July 2020

METHOD AND SYSTEM FOR ANOMALY DETECTION IN LARGE-SCALE NETWORKS

Fang Chen

John Garrett

Dave Zacks

Vladimir Yashin

METHOD AND SYSTEM FOR ANOMALY DETECTION IN LARGE-SCALE NETWORKS

AUTHORS:
Fang Chen
John Garrett
Dave Zacks
Vladimir Yashin

ABSTRACT

Effectively spotting anomalies in network or application operations can be challenging in very large, complex networks, making it difficult to be alerted to their presence in order to take action to remediate such anomalies. Proper anomaly detection is impeded by too much data from too many disparate sources, which may manifest in too many different ways on various network devices ("the curse of dimensionality" familiar to many machine learning (ML) practitioners).

Proposed herein is a novel implementation of a Long Short-Term Memory based Variational Autoencoder (LSTM-VAE) to detect such anomalies, and an associated visualization technique to display them to the network manager for subsequent remediation. By so doing, the described technique provide a novel method of anomaly detection in large, complex networks in a way not otherwise possible.

DETAILED DESCRIPTION

A typical modern network can have more than 30,000 devices. For some larger scale networks, there can be more than 100,000 devices. The anomalies in a network can lead to disruptive effects in network environments. Further, the number of non-impacting, false positive anomalies can mask disruptive anomalies in the noise. To monitor a network, the existing anomaly detection (AD) approach requires monitoring activities for each device and each telemetry in real time. Previously, this kind of monitoring was performed with human involvement, e.g., looking into dashboards and interacting with users. This type of monitoring suffers from low efficiency and high cost. With powerful artificial intelligence (AI) and machine learning, monitoring without human involvement is now possible. Systems often will deploy separate machine learning engines for different devices and different variables (e.g., CPU, memory, interfaces, etc.). This approach is not

feasible for large-scale networks: there can be ~10,000 variables (CPU, memory, interfaces, etc.) for each device. In this case, 30,000*10,000 different ML solutions could potentially exist. This number of solutions is infeasible for deployment and operation/execution. An efficient system that is lean, easy to monitor, track, and operate is needed. The system needs to be able to identify both new and known (based on prior learning of this same system) anomalies while reducing false alarms introduced by device status changes such as software updates, system reboots, and dynamic network environment changes.

Moreover, no ML solutions are known that address the correlation of multiple anomalies at the device level using telemetry time series data. Consider as an example the malicious Distributed Denial of Services (DDoS) attacks. The network element being targeted is attacked using many machines that send similar IP packets with strong time synchronicity to drain resources at the victim end-point. In this case, investigating one device and one telemetry will not raise an alert.

In another example, a common network loop condition will cause anomalous counters across many variables per device throughout the loop domain. Each device will show anomalies, and the anomalies will show in variables from the data path of the loop as well as the resource variables of individual devices.

While these are known conditions, the problem is further exacerbated because different device types react differently to the same condition. In the cases of DDoS or loop, some devices have CPU or control plane protection, while others do not. Some devices will display high resource utilization for CPU or bandwidth, while others will display anomalous counters for CPU or bandwidth protection mechanisms. Learning how a device manifests anomalies to adverse conditions is necessary to identify which type of behavior group the device falls into.

When displaying time series telemetry anomalies for human consumption, one can observe anomalies on multiple telemetry streams or multiple devices happened at the same time or in series. In order to do this, humans must be given the right set of anomalous streams displayed in the right sequence on a common timeline (assuming the behavior group is known). Only then can a human observer identify a root cause. But humans are not able to build and view all possible combinations, permutations, and behavior groups of

thousands of potential telemetry data streams across hundreds of devices in the network coverage domain.

The proposed novel anomaly detection system includes the following innovations:

1. a proposed machine learning model architecture for clustering network devices into behavior groups; and
2. a novel, unsupervised deep learning algorithm structure for multivariate anomaly detection, the output of which will provide the following:
   - one ML model for each device or multiple devices;
   - one anomaly score for each device;
   - one anomaly score for the entire network; and
   - a normality band of each telemetry of interest for each time *t*.

This approach is easy for dashboarding and visualization and does not require any labeled data given the fact that data labeling can be extremely expensive. Combining features 1 and 2 above, the complexity of the AD solution can be substantially reduced, with the following results:

1. reduction of time for training and inference;
2. reduction of cost for system deployment; easy system monitoring and maintenance;
3. explainable key performance indicators (KPI) on the device level and network level; and
4. reduction in false alarms.

Fig. 1 shows an overview of the anomaly detection system:

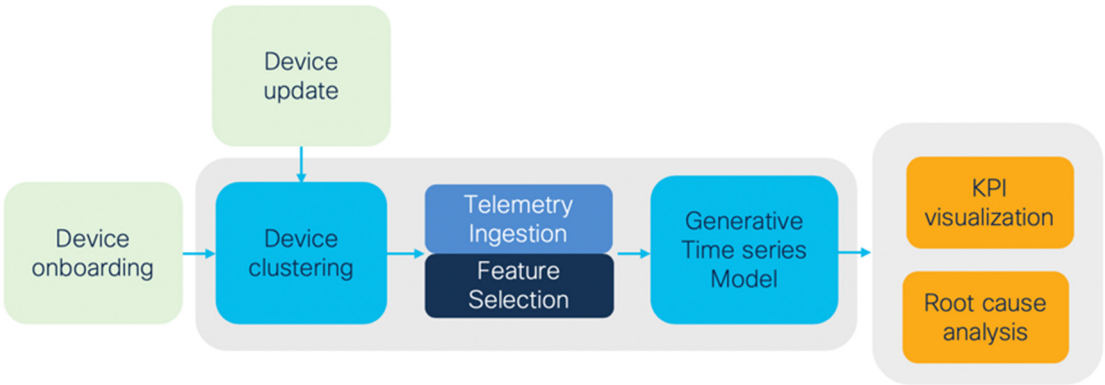

Fig. 1 - Anomaly detection system overview

The proposed system overcomes the problem of dimensionality by grouping the devices based on their similarity, as shown in the diagram of Fig. 2. During times when changes/updates happen to a device, the system will re-evaluate which cluster the device belongs to. The grouping action provides the benefits of a lean solution space and a frictionless user experience.

