HEAT-RL: ONLINE MODEL SELECTION FOR STREAMING TIME-SERIES ANOMALY DETECTION

Yujing Wang[1,*], **Luoxing Xiong**[2,*], **Mingliang Zhang**[1], **Hui Xue**[3], **Qi Chen**[3], **Yaming Yang**[2], **Yunhai Tong**[1], **Congrui Huang**[2], **Bixiong Xu**[2]
[1]Key Laboratory of Machine Perception, MOE, School of Artificial Intelligence, Peking University, Beijing, China
[2]Microsoft, Beijing, China; [3]Microsoft Research Asia, Beijing, China
{yuwang,zml24,yhtong}@pku.edu.cn; {luxiong,xuehui,cheqi,yayaming,conhua,bix}@microsoft.com

ABSTRACT

Time-series anomaly detection plays an important role in various applications. In a commercial system, anomaly detection models are either unsupervised or pre-trained in a self-supervised manner offline; while in the online serving stage, an appropriate model should be selected to fulfill each customer's requirement with only a few human interactions. Existing online model selection methods do not have good data efficiency, failing to achieve good performance with limited number of manual feedbacks. In this paper, we propose Heat-RL, a novel reinforcement learning algorithm tailored to online model selection for streaming time-series data. Specifically, we design a new state based on metric-oriented heatmaps and apply ResNet for policy and value networks to capture the correlations among similar model configurations. Experiments demonstrated the effectiveness of Heat-RL on both academic and industrial datasets. On all datasets, the average F_1 and last F_1 scores have been improved by 5.5% and 14.6% respectively compared to the best state-of-the-art solution.

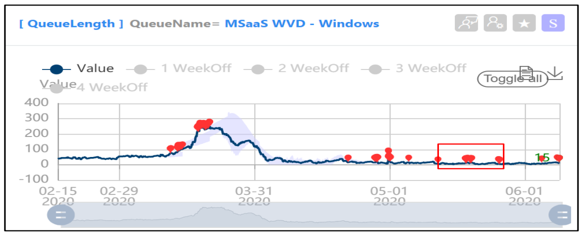
1 INTRODUCTION

Modern business benefits from a booming of data analyzing techniques. Organizations started to establish business metrics to ensure the success of their products (Hochenbaum et al., 2017; Ren et al., 2019). Their ubiquitous requirement is to monitor online streaming metrics continuously and alert for abnormal events in time. In an online monitoring system, time-series anomaly detection is one of the key components. Figure 1 demonstrates two real applications in a commercial metrics monitoring system. In Figure 1(a), the metric denotes how many people are waiting for customer support. Here, anomalies indicate potential downtime of the product, so both large and small spikes are critical and require careful examination. In Figure 1(b), the metric stands for the count of remaining messages in a Kafka queue (Kreps et al., 2011). In this case, only large spikes reflect throughput issues and are considered as anomalies, while small spikes are usually caused by network latency and do not raise significant problems.

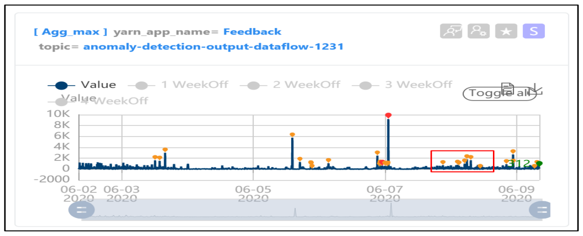
Because we do not know the data distribution and preference for a new customer in advance, the time-series anomaly detection models are usually unsupervised or pre-trained via self-supervised targets in the offline stage. However, in the online stage, a single anomaly detection model with a fixed hyper-parameter setting is not sufficient to handle diverse business scenarios. Also, the data distribution of time-series may drift over time. For instance, we can see from Figure 1(a) that the spikes become smaller as time goes on. Thus, we need to select a tailored model with the optimal hyper-parameters for each time-series and update them continuously online. However, designing an effective algorithm for this purpose is non-trivial because of the following challenges:

- A straightforward solution is to adopt existing Bayesian optimization algorithms, such as TPE (Bergstra et al., 2011). However, Bayesian-based methods are difficult to encode the high-dimensional environment states and adapt to dynamic changes in data distribution. Thus, they are not applicable or perform poorly in the scenario of online model selection.
- Reinforcement Learning (RL) is more appropriate for solving this kind of problems. Nevertheless, according to a preliminary study, traditional RL algorithms (e.g., A2C (Mnih et al., 2016), DQN (Mnih et al., 2013), and Hyp-RL (Jomaa et al., 2019)) do not yield satisfactory results in our scenario. The reason is that there are only a few manual labels given by the customers, while existing RL algorithms need a large amount of user feedbacks to achieve good effectiveness.

[*] Equal Contribution



(a) Customer Support Queue



(b) Kafka Lag Queue

Figure 1: Motivating cases for online model selection

- Other RL-based algorithms designed for anomaly detection also do not solve the specific challenge in online learning. For instance, DPLAN (Pang et al., 2021) is a recent reinforcement learning approach for semi-supervised anomaly detection. Unfortunately, it needs to know all labels in advance and interacts with a simulated environment for many times. Thus, it is designed for offline optimization and is not applicable to online model selection.

