Detecting Anomalies with Autoencoders on Data Streams

Lucas Cazzonelli ✉[0000−0003−2886−1219] and
Cedric Kulbach[0000−0002−9363−4728]

FZI Research Center for Information Technology,
Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany
{cazzonelli,kulbach}@fzi.de

Abstract. Autoencoders have achieved impressive results in anomaly detection tasks by identifying anomalous data as instances that do not match their learned representation of normality. To this end, autoencoders are typically trained on large amounts of previously collected data before being deployed. However, in an online learning scenario, where a predictor has to operate on an evolving data stream and therefore continuously adapt to new instances, this approach is inadequate. Despite their success in offline anomaly detection, there has been little research leveraging autoencoders as anomaly detectors in such a setting. Therefore, in this work, we propose an approach for online anomaly detection with autoencoders and demonstrate its competitiveness against established online anomaly detection algorithms on multiple real-world datasets. We further address the issue of autoencoders gradually adapting to anomalies and thereby reducing their sensitivity to such data by introducing a simple modification to the models' training approach. Our experimental results indicate that our solution achieves a larger gap between the losses on anomalous and normal instances than a conventional training procedure.

Keywords: Anomaly Detection · Autoencoders · Data Streams · Unsupervised Learning

1 Introduction

*Autoencoder*s (*AE*s) were found to enable the unsupervised learning of useful non-linear data representations. Consequently, they sparked significant progress in many areas of machine learning, including the area of unsupervised anomaly detection that deals with the identification of unusual events.

For this purpose, *AE*s are typically trained on large amounts of normal, non-anomalous data instances, reducing the networks' reconstruction error for this kind of data. Due to the absence of anomalies in the training data as well as the constraints imposed on *AE*s, losses for anomalies remain at a higher level and can therefore be distinguished from losses of normal instances. However, collecting and fitting large amounts of data can be an obstacle in many real-world applications. Data might for example not be available from the get-go but rather arrive little by little in small chunks or even in continuous streams of individual samples. In many cases, a model may also be subject to distributional changes over time in the form of concept drift [27]. This is especially true for anomaly detection, as the definition of what constitutes an anomaly is not definitive [31] and may for instance depend heavily on the temporal context in which the data appears. As a result, incremental model updates can often be required to adapt the model to such variations after the initial training.

Based on this observation, an anomaly detector operating on a continuous and potentially infinite stream of individual samples should fulfill the requirements proposed by [4] to

R1: be able to process a single instance at a time,
R2: be able to process each instance in a limited amount of time,
R3: use a limited amount of memory,
R4: be ready to predict at any time,
R5: be able to adapt to changes in the data distribution.

To establish a unified view, we additionally formalize the task of online *Anomaly Detection* (AD) in Section 2.

As feed-forward neural networks, AEs inherently feature a fixed time (**R2**) complexity and, due to their fixed structure, a constant memory (**R3**) consumption with respect to the number of processed examples. Since they are conventionally optimized with gradient-based methods, AEs also allow processing individual examples (**R1**) to adapt to the most recent data instance (**R5**) and predicting at any time (**R4**). Even their tendency to forget previously learned data patterns [19] when being exposed to new tasks does not necessarily inhibit the performance of AE for AD. This can even be beneficial that forgetting old concepts or distributions of normality may often even be desired, as the sudden reappearance of data that would have previously been considered normal could be deemed as abnormal.

In conclusion, AEs seem well suited for the task of online AD. Nevertheless, only a handful of studies attempted to leverage the remarkable representation learning capabilities of AEs to detect anomalies in data streams. Rather, previous contributions focused on adapting conventional offline learning AD algorithms to fulfil the requirements of online learning (see e.g. [15, 24, 26]). In this work, we present an AE-based anomaly detector for the above requirements and demonstrate its ability to outperform previous state-of-the-art anomaly detectors on multiple real-world datasets. Further, we address the problem of contaminated training data as studied by [5]. Unlike in traditional offline learning, the training data of an online AD will inevitably be contaminated with anomalous instances, since the ability to remove such data from the data stream would make the model redundant in the first place. Thus, depending on the proportion and arrival of anomalies and other external circumstances, reconstruction losses for abnormal data can be expected to decrease to a greater extent than when training on non-contaminated data, degrading the model's detection accuracy [32]. To mitigate this effect on our approach, we propose an improved optimization objective for AE-based AD, that leads to increases in discrimination performance. To the best of our knowledge, we are the first to address the issue of training data contamination in the context of AE-based online AD.

To enable to enable the usage of our results in future applications, we implemented our approach as an extension [1] to the online learning framework River [17] [2]. In conclusion, we provide the following contributions:

C1: A formalization of streaming AD.
C2: A competitive approach for AE-based anomaly detection on data streams.
C3: An alternative optimization technique to improve the performance of AEs anomaly detectors under the influence of anomalous training examples.
C4: An in-depth evaluation of our approaches with established real-world datasets.
C5: A Python package [1] extending River [17] that facilitates the reproducibility and reuse of our work.

