19] (UCR). We adopt datasets including NYC taxicab (NYC), CPU utilization (CPU), Machine temperature (M.T.) and Ambient temperature (A.T.) from NAB, commonly employed for evaluating streaming anomaly detection algorithms. As for UCR, we selectively adopt datasets obtained from natural sources, specifically EPG and ECG. We also adopt real-world streaming datasets INSECTS [74] for simulating concept drift, consisting of optical sensor values collected while monitoring flying insects, with temperature level as the controlled concept.

4.1.2 Baseline Methods. We compare METER with 15 baselines in three categories: (1) Representative and widely used anomaly detection algorithms, namely Local Outlier Factor (LOF) [12], Isolation Forest (IF) [47], and k-Nearest Neighbors (KNN). STORM [5] is also an important stream data anomaly detection work that is often used as a baseline [8, 68, 80]; (2) Incremental learning-based approaches, namely RRCF [26], MStream [7], and MemStream [8]; (3) Ensemble-based approaches, namely HS-Trees [75], iForestASD [21], RS-Hash [69], LODA [62], Kitsune [53], xStream [52], PIDForest [25], and ARCUS [87].

Table 2: Overall performance comparison of Unknown drifts in a continuous setting.

Model Class	Model	M.T. AUCROC	M.T. AUCPR	A.T. AUCROC	A.T. AUCPR	NYC. AUCROC	NYC. AUCPR	CPU. AUCROC	CPU. AUCPR	EPG AUCROC	EPG AUCPR	ECG AUCROC	ECG AUCPR	Average Rank
Traditional	LOF [12]	0.501	0.141	0.563	0.126	0.671	0.211	0.560	0.112	**0.934**	0.679	0.670	0.016	9.667
	IF [47]	0.829	0.573	0.762	0.362	0.624	**0.331**	0.817	**0.760**	0.811	0.552	0.668	0.005	5.500
	KNN	0.759	0.255	0.634	0.200	**0.697**	0.202	0.724	0.452	0.083	0.001	0.247	0.002	10.000
	STORM [5]	0.604	0.127	0.518	0.105	0.460	0.097	0.667	0.605	0.578	0.436	0.662	**0.523**	11.667
Incremental	RRCF [26]	0.628	0.153	0.519	0.110	0.502	0.121	0.617	0.368	0.814	0.498	0.387	0.002	11.667
	MStream [7]	**0.860**	0.505	0.619	0.156	0.639	0.168	0.794	0.443	0.824	0.621	0.721	0.284	6.417
	MemStream [8]	0.825	**0.573**	0.722	0.334	**0.731**	0.311	0.831	0.227	0.930	0.656	**0.780**	0.007	4.500
Ensemble	HS-Trees [75]	0.617	0.359	0.522	0.310	0.558	0.269	0.678	0.585	0.531	0.334	0.621	0.439	9.083
	iForestASD [21]	0.738	0.231	0.514	0.167	0.501	0.117	0.755	0.153	0.782	0.470	0.733	0.006	10.167
	RS-Hash [69]	0.607	0.549	0.742	0.180	0.524	0.106	0.712	0.467	0.552	0.186	0.584	0.203	9.833
	LODA [62]	0.503	0.100	0.499	0.101	0.499	0.101	0.500	0.083	0.595	0.182	0.721	0.077	13.500
	Kitsune [53]	0.684	0.416	0.599	0.274	0.465	0.124	0.824	0.669	0.897	0.020	0.726	0.231	7.833
	xStream [52]	0.696	0.596	0.567	0.319	0.586	0.121	0.730	0.195	0.687	0.158	0.705	0.365	8.250
	PIDForest [25]	0.789	0.389	**0.797**	0.320	0.513	0.113	**0.881**	0.439	0.836	0.238	0.683	0.006	7.667
	ARCUS [87]	0.376	0.511	0.518	**0.389**	0.470	0.317	0.678	0.128	0.885	**0.724**	0.682	0.340	8.417
Ours	METER	**0.842**	**0.652**	**0.782**	**0.399**	0.688	**0.386**	**0.908**	**0.715**	**0.968**	0.625	**0.778**	**0.495**	1.833

Table 3: Overall performance comparison of known drifts in a continuous setting.

Model Class	Model	INSECTS -Abr	INSECTS -Inc	INSECTS -IncGrd	INSECTS -IncRec	Average Rank	Time (s)
Traditional	LOF [12]	0.578	0.556	0.589	0.526	11.250	180
	IF [47]	0.679	0.632	0.697	0.593	6.500	67
	KNN	0.666	0.597	0.673	0.553	8.000	105
	STORM [5]	0.408	0.441	0.446	0.449	16.500	122
Incremental	RRCF [26]	0.600	0.579	0.624	0.593	9.250	121
	MStream [7]	0.703	**0.698**	**0.788**	**0.672**	3.250	18
	MemStream [8]	0.753	0.348	0.728	0.361	10.750	109
Ensemble	HS-Trees [75]	0.499	0.507	0.497	0.499	14.250	302
	iForestASD [21]	0.599	0.589	0.616	0.575	9.500	7985
	RS-Hash [69]	0.484	0.509	0.459	0.506	14.250	225
	LODA [62]	0.498	0.503	0.496	0.499	14.750	831
	Kitsune [53]	**0.759**	0.584	0.730	0.594	5.250	164
	xStream [52]	0.514	0.516	0.533	0.504	12.750	408
	PIDForest [25]	0.757	0.675	**0.748**	0.631	3.500	18047
	ARCUS [87]	0.601	0.597	0.576	0.632	8.000	79
Ours	METER	**0.816**	**0.795**	0.712	**0.794**	2.250	88

4.1.3 Evaluation Metrics. We adopt AUCROC and AUCPR as evaluation metrics. AUCROC is the area under ROC curve, which plots the false-negative rate (FNR) as the x-axis and the true-positive rate (TPR) as the y-axis at different thresholds. AUCPR is the area under PR curve, which plots the precision against recall at different thresholds. The metrics fall within the range $[0, 1]$, and a higher value indicates better detection performance.