As stated above, there remains an open question: *How to design a more efficient method for online model selection of streaming time-series anomaly detection?* To tackle this problem, we propose a novel reinforcement learning algorithm, Heat-RL, which adopts an actor-critic framework equipped with novel policy and value networks to improve the data efficiency of online model selection. Specifically, we introduce the heatmap of target metrics (F_1 scores in this paper) as a novel state, and leverage a ResNet architecture to predict action and value functions based on the current state. The proposed heatmap aims to explicitly memorize the historical performance for each model configuration and facilitate learning of policy and value networks through only a few user feedbacks. Moreover, ResNet captures the correlations among similar model configurations and brings a local-smoothing inductive bias to improve the generalization capability of the model selector. In each iteration, the policy network chooses one model configuration (including model type and hyper-parameters), then a potential reward is estimated by the value network. The policy and value networks are updated regularly to reflect users' new feedbacks. Finally, the well-trained Heat-RL model will understand human preferences and generate a customized model configuration for each time series. In addition, we design a streaming simulator to evaluate different algorithms using historical data streams. The effectiveness of Heat-RL has been verified on both academic and industrial datasets. Compared to the best state-of-the-art solution, the average F_1 and last F_1 scores have been improved by 5.5% and 14.6% respectively.

The **contributions** of this paper are summarized as follows:

- We propose a novel reinforcement learning method, Heat-RL, which performs online model selection for streaming time-series anomaly detection with both tailored states and policy/value networks. Notably, we use metric-oriented heatmaps as states and adopt a ResNet architecture for policy and value networks to capture the correlations among similar model configurations. Ablation study verifies the superiority of these designs for improving data efficiency.
- We design a simulation pipeline to evaluate the performance of different algorithms, which imitates the online scenario as real as possible based on historical time-series data streams. Experimental results demonstrate the effectiveness of Heat-RL on both academic datasets and an internal dataset from a commercial metrics monitoring system. To advocate reproduction, we will release our code after acceptance of this paper.
- To address the problem of limited number of user feedbacks, we analyze the performance of Heat-RL model with different feedback ratios. As a result, we achieve satisfactory performance with different feedback ratios, demonstrating its generality in numerous business scenarios. We believe that Heat-RL will also benefit other online optimization problems by leveraging domain-specific states and feedbacks.

2 Related Works

2.1 Time-series Anomaly Detection

Time-series anomaly detection approaches can be divided into two categories, unsupervised and supervised (including semi-supervised). Supervised methods can achieve superior accuracy, but continuous labels are usually unavailable in industrial environments, making these approaches insufficient for online applications. Thus, we mainly introduce unsupervised anomaly detection methods that are more relevant to this paper. Traditional unsupervised approaches are based on statistical characteristics to model normal/abnormal patterns, such as HBOS (Histogram-Based Outlier Score) (Goldstein & Dengel, 2012), Wavelet Analysis (Lu & Ghorbani, 2009), SVD (Mahimkar et al., 2011), ARIMA (Zhang et al., 2005), SPOT and DSPOT (Siffer et al., 2017). In the industry, Twitter publishes an anomaly detection method using Seasonal Hybrid Extreme Study Deviation test (S-H-ESD) (Vallis et al., 2014); Microsoft utilizes Spectral Residual (SR) (Ren et al., 2019), which filters out anomalies in the frequency domain. In addition, Isolation Forest (Liu et al., 2012) is another widely-used unsupervised anomaly detector. It isolates abnormal observations from normal ones through half-space data partition. Recent advances in neural networks also lay a strong foundation for time-series anomaly detection. For instance, DONUT (Xu et al., 2018) is an unsupervised anomaly detection method based on Variational Auto-Encoder (VAE), but it is not efficient enough to serve large-scale time-series data steams.

2.2 Online Model Selection

Bayesian Optimization (BO) (Snoek et al., 2012) is an efficient method for model selection. The most commonly used Bayesian optimization methods include Gaussian Process (GP) (Snoek et al., 2015) and Tree-structured Parzen Estimator (TPE) (Bergstra et al., 2011). GP leverages a similarity measure between input points (the kernel function) to estimate the value for an unseen point. TPE takes a tree-structured parameter space and utilizes Gaussian Mixture Model (Reynolds, 2009) to optimize towards the global maximum/minimum. However, Bayesian-based methods are difficult to encode the environmental state and hard to adapt to online drifting data, thus yielding unsatisfactory performance in online model selection scenarios. Hyp-RL (Jomaa et al., 2019) models the online hyper-parameter selection problem as a sequential decision process and addresses it with reinforcement learning. It shows promising results and outperforms traditional BO-based methods, but it requires a large amount of labeled data to generate effective results.

2.3 Reinforcement Learning for Anomaly Detection

The combination of Deep Neural Networks (DNNs) and Reinforcement Learning (RL) has achieved notable success in many domains (Zoph & Le, 2017), including time-series anomaly detection. (Oh & Iyengar, 2019) propose an end-to-end framework for sequential anomaly detection using Inverse Reinforcement Learning (IRL), the goal of which is to understand the underlying motive of observed behaviors when anomalous actions are intentional. (Zha et al., 2020) propose Active Anomaly Detection with Meta Policy (Meta-AAD), which learns a meta policy to approximate the sequential decision process of anomaly re-ranking. (Huang et al., 2018) present a RNN-based anomaly detector that is trained consistently through Q-learning to meet the objectives with supervised labels. Furthermore, DPLAN (Pang et al., 2021) tackles the semi-supervised anomaly detection problem with partially labeled data by an adapted version of Deep Q-Network (DQN) (Mnih et al., 2013). Our work is different from previous ones as it addresses the online serving scenario of time-series anomaly detection. For offline models, the metric is usually F_1 score at convergence, while for online serving, we need to optimize both F_1 at convergence and the average F_1 score in the entire serving life-time to maintain a good user experience. To the best our knowledge, none of RL-related previous works in the time-series domain have solved the online learning problem.