2 Problem Statement

When using an offline learning approach AEs infringe most of the defined requirements. Predictions for new data instances have to be calculated immediately at the time of their arrival to conform with **R1**. The requirements **R4** and **R5** pose the problem that online AD models also need to continuously adapt to their stream of inputs. In addition, data streams often have high frequency, requiring efficient resource utilization, as required by **R2 - R4**. Therefore, the separation of training and testing stages as performed in most offline scenarios is not applicable when evaluating data streams. Accordingly, we define the problem of anomaly detection as follows:

Definition 1. *Incremental Anomaly Detection*
Let $\mathcal{S}^+ = \{(\boldsymbol{x}^{(i)}, y^{(i)}) \,|\, \forall i \in \mathbb{N}\}$ *be a potentially infinite sequence of tuples each consisting of a feature-vector* $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$, $d \in \mathbb{N}$ *and a corresponding anomaly label* $y^{(i)} \in \{0,1\}$. *Further, let* $\mathcal{S} = \{(\boldsymbol{x}^{(i)}, y^{(i)}) \,|\, \forall i \in \{1,\dots,I\}\}$ *be a sub-sequence of previously observed instances. Let* $\mathcal{A} = \{\boldsymbol{x}^{(i)} \,|\, y^{(i)} = 1\, i \in \{1,\dots,I\}\}$ *be the set of anomalies in* $\mathcal{S}$, *where* $\frac{|\mathcal{A}|}{I} \ll 0.5$.
Also, let $\mathcal{S}_{train}$ *and* $\mathcal{S}_{valid}$ *be disjoint subsets of* $\mathcal{S}$. *Given a validation protocol* $\mathcal{V}$ *and an arbitrary anomaly detection function* $g : \mathbb{R}^d \to \{0,1\}$ *that was trained on* $\mathcal{S}_{train}$, *the objective of incremental anomaly detection can then be defined as*

$$\min_g \mathcal{V}(\mathcal{L}_c, \mathcal{S}_{valid}, g(\,\cdot\,; \mathcal{S}_{train})), \tag{1}$$

where $\mathcal{L}_c : \{0,1\} \times \mathbb{R} \to \mathbb{R}$ *denotes a classification loss that quantifies the dissimilarity of the true label* $y^{(i)}$ *and the prediction* $\hat{y}^{(i)} = g(\boldsymbol{x}^{(i)})$.

[1] Available at https://github.com/lucasczz/DAADS
[2] Available at https://github.com/online-ml/river

As a validation protocol $\mathcal{V}$ we employ a *test-then-train* [4] protocol that is sensitive to *concept drifts*. In a *test-then-train* evaluation, models are first validated based on a performance metric $\mathcal{L}_c$ and then trained on each instance in $\mathcal{S}$ at arrival-time. Although our definition of incremental AD includes supervised approaches, we limit our analysis to unsupervised AD, where only the feature instances $\boldsymbol{x}^{(i)}$ are available for training.

3 Related Work

In this section, we depict the work related to our approach by presenting online learning approaches for the AD task as well as offline learning AD approaches that are based on AE. Finally, we summarize previous work on online learning and AE-based AD.

3.1 Offline Anomaly Detection

Anomaly Detection is a very active area of research that has produced a variety of different approaches for identifying anomalous data examples. Early studies focused on statistical measures in this context (see [21]). Machine learning approaches have relied on the assumption that anomalies are found in low-density areas. Highly successful examples of such approaches include *One Class Support Vector Machines* (*OC-SVM*s) [25], which learn a hyperplane that separates high-density normal data from low-density anomalous data, and *Local Outlier Factor* (*LOF*) [6], which scores data points based on the relative densities of their neighborhoods. Rather than directly assessing differences in density, *Isolation Forests* [25] separate anomalies with an ensemble of trees that iteratively splits the data along random attribute thresholds. Since the algorithm assumes that anomalies are easily separable from the rest of the data, they can be identified as instances isolated by a small number of splits. According to [14], *Isolation Forests* provide better scalability compared to previous techniques such as *OC-SVM*s and *LOF*.

In more recent years, significant progress in the area of AD has been achieved with deep learning models. Some of the most successful model types in this regard are AEs. Inspired by the functioning principle of PCA-based AD , Sakurada and Yairi [23] proposed to exploit the limited generalization of AE feature mappings, which lead to higher reconstruction losses when facing previously unobserved data patterns. Numerous studies subsequently improved the original AD algorithm's performance by introducing advanced model variations. Zhou and Paffenroth [29] for instance introduced a robust AE that iteratively separates noise from the underlying clean data whereas Gong et al. [11] mapped the AE's representations to a fixed number of memorized vectors to avoid learning representations that generalize to anomalous data.

3.2 Online Anomaly Detection

Despite the success of *AE*s in offline learning, only a few studies investigated their usage as online anomaly detectors for data streams.

Mirsky et al. [16] adapted the concept of *AE* ensembles [7] to the task of online learning and added a secondary downstream *AE* which reconstructs the loss values of the ensemble, forming the *Kit-Net* model. To assign data features to individual *AE*s in the ensemble, *Kit-Net* requires initial training data.

In the realm of online *AD* previous research has been focused on adaptations of well-established conventional machine learning techniques. Particularly popular in this context are streaming variants of *LOF* and *IsolationForest*. Based on *Isolation Forests*, Tan et al. [26] for instance proposed the *Half Space Tree* (*HST*) *AD* approach that builds an ensemble of trees by iteratively halving random attribute subspaces of an initial data subset. Other tree-based methods include *RS-Forests* [28] and *Robust Random Cut Forest* (*RRCF*) [12]. The *xStream* anomaly detector introduced by [15] elaborates on the concept of iterative subspace cuts by applying random matrix projections before dividing examples based on randomly selected features that were created in the process. *LOF*-based streaming *AD* techniques include *Incremental Local Outlier Factor* (*iLOF*) [20] and its advancements, *MiLOF* [24] and *DILOF* [18], both of which aim to overcome the loss of density data that would result from simply removing the oldest instances to maintain fixed memory and run time usage.

There are several studies on training *AE*s and other neural networks in a more general online learning setting [1, 22, 30]. However, none have investigated the online learning of *AE*s in the specific context of *AD*, which differs considerably from most conventional scenarios, since the often targeted minimization of the reconstruction loss may even be disadvantageous in the case of *AD* if losses for anomalies are affected by it.

To overcome the high sensitivity of neural networks to the learning rate, Baydin et al. [3] proposed *Hypergradient Descent* (*HD*) optimization algorithms, which simultaneously optimize network parameters as well as the learning rate using gradient descent. We evaluate the *HD* modification of *Stochastic Gradient Descent* (*SGD*) for our approach in Section 6.