4.1.4 Implementation Details. We implement LOF and IF using the *scikit-learn* library [61], and KNN using the pyod library [92]. The open-source PySAD library [85] is used to implement STORM [5], RRCF [26], HS-Trees [75], iForestASD [21], RS-Hash [69], LODA [62], and xStream [52] with default parameters. For other baseline methods such as MStream [7], MemStream [8], Kitsune [53], PIDForest [25], ARCUS [87], we adopt the official implementations, using the recommended parameter settings. For ARCUS, we use the base model RAPP [38]. In cases where default parameter values are not provided, we conduct a grid search to find the optimal parameters that yield the best performance. Adam [39] is used as an optimizer in all learning-based models with a learning rate searched in $0.1 \sim 1e\text{-}3$.

For METER, the encoder and decoder are implemented as 2 to 10-layer DNNs with a symmetric structure, where the autoencoder's latent space dimension is set to the number of principal components to ensure a minimum of 70% explained variance following prior research [38, 87]. IEC is a two-layer DNN with the ReLU activation function. We use an Adam optimizer with a learning rate of $1e\text{-}2$ with an exponential decay rate of 0.96. The number of epochs is set to 1000. We utilize grid search for hyperparameter tuning. Specifically, the threshold rate μ_p for the pseudo labels from SCD is searched within the range of 0.05 to 0.5 with an interval of 0.05. The threshold of the concept uncertainty μ_e is searched in {0.001, 0.005, 0.01, 0.1, 0.2, 0.4}. While the threshold of the offline updating strategy μ_o is in the range of 0.1 to 1 of the window size, and the window size ΔL is set to 64. The historical data ratio h_r is set to 0.2, meaning that 20% of the dataset is utilized as a historical data stream $\mathcal{X}_s$ for training purposes. We conduct a sensitivity analysis on the three key threshold hyperparameters of our framework, along with the window size ΔL and the historical data ratio h_r in Section 4.6. We create a data generator to simulate the generation of streaming data and report the average value of 5 independent runs for all baselines. All the experiments are conducted in a server with Xeon(R) Silver 4114 CPU @ 2.2GHz (10 cores), 256G memory, and GeForce RTX 2080 Ti. All the models are implemented in PyTorch 1.10.0 with CUDA 10.2.

Table 4: Performance of ablation experiments, where AUC is the average AUCROC value on the four INSECTS datasets.

Variant	SCD	DSD	IEC	OUS	AUC
METER-S	✓	×	×	×	0.604
METER-D	×	✓	×	×	0.681
METER-S+D	✓	✓	×	×	0.696
METER w/o IEC	✓	✓	×	✓	0.713
METER w/o OUS	✓	✓	✓	×	0.741
METER	✓	✓	✓	✓	0.779

4.2 Effectiveness

We compare METER with other baselines on the 17 real-world datasets with results reported in Table 1, 2 and 3. Our METER achieves the best performance in most scenarios, across different concept drift types and problem settings (discrete or continuous). Notably, as shown in Table 1, METER outperforms the top-performing baseline by 2.2% on Ionosphere and 4.3% on Pima in terms of AUCROC. Meanwhile, METER obtains the highest AUCROC of 0.968 and the second-highest AUCROC of 0.778 (very close to the best AUCROC of 0.780 achieved by MemStream on ECG) on the more challenging dataset EPG and ECG, respectively. Likewise, drawing insights from Table 2, METER performs exceptionally well

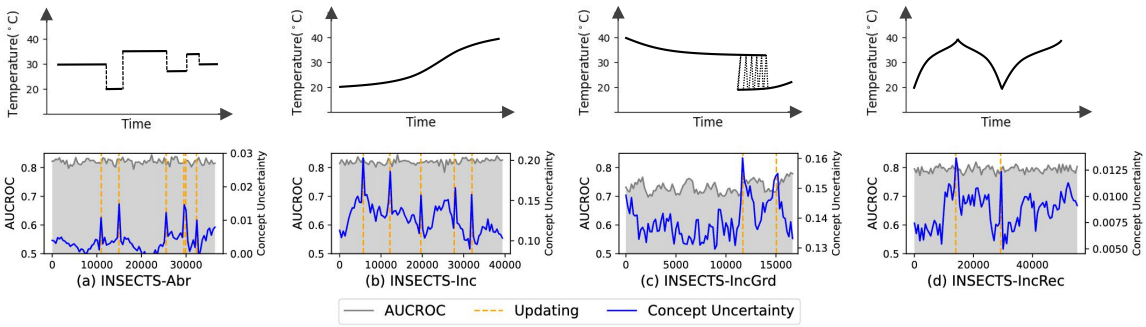

Figure 3: Analysis of concept drift adaptation on INSECTS dataset.

on time series, which is the top-ranked model on average. We adopt the shingling approach following the convention [25] with a window width of 10. The preprocessed vectors are then input to METER, enabling the model to capture short-term temporal dependencies within the time window. Further, the hypernetwork-based DSD is designed to learn and capture long-term variations in time series by dynamically generating the parameter shifts of SCD weights. Together, they enable METER to handle the temporal dependencies and achieve overall the best performance across different settings.

While some methods demonstrate remarkable performance on certain datasets, they lack consistency. For instance, ARCUS achieves the highest AUCPR on EPG, while it only obtains an AUCPR of 0.340, i.e., 34.99% worse than STORM with the highest AUCPR of 0.523. Table 3 illustrates that about half of the methods exhibit subpar performance on real-world datasets with known concept drift. Notably, although methods like MemStream obtain competitive performance on INSECT-Abr and INSECT-IncRec, they perform worse on the other two datasets, whose performance is even worse than METER using only the static concept-aware detector as shown in Table 4. This demonstrates the complexity of OAD when dealing with real-world datasets characterized by distinct concept drifts. Different kinds of concept drifts require very different modeling strategies, and thus resilient models that can adapt to a wide spectrum of concept drift scenarios are much needed. In this context, consistently outperforms baseline models across various settings and types of concept drift, achieving overall the highest rank across datasets. Also, METER shows high computational efficiency. As shown in Table 3, the average running time of METER on INSECTS is only 88s, which is substantially lower than most of the baselines, while still delivering superior performance.