3 Methodology

3.1 Overview

In an online time-series anomaly detection service, different customers have varying data patterns and definitions of anomalies. It is important to adjust the anomaly detection models in time based on customers' feedbacks to achieve better user experiences. In this work, we focus on how to leverage user feedbacks to select an appropriate time-series anomaly detection model effectively and efficiently for online serving. The process can be formulated as a sequential decision-making problem and solved by the proposed Heat-RL algorithm. Periodically, the policy and value networks in Heat-RL model are updated using recent feedbacks. After the policy network is changed, a new model configuration will be selected for each time-series. Once new data points are received, the customer can alternatively give feedbacks about whether the detection result for a specific data point is wrong; then these feedbacks will be used to calculate rewards in the next iteration.

3.2 Heat-RL Algorithm

In this section, we first introduce the definitions of action, state and reward. Then, we describe the framework of Heat-RL as well as its network architecture and loss function. At last, we will summarize the benefits of Heat-RL algorithm.

Action. The action in our environment is to select one model configuration, including the model type and corresponding hyper-parameters. Suppose the model type is $M = \{m_1, m_2, ..., m_n\}$, and the possible hyper-parameter set for model m_i is $\{H_{i,1}, H_{i,2}, ..., H_{i,k}\}$. In each timestamp t, the action selects a specific model configuration $\mathbf{c}_t =< m_i, h_{i,1}, h_{i,2}, ..., h_{i,k} >, (h_{i,j} \in H_{i,j})$, and deliver its corresponding anomaly detection results to the end customer. For hyper-parameters of real values, we discretize them in the action space because subtle changes in hyper-parameters usually does not change the detection result. Details of the search space can be found in appendix A.

Heatmap-based State. The state consists of time-series features and metric-oriented heatmaps. We use the same time-series features as Hyp-RL (Jomaa et al., 2019), and each time-series also maintains a metric-oriented heatmap for each type of anomaly detection model, as shown in Figure 2. The dimensions of heatmaps correspond to key hyper-parameters of each anomaly detection model. For example, assume that we have two hyper-parameters for model m_i, namely *sliding window size* and *detection threshold*. Then, the corresponding heatmap should be a two-dimensional matrix, $W_{h_{i,1},h_{i,2}}$, where each row $h_{i,1}$ represents a specific window size in $H_{i,1}$, and each column $h_{i,2}$ stands for a certain threshold in $H_{i,2}$. The value at a certain row and column in the heatmap is the average F_1 score of the detection results for the corresponding model configuration.

Reward. The goal of Heat-RL is to improve the accuracy of anomaly detection for the entire life-time of metrics monitoring. At a certain update iteration t, the reward r_t is defined as the improvement of F_1 score by using the current model configuration during time interval $(t-1, t]$.

$$r_t = F_1(c_t) - \text{MA}(F_1(c_{t-1}), F_1(c_{t-2}), ..., F_1(c_{t-s})) \tag{1}$$

where MA is the moving average of historical performance within a sliding window of size s, and we set $s = 5$ empirically. Assume all data points in the time interval $(t-1, t]$ is $\mathbf{x} = \{x_i\}$, and the corresponding label is $\mathbf{y} = \{y_i\}$, where $y_i \in \{0, 1\}$ (0 denotes normal and 1 denotes anomaly). We assign a pseudo label for each data point. If no feedback is given for a specific point, we trust the anomaly detection result as the pseudo label. If there exists a feedback, then we use the manual label as ground truth. Based on the pseudo labels, we calculate an average F_1 score for each model configuration as the reward in iteration t.

$$F_1(c_t) = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{tp}{tp + 0.5(fp + fn)} \tag{2}$$

where tp, fp, fn stand for the number of true positive, false positive and false negative points respectively.

RL Framework. We adopt Advantage Actor-Critic (A2C) (Sutton et al., 1999) as the overall RL framework. A2C is an on-policy algorithm, the key components of which include a policy network and a value network. In A2C, both policy and value networks take the state s_t as input. Based on the current state, the policy network $P(a_t|s_t)$ generates an action a_t, and the value network $V(s_t)$ calculates the estimated reward of current state. Then, upon acquisition of rewards, both policy and value networks are updated. In each iteration, an action a_t is selected based on the policy network. Then, we calculate Q-Function $Q(s_t, a_t)$, which denotes k-step estimation of the reward value on taking the action a_t; as well as the advantage function, $A(s_t, a_t)$, reflecting the marginal gain of taking a_t w.r.t the estimated average value of all actions on the current state, $V(s_t)$.

$$Q(s_t, a_t) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k * V(s_{t+k}) \tag{3}$$

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \tag{4}$$

where k is the step of reward estimation, for which we simply set $k = 1$ in our experiments; γ stands for the decay factor of accumulative reward, r_{t+i} is the actual reward received at timestamp $t + i$, s_t is the current state at timestamp t, $V(.)$ stands for the value function, and $V(s_{t+k})$ is the value estimation of state s_{t+k}.