4 Streaming Anomaly Detection with Autoencoders

According to Definition 1, we define an online *AD* system as depicted in Section 4. We define an anomaly detector as $g = \tau \circ f$, where $f : \mathbb{R}^d \to \mathbb{R}$ is an anomaly scoring function that judges the abnormality of its input in the form of an anomaly score $z^{(i)}$, and $\tau : \mathbb{R} \to \{0, 1\}$ is a *thresholding* function that decides whether to label the current input as an anomaly given $z^{(i)}$. Since the choice of an appropriate threshold mostly depends on the characteristics of the particular application, e.g., the relative cost of false-positive versus false-negative classifications, we omit the thresholding problem and focus on the computation of anomaly scores in this work.

While arbitrary *AD* models can implement the scoring function f, we aim to improve upon the performance of existing techniques by introducing basic-*AE* and *Denoising AE* (*DAE*) online anomaly scorers as well as a *Probability Weighted AE* (*PW-AE*) which we specifically develop to address the issue of contamination. In the following, we briefly explain the functional concept of these models. In accordance with **R1**, we begin the score calculation for the latest sample with calling the reconstruction function $r_{\boldsymbol{\theta}}$ on the model input $\boldsymbol{x}^{(i)}$ by performing a forward pass through the *AE*, generating a reconstruction $\hat{\boldsymbol{x}}^{(i)}$. Subsequently, we calculate the reconstruction loss $l^{(i)} = \mathcal{L}(\boldsymbol{x}^{(i)}, \hat{\boldsymbol{x}}^{(i)})$. To accommodate for fluctuations of average reconstruction losses, we scale $l^{(i)}$ by applying the post-processing function π, yielding an anomaly score $z^{(i)}$. For simplicity, we scale the losses by the average of a sliding window μ_ω to obtain the anomaly score $z^{(i)}$. We further use the loss value $l^{(i)}$ to optimize the *AE*'s parameters $\boldsymbol{\theta}$ for every streaming instance by calculating a parameter update $\Delta^{(i)}\boldsymbol{\theta}$ with an optimization function *opt*, which we subsequently subtract from the current model parameters $\boldsymbol{\theta}$. For the optimization function *opt* any gradient-based optimization technique can be used, although we will subsequently assume standard *SGD* for the sake of simplicity. In Algorithm 1, we describe our approach towards online *AD* with the basic *AE* variant. For this type of model, we calculate the weight updates $\Delta^{(i)}\boldsymbol{\theta}$ according to the conventional *SGD* approach as

$$\Delta^{(i)}\boldsymbol{\theta} = \gamma \nabla_{\boldsymbol{\theta}}\, l^{(i)}, \tag{2}$$

where γ represents the learning rate and $\nabla_{\boldsymbol{\theta}}\, l^{(i)}$ the gradient of $l^{(i)}$ with respect to the model parameters $\boldsymbol{\theta}$. Apart from calculating a new reconstruction and training loss using dropout, we use the same update rule for the *DAE* as for the base variant. Through training with corrupted input data, *DAE*s were shown to

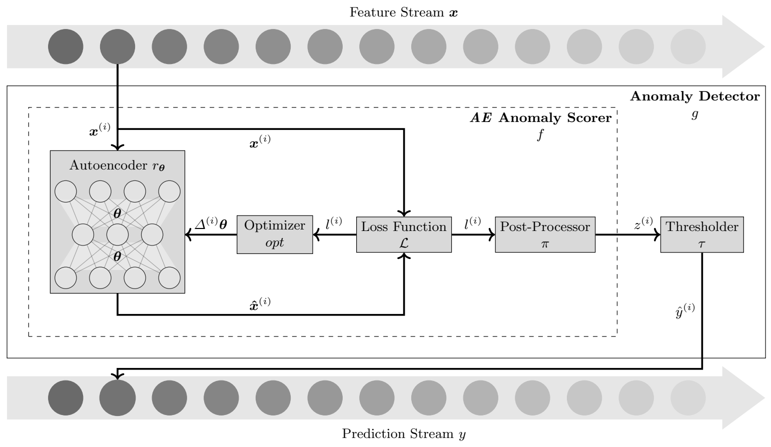

Fig. 1: Conceptual overview of proposed *AE*-anomaly scorer f within an incremental anomaly detector g.

Algorithm 1 Basic anomaly detection with *AE*

```
1: Input:
2: stream of inputs $X$, window size $w\_size$, learning rate $\gamma$,
3: thresholding function $\tau$, architecture $arch$
4: Output: stream of anomaly classifications $\hat{Y}$
5: $\boldsymbol{\theta} \leftarrow \text{init}(arch)$    ▷ Perform Glorot weight initialization [10]
6: $\omega \leftarrow \text{Window}(w\_size)$    ▷ Initialize sliding window
7: for all $\boldsymbol{x}^{(i)} \in X$ do    ▷ Start Stream $X$
8:     $l^{(i)} \leftarrow \mathcal{L}(\boldsymbol{x}^{(i)}, r_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))$    ▷ Calculate reconstruction loss
9:     $z^{(i)} \leftarrow \frac{l^{(i)}}{\mu_\omega}$    ▷ Apply post-processing
10:    $\hat{y}^{(i)} \leftarrow \tau(z^{(i)})$    ▷ Apply thresholding
11:    $\Delta^{(i)}\boldsymbol{\theta} \leftarrow opt(l^{(i)})$    ▷ Calculate parameter update
12:    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \Delta^{(i)}\boldsymbol{\theta}$    ▷ Apply parameter update
13:    $\omega.append(l^{(i)})$    ▷ Add loss to sliding window
14:    yield: $\hat{y}^{(i)}$
15: end for
```

be able to achieve more robust representations and thus better *AD* accuracy in previous studies [23].

