Journal of the Franklin Institute 361 (2024) 107199

Contents lists available at ScienceDirect

Journal of the Franklin Institute

journal homepage: www.elsevier.com/locate/fi

DeepQCD: An end-to-end deep learning approach to quickest change detection

Mehmet Necip Kurt [a,*], Jiaohao Zheng [b], Yasin Yilmaz [c], Xiaodong Wang [a]

[a] Columbia University, Department of Electrical Engineering, New York, NY 10027, USA
[b] Chinese Academy of Sciences, Shenzhen Institutes of Advanced Technology, Shenzhen, 518055, China
[c] University of South Florida, Department of Electrical Engineering, Tampa, FL 33620, USA

ARTICLE INFO

Keywords:
Quickest change detection
Deep learning
Temporal correlation
Transient change
Internet of things
Surveillance videos

ABSTRACT

This paper aims to generalize the quickest change detection (QCD) framework via a data-driven approach. To this end, a generic neural network architecture is proposed for the QCD task, composed of feature transformation, recurrent, and dense layers. The neural network is trained end-to-end to learn the change detection rule directly from data without needing the knowledge of probabilistic data models. Specifically, the feature transformation layers can perform a broad range of operations including feature extraction, scaling, and normalization. The recurrent layers keep an internal state summarizing the time-series data seen so far and update the state as new information comes in. Finally, the dense layers map the internal state into a decision statistic, defined as the posterior probability that a change has taken place. Comparisons with the existing model-based QCD algorithms demonstrate the power of the proposed data-driven approach, called DeepQCD, under several scenarios including transient changes and temporally correlated data streams. Experiments with real-world data illustrate superior performance of DeepQCD compared to state-of-the-art algorithms in real-time anomaly detection over surveillance videos and real-time attack detection over Internet of Things (IoT) networks.

1. Introduction

Suppose that a random process generates time-series data $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots$, where $\mathbf{x}_t \in \mathbb{R}^p$ is the observation made at time t and $p \geq 1$ is the data dimensionality. An abrupt change happens in the observed process at an unknown time τ, called the change-point, such that

$$\mathbf{x}_t \sim \begin{cases} f_0, & \text{if } t < \tau, \\ f_1, & \text{if } t \geq \tau, \end{cases} \tag{1}$$

where f_0 and f_1 denote the pre- and post-change probability density functions (PDFs) of $\mathbf{x}_t$, respectively. In the quickest change detection (QCD) framework, the objective is to develop a sequential procedure for detecting changes as quickly as possible while limiting the risk of false alarm.

In the QCD procedures, at each time step, a decision is made based on the available information on whether to stop and declare a change, or continue to acquire a further observation in the next time interval. The stopping time Γ is a random variable depending on the observations made so far. That is, $\{\Gamma = t\} \in \mathcal{F}_t, \forall t \geq 1$, where $\mathcal{F}_t = \sigma(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t)$ denotes the sigma-algebra generated by

* Corresponding author.
E-mail addresses: m.n.kurt@columbia.edu (M.N. Kurt), jh.zheng@siat.ac.cn (J. Zheng), yasiny@usf.edu (Y. Yilmaz), wangx@ee.columbia.edu (X. Wang).

https://doi.org/10.1016/j.jfranklin.2024.107199
Received 19 June 2023; Received in revised form 2 June 2024; Accepted 20 August 2024
Available online 24 August 2024

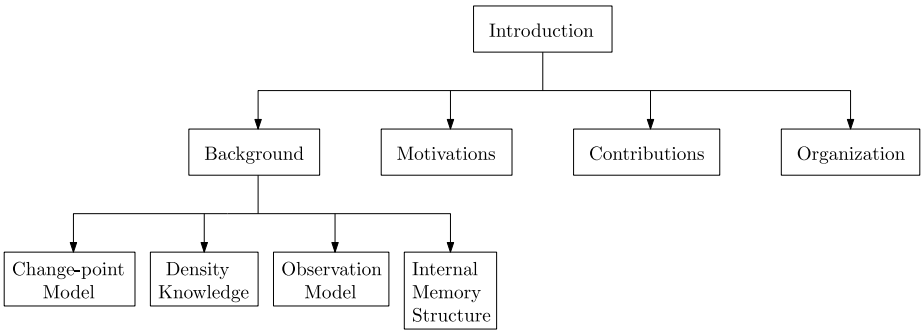

Fig. 1. Overview of the Introduction section.

the observations up to time t. The event $\{T < \tau\}$ corresponds to a false alarm event and $(T - \tau)^+$ is called the detection delay, where $(\cdot)^+ \triangleq \max\{0, \cdot\}$. While the goal is to minimize both the average detection delay and the frequency of false alarms, there is an inherent tradeoff between these two objectives. Ideally the stopping time is chosen to optimally balance the detection speed and the detection accuracy.

The QCD problem finds diverse applications in many fields such as critical infrastructure monitoring, industrial process control, environmental monitoring, cybersecurity, finance, video surveillance, biomedical signal processing, neuroscience, cognitive radio, onset of a disease outbreak, social networks, sensor networks, and military [1–6]. This paper aims to generalize the QCD framework via a novel data-driven solution approach, which is applicable to a variety of real-world settings.

In the remainder of this section, we first provide a background of the QCD framework. Subsequently, we discuss the motivations behind our work. We then outline our key contributions and present the organization of the paper. Fig. 1 illustrates a roadmap of the Introduction section.

1.1. Background

The QCD framework can be classified in terms of assumptions on the change-point model, density knowledge, observation model, and internal memory structure, as detailed next.

1.1.1. Change-point model

Based on the change-point model, two common problem formulations exist. First, in the Bayesian formulation, the change-point is random with a known prior distribution [7]. The change-point is usually assumed to be a geometric random variable, denoted with $\tau \sim \text{geo}(\rho)$, where

$$\pi_t \triangleq \text{P}(\tau = t) = \rho\,(1-\rho)^{t-1},\ t = 1, 2, 3, \ldots \tag{2}$$

The design goal is to minimize the average detection delay (ADD) subject to an upper bound on the probability of false alarm (PFA).

Second, in the minimax formulation, location of the change-point is deterministic but unknown. The design goal is to minimize the worst-case ADD subject to an upper bound on the average false alarm period (FAP). The Lorden's problem [8] and Pollak's problem [9] are two well-known problem formulations depending of the definition of the worst-case ADD.

1.1.2. Density knowledge

In terms of density knowledge, three cases are common: (i) known pre- and post-change PDFs, (ii) known pre-change PDF, unknown post-change PDF, and (iii) unknown pre- and post-change PDFs.

The first case has been extensively studied, especially for the independent and identically distributed (IID) observations, and the optimality properties of the well-known QCD algorithms have been shown [1,3]. In fact, many existing QCD algorithms such as Shiryaev [10] and cumulative sum (CUSUM) [11] require the computation of the likelihood ratio (LR), which is possible only if the pre- and post-change PDFs are known.

In the second case, if the post-change PDF belongs to a certain parametric family, either the unknown parameters can be estimated and incorporated into the change detection process, which is called the generalized likelihood ratio (GLR) approach [12,13] or a robust detection approach can be employed where the worst-case parameters are determined for the post-change PDF [14]. However, if the post-change PDF is completely unknown, observed data stream can be evaluated on whether it statistically fits or deviates from the pre-change PDF [15–18].

In the third case, if both the pre- and post-change PDFs belong to a parametric family, then either the unknown parameters can be estimated or the worst-case parameters can be determined. Alternatively, a nonparametric approach can be used if the data distributions are completely unknown [19,20]. Furthermore, neural network-based LR estimation was investigated in [21], enabling a two-step solution that leverages an existing QCD algorithm based on the LR estimates.

1.1.3. Observation model

There exist many time series models: IID [1], autoregressive (AR) [22], autoregressive moving average (ARMA) [23], state-space [24], hidden Markov model (HMM) [25,26], and so forth. Almost all prior work in the QCD framework studied the IID setting, for both the pre- and post-change observations. This is mainly because it gets more complicated (i.e., with increasing space and time complexity) to compute the LR as the observations get more correlated over time. The assumption of temporally independent observations greatly simplifies the design of QCD algorithms and enables low-complexity recursive procedures that are well suitable for real-time processing.

In general non-IID observation settings, asymptotically optimal model-based QCD algorithms are extended from their counterparts in the IID observation setting [3,12,27]. However, low-complexity recursive algorithm design is no longer possible in the non-IID settings.

1.1.4. Internal memory structure

In terms of utilizing the observation history in the decision making process, there exist two common approaches. First, window-limited QCD procedures restrict the internal memory to a fixed number of recent observations while ignoring the further past, see [28] for example. There is an inherent detection delay caused by the window size, which is itself a design parameter. Specifically, a larger window size leads to a larger ADD whereas a smaller window size ignores more information from the past. In the extreme case, the past observations are completely ignored and only the most recent observation is utilized, see the Shewhart test [29] for example. Second, fully-sequential QCD procedures utilize the entire observation history via keeping a summary of the observations made so far. The fully-sequential procedures are ideally preferred over the window-limited procedures if the relevant information can be well extracted from the observation history. However, to reduce memory requirements, existing fully-sequential QCD procedures usually make simplified assumptions such as temporally independent observations.