ResNet-based Policy/Value Networks. We design a tailored network architecture for both the policy and value networks in Heat-RL. Specifically, a ResNet (He et al., 2016) model is applied to the heatmap to capture the continuous relationship among hyper-parameters. As shown in Figure 2, both the policy and value networks share the same encoder architecture, which is composed of 6 ResNet blocks with 256 channels. Each block is composed of two convolutional layers, two normalization and activation layers. We replace BatchNorm in the original ResNet block with instanceNorm, because BatchNorm is found to be less effective in cases where samples are highly correlated (Hoffer et al., 2018). With

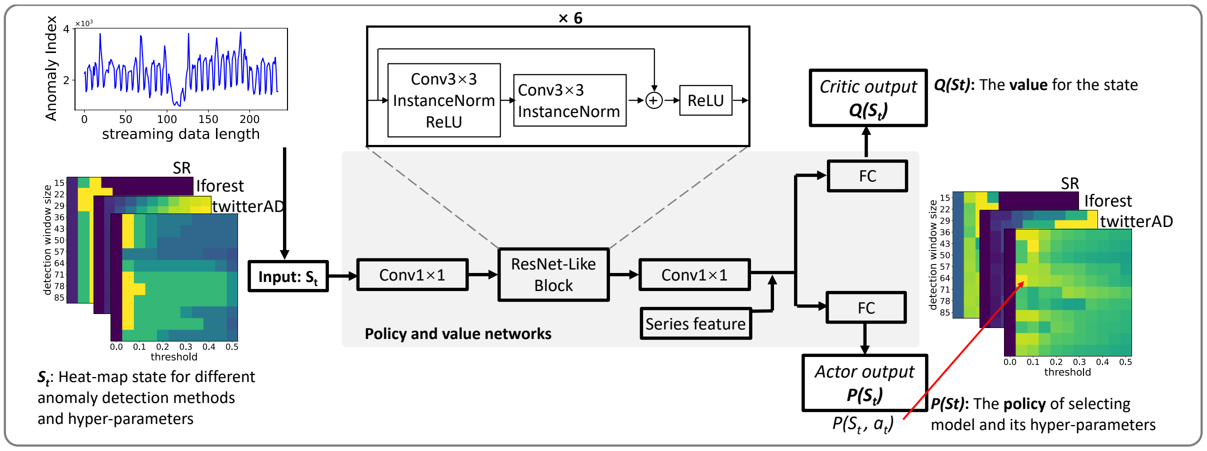


Figure 2: Network architecture of the policy and value networks.

the ResNet architecture over heatmap-based state, configurations with similar hyper-parameters can collaborate with each other to improve data efficiency and result in a more generalizable model selector. As similar configurations are at the adjacent position in a heatmap, ResNet can capture local smoothing patterns via convolutional kernels and improve the data efficiency for training both policy and value networks.

Loss Function. The overall loss function at timestamp t is defined in Equation (5), which consists of the policy loss (L_{P_t}), the value loss (L_{Q_t}), and the entropy of action probabilities (E_t). The action loss is to increase the probabilities of actions with high advantages and decrease the ones with low advantages. The value loss aims to make the critic network generate a more accurate reward estimation for a given state. The distribution entropy is calculated based on action probabilities to let the actor network explore sufficient actions before convergence.

$$L_t = L_{P_t} + \alpha * L_{Q_t} - \beta * E_t \tag{5}$$
$$L_{P_t} = -A(s_t, a_t) * \log P(a_t|s_t) \tag{6}$$
$$L_{Q_t} = (Q(s_t, a_t) - V(s_t))^2 \tag{7}$$
$$E_t = -\sum_{a_t \in \mathcal{A}} P(a_t|s_t) * \log P(a_t|s_t) \tag{8}$$

where $\mathcal{A}$ denotes the action space (all model configurations in our case); α and β adjust the relative importance of different losses.

Benefits of Heat-RL. We emphasize the benefits of Heat-RL with novel designs of heatmap-based states and ResNet architectures. First, the metrics-oriented heatmaps explicitly memorize the historical performance of each model configuration. Because the online training time is strictly limited, it is ineffective to capture historical performance implicitly through policy and value networks. Instead, we utilize a heatmap as input to the networks, making them easier to learn tailored action and critic functions. Second, configurations with similar hyper-parameters can collaborate with each other to improve data efficiency and result in a more generalizable model selector. As similar configurations are at the adjacent position in a heatmap, ResNet can capture local smoothing patterns via convolutional kernels and improve the data efficiency for learning both policy and value networks.

4 EXPERIMENTS

4.1 DATASETS

We use three datasets for evaluation. The statistics of these datasets are summarized in Table 1.