4.3 Concept Drift

To validate the effectiveness of METER in real-time detection and rapid response to concept drift, we monitor the evolution of concept uncertainty of METER (depicted as a blue line) on real data streams subject to concept drift, specifically on INSECTS. In this experiment, the changes in temperature are used as indicators of concept drift. In addition, we plot the timing of model updates (indicated by the orange dotted line) and the corresponding changes in AUCROC (indicated by the gray line) over time. This tracks the performance of METER as it adapts to evolving data streams. For a comprehensive understanding of the monitoring and adaptation effects of concept drift, we adopt equidistant sampling across the entire dataset, given that our method operates at the instance level. By selecting 100 equidistant points for both uncertainty and AUCROC analysis, we ensure a representative snapshot of the data stream's evolution. As shown in Figure 3, despite some volatility in uncertainty across the data stream, there is generally a sharp increase in uncertainty when concept drift occurs, with virtually no delay. We note that the initial drift results in a sharp spike in the concept uncertainty at the beginning of concept drift, e.g., the time step around 7000 in INSECTS-Inc, and more moderated concept uncertainty during the subsequent concept drifts, even if an increasing trend is observed, such as the concept shift occurring at time step 20000. This phenomenon is also consistent with the observation in INSECTS-IncGrd and INSECTS-IncRec. Further, the timing of model updates aligns with these sharp increases in uncertainty, corroborating the rapid response of METER to the emergence of anomalies. Crucially, METER consistently maintains high AUCROC scores on real data streams, showing stability and reliability without significant fluctuations in the presence of concept drift.

4.4 Ablation Study

Effectiveness of each module: Extensive ablation studies are conducted on four real-world datasets with different types of concept drifts to evaluate the contribution of individual modules in METER. The composition of the variants and their average AUCROC results on INSECTS are detailed in Table 4, with separate results provided in Figure 4. The results show that each module contributes considerably to improving the detection performance. Notably, the experiments suggest that DSD plays a more important role than SCD. Also, both IEC and OUS, introduced to detect and address concept drift respectively, are critical to support effective OAD. Similar results can be observed in Figure 4. Further, the increase in the detection performance of METER becomes more prominent as the data stream's sample size grows (e.g., the gain curve on INSECTS-IncRec with 67,455 samples is steeper than that on INSECTS-IncGrd with 20,367 samples), demonstrating the scalability of METER to large-scale data stream scenarios.

Instance-specific information: A dedicated ablation study is conducted to assess the superiority of adopting the instance-aware input for hypernetwork over the conventional approach of using

Table 5: Effects of instance-specific information, prior knowledge and various SCD implementations on METER.

Variant	INSECTS -Abr	INSECTS -Inc	INSECTS -IncGrd	INSECTS -IncRec	Average
METER-re	0.640	0.482	0.526	0.617	0.566
METER-pl	0.827	0.806	0.762	0.782	0.794
METER	0.814	0.795	0.713	0.795	0.779
METER-lstm	0.915	0.844	0.765	0.827	0.838
METER-conv	0.893	0.825	0.693	0.816	0.807
Res-METER-conv	0.910	0.832	0.742	0.821	0.826

Table 6: Validation of recurring patterns.

Variant	p_{1-test}	p_{2-test}	p_{3-test}	p_{4-test}	p_{5-test}	Average
Group I	0.804	0.681	0.711	0.702	0.801	0.740 ± 0.052
Group II	0.804	0.710	0.734	0.754	0.837	0.768 ± 0.046

random embeddings. We denote the variant by METER-re. Remarkably, instance-specific information significantly improves the detection performance of METER. In contrast to the original hypernetwork, which uses random embeddings and thus has a weak correlation between parameter generation and the current input instance, METER takes into account instance-specific inputs, leading to instance-aware modeling. This enables METER to effectively capture the dynamic changes in streaming data, thereby making it more efficient and effective for online anomaly detection.

Prior knowledge: We introduce a variant of METER to evaluate its adaptability and generalization in situations with limited labeled samples. Table 5 reports the experimental results, where METER-pl denotes the pseudo-labeling strategy enhanced by incorporating 1% of labeled anomalies into the training set. The results in Table 5 also reveal that the evidential IEC successfully leverages prior knowledge and considerably enhances the learning capacity of METER by incorporating only a small number of labeled samples.

Flexibility: We conducted comprehensive testing of various SCD designs to thoroughly explore the flexibility of METER, including LSTM, 1D convolution, and 1D convolution with residual connections. For a fair comparison, all the variants share the same base structure as METER's DNNs implementation. Results in Table 5 show that all the three enhanced SCD modules can achieve noticeably better detection performance, with an increase in AUCROC of 7.57%, 3.59%, and 6.03% respectively, as compared to the original SCD module based on a canonical DNNs. This not only confirms the adaptability and flexibility of METER but also points out enhancement directions for further improving METER's performance.

Reccurring Patterns: We partition INSECTS-Abr dataset into five sequential and equal subsets, denoted as p_1 to p_5. Then, we further split each subset into one training set and one test set, and derive $p_{1-train}/p_{1-test}$ to $p_{5-train}/p_{5-test}$ correspondingly. Using these subsets, we conduct two groups of experiments. In Group I, we train METER on $p_{1-train}$ and then fine-tune METER on $p_{2-train}$ to $p_{5-train}$ respectively, and report the detection performance of METER. As for the experiments of Group II, we only train METER on $p_{1-train}$ once, and then report the trained METER on all test sets, namely p_{1-test} to p_{5-test}. Results summarized in Table 6 show that (1) the fine-tuning strategy is not effective in OAD, which may lead to worse performance than the model initially trained but without fine-tuning, e.g., the Group I model archives an AUCROC of 0.702 on p_{4-test}, which is much worse than 0.754 of the Group II model without further fine-tuning. (2) The model of Group II archives an AUCROC of 0.768, which is only slightly worse than 0.804 obtained on p_{1-test}, and obtains an even higher AUCROC of 0.837 on p_{5-test}. These two findings suggest that the majority of anomaly patterns are indeed already encompassed on the status historical data, namely $p_{1-train}$, and the model of Group II only trained on $p_{1-train}$ can detect most anomalies on the subsequent unseen test sets, which achieve so by learning the recurring central concepts encompassed in the static historical data.