From a probabilistic point of view, *SGD* optimizes the objective function

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\hat{p}_{\mathcal{S}}} \mathcal{L}(\boldsymbol{x}, r_{\boldsymbol{\theta}}(\boldsymbol{x})), \tag{3}$$

where $\boldsymbol{x}$ follows the empirical distribution defined by previously observed data $\hat{p}_{\mathcal{S}}$. Due to the fact, that Equation (3) rewards decreasing loss values for both normal and anomalous instances, minimizing the above objective does not directly correspond with maximizing the usefulness of an *AE*-model for the purpose of *AD*. Therefore, to establish a closer relationship between the training objective of the *AE* and its usefulness as an anomaly detector, we propose the alternative objective function

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\hat{p}_{\mathcal{S}}} \mathcal{L}(\boldsymbol{x}, r_{\boldsymbol{\theta}}(\boldsymbol{x}) \mid y=1) - \mathcal{L}(\boldsymbol{x}, r_{\boldsymbol{\theta}}(\boldsymbol{x}) \mid y=0), \tag{4}$$

that is equivalent to the maximization of the expected margin between the scores of anomalous- and normal data. Empirically, this objective estimates to

$$\begin{aligned} &\max_{\boldsymbol{\theta}} \frac{1}{I} \sum_{i=1}^{I} y^{(i)} \mathcal{L}(\boldsymbol{x}^{(i)}, r_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) - (1-y^{(i)}) \mathcal{L}(\boldsymbol{x}^{(i)}, r_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) \\ \Leftrightarrow &\min_{\boldsymbol{\theta}} \frac{1}{I} \sum_{i=1}^{I} (1-2y^{(i)}) \mathcal{L}(\boldsymbol{x}^{(i)}, r_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})). \end{aligned} \tag{5}$$

Most *AE*-based *AD* applications assume that training is performed on data, where the number of anomalous instances ($y^{(i)} = 1$) is close to zero. Under this circumstance Equation (5) approximates the standard *AE* optimization goal. However, this assumption is not necessarily valid for every phase of a data stream. Anomalies can, for example, concentrate in small time windows and therefore negatively affect the performance of a model trained to optimize the objective in Equation (3), as losses decrease for such instances. In the following, we derive an improved update rule from the objective defined in Equation (5). We refer to *DAE*s trained with this update rule as *PW-AE*s.

Since the true value of $y^{(i)}$ is unknown at the time of training, we substitute $y^{(i)}$ with an estimate of the probability $\hat{p}(y^{(i)} = 1)$, that we calculate by assuming $l^{(i)} \sim \mathcal{N}(\mu_\omega, \sigma_\omega)$, with μ_ω and σ_ω being the average and standard deviation of a sliding window ω of previous losses. Under these assumptions, we estimate

$$\hat{p}_{\mathcal{N}}(y^{(i)} = 1) = \Phi(\frac{l^{(i)} - \mu_\omega}{\sigma_\omega}), \tag{6}$$

where Φ denotes the cumulative standard-normal distribution function. Although reconstruction losses cannot be accurately represented by a normal distribution, they could improve the sensitivity of *AD* to anomalies. To incorporate the prior knowledge that on average $p(y^{(i)} = 1) \ll 0.5$ (see Definition 1), we calculate the final probability estimate $\hat{p}(y^{(i)} = 1)$ as

$$\hat{p}(y^{(i)} = 1) = \hat{p}_{\mathcal{N}}(y^{(i)} = 1) * \frac{1}{2p_0}, \tag{7}$$

where p_0 is a hyperparameter that determines the value of $\hat{p}_{\mathcal{N}}(y^{(i)} = 1)$ for which the weight update $\Delta^{(i)}$ assumes a value of 0. Using this probability estimate, the weight update for optimizing the objective in Equation (5) is given by

$$\begin{aligned} \Delta^{(i)}\boldsymbol{\theta} &= (1 - 2\hat{p}_{\mathcal{N}}(y^{(i)} = 1) * \frac{1}{2p_0})\gamma\nabla_{\boldsymbol{\theta}} l^{(i)} \\ &= (1 - \frac{\hat{p}_{\mathcal{N}}(y^{(i)} = 1)}{p_0})\gamma\nabla_{\boldsymbol{\theta}} l^{(i)} \end{aligned} \tag{8}$$

5 Experiments

We evaluated our approaches (*AE*, *DAE*, and *PW-AE*) based on 3 commonly used real-world datasets in comparison with state-of-the-art approaches for online *AD* in the following experiments:

Performance We compare all approaches based on the area-under-the-curve measures of their *Receiver Operating Characteristic* (*ROC-AUC*) and *Precision Recall* (*PR-AUC*) curves as well as their runtimes.

Contamination Robustness To evaluate the robustness against the degree of contamination, we present performance measures for different proportions of anomalies.

Parameters Finally, we present an evaluation based on different design choices for the latent space, the optimizer, and the learning rate of *AD*.

In the following, we describe the used datasets and the experimental setting in which we evaluated our approach along with established online *AD* algorithms.

5.1 Data Streams

We conducted our experiments on incremental data streams which we emulated using the Covertype, Shuttle [3] and Creditcard [8] real-world data streams. The key characteristics of the selected streams are presented in the following.

Covertype originally consists of 7 classes representing different types of forest cover. Due to its non-stationary data distribution, Covertype is frequently used to emulate data streams in online learning (see e.g. [24,26]. We modified the dataset according to the standard procedure from previous work on *AD* (e.g. [26]), where the most frequent class is selected as normal and the rarest class as anomalous, resulting in an anomaly share of 0.96%. [4] We removed the remaining classes and categorical attributes, leaving a total of 10 numerical attributes. We then grouped the anomalous samples in short sequences of 2 to 9 samples and randomly distributed them throughout the data stream.