Production is collected from our metric monitoring system. We collect 781 time series with daily intervals. These time series mainly indicate usage metrics of products, such as revenue, number of active users, and number of new users. Feedbacks of data points have been annotated by customers themselves. *KPI* is an open dataset released by AIOps data competition[1]. The dataset consists of multiple KPI time-series with anomaly labels collected from various Internet

[1]https://github.com/NetManAIOps/KPI-Anomaly-Detection

Table 1: Statistics of Datasets

	Production	**KPI**	**Artificial**
# of Time-series	781	29	252
# of Points	134,676	3,004,066	391,692
Length of Time-series	172	103,588	1,554
# of Anomalies	12,596/8.66%	79,554/2.65%	2,607/0.72%

companies. Most time-series have an interval of 1 minute between two adjacent data points, while some of them have an interval of 5 minutes. *Artificial* is a synthetic dataset. Following (Nikolay et al., 2015), we generate time series with seasonal oscillation and trend of a random slope. Anomalies are sampled from a normal distribution and inserted at random positions at a ratio of 0.75%. We publish this dataset[2] to facilitate a fair comparison of future works.

4.2 Metrics

F_1 score is a common metric to evaluate the performance of anomaly detection methods. We follow the evaluation protocol in (Xu et al., 2018), which considers the timeliness of streaming time-series anomaly detection. To examine the quality of anomaly detection results during the entire monitoring process, we calculate both last-F_1 and average-F_1 scores. Last-F_1 is the corresponding F_1 score of selected anomaly detection model at the last timestamp of each time-series, indicating the model performance when RL is converged. Average-F_1 is the average F_1 score of selected models for all historical timestamps, reflecting customer experiences in the whole monitoring process. Assume the data points in time interval $(t-1, t]$ is $\mathbf{x}_t$, and the set of all data points is $\mathbf{x} = \bigcup\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$. The two evaluation metrics are defined formally as below.

$$\text{last-}F_1 = F_1(c_T, \mathbf{x}), \;\; \text{average-}F_1 = \frac{1}{T}(F_1(c_1, \mathbf{x}_1) + F_1(c_2, \mathbf{x}_2), ..., F_1(c_T, \mathbf{x}_T)) \tag{9}$$

We also report precision and recall at the last timestamp for comparison. We treat the whole segment as correct if any observation in this segment is detected as an anomaly in a reasonable delay following (Ren et al., 2019). According to specific product requirements (Ren et al., 2019), the delay for the *KPI* dataset is set as 7 timestamps. This means if the anomaly segment could be detected within 7 minutes since it first occurs, then the detection result will be treated as correct. The delay for both *Production* and *Artificial* datasets is set as 1, as the time intervals of these datasets are longer.

4.3 Streaming Simulator

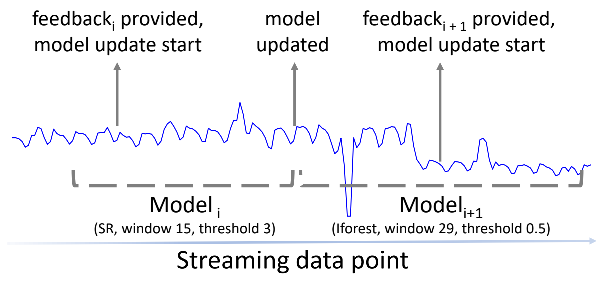

feedback$_i$ provided, model update start
model updated
feedback$_{i+1}$ provided, model update start
Model$_i$
(SR, window 15, threshold 3)
Model$_{i+1}$
(Iforest, window 29, threshold 0.5)
Streaming data point

Figure 3: Illustration of the streaming simulator.

We build a streaming simulator to evaluate each model's performance in an online environment, which is illustrated in Figure 3. At regular intervals, a new data point is ingested into the simulator and processed with the current model, *i.e.*, $model_i$. The customers may give feedbacks if they think the detection results of some points are wrong. In the next iteration, the online model selection module will be trained via feedbacks in time interval $(t, t+1]$. Then, the updated model will choose a new model configuration, *i.e.*, $model_{i+1}$. Hereafter, the following time-series points will be detected by $model_{i+1}$ until it is updated in the next iteration. We simulate feedbacks as close as possible to the real online scenario by controlling the overall feedback ratios, as well as the ratios of *false positive*, *false negative*, *true positive* and *true negative* points. The feedback ratios are naturally different in various business scenarios with

[2]https://github.com/anonymous4doubleblind1/dataset-for-time-series-anomaly-detection

distinct timestamp intervals. Thus, we set the overall feedback ratios differently for each dataset, which imitates the real feedback ratio for a specific time interval in our online environment. For *Production*, *Artificial*, and *KPI* datasets, the feedback ratios are set to 10%, 2%, and 0.2% respectively. In addition, users usually provide feedbacks when the prediction results are wrong, and taking no actions if the anomaly detection results are satisfactory. Thus, we give high sampling probabilities (49%) to *false positive* and *false negative* points, while giving low probabilities (1%) to *true positive* and *true negative* ones. We will analyze the model performance with different feedback ratios in Section 4.7.

4.4 Experimental Setup

4.4.1 Anomaly Detection Models.

We choose three anomaly detection models to construct the candidate space for online model selection. They are the most commonly-used models in industrial anomaly detection systems, owing to their outstanding performance for a balance between accuracy and efficiency.

- **Iforest** (Liu et al., 2012) is a widely-used model for unsupervised time-series anomaly detection. It has also been used in an offline reinforcement learning method for time-series anomaly detection (Pang et al., 2021), serving as the intrinsic reward function.
- **TwitterAD** (Vallis et al., 2014) is an industrial model released by Twitter. It detects anomalies by Extreme Studentized Deviate test (ESD) after removing trends and seasonal bias.
- **SR** (Ren et al., 2019) is an industrial anomaly detector published by Microsoft, which captures time-series anomalies through spectral residual in the frequency domain.