Table 7: Training and inference efficiency of METER.

Dataset	Throughput Training	Throughput Inference	Training Time(s)	Memory(MiB)
Ion.	9,044	181,225	0.008	6.18
NSL	142,856	85,435,573	0.020	16.28
M.T.	303,025	2,240,917	0.015	9.70
CPU	206,337	4,601,219	0.015	9.22
INSECTS-Abr	520,843	24,662,015	0.017	38.08

Table 8: The performance under different ΔL.

Window Size ΔL	32	48	64	96	128	256
AUCROC	0.875	0.862	0.856	0.832	0.805	0.798
Time(s)	253	248	188	173	133	98

4.5 Efficiency

We evaluate the training and inference efficiency of METER by measuring throughput, denoted as the number of processed samples per second, on diverse benchmark datasets. Results in Table 7 show that METER is rather efficient in both training and inference on datasets of various sizes and dimensions. This indicates that METER exhibits high computational efficiency and thus supports rapid response rates for real-time OAD applications. Another issue worth concerning is the training time and the maximum memory usage (peak memory). Considering METER's adaptive offline update capability to handle concept drift, achieving a shorter training time and a smaller peak memory becomes especially desirable. Table 7 reports the average training time for each epoch and the peak memory. The results demonstrate that METER requires negligible training time and takes low memory usage across these datasets. This is mainly due to the lightweight design of the key modules of METER, as discussed in detail in Section 3.4. To provide further insights into this matter, we conduct tests to assess the efficiency impact of the IEC and DSD modules. Specifically, we compare peak memory usage and average training time per epoch on CPU of METER with and without IEC and DSD. Results show that the introduction of the IEC and DSD modules incurs a negligible increase in training time (0.014s and 0.007s, respectively) and peak memory usage (5.17MB and 2.61MB, respectively). Further, our ablation studies in Table 4 show that integrating these two modules into METER enhances the performance by a large margin. These findings corroborate the efficiency and effectiveness of the IEC and DSD in our framework, which is well-suited for real-time applications and supports high-performance OAD with efficiency.

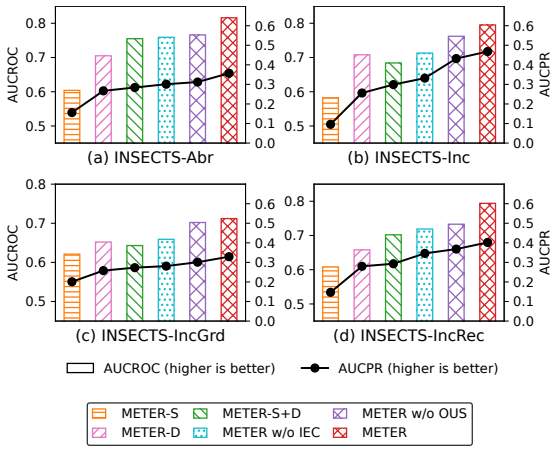

Figure 4: Ablation analysis. The AURROC and AUCPR performance of six METER variants on INSECTS datasets.

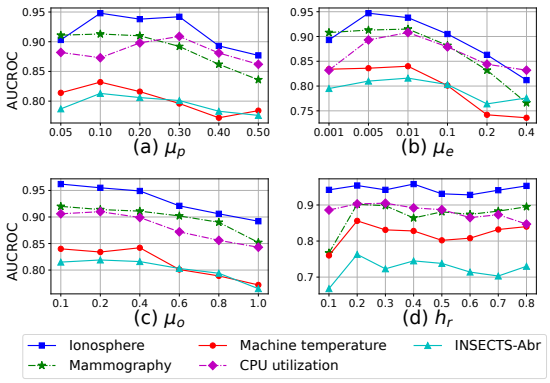

Figure 5: Sensitivity test of parameters μ_p, μ_e, μ_o, and ΔL.

4.6 Sensitivity Study

First, we focus on the three critical threshold hyperparameters: μ_p, μ_e, and μ_o. These hyperparameters control dynamic concept adaptation and the frequency of model updates. Adopting low thresholds for μ_p and μ_e may result in an excessive number of samples being allocated to the Dynamic Shift Detector (DSD), even when the current concept remains unchanged. Conversely, excessively high thresholds could hinder the detection of concept drift when it arises. Experimental results, as depicted in Figure 5, suggest that the optimal settings for μ_p lie between 0.1 and 0.2, and for μ_e between 0.005 and 0.01. In the absence of prior knowledge about a dataset, we recommend these default parameters as initial values, followed by a grid search to find better hyperparameters. As for μ_o, this hyperparameter denotes the number of samples that exceed the update threshold within the sliding window. To evaluate the effect of the offline updating strategy, we scrutinized the ratio of μ_o to the window size. Our results indicate that a smaller μ_o value improves performance. However, a small μ_o can lead to frequent offline updates, and thus reduce model efficiency. Therefore, a balance between performance and efficiency can be attained by setting μ_o between 0.1 and 0.4, as shown in Figure 5.

Next, we evaluate the historical data ratio h_r within the range of 0.1 to 0.8. One key observation is that when h_r is too small, the performance notably declines due to the inadequacy of historical data. This inadequacy results in a failure to acquire adequately informative central concepts. As the ratio increases, there is a noticeable improvement in performance. However, a higher ratio does not consistently guarantee better performance. For instance, in the CPU and INSECTS-Abr datasets, increasing the ratio actually leads to a decline in performance. This observation suggests that the model might become more susceptible to overfitting the training data, thus causing a reduction in performance during inference. While the targeted OUS is more efficient and performs better than training models directly on more data.