Shuttle originally contains 8 classes. According to [26], we used the first class as normal data and samples of the remaining classes, except for the excluded fourth class, as anomalous. The share of anomalies in this modified dataset is 7.15%. Originally in time order, the dataset donor later randomized the order of data examples causing the dataset to exhibit a largely stationary data distribution [26].

Creditcard [8] consists of credit card transactions performed in September 2013 out of which only 0.17% were marked as fraudulent, causing a highly imbalanced class distribution. Due to the confidentiality of the original transactions, the data is provided in the form of the first 28 principal components as well as the timestamp and monetary amount of each transaction. Since Creditcard features stream-typical characteristics such as concept drift, as well as a close relation to real-world fraud detection, the dataset was intensively investigated regarding online *AD* (see [8]).

Due to their different properties regarding the occurrence of concept drifts as well as their share of anomalous data, the selected data streams cover the necessary range of possible streaming-*AD* scenarios.

5.2 Setup

In this section, we present the experimental setup, necessary to obtain the results presented in Section 6. While the proposed *AD* algorithms can, in theory, be executed using any *AE* architecture and optimization algorithm, we used shallow networks with coupled encoder- and decoder weights and *Scaled Exponential Linear Unit* (*SELU*) activations along with basic *SGD* optimization for the sake of simplicity and computational efficiency. For calculating the models'

[3] Both published at the UCI ML Repository https://archive.ics.uci.edu/ml (last accessed August 30, 2022)

[4] All modified datasets are available at https://github.com/lucasczz/DAADS (last accessed August 30, 2022).

reconstruction errors, we used a smooth *Mean Absolute Error* (*MAE*) function, which we selected due to its advantage of being less prone to outliers compared to *Mean Squared Error*s (*MSE*s) [9].

For the performance evaluation, we used a learning rate of 0.02 for *AE* and *DAE*, and a higher value of 0.1 for the *PW-AE* to account for the adaptive reduction of its learning rate. Like [16], we determined the network width by using a *latent layer ratio* specifying the number of units in the hidden layer relative to the number of input features to account for the varying dimensionality between data streams. We evaluated the basic *AE* with a latent layer ratio of 10% and, due to the regularization induced by applying 10% dropout, we chose 100% for the *DAE* and *PW-AE* models. For the p_0 parameter of the latter variant, we used a value of 0.9.

We evaluate the proposed incremental *AE* anomaly detectors along with several well-established online anomaly detectors such as *iLOF* [20], *HST* [26], *RRCF* [12], *xStream* [15] and *Kit-Net* [16]. Except for *xStream*, we executed all reference algorithms using the authors' suggested parameter values and restricted the sizes of all sliding windows to a maximum of 250 data points. We ran *xStream* with an improved configuration due to poor performance with its default setting. For *Kit-Net*, we used the first 100 examples in each data stream to map the input features to the individual *AE*s.

6 Results

In this section, we present the results for the experiments depicted in Section 5. Fig. 2 shows an example of the distribution of individual anomaly scores generated by different *AD* models for Shuttle. Denote that the value range of the anomaly scores differs significantly between the individual models, which can be remedied by adjusting the value range of the anomaly threshold. While all models yield higher average scores for anomalous examples, it can be seen that the *AE* models provided the most significant gap between scores of anomalous and normal samples throughout most of the data stream segment. The *ROC-AUC* and *PR-AUC* scores shown in Table 1 reflect this observation. While the tree-based *RRCF* and *HST* come close to the *AE* models in terms of *ROC-AUC*, the latter clearly outperformed all other approaches in terms of *PR-AUC*. For Covertype and Creditcard, we found similar results in that the *DAE* and *PW-AE* models yielded significantly higher *PR-AUC* values than the baseline approaches while being among the highest performing models in terms of *ROC-AUC*. Their high performance on all three of the investigated datasets highlights the models' ability to accurately identify anomalies for both data with and without concept drift and different degrees of class imbalance. While providing overall slightly less accurate anomaly scores, the basic *AE* also managed to exceed the baseline models' *PR-AUC* for all datasets except for Covertype. The *AE*s were among the most computationally efficient models, using only a fraction of the runtime (**R2** and **R4**) of most competing algorithms. Although being slower than *Kit-Net* on Covertype and Shuttle, the *AE* models' runtimes were still within the

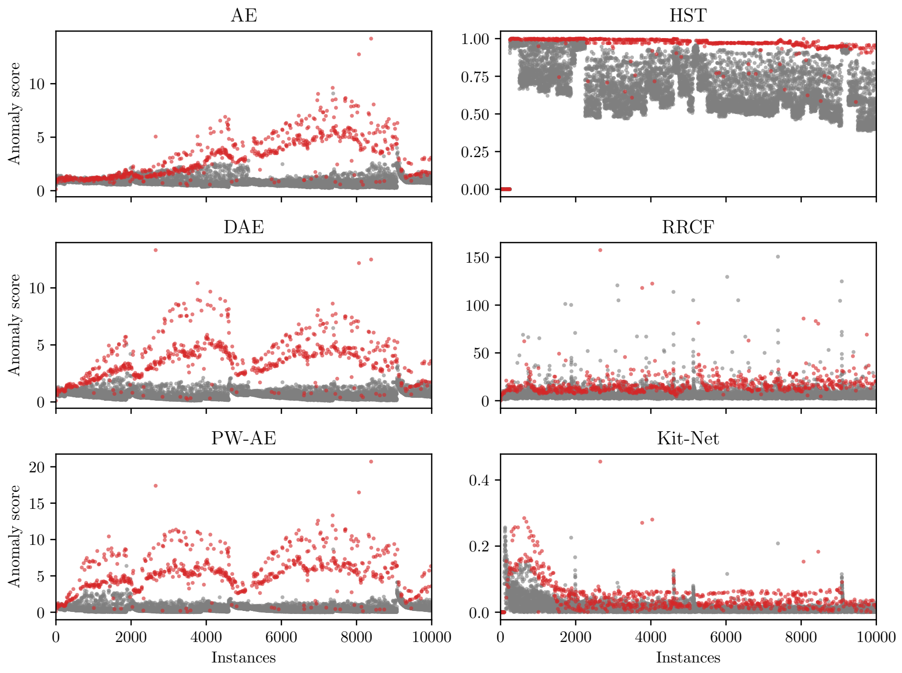

Fig. 2: Distribution of anomaly scores as a function of the number of processed instances on Shuttle. Anomalies are drawn in red.