1.2. Motivations

Real-world data streams often have unknown properties. Hence, either the underlying probabilistic data model can be learned and a model-based QCD procedure is employed or the change detection rule can be directly learned from data. The former approach requires fitting a parametric model to the observed data and the estimation of unknown model parameters. This can be costly or even intractable for complex, temporally correlated, and high-dimensional data streams. Moreover, it is vulnerable to model mismatch as the real-world data may not fit into existing parametric models and even the lack of parametric models is common in high-dimensional settings [30]. Hence, in general, only an approximate model is learned, which may degrade the corresponding model-based QCD procedure. The latter approach is more direct, robust to model mismatch, and more widely applicable. Hence, model-free QCD algorithms are better suited for real-world data streams.

In [31], responding to changes in a potentially adversarial environment rapidly yet accurately is addressed as a problem faced by animal brains at every level from neurons to behavior. It is argued that neurons are optimized for tracking abrupt changes in the input spike statistics and moreover, intracellular dynamics of the spiking neurons, specifically the leaky-integrate-and-fire neurons, remarkably resemble the information accumulation and decision process of the Bayesian-optimal Shiryaev algorithm [10] in the QCD framework. It is also argued that neurons have a speed-accuracy tradeoff in their decision making process, just like the QCD procedures. These well motivate a new QCD algorithm based on the neural networks and the Shiryaev algorithm. Finally, end-to-end deep learning (DL) does not require explicit data modeling, feature engineering, or procedure design, and hence it enables a completely model-free QCD algorithm.

1.3. Contributions

We list our main contributions as follows:

- We propose a novel fully-sequential model-free QCD algorithm, called DeepQCD, which learns the change detection rule directly from observed raw data via neural networks.
- To the best of our knowledge, DeepQCD incorporates the entire QCD procedure through end-to-end deep learning for the first time in the literature.
- DeepQCD unifies the QCD framework. With sufficient training data, DeepQCD can be effective for various kinds of change-point models, data distributions, and observation models.
- DeepQCD achieves comparable or superior performance compared to the existing model-based QCD procedures, which are designed with complete statistical knowledge of the observed data stream.
- In challenging real-world applications, real-time video anomaly detection and cyber-attack detection over Internet of Things (IoT) networks, DeepQCD outperforms the state-of-the-art.

1.4. Organization

The remainder of the paper is organized as follows. Section 2 reviews the related work. Section 3 presents the generic QCD procedure, which is the backbone of the proposed solution approach. Section 4 describes the DeepQCD algorithm. Section 5 justifies DeepQCD through comparisons with the existing model-based QCD algorithms. Section 6 evaluates the performance of DeepQCD in real-world applications. Finally, Section 7 concludes the paper.

2. Related work

In the nonparametric detection, classical procedures are the goodness-of-fit tests such as the Kolmogrov–Smirnov test and the Pearson's chi-squared test [32], which are mainly designed for batch processing of univariate data, where their window-limited versions can be employed for real-time detection as well [15,16]. Moreover, for multivariate data streams, support vector machine (SVM)-based one-class classification algorithms [33,34], nearest neighbor graph-based algorithms [19,35,36], subspace-based algorithms [19,37], and information theoretic algorithms [38,39] have been proposed for real-time multivariate change and anomaly detection. Furthermore, in [21], the LR was estimated using neural networks, and the CUSUM algorithm was applied based on these estimates. Furthermore, in [21], the LR was estimated using neural networks, and the CUSUM algorithm was applied based on these estimates. While the approach in [19] focuses solely on LR estimation, DeepQCD offers an end-to-end deep learning solution for QCD.

A fundamental building block of DeepQCD is the recurrent memory cells. Hence, we next explain how the recurrent neural networks (RNNs) have been used in the relevant literature. The first approach is based on learning a pre-change prediction model [40–42], where the trained RNN predicts the future observations and an anomaly is declared if large prediction errors occur. The second approach builds a classifier on the entire sequence [43]. After observing the entire data stream, the trained RNN makes a binary decision on whether the data stream is normal or anomalous. In the third approach, an RNN-based autoencoder is trained to learn a pre-change reconstruction model [44,45], where an anomaly is declared if large reconstruction errors occur. In addition to these approaches, [46] introduces a specialized RNN architecture, called the pyramid RNN, incorporating wavelet layers. This design aims to enhance the detection of gradual and multi-scale changes.

3. Generic QCD procedure

In the QCD framework, the stopping time Γ is decided based on the information derived from the observation history. Let $\mathbf{s}_t$ be the internal state (i.e., memory) of the decision maker at time t, based on the observations made so far, that is, $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t\}$. Ideally, the internal state captures and summarizes all the relevant information from the observation history and as new observations are made, the internal state is updated accordingly. Let $\phi(\cdot,\cdot)$ be the state update function that processes the most recent observation $\mathbf{x}_t$ and updates the internal state, as given by

$$\mathbf{s}_t = \phi(\mathbf{x}_t, \mathbf{s}_{t-1}). \tag{3}$$

Based on the internal state, the decision maker either stops and declares a change, or continues to acquire a further observation in the next time interval. More particularly, the internal state $\mathbf{s}_t$ is mapped to a decision statistic d_t via a function $\omega(\cdot)$ as follows:

$$d_t = \omega(\mathbf{s}_t). \tag{4}$$

The decision is then made based on a threshold-crossing mechanism. In other words, the decision maker detects a change as soon as the decision statistic exceeds a predetermined threshold h. Hence, the stopping time is given by

$$\Gamma = \inf\{t : d_t \geq h\}. \tag{5}$$

This procedure (see Fig. 2) is common for all QCD algorithms. What makes algorithms different are the state update function $\phi(\cdot,\cdot)$ and the decision mapping $\omega(\cdot)$. In particular, the existing QCD algorithms characterize $\phi(\cdot,\cdot)$ and $\omega(\cdot)$ in special forms depending on the assumptions on change-point model, pre- and post-change PDFs, observation model, and internal memory structure (see Appendix for examples). In this paper, we consider the most general case where no assumptions are imposed on the observed data stream. Instead, we aim to learn both $\phi(\cdot,\cdot)$ and $\omega(\cdot)$ from data and thereby to achieve a completely data-driven QCD procedure.

4. Data-driven QCD via end-to-end deep learning

Suppose that a dataset is available including both pre-change and post-change samples. To obtain an effective data-driven QCD procedure, we propose to train a neural network composed of three components: feature transformation layers, recurrent layers, and dense layers (see Fig. 3). The purpose of this network architecture is to learn both the state update and the decision mapping functions without imposing statistical model assumptions on the observed data stream. As illustrated in Fig. 3, the feature transformation and the recurrent layers together correspond to the state update function $\phi(\cdot,\cdot)$, whereas the dense (i.e., fully connected) layers correspond to the decision mapping function $\omega(\cdot)$.

4.1. Feature transformation

The neural network may perform initial processing on each observation $\mathbf{x}_t$ to extract useful features and filter out noise, which is particularly necessary for complex and high-dimensional data streams. To ensure the most relevant and useful features are extracted, the feature transformation layers should be tailored to the specific type of observed data.

For instance, in the case of video data streams, convolutional layers are employed to effectively capture spatial patterns (see Section 6.1). For other types of data, the feature transformation may involve basic operations such as data scaling or normalization to standardize the inputs (see Section 6.2).

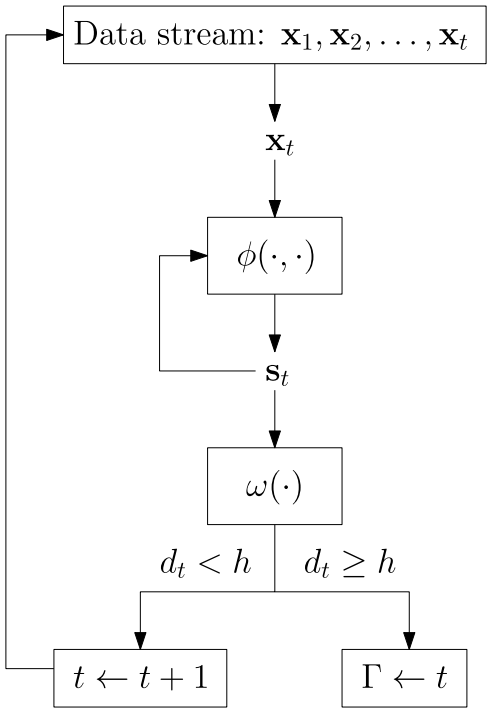

Fig. 2. Generic QCD procedure.

In some scenarios, particularly with low-dimensional data streams, feature transformation may not be necessary, and observations can be directly fed into the recurrent layers (see Section 5.1). This direct approach can be beneficial when the data is already clean and well-structured, allowing the model to learn the temporal dependencies without additional preprocessing.

Overall, feature transformation may involve a broad range of operations and it is a crucial component of the proposed neural network, enhancing the ability of the network to learn from diverse data types.

4.2. Building internal state

The recurrent layers function as the network memory, summarizing the data stream seen so far. At each time t, the internal state is updated after a new observation $\mathbf{x}_t$ is made. The recurrent layers are the most fundamental component of the proposed neural network architecture. This is because how effectively they capture the observation history has a critical impact on the QCD performance. In practice, advanced recurrent memory cells such as the long short-term memory (LSTM) [47] and gated recurrent unit (GRU) [48] can be used in the recurrent layers. LSTMs are more powerful to capture long-term dependencies in data streams but are computationally heavier due to higher number of parameters and gates. GRUs are simpler, faster, and often perform comparably, making them a good choice when computational efficiency is a priority [49,50]. Therefore, GRUs may be favored for processing high-dimensional data streams to maintain manageable computational complexity.