Furthermore, we evaluate the impact of the window size ΔL on both effectiveness and efficiency. This parameter is central for OUS. The results in Table 8 show that an excessively small ΔL results in frequent model updates, which can somewhat ensure a certain level of AUCROC performance but at the expense of efficiency. Conversely, an overly large ΔL reduces inference time but compromises the model's capacity to timely capture concept changes, resulting in performance degradation. Fortunately, as discussed in Section 4.5, the update time of METER is minimal, and it maintains commendable performance across a wide range of ΔL. This allows users to tailor ΔL to their specific requirements in practical applications.

4.7 Interpretability

To provide a clearer understanding of the uncertainty modeled by IEC, we offer an interpretation from a semantic perspective. High uncertainty in anomaly detection often arises when the model confronts unfamiliar data concepts. Figure 6 serves as an illustrative example of how METER generates interpretable results for different time steps and concepts, and how it accurately identifies and rapidly adapts to concept drift based on probability distribution and concept uncertainty as observed on the real-world dataset INSECTS. We identify three representative data points from two different concepts and their concept drift point for illustration. For the first data point, METER generates a probability distribution characterized by small entropy and low uncertainty, enabling the model to correctly classify it as a normal sample. For the second data point, METER produces a probability distribution with high concept uncertainty, indicating the presence of concept drift. In this case, METER transitions into the dynamic mode via the IEC, and the DSD learns the parameter drift of the base detection model and accurately classifies it as a normal sample. By examining the output of the third point, we can notice that the model effectively adapts to the new concept, indicated by the prediction with low concept uncertainty. These findings validate the capability of METER to provide more interpretable and trustworthy detection for better user understanding.

4.8 Integration on Flink

Stream processing engines play a pivotal role in deploying OAD frameworks. To illustrate how METER can function as a component of a larger-scale system, we integrate METER into Apache Flink [14], a framework and distributed processing engine for stateful computations over unbounded and bounded data streams. Flink

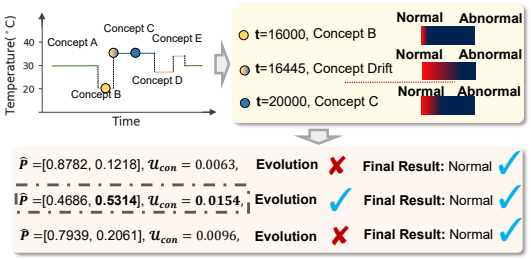

Figure 6: Interpretability with respect to model evolution.

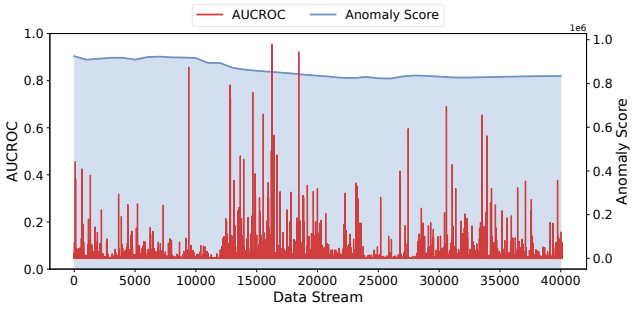

Figure 7: Timely output results of METER on Flink.

wraps METER inside a Flink operator, where Flink helps to establish the required environment, manage resources, read/write the data with versatile connectors, and handle failures. Building streaming workloads upon Flink empowers METER with real-time data processing pipelines, large-scale exploratory data analysis, and ETL processes. The experiments are conducted on the INSECT-Abr dataset. The timely output results, as illustrated in Figure 7, validate a seamless integration of METER with a stream processing engine, confirming its efficacy in supporting real-time anomaly detection.

5 RELATED WORK

Anomaly Detection. Anomaly detection (AD) has been extensively studied in various fields such as computer networks [55, 76], intrusion detection [22, 34], healthcare [65, 83], and finance [32]. Traditional approaches for AD include statistical methods [12], clustering algorithms [5, 27], classification-based techniques [47], and nearest neighbor-based methods [10, 78]. With the advent of deep learning, autoencoder-based techniques have become popular for AD [4]. Autoencoders learn to reconstruct input data by minimizing the reconstruction error, where a higher error is an indication of an anomaly. [67] proposes a stacked autoencoder for detecting anomalies in credit card transactions. The method achieved good performance but suffered from high false positive rates. To address this issue, [51] introduces a sparse autoencoder, which achieves better performance in terms of false positive rates. There are also some extensions for traditional autoencoders on AD. ADAE [82] proposes an Adversarial Dual Autoencoder framework for AD, which uses two autoencoders in a dual structure to improve the representation power of the model. [45] presents a smoothness-inducing sequential variational auto-encoder (VAE) model for the robust estimation and AD of multidimensional time series. However, the above approaches have not considered processing data in a streaming fashion and usually require large amounts of training data in an offline setting, and thus cannot be applied to online AD.

Online Anomaly Detection. Online anomaly detection (OAD) aims to promptly identify abnormal behavior or events in real-time data streams. Notably, compared to AD, OAD faces the challenge of concept drift. Previous works [9, 15, 33] rely on sliding windows and primarily focus on detecting alterations in specific statistical characteristics (e.g., mean values) of streaming data or its individual features to identify concept drift. However, these approaches prove insufficient for OAD, since concept drift within OAD can encompass more intricate changes in data distribution, including shifts, correlations, and transformations of data patterns, extending beyond the scope of basic statistical features like mean values.