same order of magnitude. For Creditcard, the runtimes were, on the other hand, significantly lower, which made the *AE*s on average faster than the competing algorithms when considering all datasets. Since the memory requirement of an *AE* is influenced by the choice of architecture rather than by data instances occurring over time, the memory consumption is constant and can be intentionally limited/selected by adjusting the network's size. When comparing individual *AE* models, the *DAE* and *PW-AE* demonstrated a distinct advantage in terms of accuracy metrics. Especially for Covertype, the alternative model architecture led to significantly higher *PR-AUC* scores, which came at the cost of longer runtimes due to the additional forward pass needed to reconstruct the corrupted input for each training step. This advantage is also apparent by the distribution of scores displayed in Fig. 3, where the scores of anomalous data examples separate from the scores of normal data at a much higher rate at the start of the data stream. We achieved further improvements in discriminative performance by using the *PW-AE* update rule for Creditcard and – likely due to its higher grade of contamination – especially for Shuttle. With regard to **R5**, Fig. 3 shows, that the *PW-AE* adapted more quickly to normal samples in Shuttle than the *DAE* due to its higher default learning rate. While the *DAE*'s losses briefly collapse to a single range of values, the *PW-AE*'s losses remain almost perfectly

Table 1: Average benchmarking results for 10 random seeds. Best results are displayed in bold. Runtimes are given in core-minutes of an Intel(R) Xeon(R) Platinum 8180M CPU.

	Covertype			Creditcard			Shuttle		
Model	ROC-AUC	PR-AUC	Runtime	ROC-AUC	PR-AUC	Runtime	ROC-AUC	PR-AUC	Runtime
iLOF [20]	0.934	0.339	7.44	0.922	0.127	8.96	0.551	0.193	1.52
HST [26]	0.905	0.141	21.07	0.931	0.171	21.47	0.975	0.78	3.92
RRCF [2]	0.968	0.301	240.82	**0.950**	0.103	354.46	0.947	0.461	43.54
xStream [15]	0.754	0.033	13.85	0.711	0.005	15.38	0.724	0.118	2.35
Kit-Net [16]	0.905	0.205	**0.95**	0.943	0.141	6.02	0.803	0.259	**0.17**
AE	0.956	0.266	1.56	0.940	0.234	**1.67**	0.973	0.886	0.26
DAE	**0.984**	**0.501**	2.60	0.943	0.247	2.66	0.981	0.922	0.42
PW-AE	0.982	0.451	2.87	0.945	**0.258**	2.95	**0.986**	**0.955**	0.47

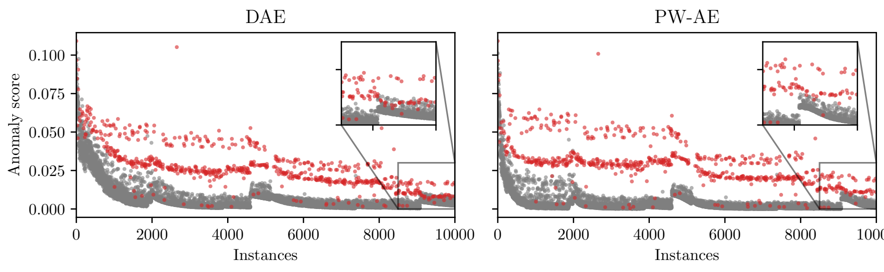

Fig. 3: Distribution of unscaled *AE* anomaly scores concerning the number of processed instances on Shuttle. Anomalies are drawn in red.

linearly separable throughout the whole stream segment and separate at a much faster rate after the sudden increase of normal losses due to a faster adaptation to normal data and even a slight increase in losses for anomalies. Since the *PW-AE*'s training procedure reinforces the model's concept of abnormality, its benefit likely depends on the accuracy of the underlying base model, which is supported by the performance gains on Shuttle for which the *DAE* and *AE* models are already able to label the majority of data accurately. Nevertheless, the distribution of anomaly scores demonstrates that our alternative *PW-AE* training approach can significantly widen the gap between the scores of anomalous and normal instances and therefore improve the separability of anomalies under the influence of contaminated training data as reflected by its increases in *ROC-AUC*, and *PR-AUC*.

Regarding the robustness of different contamination shares, Fig. 4 shows that the *PW-AE* appears to benefit from higher levels of contamination. As the proportion of anomalies randomly distributed in the Covertype or Shuttle data increases, the benefit of the *PW-AE* training procedure increases, leading to an increase in the *ROC-AUC* of the model compared to the architecturally identical *DAE* trained at a fixed learning rate. This observation supports the premise that our modified procedure can mitigate the effects of training on anomalous data. To investigate the effect of the parameterization, we ran evaluations for