The RNNs encode the observation history into an internal state via sequential data processing, similar to how QCD algorithms build an internal state from sequential observations. Although, in principle, the latest RNN state captures all the past observations, remembering arbitrarily long history can be practically difficult [51]. Nevertheless, from the QCD perspective, recent observation history is more important since observations made after the change-point (i.e., $\{\mathbf{x}_t\}_{t\geq\tau}$) are more informative about the change event. Moreover, advanced RNN cells can make a selective use of the observation history [47,48]. Hence, recurrent layers can well learn and build the internal state of a QCD procedure.

4.3. Decision mapping

The recurrent layers are followed by dense layers that map the internal state $\mathbf{s}_t$ into a decision statistic d_t. Motivated by the Bayesian-optimal Shiryaev algorithm [10], decision statistic is defined as the posterior probability that a change has happened given all the observations made so far:

$$d_t = \mathrm{P}\left(t \geq \tau \,|\, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t\right). \tag{6}$$

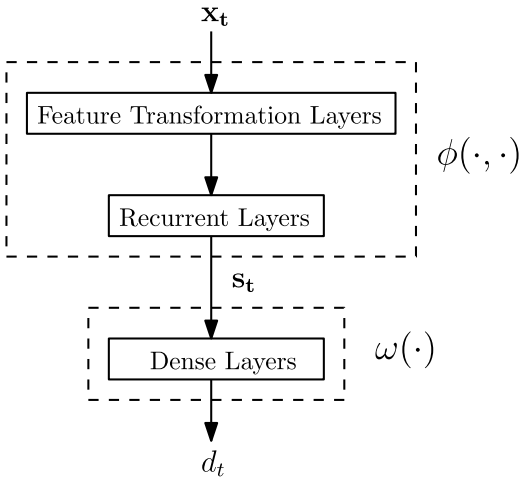

Fig. 3. Proposed generic neural network architecture for DeepQCD. At each time t, new observation $\mathbf{x}_t$ is processed and the internal state $\mathbf{s}_t$ is updated. The internal state is then mapped to the decision statistic d_t.

In the offline training phase, it is convenient to set the ground truth labels for the decision statistic. In the pre-change period, the ground truth label $\hat{d}_t$ is given by, see Eq. (6),

$$\hat{d}_t = 0, \ t < \tau, \tag{7}$$

whereas in the post-change period, the ground truth label is given by, see Eq. (6),

$$\hat{d}_t = 1, \ t \geq \tau. \tag{8}$$

4.4. DeepQCD algorithm

The DeepQCD algorithm comprises two phases: offline training and real-time detection. Let ϕ denote the parameters of the DeepQCD network. In the offline training phase (see Alg. 1), the network is trained end-to-end with the objective of minimizing the binary cross entropy loss, given by

$$L(\phi) \triangleq -\mathrm{E}_{\mathcal{D}} \left[\hat{d}_t \log(d_t) + (1 - \hat{d}_t) \log(1 - d_t) \right],$$

where d_t is a function of ϕ and $\mathcal{D}$ denotes the training dataset. At each training iteration, the network parameters ϕ are adjusted towards minimizing the loss function $L(\phi)$:

$$\phi \leftarrow \phi - \alpha \nabla_{\phi} L(\phi),$$

where α denotes the learning rate. The training process incorporates early stopping regularization, whereby training is halted when there is no further reduction observed in the binary cross-entropy loss on an independent validation dataset, denoted by $\mathcal{D}_{\mathrm{val}}$.

After the training phase is over, the network is used for the QCD task. Specifically, in the real-time detection phase (see Alg. 2), at each time t, a new observation $\mathbf{x}_t$ is input to the network and the network outputs the decision statistic d_t (see Fig. 3). The algorithm declares a change at the first time instant the decision statistic exceeds a predetermined threshold $h \in (0, 1)$:

$$\mathit{\Gamma} = \inf \left\{ t : d_t \geq h \right\}.$$

The test threshold h controls the speed-accuracy tradeoff in the decision-making process. A larger threshold value leads to a reduction in the false alarm rate (FAR). However, this comes at the cost of a higher average detection delay (ADD), as the algorithm requires stronger evidence on change before making a decision. Conversely, a smaller threshold value results in a lower ADD, enabling quicker detection of changes. However, this also leads to a higher FAR, as the algorithm becomes more prone to making false detections in noisy or uncertain conditions. Thus, the test threshold h is chosen based on the specific constraints of the application, balancing the need for prompt detection with the desire to minimize false alarms.

5. Numerical justification of DeepQCD

In this section, we consider several cases within the QCD framework and evaluate the existing (optimal) model-based QCD algorithms and the proposed data-driven DeepQCD algorithm through experiments. Our purpose is to justify the power of DeepQCD compared to the model-based algorithms.

Algorithm 1 DeepQCD: Offline Training

Inputs:

- Training data: $\mathcal{D} = \{(\mathbf{x}_t^i, \hat{d}_t^i),\ t = 1, 2, \dots, \mathrm{T},\ i = 1, 2, \dots, \mathrm{I}\}$
- Validation data: $\mathcal{D}_{\mathrm{val}} = \{(\mathbf{x}_t^i, \hat{d}_t^i),\ t = 1, 2, \dots, \mathrm{T},\ i = 1, 2, \dots, \mathrm{I}_{\mathrm{val}}\}$
- Learning rate: α
- Early stopping threshold: C

Outputs:

- Trained neural network parameters ϕ

Initialization:

Randomly initialize ϕ
Initialize the best validation loss: $L_{\mathrm{val}}^{*} \leftarrow \infty$
Initialize the training epoch number: $e \leftarrow 1$
Initialize the early stop counter: $c \leftarrow 0$

Training:

for each training epoch e **do**
  Update the network parameters: $\phi \leftarrow \phi - \alpha \nabla_{\phi} L(\phi)$
  Validation loss: $L_{\mathrm{val}}^{e} = -\mathrm{E}_{\mathcal{D}_{\mathrm{val}}} \left[\hat{d}_t \log(d_t) + (1 - \hat{d}_t) \log(1 - d_t)\right]$
  if $L_{\mathrm{val}}^{e} < L_{\mathrm{val}}^{*}$ **then**
    Update the best validation loss: $L_{\mathrm{val}}^{*} \leftarrow L_{\mathrm{val}}^{e}$
    Reset the early stop counter: $c \leftarrow 0$
  else
    Increment the early stop counter: $c \leftarrow c + 1$
    if $c \geq C$ **then**
      Terminate the training procedure
    end if
  end if
  Proceed to the next epoch: $e \leftarrow e + 1$
end for

Algorithm 2 DeepQCD: Real-Time Detection

Input:

- Observed time series data: $\{\mathbf{x}_t, t = 1, 2, 3, \dots\}$
- Trained neural network parameters ϕ

Output:

- Stopping time: Γ

Initialization:

Initialize time: $t \leftarrow 0$
Initialize the decision statistic: $d_0 \leftarrow 0$

Real-Time Detection:

while $d_t < h$ **do**
  $t \leftarrow t + 1$
  $\mathbf{x}_t$ is input to the neural network
  The neural networks outputs d_t
end while
Declare a change and stop the QCD procedure: $\Gamma \leftarrow t$

5.1. IID observation process

Consider a data stream described through Eq. (1) and let the observations be IID over time in both the pre- and post-change periods.

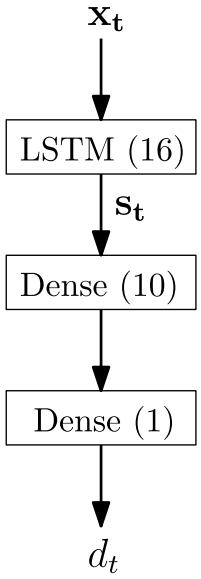

Fig. 4. DeepQCD network architecture for the experiments in Section 5.

5.1.1. IID Bayesian setting

In the IID Bayesian setting where $\tau \sim \text{geo}(\rho)$, see Eq. (2), we compare DeepQCD with the optimal Shiryaev algorithm [10]. As an example, suppose $\rho = 0.001$, the data dimension is $p = 7$, and the pre- and post-change PDFs are $f_0 \sim \mathcal{N}(\mathbf{0}_p, \mathbf{I}_p)$ and $f_1 \sim \mathcal{N}(\mathbf{1}_p, \mathbf{I}_p)$, respectively, where $\mathbf{0}_p$ denotes a p-dimensional vector of all zeros, $\mathbf{1}_p$ denotes a p-dimensional vector of all ones, and $\mathbf{I}_p$ denotes a $p \times p$ identity matrix.

In this setup, we generate a training dataset consisting of 3200 data streams where the length of each is 2000. That is,

$$D = \{(\mathbf{x}_t^i, \hat{d}_t^i),\ t = 1, 2, \dots, 2000,\ i = 1, 2, \dots, 3200\},$$

such that for the ith data stream (i.e., $\{\mathbf{x}_1^i, \mathbf{x}_2^i, \dots, \mathbf{x}_{2000}^i\}$), we have

$$\mathbf{x}_t^i \sim \begin{cases} f_0, & \text{if } t < \tau, \\ f_1, & \text{if } t \geq \tau, \end{cases}$$

where $\tau \sim \text{geo}(0.001)$.