Different anomaly detection approaches span various paradigms, such as isolation forest [47] (IF), k nearest neighbors (kNN), local outlier factor [12] (LOF), and deep neural networks (DNNs). These approaches have been extensively adapted for OAD tasks, leading to the rise of two prevailing OAD approaches: incremental learning-based [7, 8, 11, 26, 54] and ensemble-based approaches [21, 25, 52, 53, 62, 69, 75, 87]. However, these methods suffer from limitations in their ability to adapt to evolving data streams, since they need to update their detection models to accommodate the changing concepts, typically by retraining, fine-tuning, or updating certain model statuses. Moreover, the effectiveness of these techniques is inherently bounded by their quantity, such as the number of pre-trained models in ensemble-based methods, which necessitates a trade-off between efficiency and performance. Our work aims to address these challenges by proposing a novel framework that dynamically adapts to concept drift without the need for additional fine-tuning or retraining, which guarantees both effectiveness and efficiency while providing interpretable decision-making.

6 CONCLUSIONS

In this paper, we present a novel framework METER for online anomaly detection (OAD), which addresses the challenge of concept drift in an effective, efficient, and interpretable manner. By leveraging a static concept-aware detector trained on historical data, METER captures and handles recurring central concepts, while dynamically adapting to new concepts in evolving data streams using a lightweight drift detection controller and a hypernetwork-based parameter shift technique. The evidential deep learning-based drift detection controller enables efficient and interpretable concept drift detection. Our experimental study demonstrates that METER outperforms existing OAD approaches in various scenarios and facilitates valuable interpretability.

7 ACKNOWLEDGMENTS

This work is supported by the National Research Foundation, Singapore under its Emerging Areas Research Projects (EARP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. The work is also supported by the National Natural Science Foundation of China National Science Fund for Distinguished Young Scholars 62025301, and the National Natural Science Foundation of China under Grant 61933002.

REFERENCES

[1] 1999. KDD Cup Dataset. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed:2023-07.

[2] Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. 2021. Practical approach to asynchronous multivariate time series anomaly detection and localization. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2485–2494.

[3] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. 2017. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing* 262 (2017), 134–147.

[4] Jinwon An and Sungzoon Cho. 2015. Variational autoencoder based anomaly detection using reconstruction probability. *Special lecture on IE* 2, 1 (2015), 1–18.

[5] Fabrizio Angiulli and Fabio Fassetti. 2007. Detecting distance-based outliers in streams of data. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. 811–820.

[6] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. 2020. Usad: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3395–3404.

[7] Siddharth Bhatia, Arjit Jain, Pan Li, Ritesh Kumar, and Bryan Hooi. 2021. Mstream: Fast anomaly detection in multi-aspect streams. In *Proceedings of the Web Conference 2021*. 3371–3382.

[8] Siddharth Bhatia, Arjit Jain, Shivin Srivastava, Kenji Kawaguchi, and Bryan Hooi. 2022. MemStream: Memory-Based Streaming Anomaly Detection. In *Proceedings of the ACM Web Conference 2022*. 610–621.

[9] Albert Bifet and Ricard Gavalda. 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 443–448.

[10] Paul Boniol, Michele Linardi, Federico Roncallo, Themis Palpanas, Mohammed Meftah, and Emmanuel Remy. 2021. Unsupervised and scalable subsequence anomaly detection in large data series. *The VLDB Journal* (2021), 1–23.

[11] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J Franklin. 2021. SAND: streaming subsequence anomaly detection. *Proceedings of the VLDB Endowment* 14, 10 (2021), 1717–1729.

[12] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 93–104.

[13] Shaofeng Cai, Kaiping Zheng, Gang Chen, HV Jagadish, Beng Chin Ooi, and Meihui Zhang. 2021. Arm-Net: Adaptive relation modeling network for structured data. In *Proceedings of the 2021 International Conference on Management of Data*. 207–220.

[14] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2015. Apache flink: Stream and batch processing in a single engine. *The Bulletin of the Technical Committee on Data Engineering* 38, 4 (2015).

[15] Rodolfo C Cavalcante, Leandro L Minku, and Adriano LI Oliveira. 2016. Fedd: Feature extraction for explicit concept drift detection in time series. In *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 740–747.

[16] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 1–58.

[17] Sudarshan S Chawathe and Hector Garcia-Molina. 1997. Meaningful change detection in structured data. *ACM SIGMOD Record* 26, 2 (1997), 26–37.

[18] Zhuangbin Chen, Jinyang Liu, Yuxin Su, Hongyu Zhang, Xiao Ling, Yongqiang Yang, and Michael R Lyu. 2022. Adaptive performance anomaly detection for online service systems via pattern sketching. In *Proceedings of the 44th International Conference on Software Engineering*. 61–72.

[19] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. 2021. The UCR Time Series Classification Archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/UCR_TimeSeriesAnomalyDatasets2021.zip. Accessed:2023-07.

[20] Hanqiu Deng and Xingyu Li. 2022. Anomaly detection via reverse distillation from one-class embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9737–9746.

[21] Zhiguo Ding and Minrui Fei. 2013. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes* 46, 20 (2013), 12–17.

[22] Filipe Falcão, Tommaso Zoppi, Caio Barbosa Viera Silva, Anderson Santos, Baldoino Fonseca, Andrea Ceccarelli, and Andrea Bondavalli. 2019. Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 318–327.

[23] Zheyao Gao, Yuanye Liu, Fuping Wu, NanNan Shi, Yuxin Shi, and Xiahai Zhuang. 2023. A Reliable and Interpretable Framework of Multi-view Learning for Liver Fibrosis Staging. *arXiv preprint arXiv:2306.12054* (2023).

[24] Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. 2019. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1705–1714.

[25] Parikshit Gopalan, Vatsal Sharan, and Udi Wieder. 2019. Pidforest: anomaly detection via partial identification. *Advances in Neural Information Processing Systems* 32 (2019).

[26] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. 2016. Robust random cut forest based anomaly detection on streams. In *International conference on machine learning*. PMLR, 2712–2721.

[27] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. 2003. KNN model-based approach in classification. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*. Springer, 986–996.

[28] David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106* (2016).

[29] Zongbo Han, Changqing Zhang, Huazhu Fu, and Joey Tianyi Zhou. 2022. Trusted multi-view classification with dynamic evidential fusion. *IEEE transactions on pattern analysis and machine intelligence* 45, 2 (2022), 2551–2566.