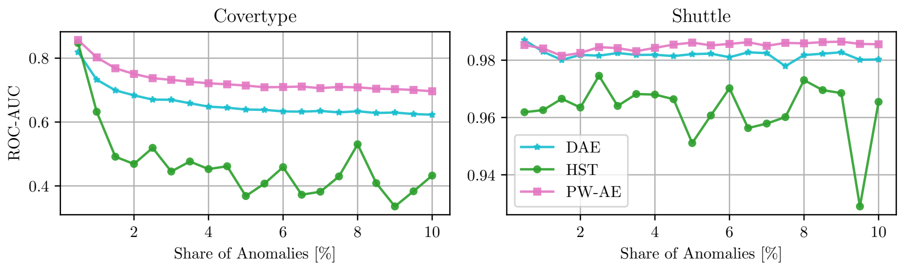

Fig. 4: *ROC-AUC* scores for varying anomaly percentages, which we generated by randomly sampling and inserting anomalies into the data. To allow for larger anomaly percentages, we used sampling with replacement. Each dataset created in this manner consisted of 50,000 examples.

learning rates ranging from 0.001 to $0.001 \cdot 2^8$ and different optimizers, the results of which are illustrated in Fig. 5. When using conventional *SGD*, the basic- and denoising-*AE* on average produced the highest *ROC-AUC* for learning rates between 0.01 and 0.04, with both increases and decreases resulting in declines. Due to its adaptively decreasing step sizes, the *PW-AE* appears to benefit from larger base learning rates than the models using conventional *SGD* updates. While yielding marginally lower *ROC-AUC* at learning rates below 0.05, the *PW-AE*'s average *ROC-AUC* for higher learning rates exceeded those of the best-performing *AE* and *DAE*, indicating that the *PW-AE*'s performance advantage must be caused by its alternative training approach rather than a difference in learning rate. The optimal learning rates are relatively consistent across datasets, supporting the transferability of our results. Overall, the *Adam* [13] and *HD-SGD* [3] optimization algorithms did not provide significant advantages over basic *SGD*, since neither approach was able to improve the *ROC-AUC* or contribute to the robustness with respect to the base learning rate. Considering the additional computational complexity of *Adam* and *HD-SGD* compared to plain *SGD*, the latter is therefore likely the best suited out of the investigated optimization algorithms for online *AD*.

To investigate the impact of the network width on *AD* performance, we performed test-then-train evaluations with latent layer ratios between 10% and 200%. As displayed in Fig. 6 all model variants achieved *ROC-AUC* scores close to or greater than 0.96 for all investigated latent layer ratios, indicating that the performance of the proposed approach is rather robust to the choice of network width. In terms of differences between model variants, it appears that the basic

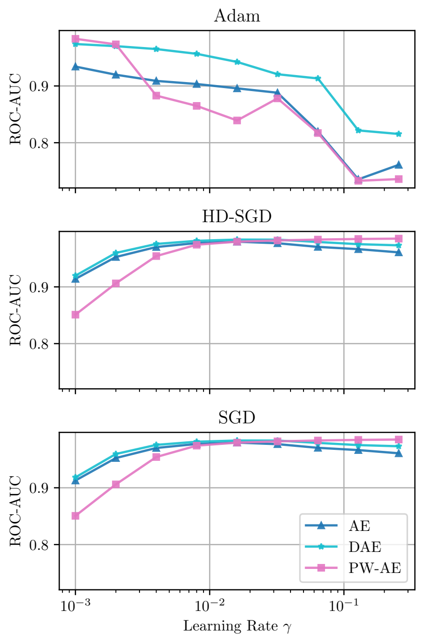

Fig. 5: *ROC-AUC* scores for variable learning rates γ averaged over the first 50,000 samples of each dataset.

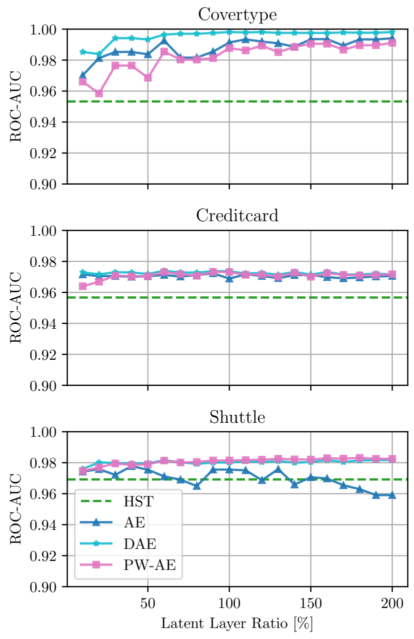

Fig. 6: *ROC-AUC* scores for *AE* anomaly detectors with varying latent layer ratios on the first 50,000 samples of each dataset.

AE favors lower latent layer ratios in some cases, which its lack of regularization could explain.

7 Conclusion and Future Work

In this work, we formalized the task of online *AD* based on predefined requirements for *AD* on streaming data, for which we introduced an *AE*-based framework. Our experimental results on three real-world datasets showed the proposed approach to yield significantly more accurate predictions than established baseline models, as is evident by their at least competitive or often superior *ROC-AUC* scores and their *PR-AUC* scores, which in the case of the most performant *PW-AE* and *DAE* regularly exceeded the scores of the next best non-*AE* model by more than 20%. Using only a fraction of the processing time required by most baseline models, the proposed models were also among the most computationally efficient models.

To mitigate the drawback of the conventional *AE* training objective in the case of contaminated training data, we further proposed an alternative *AE* objective for *AD*, which we used to derive a modified simple yet effective modification of the *SGD* update rule that reduces the learning rate for data instances that the *AE* deems to have a high chance of being an outlier. Our experiments indicate that the *PW-AE* models using this technique can reduce losses on normal data while maintaining higher losses on anomalous data compared to the models.

To investigate the effect of different design choices on the predictive performance of the proposed approach, we performed additional experiments, in which the online *AE* models showed relatively high robustness towards the choice of the learning rate and the network width (see Figs. 5 and 6). All in all, our experimental results also suggest that the proposed approach provides a promising basis for the development of even more effective online *AD* techniques. In this work, we showed a promising approach for online *AD* based on *AE*s that outperforms existing methods and can be further improved by tuning the model's learning rate according to an estimate of the anomaly probability as demonstrated by our *PW-AE*. While we calculated this estimate under the assumption that the *AE*'s losses are normally distributed, a more tailored approach would most likely lead to more accurate estimates and therefore decrease the risk of selecting an incorrect learning rate. If available, label information could also be exploited to more precisely adjust the learning rate in a semi-supervised approach.