For DeepQCD, we choose a neural network architecture as shown in Fig. 4, which is composed of recurrent and dense layers. In this case, for the observed 7-dimensional multivariate Gaussian data stream, we find that feature transformation is not necessary. The recurrent layers contain an LSTM cell with 16 units (i.e., $\mathbf{s}_t \in \mathbb{R}^{16}$) and there are two dense layers with 10 neurons and 1 neuron, respectively. We use the rectified linear unit (ReLU) and the sigmoid activation functions in the first and the second dense layers, respectively. Moreover, we use the adaptive moment (Adam) optimizer [52] with a learning rate of 0.001 during the offline training phase.

After the training phase is over, we evaluate DeepQCD and the Shiryaev algorithms in the same Bayesian setting. Fig. 5 presents the ADD-PFA tradeoff curves of both algorithms, demonstrating that DeepQCD converges to a near-optimal solution. Note that while DeepQCD is data-driven, the Shiryaev algorithm is designed with the complete statistical knowledge of the observed time-series data, including the IID observation model, the pre- and post-change PDFs f_0 and f_1, the geometric change-point model with a known parameter ρ.

5.1.2. IID minimax setting

In the IID minimax setting, the CUSUM algorithm is the optimal solution to the Lorden's problem [53] and an asymptotically optimal solution to the Pollak's problem [3]. Furthermore, the Shiryaev-Roberts (SR) algorithm is an asymptotically optimal solution to the Pollak's problem [3]. Note that since both CUSUM and SR algorithms have their optimality properties based on some pessimistic worst-case ADD measures [3,53], it is, in principle, possible to design better QCD algorithms in the minimax setting.

We evaluate DeepQCD, CUSUM, and SR algorithms using the same experimental setup as in the Bayesian setting, except for the change-point model. Firstly, assuming $\tau = \infty$ (i.e., no change at all), we compute the FAP of all three algorithms for various thresholds. Next, assuming $\tau = 1$, we compute the ADD of all algorithms for the same set of thresholds. Fig. 6 illustrates that DeepQCD achieves smaller ADD at the same FAP levels, and hence outperforms the CUSUM and SR algorithms. This implies that DeepQCD converges to a better solution than the existing minimax-optimal solutions.

In this experiment, we use the same DeepQCD network trained in the Bayesian setting. Hence, DeepQCD performs well even if there is a mismatch between the training data and the observed data regarding the change-point model.

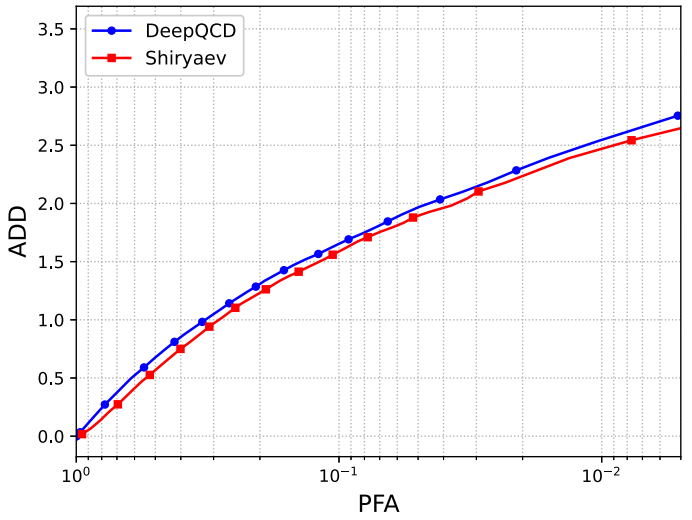

Fig. 5. ADD vs. PFA curves of DeepQCD and the Shiryaev algorithms in an IID Bayesian setting.

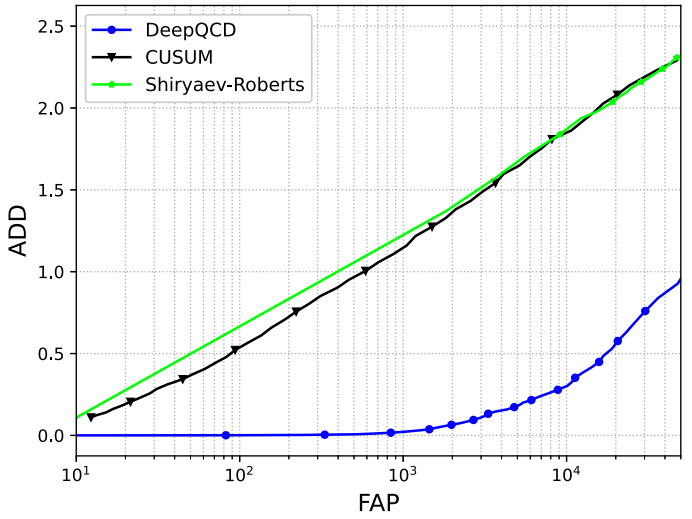

Fig. 6. ADD vs. FAP curves of DeepQCD, CUSUM, and SR algorithms in an IID minimax setting.

5.2. AR observation process

Consider a first-order AR observation process driven by a temporally independent Gaussian noise:

$$x_t = \begin{cases} \mu_0 + \lambda_0 x_{t-1} + \epsilon_t, & \text{if } t < \tau, \\ \mu_1 + \lambda_1 x_{t-1} + \epsilon_t, & \text{if } t \geq \tau, \end{cases} \tag{9}$$

where $\epsilon_t \sim \mathcal{N}(0,1)$ and $|\lambda_0|, |\lambda_1| \leq 1$. Our goal is to timely detect a change in the drift of the AR(1) observation process (i.e., from μ_0 to μ_1) and its correlation coefficient (i.e., from λ_0 to λ_1). Notice that the observations are correlated over time except for the special case where $\lambda_0 = \lambda_1 = 0$.

In [22], the CUSUM and SR algorithms are adapted to the AR(1) observation process. Specifically, the LR is derived as follows [22]:

$$\ell_t = e^{[x_t - 0.5(x_{t-1}(\lambda_0+\lambda_1)+\mu_0+\mu_1)][x_{t-1}(\lambda_1-\lambda_0)+\mu_1-\mu_0]}, \tag{10}$$

and the CUSUM and SR algorithms are accordingly extended from the IID setting to the AR(1) setting using the LR formula in Eq. (10). It is then shown that the modified CUSUM and SR algorithms are asymptotically minimax-optimal [22].

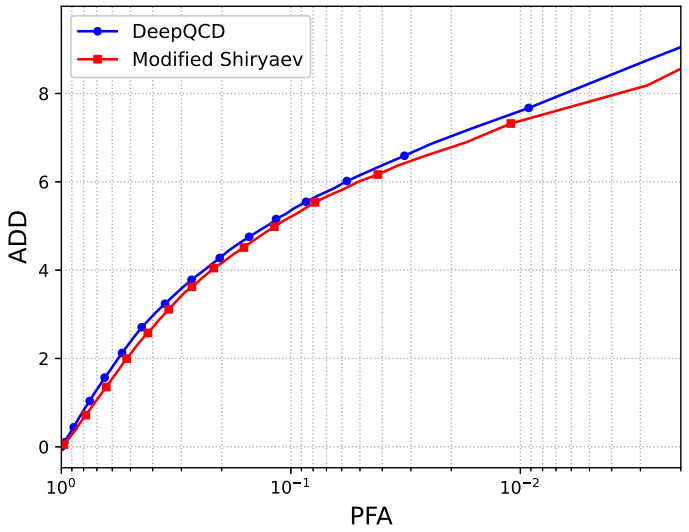

Fig. 7. ADD vs. PFA curves of DeepQCD and the modified Shiryaev algorithms in an AR(1) Bayesian setting.

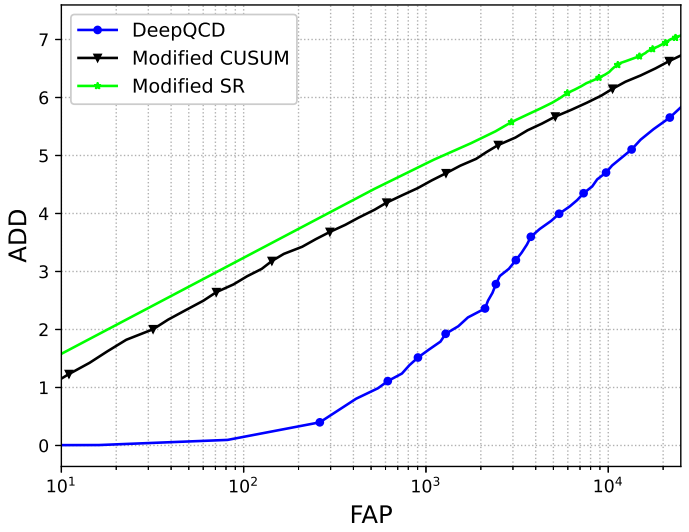

Fig. 8. ADD vs. FAP curves of DeepQCD, modified CUSUM, and modified SR algorithms in an AR(1) minimax setting.

In this experiment, we compare DeepQCD with the modified CUSUM and modified SR procedures in the AR(1) minimax setting. Furthermore, we extend the Shiryaev algorithm to the AR(1) observation setting using the LR formula given in Eq. (10), and then compare it with DeepQCD in the AR(1) Bayesian setting.