[30] Matan Haroush, Tzivel Frostig, Ruth Heller, and Daniel Soudry. 2021. Statistical testing for efficient out of distribution detection in deep neural networks. *arXiv preprint arXiv:2102.12967* (2021).

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[32] Waleed Hilal, S Andrew Gadsden, and John Yawney. 2022. Financial Fraud:: A Review of Anomaly Detection Techniques and Recent Advances. (2022).

[33] David Tse Jung Huang, Yun Sing Koh, Gillian Dobbie, and Russel Pears. 2014. Detecting volatility shift in data streams. In *2014 IEEE International Conference on Data Mining*. IEEE, 863–868.

[34] Hao Huang and Shiva Prasad Kasiviswanathan. 2015. Streaming anomaly detection using randomized matrix sketching. *Proceedings of the VLDB Endowment* 9, 3 (2015), 192–203.

[35] Peng Jia, Shaofeng Cai, Beng Chin Ooi, Pinghui Wang, and Yiyuan Xiong. 2023. Robust and Transferable Log-based Anomaly Detection. 1, 1 (2023), 64:1–64:26.

[36] Audun Jsang. 2018. *Subjective Logic: A formalism for reasoning under uncertainty*. Springer Publishing Company, Incorporated.

[37] Tung Kieu, Bin Yang, Chenjuan Guo, and Christian S Jensen. 2019. Outlier Detection for Time Series with Recurrent Autoencoder Ensembles.. In *IJCAI*. 2725–2732.

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

[39] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR*.

[40] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[41] Frank Klinker. 2011. Exponential moving average versus moving exponential average. *Mathematische Semesterberichte* 58 (2011), 97–107.

[42] Marius Kloft and Pavel Laskov. 2010. Online anomaly detection under adversarial impact. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 405–412.

[43] Marius Kloft and Pavel Laskov. 2012. Security analysis of online centroid anomaly detection. *The Journal of Machine Learning Research* 13, 1 (2012), 3681–3724.

[44] Chieh-Hsin Lai, Dongmian Zou, and Gilad Lerman. 2019. Robust subspace recovery layer for unsupervised anomaly detection. *arXiv preprint arXiv:1904.00152* (2019).

[45] Longyuan Li, Junchi Yan, Haiyang Wang, and Yaohui Jin. 2020. Anomaly detection of time series with smoothness-inducing sequential variational auto-encoder. *IEEE transactions on neural networks and learning systems* 32, 3 (2020), 1177–1191.

[46] Sainan Li, Qilei Yin, Guoliang Li, Qi Li, Zhuotao Liu, and Jinwei Zhu. 2022. Unsupervised Contextual Anomaly Detection for Database Systems. In *Proceedings of the 2022 International Conference on Management of Data*. 788–802.

[47] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 413–422.

[48] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering* 31, 12 (2018), 2346–2363.

[49] Yue Lu, Renjie Wu, Abdullah Mueen, Maria A Zuluaga, and Eamonn Keogh. 2022. Matrix profile XXIV: scaling time series anomaly detection to trillions of datapoints and ultra-fast arriving data streams. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1173–1182.

[50] Zhaojing Luo, Shaofeng Cai, Yatong Wang, and Beng Chin Ooi. 2023. Regularized Pairwise Relationship based Analytics for Structured Data. *Proceedings of the 2023 ACM SIGMOD International Conference on Management of Data* 1, 1 (2023), 82:1–82:27.

[51] Alireza Makhzani and Brendan J Frey. 2015. Winner-take-all autoencoders. *Advances in neural information processing systems* 28 (2015).

[52] Emaad Manzoor, Hemank Lamba, and Leman Akoglu. 2018. xstream: Outlier detection in feature-evolving data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1963–1972.

[53] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. 2018. Kitsune: an ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089* (2018).

[54] Gyoung S Na, Donghyun Kim, and Hwanjo Yu. 2018. Dilof: Effective and memory efficient local outlier detection in data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1993–2002.

[55] Mukrimah Nawir, Amiza Amir, Naimah Yaakob, and Ong Bi Lynn. 2019. Effective and efficient network anomaly detection system using machine learning algorithm. *Bulletin of Electrical Engineering and Informatics* 8, 1 (2019), 46–51.

[56] Kai Wang Ng, Guo-Liang Tian, and Man-Lai Tang. 2011. Dirichlet and related distributions: Theory, methods and applications. (2011).

[57] Adam Oliner and Jon Stearley. 2007. What Supercomputers Say: A Study of Five System Logs. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*. 575–584. https://doi.org/10.1109/DSN.2007.103

[58] Beng Chin Ooi, Kian-Lee Tan, Sheng Wang, Wei Wang, Qingchao Cai, Gang Chen, Jinyang Gao, Zhaojing Luo, Anthony K. H. Tung, Yuan Wang, Zhongle Xie, Meihui Zhang, and Kaiping Zheng. 2015. SINGA: A Distributed Deep Learning Platform. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference, MM*. ACM, 685–688.

[59] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. 2021. Deep learning for anomaly detection: A review. *ACM computing surveys (CSUR)* 54, 2 (2021), 1–38.

[60] John Paparrizos, Yuhao Kang, Paul Boniol, Ruey S Tsay, Themis Palpanas, and Michael J Franklin. 2022. TSB-UAD: an end-to-end benchmark suite for univariate time-series anomaly detection. *Proceedings of the VLDB Endowment* 15, 8 (2022), 1697–1711.

[61] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[62] Tomáš Pevnỳ. 2016. Loda: Lightweight on-line detector of anomalies. *Machine Learning* 102 (2016), 275–304.

[63] Shebuti Rayana. 2016. ODDS Library. https://odds.cs.stonybrook.edu. Accessed:2023-07.

[64] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. 2021. A unifying review of deep and shallow anomaly detection. *Proc. IEEE* 109, 5 (2021), 756–795.