Hyper-parameters. We represent the hyper-parameters of each model in a discretized value space. In practice, we need to find the best trade-off between model accuracy and the latency of online model selection. For this consideration, we select the most important hyper-parameters, *i.e.*, *sliding window size* and *detection threshold* for tuning in all three models. The value range of each parameter is decided based on its original definition. Specifically, the threshold of SR is set between 0.0 and 10.0 with step 0.2. Meanwhile, the threshold of normalized average path length in Iforest is set between 0.1 and 1.0 with step 0.01. The threshold of anomaly ratio of TwitterAD is set between 0 and 0.49 with step 0.05. The range of the detection window size has been set according to the time intervals of different time-series. In *KPI* dataset, the window size is in $[2881, 8641]$ with step 1440. The window size of *Production* dataset is in $[15, 85]$ with step 7. In *Artificial* dataset, the window size is in $[201, 1201]$ with step 100. Note that our methodology is general and more hyper-parameters can be added to the hyper-parameter search space.

Baselines. We consider the following baselines: Brute Search, a widely-used Bayesian optimization method (TPE (Bergstra et al., 2011)), and existing state-of-the-art RL-based methods for online model selection (DQN (Mnih et al., 2013), A2C (Mnih et al., 2016) and Hyp-RL (Jomaa et al., 2019)). Note that GP (Snoek et al., 2015) performs even worse than TPE, so we do not include the results here. Other RL-based SOTAs are either orthogonal to our work (*e.g.*, DPLAN (Pang et al., 2021), Inverse RL (Oh & Iyengar, 2019)) or do not fit in the online learning scenario (*e.g.*, DPLAN (Pang et al., 2021), Inverse RL (Oh & Iyengar, 2019)).

- **Brute Search** utilizes the heatmap proposed in this paper. In each iteration, it selects the anomaly detection model and hyper-parameters with the highest F_1 score in the current heatmap. The heatmap is updated periodically with user feedbacks. Note that brute searching all configurations in the heatmap for each time series is costly, which is unrealistic in online serving.
- **TPE** (Bergstra et al., 2011) selects anomaly detection model and hyper-parameters with the highest expected improvement (EI) of F_1 score by modeling the conditional probabilities with Bayesian method.
- **DQN** (Mnih et al., 2013) is a value-based reinforcement leaning method, in which we use two feed-forward layers as the value network.
- **A2C** (Mnih et al., 2016) is an actor-critic-based reinforcement learning method. Different from Heat-RL, A2C only uses the time-series features and current model configuration to represent a state. Both the policy and value networks are composed of two feed-forward layers.
- **Hyp-RL** (Jomaa et al., 2019) is a DQN-based reinforcement learning method for model selection and hyper-parameter tuning. It uses LSTM to model time series features and historical anomaly detection performance. The hidden states generated by LSTM as well as the current model configuration are used to construct a state.

In our experiments, the same time-series features are used as in the Hyp-RL (Jomaa et al., 2019) work. All reinforcement learning methods use the RMSprop optimizer. The values of γ, α, and β are set to 0.8, 0.5 and 0.01 respectively. We try

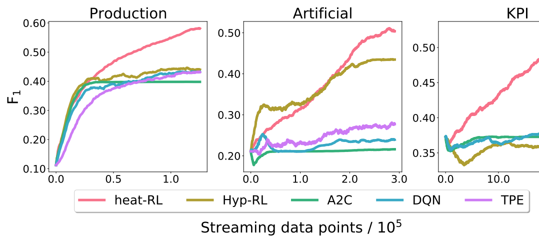


Figure 4: Comparison of F_1 scores over time (timestamps are normalized at x axis).

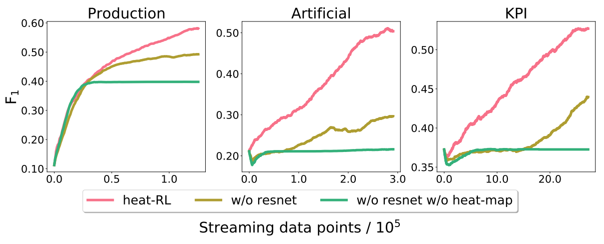


Figure 5: Ablation study.

Table 2: Comparison of Heat-RL with baseline models.

	Production				Artificial				KPI			
	avg-F_1	last-F_1	precision	recall	avg-F_1	last-F_1	precision	recall	avg-F_1	last-F_1	precision	recall
Heat-RL	**0.521**	**0.581**	0.590	0.660	**0.389**	**0.504**	0.435	0.656	**0.455**	**0.527**	0.539	0.615
Brute Search	0.509	0.569	0.571	0.668	0.378	0.425	0.351	0.770	0.409	0.428	0.382	0.646
Hyp-RL	0.420	0.440	0.418	0.550	0.376	0.435	0.363	0.617	0.359	0.366	0.301	0.646
A2C	0.386	0.398	0.345	0.560	0.211	0.216	0.133	0.816	0.358	0.373	0.306	0.648
DQN	0.405	0.433	0.405	0.578	0.228	0.239	0.149	0.795	0.370	0.377	0.313	0.645
TPE	0.394	0.431	0.410	0.601	0.250	0.277	0.205	0.692	0.294	0.268	0.246	0.634

the learning rate from 10^{-5} to 10^{-6} and show the result of the best learning rate for each method. For value-based methods, actions are sampled by ϵ-greedy strategy, where ϵ decreases from 1 to 0.1 during the first million steps linearly, and is fixed at 0.1 afterwards (Mnih et al., 2013). Our work is implemented in Pytorch and Jittor[3] frameworks. Every experiment has been repeated three times to mitigate the effects of random seeds.