As an example, let $\mu_0 = 0$, $\lambda_0 = -0.3$, $\mu_1 = 1$, and $\lambda_1 = 0.2$. Then, for the AR(1) observation process in Eq. (9), we generate a training dataset consisting of 3200 independent data streams where each stream length is 2000. As before, to obtain various change-points in the training data, we use a random change-point $\tau \sim \text{geo}(0.001)$. Finally, we use the same DeepQCD network architecture in Fig. 4 and the same hyperparameters described in Section 5.1.

After the training phase is over, firstly in the Bayesian setting where $\tau \sim \text{geo}(0.001)$, we compare DeepQCD and the modified Shiryaev algorithms. Next, in the minimax setting, we compare DeepQCD, modified CUSUM, and modified SR algorithms, where we set $\tau = \infty$ for the FAP and $\tau = 1$ for the ADD calculations. Figs. 7 and 8 demonstrate that DeepQCD performs comparably to the modified Shiryaev algorithm in the Bayesian setting and outperforms the modified CUSUM and SR algorithms in the minimax setting, respectively.

5.3. Transient QCD

In the QCD framework, it is typically assumed that change is persistent, meaning that the post-change period is infinitely long after the change happens, see Eq. (1). However, in some applications such as radar, sonar, intrusion detection, and industrial

monitoring [28], the change might be transient. That is,

$$\mathbf{x}_t \sim \begin{cases} f_0, & \text{if } t < \tau_1, \\ f_1 \neq f_0, & \text{if } \tau_1 \leq t < \tau_2, \\ f_0, & \text{if } t \geq \tau_2, \end{cases}$$

where the change lasts for a finite time interval $[\tau_1, \tau_2)$.

In the transient QCD, a common objective is to detect the change before it disappears. Let Γ be the stopping time for detecting a change from f_0 to f_1. Then, $\{\tau_1 \leq \Gamma < \tau_2\}$, $\{\Gamma \geq \tau_2\}$, and $\{\Gamma < \tau_1\}$ are defined as the detection, missed detection, and the false alarm events, respectively. Moreover, let $\mathrm{P}_{\tau_1,\tau_2}$ be the probability measure when a change happens at time τ_1 and disappears at time τ_2. Then, $\mathrm{P}_{\tau_1,\tau_2}(\{\tau_1 \leq \Gamma < \tau_2\})$, $\mathrm{P}_{\tau_1,\tau_2}(\{\Gamma \geq \tau_2\})$, and $\mathrm{P}_{\tau_1,\tau_2}(\{\Gamma < \tau_1\})$ denote the probability of detection (PD), probability of missed detection (PMD), and the PFA, respectively. Note that these events are disjoint and mutually exclusive, and hence

$$\mathrm{PD} + \mathrm{PMD} + \mathrm{PFA} = 1.$$

The goal is to maximize the PD or equivalently minimize the PMD subject to an upper bound on the PFA. The transient QCD problem can then be written as follows:

$$\max_{\Gamma} \ \mathrm{PD} \ \ \text{subject to} \ \ \mathrm{PFA} \leq \alpha, \tag{11}$$

where $\alpha \in (0, 1)$ is a desired upper bound on the PFA.

Suppose that the observations are IID over time given τ_1 and τ_2. For the transient QCD problem given in Eq. (11), no optimal solutions exist except for a special case where only a single observation is made from f_1, that is, $\tau_2 = \tau_1 + 1$, for which the Shewhart algorithm is optimal in maximizing the worst-case PD [29]. Moreover, the well-known QCD algorithms such as the CUSUM and SR loose their optimality properties in the transient QCD as they are designed for the detection of persistent changes.

In [28], a suboptimal window-limited CUSUM algorithm is designed to minimize the worst-case PMD, assuming a certain transient change period $K \geq 1$ (i.e., $\tau_2 = \tau_1 + K$). In this algorithm, the most recent K observations are utilized. In terms of the generic QCD procedure (see Fig. 2), the algorithm can be written as follows:

$$\mathbf{s}_t = [\mathbf{x}_{t-K+1}, \mathbf{x}_{t-K+2}, \dots, \mathbf{x}_t],$$

$$d_t = \max_{t-K+1 \leq k \leq t} \sum_{i=k}^{t} \log\left(\frac{f_1(\mathbf{x}_i)}{f_0(\mathbf{x}_i)}\right),$$

$$\Gamma = \inf\left\{t : d_t \geq h\right\}.$$

For a transient change, two consecutive distribution changes occur: the first from f_0 to f_1 at time τ_1 and the second from f_1 back to f_0 at time τ_2. For example, we assume both change-points are geometric random variables such that $\tau_1 \sim \mathrm{geo}(0.001)$ and $\tau_2 - \tau_1 \sim \mathrm{geo}(0.002)$, and generate the training data as follows:

$$D = \{(\mathbf{x}_t^i, \hat{d}_t^i),\ t = 1, 2, \dots, 3000,\ i = 1, 2, \dots, 3200\},$$

and for the ith data stream,

$$\mathbf{x}_t^i \sim \begin{cases} f_0, & \text{if } t < \tau_1, \\ f_1, & \text{if } \tau_1 \leq t < \tau_2, \\ f_0, & \text{if } t \geq \tau_2, \end{cases}$$

where $f_0 = \mathcal{N}(0, 1)$ and $f_1 = \mathcal{N}(1, 1)$. In this case, the ground truth labels are given by

$$\hat{d}_t^i = \begin{cases} 0, & \text{if } t < \tau_1, \\ 1, & \text{if } \tau_1 \leq t < \tau_2, \\ 0, & \text{if } t \geq \tau_2. \end{cases}$$

We use the same DeepQCD network architecture and the same hyperparameters described in Section 5.1.

After the training phase is over, assuming that $\tau_1 = 1000$, $K = 25$, and $\tau_2 = \tau_1 + K$, we evaluate the performance of DeepQCD and the window-limited CUSUM algorithms. Fig. 9 presents the PD versus the PFA curves of both algorithms. Although the window-limited CUSUM algorithm is designed with the full statistical knowledge of the observed data stream including the pre- and post-change PDFs, IID observation model, and the transient change period K, it is still slightly outperformed by DeepQCD.

6. Experiments with real-world data

In this section, we experiment with two real-world QCD tasks: video anomaly detection and cyber-attack detection in IoT networks. The experiments are executed on an Ubuntu 18 LTS server with one Intel Xeon Gold 5118 CPU @2.30 GHz, 288 GB memory, and one NVIDIA RTX 2080Ti GPU.

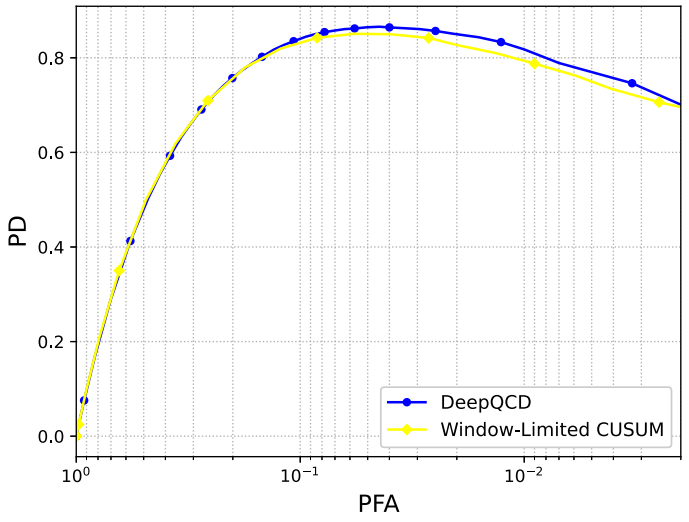

Fig. 9. PD vs. PFA curves of DeepQCD and the window-limited CUSUM algorithms for transient QCD.

Fig. 10. An example car accident. A bus drives on the sidewalk.

Fig. 11. Another example car accident. The white car hits the black car.

6.1. Real-time anomaly detection over surveillance videos

In [54,55], a large-scale video surveillance dataset is provided, including normal activities and 13 anomalies such as road accidents, burglary, and fighting, where each video frame can be resized to 240 × 320 pixels (i.e., $\mathbf{x}_t \in \mathbb{R}^{76800}$). In this dataset, ground truth labels are available at the video-level, that is, the videos are labeled as either normal or anomalous. For anomalous videos, the exact timing of the anomaly occurrence is not specified. However, DeepQCD training requires the frame-level ground truth, see Alg. 1. Moreover, performance evaluation, specifically ADD calculation, requires the knowledge of the change-point. For these reasons, we manually label the frames for a subset of the anomalous videos, specifically for the road accident videos. This is primarily due to the prevalence of road accident videos among the available dataset, which often exhibit clear change-points. Please refer to Figs. 10 and 11 for examples. Additionally, as the normal videos, we use vehicle traffic footage without anomalies.

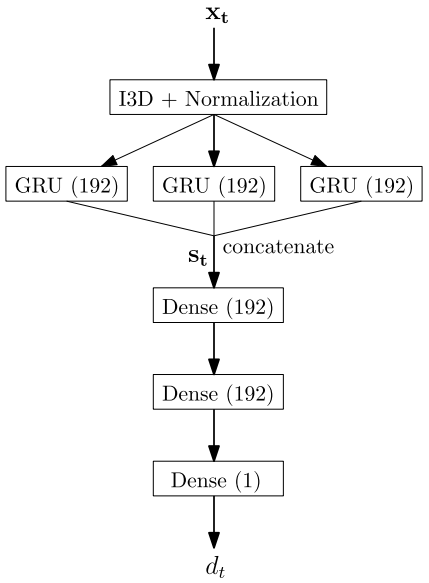

Fig. 12. DeepQCD network architecture for the real-time anomaly detection over surveillance videos.