[65] Edin Šabić, David Keeley, Bailey Henderson, and Sara Nannemann. 2021. Healthcare and anomaly detection: using machine learning to predict anomalies in heart rate data. *AI & SOCIETY* 36, 1 (2021), 149–158.

[66] Mohammad Sabokrou, Mahmood Fathy, and Mojtaba Hoseini. 2016. Video anomaly detection and localisation based on the sparsity and reconstruction error of auto-encoder. *Electronics Letters* 52, 13 (2016), 1122–1124.

[67] Mayu Sakurada and Takehisa Yairi. 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*. 4–11.

[68] Mahsa Salehi, Christopher Leckie, James C Bezdek, Tharshan Vaithianathan, and Xuyun Zhang. 2016. Fast memory efficient local outlier detection in data streams. *IEEE Transactions on Knowledge and Data Engineering* 28, 12 (2016), 3246–3260.

[69] Saket Sathe and Charu C Aggarwal. 2016. Subspace outlier detection in linear time with randomized hashing. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 459–468.

[70] David Savage, Xiuzhen Zhang, Xinghuo Yu, Pauline Chou, and Qingmai Wang. 2014. Anomaly detection in online social networks. *Social networks* 39 (2014), 62–70.

[71] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. 2022. Anomaly detection in time series: a comprehensive evaluation. *Proceedings of the VLDB Endowment* 15, 9 (2022), 1779–1797.

[72] Murat Sensoy, Lance Kaplan, and Melih Kandemir. 2018. Evidential deep learning to quantify classification uncertainty. *Advances in neural information processing systems* 31 (2018).

[73] Ava P Soleimany, Alexander Amini, Samuel Goldman, Daniela Rus, Sangeeta N Bhatia, and Connor W Coley. 2021. Evidential deep learning for guided molecular property prediction and discovery. *ACS central science* 7, 8 (2021), 1356–1367.

[74] Vinicius MA Souza, Denis M dos Reis, Andre G Maletzke, and Gustavo EAPA Batista. 2020. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery* 34 (2020), 1805–1858.

[75] Swee Chuan Tan, Kai Ming Ting, and Tony Fei Liu. 2011. Fast anomaly detection for streaming data. In *Twenty-second international joint conference on artificial intelligence*. Citeseer.

[76] Alexander G Tartakovsky, Aleksey S Polunchenko, and Grigory Sokolov. 2012. Efficient computer network anomaly detection by changepoint detection methods. *IEEE Journal of Selected Topics in Signal Processing* 7, 1 (2012), 4–11.

[77] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 1–6.

[78] Jing Tian, Michael H Azarian, and Michael Pecht. 2014. Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm. In *PHM society European conference*, Vol. 2.

[79] Theodoros Toliopoulos, Christos Bellas, Anastasios Gounaris, and Apostolos Papadopoulos. 2020. PROUD: parallel outlier detection for streams. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2717–2720.

[80] Luan Tran, Liyue Fan, and Cyrus Shahabi. 2016. Distance-based outlier detection in data streams. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1089–1100.

[81] Luan Tran, Min Y Mun, and Cyrus Shahabi. 2020. Real-time distance-based outlier detection in data streams. *Proceedings of the VLDB Endowment* 14, 2 (2020), 141–153.

[82] Ha Son Vu, Daisuke Ueta, Kiyoshi Hashimoto, Kazuki Maeno, Sugiri Pranata, and Sheng Mei Shen. 2019. Anomaly detection with adversarial dual autoencoders. *arXiv preprint arXiv:1902.06924* (2019).

[83] Ziyu Wang, Nanqing Luo, and Pan Zhou. 2020. GuardHealth: Blockchain empowered secure data management and Graph Convolutional Network enabled anomaly detection in smart healthcare. *J. Parallel and Distrib. Comput.* 142 (2020), 1–12.

[84] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. 2019. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition* 90 (2019), 119–133.

[85] Selim F Yilmaz and Suleyman S Kozat. 2020. Pysad: A streaming anomaly detection framework in python. *arXiv preprint arXiv:2009.02572* (2020).

[86] Susik Yoon, Jae-Gil Lee, and Byung Suk Lee. 2020. Ultrafast local outlier detection from a data stream with stationary region skipping. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1181–1191.

[87] Susik Yoon, Youngjun Lee, Jae-Gil Lee, and Byung Suk Lee. 2022. Adaptive Model Pooling for Online Deep Anomaly Detection from a Complex Evolving Data Stream. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2347–2357.

[88] Susik Yoon, Yooju Shin, Jae-Gil Lee, and Byung Suk Lee. 2021. Multiple dynamic outlier-detection from a data stream by exploiting duality of data and queries. In *Proceedings of the 2021 International Conference on Management of Data*. 2063–2075.

[89] Houssam Zenati, Manon Romain, Chuan-Sheng Foo, Bruno Lecouat, and Vijay Chandrasekhar. 2018. Adversarially learned anomaly detection. In *2018 IEEE International conference on data mining (ICDM)*. IEEE, 727–736.

[90] Nengwen Zhao, Junjie Chen, Zhaoyang Yu, Honglin Wang, Jiesong Li, Bin Qiu, Hongyu Xu, Wenchi Zhang, Kaixin Sui, and Dan Pei. 2021. Identifying bad software changes via multimodal anomaly detection for online service systems. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 527–539.

[91] Nengwen Zhao, Honglin Wang, Zeyan Li, Xiao Peng, Gang Wang, Zhu Pan, Yong Wu, Zhen Feng, Xidao Wen, Wenchi Zhang, et al. 2021. An empirical investigation of practical log anomaly detection for online service systems. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1404–1415.

[92] Yue Zhao, Zain Nasrullah, and Zheng Li. 2019. Pyod: A python toolbox for scalable outlier detection. *arXiv preprint arXiv:1901.01588* (2019).

[93] Kaiping Zheng, Shaofeng Cai, Horng Ruey Chua, Wei Wang, Kee Yuan Ngiam, and Beng Chin Ooi. 2020. Tracer: A framework for facilitating accurate and interpretable analytics for high stakes applications. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1747–1763.

[94] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*.