4.5 Results

We evaluate different methods through the streaming simulator, and the results are shown in Figure 4. From the figure, we notice that Heat-RL generally outperforms all baselines on all three datasets. On the *Production* dataset, all RL-based methods outperform TPE when the number of streaming data points is small. Then, TPE surpasses A2C a little, but is still worse than other RL-based methods when streaming data points further accumulate. Regarding the *Artificial* dataset, the Heat-RL and Hyp-RL methods are much more accurate than other methods, while Heat-RL gradually outperforms Hyp-RL when the number of data points increase. On the *KPI* dataset, the Heat-RL method performs much better than all other SOTAs. TPE cannot converge on this dataset.

In Table 2, we further compare the metrics of different methods, where avg-$F1$ denotes average F_1 score of the selected models for all historical timestamps; last-F_1 is the corresponding F_1 score of selected model at the last timestamp; Both precision and recall are reported at the last timestamp. To summarize, Heat-RL achieves the best F_1 score both at convergence and during the entire monitoring process. This shows a good data efficiency and generality of Heat-RL when there are limited user feedbacks.

4.6 Ablation Study

We show the contribution of metric-oriented heatmap and ResNet model architecture through ablation study. In the ablation setting without ResNet, neural networks of two feed-forward layers are utilized as policy and value networks. From the results in Figure 5, we see that the model performance drops significantly when using vanilla feed-forward layers to replace the ResNet architecture. Moreover, the performance becomes even worse after removing the metric-oriented heatmaps. These results confirm the effectiveness of using metric-oriented heatmaps as states and leveraging the inductive bias of ResNet to model the local relationship of hyper-parameters.

4.7 Analysis of Different Feedback Ratios

In the online environment, the feedback ratios vary a lot for different applications. It is important to have a robust method that increases anomaly detection accuracy with different feedback ratios, especially when the feedback number

[3]http://jittor.com

Table 3: Comparison of last-F_1 scores with different feedback ratios

	Production			**Artificial**			**KPI**		
Feedback Ratio	5%	10%	25%	5%	10%	25%	5%	10%	25%
Heat-RL	**0.519**	**0.581**	**0.646**	**0.533**	**0.531**	0.510	**0.609**	**0.607**	**0.629**
Hyp-RL	0.415	0.440	0.427	0.321	0.246	0.391	0.248	0.356	0.474
A2C	0.387	0.398	0.451	0.253	0.512	**0.513**	0.373	0.547	0.560

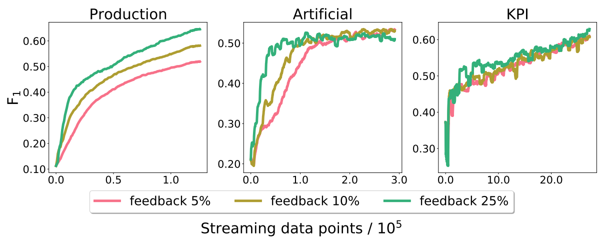


Figure 6: Model performance of Heat-RL

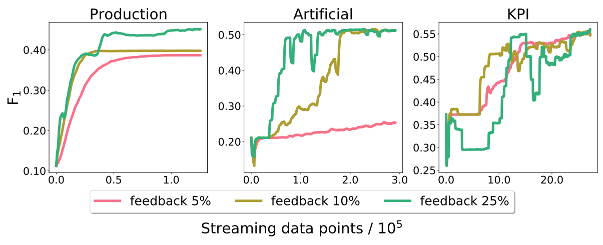


Figure 7: Model performance of A2C

is very small. We compare the last-F_1 score of Heat-RL, Hyp-RL, and vanilla A2C method when feedback ratios are 5%, 10% and 25% respectively. As shown in Table 3, Heat-RL demonstrates consistent improvement at different levels of feedback ratios.

We also analyze the model performance over time when feedback ratio changes from 5% to 25%. The model performance of Heat-RL is visualized in Figure 6. On the *Production* dataset, the model performance of Heat-RL steadily increases when the feedback ratio becomes larger, while on the *KPI* dataset, the performance of Heat-RL only improves a little. Intuitively, anomaly detection is more challenging for *Production* dataset, thus we need more feedbacks to achieve better results. Instead, patterns in the *KPI* dataset are much simpler, so only a few feedbacks could be enough to select an appropriate anomaly detection model. The performance of vanilla A2C (without heatmap and ResNet) is compared in Figure 7. Although the F_1 score of selected model is increased by time in most cases, the performance is relatively unstable and becomes very poor when the feedback ratio is small on *Artificial* dataset. This indicates that both metric-oriented heatmaps and ResNet architectures are indispensable to achieving a stable performance.

5 Shipping to Production

After shipping the Heat-RL algorithm to a commercial monitoring service, the online click-through rate (CTR) of anomaly alerts has been improved by 29.9% (Figure 8). Online CTR indicates the probability of a customer to click into the system portal after receiving an anomaly alert, which reflects how much the customers trust our anomaly detection results. Therefore, higher CTR score indicates better user satisfaction.