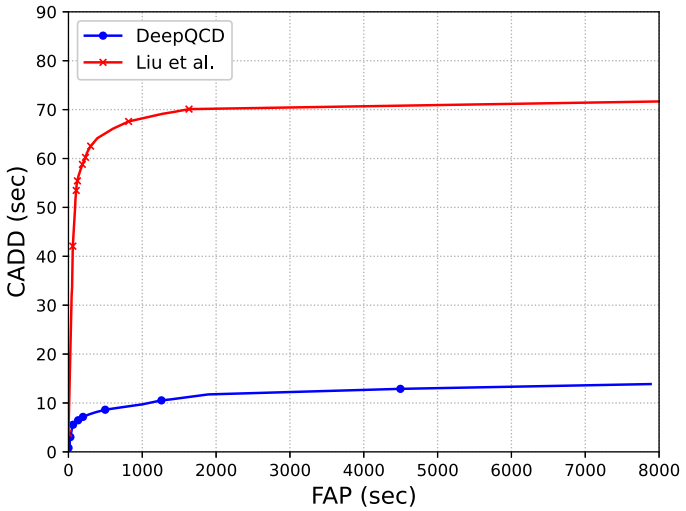

Fig. 13. CADD vs. FAP curves of DeepQCD and the benchmark algorithm from Liu et al. [60] for the real-time anomaly detection over surveillance videos.

For DeepQCD, we use the neural network architecture shown in Fig. 12. The feature transformation layers contain the Two-Stream Inflated 3D ConvNet (I3D) [56] pretrained over the ImageNet [57] dataset[1] (i.e., we use transfer learning). The I3D extracts 1024 features from the video frames. The extracted features are normalized and fed into the recurrent layers of the network, containing three GRUs with a single layer each. For the GRU layers, we apply the dropout [59] at the rate of 0.5. Finally, three dense layers are used to output the decision statistic. Before each dense layer, there is a batch normalization and before the last dense layer, we apply the dropout at the rate of 0.5. The first two dense layers use the ReLU activation, whereas the last dense layer uses the sigmoid activation function. Note that we did not extensively optimize the hyperparameters; therefore, further performance improvements are possible with hyperparameter tuning.

We use 320 normal and 141 anomalous videos in total. Among 320 normal videos, the shortest video has 203 frames, the longest video has 976,503 frames, and the average number of frames per video is $10,021.6$. Furthermore, among 141 anomalous videos, the shortest video has 137 frames, the longest video has 141,900 frames, and the average number of frames per video is $4,153.2$. We split this dataset into training, validation, and test sets with the training:validation:test ratios as 0.4:0.2:0.4. For training, we

[1] We obtain the pretrained I3D weights from [58] and choose "rgb_imagenet.pt".

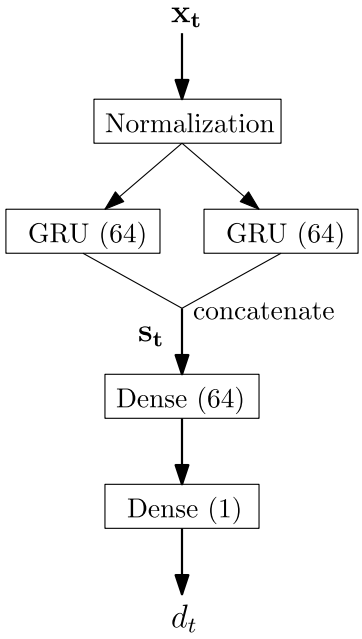

Fig. 14. DeepQCD network architecture for the real-time detection of IoT botnet attacks.

stitch the available videos to obtain longer data streams. Specifically, in each training data stream, we include both normal and anomalous videos such that the normal:anomalous ratio is approximately 0.75:0.25. We use the Adam optimizer with a learning rate of 0.0001 and a batch size of 32.

The offline training phase of DeepQCD lasts about 5.6 h in our experimental setup. After training, DeepQCD processes each frame in approximately 18.9 ms, equivalent to 53 frames per second (fps). This processing speed can be further increased with a more powerful GPU. Given that the frame rates of the videos used in our experiments range from 20 to 60 fps, the proposed DeepQCD algorithm is capable of processing streaming videos in real-time.

As the benchmark, we use the algorithm from Liu et al. [60], which predicts future frames and evaluates prediction errors for anomaly detection. We use the sequential version of this algorithm, where an anomaly is detected as soon as the prediction error exceeds a certain threshold. Let the conditional ADD (CADD) be defined as the ADD given that the anomaly is detected before the video ends. Fig. 13 presents the CADD vs. FAP curves of DeepQCD and Liu et al. [60] algorithms. DeepQCD outperforms Liu et al. [60] by achieving smaller CADD at the same FAP levels. In fact, future frame prediction is susceptible to errors, as not every novel event qualifies as an anomaly. Consequently, the benchmark algorithm [60] is prone to frequent false alarms. For instance, a flashing turn signal can be misinterpreted as an anomaly, triggering a false alarm.

6.2. Real-time detection of IoT botnet attacks

The network-based detection of IoT botnet attacks (N-BaIoT) dataset [61], sourced from the UCI Machine Learning Repository [62], offers network traffic statistics for an IoT network during both normal system operation and IoT botnet attacks. The network traffic statistics include time intervals between packet arrivals, packet sizes and counts, and so forth. Each data point is represented as a 115-dimensional vector (i.e., $\mathbf{x}_t \in \mathbb{R}^{115}$). In instances of botnet attacks, attackers inject malware into the IoT devices and control them to orchestrate larger scale attacks across network. For example, we examine a spam attack launched by the BASHLITE botnet [61] and observe the thermostat data for real-time attack detection.

For DeepQCD, we use the neural network architecture shown in Fig. 14. The feature transformation involves data normalization. The recurrent layers involve two GRUs with a single layer each, where we apply the dropout at the rate of 0.25. Finally, we include two dense layers: the first with 64 neurons, and the second with 1 neuron. We employ ReLU and sigmoid activation functions for the first and second dense layers, respectively. We apply batch normalization before the dense layers. Note that additional performance improvements can be achieved through hyperparameter tuning.

We split the available dataset such that training:validation:test ratios are 0.4:0.2:0.4. In the training phase, we uniformly sample from the normal samples before the change-point and the spam attack samples after the change-point. Each data stream has a length of 2048 and the change-point is uniform between 128 and 1920. During the training phase, we utilize the Adam optimizer with a learning rate of 0.0001 and a batch size of 256. The DeepQCD training lasts about 1.7 h. After training, DeepQCD processes each data point in approximately 6.9 ms, corresponding to a processing rate of 145 samples per second. While the data sampling frequency in the N-BaIoT dataset was not specified, DeepQCD processing time appears sufficiently low to handle streaming data

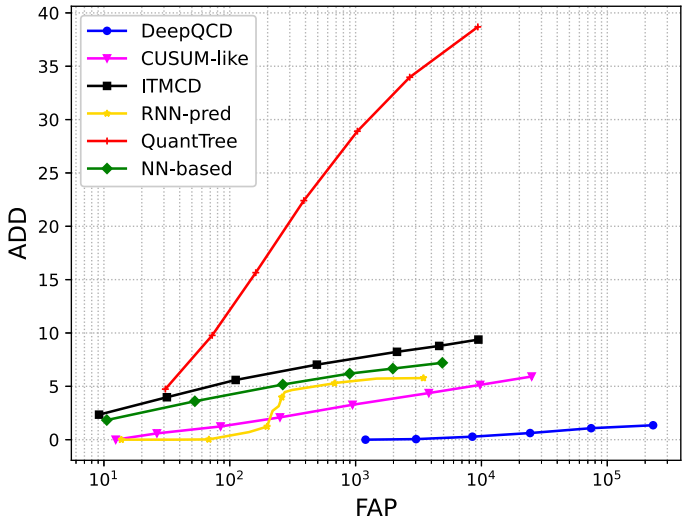

Fig. 15. ADD vs. FAP curves of DeepQCD and the benchmark algorithms for the real-time detection of an IoT botnet attack.

in real-time. Moreover, with a more powerful GPU, further reductions in processing time can be achieved for more stringent time requirements.

Despite the limited number of samples in the test dataset, to generate ADD vs. PFA performance curves, we simulate an infinite data stream by uniformly sampling from the normal dataset for the pre-change case and the spam attack dataset for the post-change case. For the ADD and the FAP calculations, we set $\tau = 1$ and $\tau = \infty$, respectively. As benchmark algorithms, we implement the principal component analysis (PCA)-based nonparametric CUSUM-like algorithm [19], QuantTree algorithm [63], Information Theoretic Multivariate Change Detection (ITMCD) algorithm [38], nearest neighbor graph-based algorithm [35], and RNN-based algorithm utilizing an observation prediction model (referred to as "RNN-pred") similar to [40–42]. While each benchmark algorithm is based on manually designed procedures, DeepQCD represents an end-to-end trained data-driven approach. Fig. 15 illustrates that DeepQCD achieves significantly smaller ADD for the same FAP levels compared to all benchmark algorithms. Note that the decision threshold of DeepQCD can be calibrated using the ADD-FAP tradeoff curve, considering application constraints such as the minimum tolerable FAP or maximum tolerable ADD.