Acknowledgements

This work was partially supported by the project *AI REGIO*, funded by Horizon 2020 research and innovation programme under Grant Agreement Number: 952003 and the project *CoPoT*, funded by the German Federal Ministry of Education and Research (BMBF) under the Grant 16SV8281.

References

1. Ashfahani, A., and Pratama, M.: Autonomous Deep Learning: Continual Learning Approach for Dynamic Environments. In: SDM (2019). DOI: 10.1137/1.9781611975673.75
2. Bartos, M.D., Mullapudi, A., and Troutman, S.C.: Rrcf: Implementation of the Robust Random Cut Forest Algorithm for Anomaly Detection on Streams. J. Open Source Softw. (2019). DOI: 10.21105/joss.01336
3. Baydin, A.G., *et al.*: Online Learning Rate Adaptation with Hypergradient Descent. In: ICLR (2018)
4. Bifet, A., *et al.*: Machine Learning for Data Streams: With Practical Examples in MOA (2018)
5. Borges, N., and Meyer, G.G.L.: Coping with Training Contamination in Unsupervised Distributional Anomaly Detection. In: CISS (2009). DOI: 10.1109/CISS.2009.5054728
6. Breunig, M., *et al.*: LOF: Identifying Density-Based Local Outliers. In: ACM SIGMOD (2000). DOI: 10.1145/342009.335388
7. Chen, J., *et al.*: Outlier Detection with Autoencoder Ensembles. In: SDM (2017). DOI: 10.1137/1.9781611974973.11

8. Dal Pozzolo, A., *et al.*: Learned Lessons in Credit Card Fraud Detection from a Practitioner Perspective. Expert Systems with Applications (2014). DOI: 10.1016/j.eswa.2014.02.026
9. Girshick, R.: Fast R-CNN. ArXiv 150408083 Cs (2015)
10. Glorot, X., and Bengio, Y.: Understanding the Difficulty of Training Deep Feedforward Neural Networks. In: AISTATS (2010)
11. Gong, D., *et al.*: Memorizing Normality to Detect Anomaly: Memory-Augmented Deep Autoencoder for Unsupervised Anomaly Detection. In: ICCV (2019). DOI: 10.1109/ICCV.2019.00179
12. Guha, S., *et al.*: Robust Random Cut Forest Based Anomaly Detection on Streams. In: ICML (2016)
13. Kingma, D.P., and Ba, J.: Adam: A Method for Stochastic Optimization. In: ICLR (2015)
14. Liu, F.T., Ting, K.M., and Zhou, Z.-H.: Isolation Forest. In: ICDM (2008). DOI: 10.1109/ICDM.2008.17
15. Manzoor, E., Lamba, H., and Akoglu, L.: xStream: Outlier Detection in Feature-Evolving Data Streams. In: ACM SIGKDD (2018). DOI: 10.1145/3219819.3220107
16. Mirsky, Y., *et al.*: Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. ArXiv 180209089 Cs (2018)
17. Montiel, J., *et al.*: River: Machine Learning for Streaming Data in Python. CoRR (2020)
18. Na, G.S., Kim, D., and Yu, H.: DILOF: Effective and Memory Efficient Local Outlier Detection in Data Streams. In: ACM SIGKDD (2018). DOI: 10.1145/3219819.3220022
19. Parisi, G.I., *et al.*: Continual Lifelong Learning with Neural Networks: A Review. Neural Networks (2019). DOI: 10.1016/j.neunet.2019.01.012
20. Pokrajac, D., Lazarevic, A., and Latecki, L.J.: Incremental Local Outlier Detection for Data Streams. In: IEEE CIDM (2007). DOI: 10.1109/CIDM.2007.368917
21. Rousseeuw, P.J., and Leroy, A.M.: Robust Regression and Outlier Detection (1987)
22. Sahoo, D., *et al.*: Online Deep Learning: Learning Deep Neural Networks on the Fly. In: IJCAI (2018). DOI: 10.24963/ijcai.2018/369
23. Sakurada, M., and Yairi, T.: Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction. In: MLSDA Workshop (2014). DOI: 10.1145/2689746.2689747
24. Salehi, M., *et al.*: Fast Memory Efficient Local Outlier Detection in Data Streams. IEEE TKDE (2016). DOI: 10.1109/TKDE.2016.2597833
25. Schölkopf, B., *et al.*: Support Vector Method for Novelty Detection. In: NeurIPS (2000)
26. Tan, S.C., Ting, K.M., and Liu, T.F.: Fast Anomaly Detection for Streaming Data. In: IJCAI (2011)
27. Widmer, G., and Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts. Mach Learn (1996). DOI: 10.1007/BF00116900
28. Wu, K., *et al.*: RS-Forest: A Rapid Density Estimator for Streaming Anomaly Detection. In: ICDM (2014). DOI: 10.1109/ICDM.2014.45
29. Zhou, C., and Paffenroth, R.C.: Anomaly Detection with Robust Deep Autoencoders. In: ACM SIGKDD (2017). DOI: 10.1145/3097983.3098052
30. Zhou, G., Sohn, K., and Lee, H.: Online Incremental Feature Learning with Denoising Autoencoders. In: AISTATS (2012)
31. Žliobaitė, I., Pechenizkiy, M., and Gama, J.: An Overview of Concept Drift Applications. In: Big Data Analysis: New Algorithms for a New Society (2016). DOI: 10.1007/978-3-319-26989-4 4

32. Zong, B., *et al.*: Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. In: ICLR (2018)