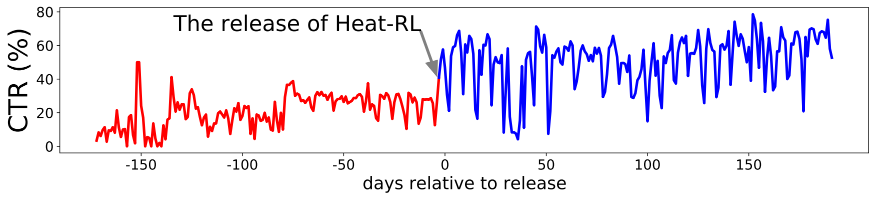


Figure 8: CTR of anomaly alerts in a commercial metrics monitoring system

6 Conclusion & Future Work

In this paper, we present a novel reinforcement learning algorithm, namely Heat-RL, which is tailored to online model selection for streaming time-series anomaly detection. Different with existing state-of-the-arts, Heat-RL utilizes metrics-oriented heatmaps as states and ResNet architectures as value and policy networks. The effectiveness of Heat-RL has been proved by extensive experiments and analyses. Due to the unavailability of large-scale labeled data, time-series anomaly detection is difficult, if not impossible, to generate satisfied results for different applications. Therefore, a framework that can learn a better model for various applications automatically is crucial in industry. We build Heat-RL based on this principle to reduce the limitation to utilize time-series anomaly detection. As of now Heat-RL has only been validated in time-series anomaly detection, in the future, we would like to extend Heat-RL to other online learning scenarios. Moreover, we would like to mention the concept of responsible AI, which guides us to integrate fairness, interpretability, privacy, security, and accountability when applying the Heat-RL algorithm to real-world applications.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2020YFB2103402).

References

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *NIPS 2011*, volume 24. Neural Information Processing Systems Foundation, 2011.

Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. 2012.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016.

Jordan Hochenbaum, Owen S Vallis, and Arun Kejariwal. Automatic anomaly detection in the cloud via statistical learning. *arXiv preprint arXiv:1704.07706*, 2017.

Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In *NeurIPS*, 2018.

Chengqiang Huang, Yulei Wu, Yuan Zuo, Ke Pei, and Geyong Min. Towards experienced anomaly detector through reinforcement learning. In *Proceedings of the Thirty-Second AAAI*, 2018.

Hadi S. Jomaa, Josif Grabocka, and Lars Schmidt-Thieme. Hyp-rl : Hyperparameter optimization by reinforcement learning. *CoRR*, abs/1906.11527, 2019.

Jay Kreps, Neha Narkhede, Jun Rao, et al. Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, volume 11, 2011.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1), 2012.

Wei Lu and Ali A Ghorbani. Network anomaly detection based on wavelet analysis. *EURASIP Journal on Advances in Signal Processing*, 2009.

Ajay Mahimkar, Zihui Ge, Jia Wang, Jennifer Yates, Yin Zhang, Joanne Emmons, Brian Huntley, and Mark Stockert. Rapid detection of maintenance induced changes in service performance. In *Proceedings of the Seventh Conference on emerging Networking Experiments and Technologies*, 2011.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*, 2016.

Laptev Nikolay, Amizadeh Saeed, and Billawala Youssef. A benchmark dataset for time series anomaly detection, 2015.

Min-hwan Oh and Garud Iyengar. Sequential anomaly detection using inverse reinforcement learning. In *Proceedings of the 25th ACM SIGKDD*, 2019.

Guansong Pang, Anton van den Hengel, Chunhua Shen, and Longbing Cao. Toward deep supervised anomaly detection: Reinforcement learning from partially labeled anomaly data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD*, 2019.

Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741, 2009.

Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouët. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD*, 2017.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *arXiv preprint arXiv:1206.2944*, 2012.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pp. 2171–2180. PMLR, 2015.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, 1999.

Owen Vallis, Jordan Hochenbaum, and Arun Kejariwal. A novel technique for long-term anomaly detection in the cloud. In *6th USENIX Workshop on Hot Topics in Cloud Computing, HotCloud '14*, 2014.

Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, Jie Chen, Zhaogang Wang, and Honglin Qiao. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web (WWW)*, 2018.

Daochen Zha, Kwei-Herng Lai, Mingyang Wan, and Xia Hu. Meta-aad: Active anomaly detection with deep reinforcement learning. In *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 771–780. IEEE, 2020.

Yin Zhang, Zihui Ge, Albert Greenberg, and Matthew Roughan. Network anomography. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, 2005.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations (ICLR)*, 2017.

A Hyper-parameter Space.

The value range of each parameter is decided based on its original definition. Specifically, the threshold of SR is set between 0.0 and 10.0 with step 0.2. Meanwhile, the threshold of normalized average path length in Iforest is set between 0.1 and 1.0 with step 0.01. The threshold of anomaly ratio of TwitterAD is set between 0 and 0.49 with step 0.05. The range of the detection window size has been set according to the time intervals of different time-series. In *KPI* dataset, the window size is in $[2881, 8641]$ with step 1440. The window size of *Production* dataset is in $[15, 85]$ with step 7. In *Artificial* dataset, the window size is in $[201, 1201]$ with step 100.