7. Concluding remarks

We have proposed a novel data-driven QCD algorithm, called DeepQCD, based on an end-to-end trained generic neural network architecture. The neural network discovers the change detection rule directly from the raw observations without requiring any prior knowledge on the statistical model of the observed time-series data or manual design of the QCD procedure. Specifically, the feature transformation layers perform a broad range of operations including feature extraction, scaling, and normalization. The recurrent layers sequentially build an internal state representation, summarizing the observation history for the QCD task. Finally, the dense layers map the internal state into the decision statistic, defined as the posterior probability that change has taken place given the observations made so far at a given time. We have demonstrated that DeepQCD performs comparably to, or even outperforms, existing optimal model-based QCD algorithms. Furthermore, experiments with real-world data have illustrated that DeepQCD outperforms the state-of-the-art in real-time video anomaly detection as well as real-time attack detection in IoT networks.

CRediT authorship contribution statement

Mehmet Necip Kurt: Conceptualization, Data curation, Methodology, Software, Writing – original draft, Writing – review & editing. **Jiaohao Zheng:** Data curation, Software. **Yasin Yilmaz:** Funding acquisition, Supervision, Writing – review & editing. **Xiaodong Wang:** Funding acquisition, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the U.S. National Science Foundation (NSF) [grant numbers ECCS-2040500 and ECCS-2040572].

Appendix. Existing QCD algorithms as a special case of the generic QCD procedure

We briefly explain below how a few existing QCD algorithms fit into the generic QCD procedure (see Fig. 2).

A.1. CUSUM algorithm

In the CUSUM algorithm, the log-likelihood ratio (LLR) is considered as the statistical evidence for change at a time and the accumulation of LLRs over time correspond to the internal state of the algorithm. In particular, let ℓ_t be the LR at time t, given by

$$\ell_t \triangleq \frac{f_1(\mathbf{x}_t)}{f_0(\mathbf{x}_t)}. \tag{A.1}$$

Assuming that observations are independent over time, the state is updated recursively as follows:

$$s_t = \max\left\{0, s_{t-1} + \log(\ell_t)\right\}, \tag{A.2}$$

where $s_0 = 0$. The decision statistic is identical to the state, that is,

$$d_t = s_t, \tag{A.3}$$

and hence the decision mapping $\omega(\cdot)$ is an identity function. The change is declared at the first time the accumulated statistical evidence is reliably high:

$$\Gamma = \inf\{t : d_t \geq h\}. \tag{A.4}$$

Notice that the CUSUM update given in Eq. (A.2) is a special form of the generic state update in Eq. (3), and the CUSUM decision mapping given in Eq. (A.3) is a special form of the generic mapping in Eq. (4). Moreover, the CUSUM stopping time given in Eq. (A.4) is identical to Eq. (5).

A.2. Shiryaev algorithm

In the Shiryaev algorithm, the state corresponds to the probability that change has taken place given the observations made so far, that is,

$$s_t = \mathrm{P}\left(t \geq \tau \,|\, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t\right). \tag{A.5}$$

The Bayesian formulation assumes $\tau \sim \mathrm{geo}(\rho)$, see Eq. (2). Moreover, assuming that the observations are independent over time, the following recursive state update rule is employed:

$$s_t = \frac{\tilde{s}_{t-1}\ell_t}{\tilde{s}_{t-1}\ell_t + (1 - \tilde{s}_{t-1})}, \tag{A.6}$$

where

$$\tilde{s}_t \triangleq s_t + \rho(1 - s_t),$$

and $s_0 = 0$. The decision statistic is identical to the state,

$$d_t = s_t, \tag{A.7}$$

and the change is declared as soon as the posterior probability of having change exceeds the test threshold:

$$\Gamma = \inf\{t : d_t \geq h\}, \tag{A.8}$$

where $h \in (0, 1)$. Notice that the Shiryaev state update given in Eq. (A.6) is a special form of Eq. (3). Similarly, the decision mapping given in Eq. (A.7) and the stopping time given in Eq. (A.8) both fit into the generic QCD procedure.

A.3. Shiryaev-Roberts (SR) algorithm

The SR procedure is obtained from the Shiryaev algorithm in a special regime where $\rho \to 0$, that is, the change is a rare event and the change-point τ is uniform over time. The internal state of the SR procedure is updated recursively as follows:

$$s_t = (1 + s_{t-1})\ell_t, \tag{A.9}$$

where $s_0 = 0$. As before, the decision statistic is the same with the internal state,

$$d_t = s_t, \tag{A.10}$$

and the stopping time is given by

$$\Gamma = \inf\{t : d_t \geq h\}. \tag{A.11}$$

The SR procedure summarized through Eqs. (A.9)–(A.11) also fits into the generic QCD procedure.

A.4. Window-limited goodness-of-fit tests

In the CUSUM, Shiryaev, and SR algorithms, the state is well aligned with the QCD task, memory requirement is low, and the state is recursively updated with low complexity. However, these procedures require that both the pre- and post-change PDFs are known so that the LR can be computed, and the observations are independent over time. In some cases, such assumptions may not hold and alternative solutions are needed. For example, if the pre-change PDF is known but the post-change PDF is completely unknown, window-limited goodness-of-fit tests [15,16] can be used to determine if the observed data stream fits or deviates from the pre-change PDF. In this case, let a sliding-window state be formed with the most recent $K \geq 1$ observations, given by

$$\mathbf{s}_t = [\mathbf{x}_{t-K+1}, \mathbf{x}_{t-K+2}, \dots, \mathbf{x}_t]. \tag{A.12}$$

At each time t, the goodness-of-fit test analyzes the sliding-window state to make a decision.

For example, the sliding-window chi-squared test [15] considers an IID univariate data stream $x_1, x_2, x_3, \dots$ with a known pre-change PDF f_0. It divides the domain of f_0 into L disjoint and mutually exclusive intervals $I_1, I_2, \dots, I_L$ such that $p_1 = \mathrm{P}(x_t \in I_1)$, $p_2 = \mathrm{P}(x_t \in I_2)$, $\dots$, $p_L = \mathrm{P}(x_t \in I_L)$ correspond to the probabilities that observations belong to each interval in the pre-change period, where $\sum_{i=1}^{L} p_i = 1$. Then, for any given state, expected number of observations in the predetermined intervals are computed by Kp_1, Kp_2, $\dots$, Kp_L. Moreover, at a given time t, let the number of observations falling into each interval be counted as $N_{1,t}$, $N_{2,t}$, $\dots$, $N_{L,t}$, where $\sum_{i=1}^{L} N_{i,t} = K$. The decision statistic is then computed by

$$d_t = \sum_{i=1}^{L} \frac{(N_{i,t} - Kp_i)^2}{Kp_i}. \tag{A.13}$$

The stopping time of this procedure is given by

$$\Gamma = \inf\left\{t : d_t \geq h\right\}.$$

The window-limited goodness-of-fit tests can also be expressed within the generic QCD procedure through a sliding-window internal state as given in Eq. (A.12) and computation of the decision statistic as given in Eq. (A.13) for the chi-squared test, for example. For the sliding-window state, the state update is performed at time t by simply removing the oldest observation $\mathbf{x}_{t-K}$ and adding the most recent observation $\mathbf{x}_t$.

A.5. Shewhart test

In the context of memory usage, an extreme scenario involves conducting QCD solely based on the most recent observation, while ignoring all past observations. In this setting, the classical QCD procedure is the Shewhart test [29]. Assuming the pre- and post-change PDFs are known, the Shewhart test computes the LLR and compares it with a certain threshold. Here, the internal state corresponds to the most recent observation, see Eq. (A.12) when $K = 1$, that is,

$$\mathbf{s}_t = \mathbf{x}_t, \tag{A.14}$$

the decision statistic is the LLR, given by

$$d_t = \log\left(\frac{f_1(\mathbf{s}_t)}{f_0(\mathbf{s}_t)}\right). \tag{A.15}$$

A change is declared whenever the LLR exceeds a certain threshold:

$$\Gamma = \inf\left\{t : d_t \geq h\right\}. \tag{A.16}$$

The Shewhart test described through Eqs. (A.14)–(A.16) also fits into the generic QCD procedure.

References

[1] M. Basseville, I.V. Nikiforov, Detection of Abrupt Changes: Theory and Application, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.
[2] H.V. Poor, O. Hadjiliadis, Quickest Detection, Cambridge University Press, 2008.
[3] V.V. Veeravalli, T. Banerjee, Quickest change detection, in: Academic Press Library in Signal Processing, Vol. 3, Elsevier, 2014, pp. 209–255.
[4] B. Moussakhani, J. Flåm, T. Ramstad, I. Balasingham, On change detection in a Kalman filter based tracking problem, Signal Process. 105 (2014) 268–276.
[5] M.N. Kurt, X. Wang, Multisensor sequential change detection with unknown change propagation pattern, IEEE Trans. Aerosp. Electron. Syst. 55 (3) (2019) 1498–1518, http://dx.doi.org/10.1109/TAES.2018.2873067.
[6] N. Ilić, S.S. Stanković, M.S. Stanković, K.H. Johansson, Consensus based distributed change detection using generalized likelihood ratio methodology, Signal Process. 92 (7) (2012) 1715–1728.
[7] G.V. Moustakides, Sequential change detection revisited, Ann. Statist. 36 (2) (2008) 787–807, http://dx.doi.org/10.1214/009053607000000938.
[8] G. Lorden, Procedures for reacting to a change in distribution, Ann. Math. Stat. 42 (6) (1971) 1897–1908, http://dx.doi.org/10.1214/aoms/1177693055.
[9] M. Pollak, Optimal detection of a change in distribution, Ann. Statist. 13 (1) (1985) 206–227.
[10] A.N. Shiryaev, Optimal Stopping Rules, Springer-Verlag, NY, 1978.
[11] E.S. Page, Continuous inspection schemes, Biometrika 41 (1954) 100–115.
[12] T.L. Lai, Information bounds and quick detection of parameter changes in stochastic systems, IEEE Trans. Inform. Theory 44 (7) (1998) 2917–2929.
[13] M.N. Kurt, Y. Yilmaz, X. Wang, Distributed quickest detection of cyber-attacks in smart grid, IEEE Trans. Inf. Forensics Secur. 13 (8) (2018) 2015–2030, http://dx.doi.org/10.1109/TIFS.2018.2800908.
[14] J. Unnikrishnan, V.V. Veeravalli, S.P. Meyn, Minimax robust quickest change detection, IEEE Trans. Inform. Theory 57 (3) (2011) 1604–1614.
[15] M.N. Kurt, Y. Yilmaz, X. Wang, Real-time detection of hybrid and stealthy cyber-attacks in smart grid, IEEE Trans. Inf. Forensics Secur. 14 (2) (2019) 498–513, http://dx.doi.org/10.1109/TIFS.2018.2854745.
[16] A.L. Toledo, X. Wang, Robust detection of selfish misbehavior in wireless networks, IEEE J. Sel. Areas Commun. 25 (6) (2007) 1124–1134.
[17] M.N. Kurt, Y. Yilmaz, X. Wang, Secure distributed dynamic state estimation in wide-area smart grids, IEEE Trans. Inf. Forensics Secur. 15 (2020) 800–815, http://dx.doi.org/10.1109/TIFS.2019.2928207.
[18] M.N. Kurt, O. Ogundijo, C. Li, X. Wang, Online cyber-attack detection in smart grid: A reinforcement learning approach, IEEE Trans. Smart Grid 10 (5) (2019) 5174–5185.
[19] M.N. Kurt, Y. Yilmaz, X. Wang, Real-time nonparametric anomaly detection in high-dimensional settings, IEEE Trans. Pattern Anal. Mach. Intell. 43 (7) (2021) 2463–2479.
[20] M.N. Kurt, Y. Yilmaz, X. Wang, Sequential model-free anomaly detection for big data streams, in: 2019 57th Annual Allerton Conference on Communication, Control, and Computing, Allerton, 2019, pp. 421–425.
[21] G.V. Moustakides, K. Basioti, Training neural networks for likelihood/density ratio estimation, 2019, arXiv preprint arXiv:1911.00405.
[22] A.S. Polunchenko, V. Raghavan, Comparative performance analysis of the cumulative sum chart and the shiryaev-roberts procedure for detecting changes in autocorrelated data, Appl. Stoch. Models Bus. Ind. 34 (6) (2018) 922–948.
[23] G.V. Moustakides, A. Benveniste, Detecting changes in the AR parameters of a nonstationary ARMA process, Stochastics 16 (1–2) (1986) 137–155.
[24] B. Brodsky, Sequential change-point detection in state-space models, Sequential Anal. 31 (2) (2012) 145–171.
[25] G.V. Moustakides, Detecting changes in hidden Markov models, in: 2019 IEEE International Symposium on Information Theory, ISIT, 2019, pp. 2394–2398.
[26] C. Fuh, A.G. Tartakovsky, Asymptotic Bayesian theory of quickest change detection for hidden Markov models, IEEE Trans. Inform. Theory 65 (1) (2019) 511–529.
[27] A.G. Tartakovsky, V.V. Veeravalli, General asymptotic Bayesian theory of quickest change detection, Theory Probab. Appl. 49 (3) (2005) 458–497.
[28] B.K. Guépié, L. Fillatre, I. Nikiforov, Sequential detection of transient changes, Sequential Anal. 31 (4) (2012) 528–547.
[29] G.V. Moustakides, Multiple optimality properties of the Shewhart test, Sequential Anal. 33 (3) (2014) 318–344.
[30] R. Laxhammar, G. Falkman, Online learning and sequential anomaly detection in trajectories, IEEE Trans. Pattern Anal. Mach. Intell. 36 (6) (2014) 1158–1173.
[31] J.Y. Angela, Optimal change-detection and spiking neurons, in: Advances in Neural Information Processing Systems, 2007, pp. 1545–1552.
[32] M.A. Stephens, EDF statistics for goodness of fit and some comparisons, J. Amer. Statist. Assoc. 69 (347) (1974) 730–737.
[33] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Neural Comput. 13 (7) (2001) 1443–1471.
[34] M. Wu, J. Ye, A small sphere and large margin approach for novelty detection using training data with outliers, IEEE Trans. Pattern Anal. Mach. Intell. 31 (11) (2009) 2088–2092, http://dx.doi.org/10.1109/TPAMI.2009.24.
[35] H. Chen, Sequential change-point detection based on nearest neighbors, Ann. Statist. 47 (3) (2019) 1381–1407, http://dx.doi.org/10.1214/18-AOS1718.
[36] K. Srichanran, A.O. Hero, Efficient anomaly detection using bipartite k-NN graphs, in: Advances in Neural Information Processing Systems, 2011, pp. 478–486.
[37] A.A. Qahtan, B. Alharbi, S. Wang, X. Zhang, A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 935–944.
[38] L. Faivishevsky, Information theoretic multivariate change detection for multisensory information processing in internet of things, in: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2016, pp. 6250–6254.
[39] T. Dasu, S. Krishnan, S. Venkatasubramanian, K. Yi, An information-theoretic approach to detecting changes in multi-dimensional data streams, in: Proc. Symp. on the Interface of Statistics, Computing Science, and Applications, Citeseer, 2006.
[40] Y.-L. Kong, Q. Huang, C. Wang, J. Chen, J. Chen, D. He, Long short-term memory neural networks for online disturbance detection in satellite image time series, Remote Sens. 10 (3) (2018) 452.
[41] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, Long short term memory networks for anomaly detection in time series, in: Proceedings, Vol. 89, Presses universitaires de Louvain, 2015.
[42] L. Bontemps, J. McDermott, N.-A. Le-Khac, et al., Collective anomaly detection based on long short-term memory recurrent neural networks, in: International Conference on Future Data and Security Engineering, Springer, 2016, pp. 141–152.
[43] C. Yin, Y. Zhu, J. Fei, X. He, A deep learning approach for intrusion detection using recurrent neural networks, IEEE Access 5 (2017) 21954–21961.
[44] D. Park, Y. Hoshi, C.C. Kemp, A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder, IEEE Robot. Autom. Lett. 3 (3) (2018) 1544–1551.
[45] E. Marchi, F. Vesperini, S. Squartini, B. Schuller, Deep recurrent neural network-based autoencoders for acoustic novelty detection, Comput. Intell. Neurosci. 2017 (2017).
[46] Z. Ebrahimzadeh, M. Zheng, S. Karakas, S. Kleinberg, Deep learning for multi-scale changepoint detection in multivariate time series, 2019, arXiv preprint arXiv:1905.06913.
[47] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[48] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, 2014, arXiv preprint arXiv:1406.1078.
[49] R. Cahuantzi, X. Chen, S. Güttel, A comparison of LSTM and GRU networks for learning symbolic sequences, in: Science and Information Conference, Springer, 2023, pp. 771–785.
[50] R. Jozefowicz, W. Zaremba, I. Sutskever, An empirical exploration of recurrent network architectures, in: International Conference on Machine Learning, PMLR, 2015, pp. 2342–2350.
[51] A.B. Dieng, C. Wang, J. Gao, J. Paisley, TopicRNN: A recurrent neural network with long-range semantic dependency, in: ICLR 2017 : International Conference on Learning Representations 2017, 2017.
[52] D.P. Kingma, J.L. Ba, Adam: A method for stochastic optimization, in: ICLR 2015 : International Conference on Learning Representations 2015, 2015.
[53] G.V. Moustakides, Optimal stopping times for detecting changes in distributions, Ann. Statist. 14 (4) (1986) 1379–1387, http://dx.doi.org/10.1214/aos/1176350164.
[54] W. Sultani, C. Chen, M. Shah, Real-world anomaly detection in surveillance videos, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6479–6488.
[55] Video anomaly detection dataset. URL https://webpages.uncc.edu/cchen62/dataset.html.
[56] J. Carreira, A. Zisserman, Quo vadis, action recognition? A new model and the kinetics dataset, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2017, pp. 4724–4733.
[57] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database, in: CVPR09, 2009.
[58] Pretrained I3D network. URL https://github.com/piergiaj/pytorch-i3d/tree/master/models.
[59] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958.
[60] W. Liu, W. Luo, D. Lian, S. Gao, Future frame prediction for anomaly detection–a new baseline, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6536–6545.
[61] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, D. Breitenbacher, A. Shabtai, Y. Elovici, N-baIoT: Network-based detection of IoT botnet attacks using deep autoencoders, 2018, arXiv preprint arXiv:1805.03409.
[62] D. Dheeru, E. Karra Taniskidou, UCI machine learning repository, 2017, URL http://archive.ics.uci.edu/ml.
[63] G. Boracchi, D. Carrera, C. Cervellera, D. Macciò, QuantTree: Histograms for change detection in multivariate data streams, in: ICML, 2018.