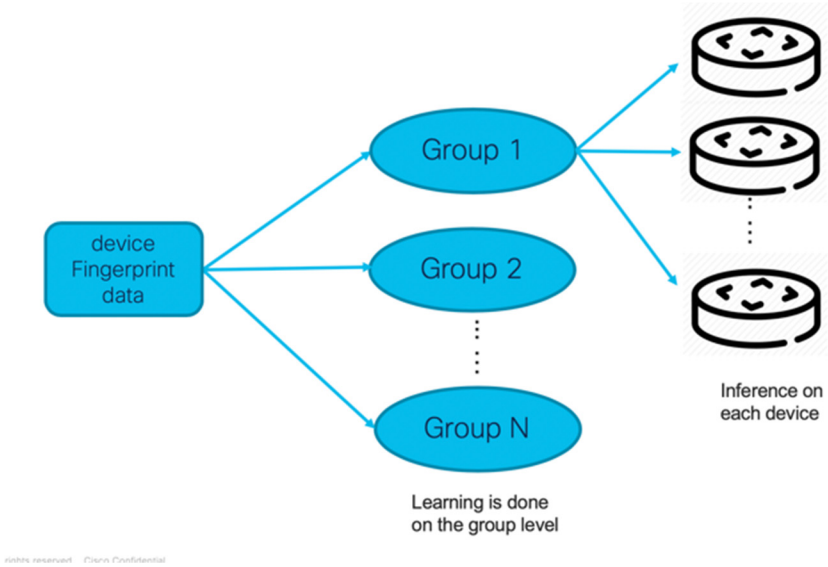

Fig. 2 - Device grouping based on similarity

The proposal system has the following benefits:

1. Lean solution space: the devices are grouped according to similarity. This excludes any redundancy of the system. For devices that are similar, one solution is designed and deployed without introducing additional solutions. The solution space is kept lean and efficient.
2. Frictionless experience: given when device conditions change, the system does not need to rebuild solutions for the updated device.

The details of grouping devices are given below. The process starts by identifying the devices of interest. Then, device information is collected including: product family, SW versions, image names, patches, all hardware PIDS, and FIS modeled features. At a high level, a first data transformation (text to digit) is performed, then dimensionality reduction, and then clustering is applied. Devices with similar profiles can be clustered for

this purpose by using only the features that are related to telemetry data sources available for the device.

In addition, the topology of the devices across the network captures how devices are being related. The topology can be pairwise, e.g., device *x* and device *y* are related. Devices can be further identified by expanding the fingerprint to include dynamic and performance features. Topology is a dynamic feature in actual network use and is learned by evaluating the traffic flow patterns of the device. This clustering feature can be added by evaluating the telemetry variables related to traffic patterns. Clustering devices by matching the available telemetry data to all possible features to get a subset of only "telemetry tracked" features related to anomaly detection is a novel approach to clustering for purposes related to building a multivariate, multi-device anomaly detection system.

After all of the relevant features are obtained, the devices are clustered using a clustering algorithm. Once the cluster is formed, all historical telemetry data is aggregated and one machine learning model is trained for each cluster. The ML solution is discussed below.

According to another novel aspect of the described system, a deep learning model structure is employed to detect anomalies on multiple telemetries at the same time, considering the co-occurrence of the anomalies on multiple telemetries. In practice, the system trains and deploys one model for one device within the same group, which can significantly reduce the dimensionality in the solution space.

Network anomalies often have the following properties:

- Temporal correlation: this part is readily evident; each telemetry is a time series. The telemetry state will evolve over time.
- Spatial correlation: Often, one anomaly manifests in a way that many telemetries are affected at the same time, such as in the network loop condition. In this case, investigating telemetries independently may not yield meaningful result and can often raising false alarms. In this case, the approach needs to properly address co-occurrence of anomalies for each device and among many devices.

Under this setup, a long short-term memory based variational autoencoder (LSTM-VAE) model structure is proposed for multimodal anomaly detection. An LSTM network

can make use of the long-term dependencies in time series and avoid a vanishing gradient. However, the result from LSTM alone is a single point estimate. The common practice for LSTM is to compare the output with the observation, and differentiate normal and abnormal with a threshold method. This approach works for univariate input. However, given the scenario of a multi-dimensional input with different meaning (e.g., CPU, memory, and interfaces), this approach is no longer effective.

The LSTM-VAE model structure is designed to learn a distribution of the input features jointly. In particular, as shown in Fig. 3, the model has two parts: encoding and decoding. For encoding, the model maps multi-dimensional observations across different time periods into a latent space. For the decoding part, it estimates the expected distribution of the time series input. The algorithm is designed such that, for each time step modeled, it has a VAE structure. This model structure enables tracing back to individual telemetry (features) over time to evaluate its behavior. Notice for each time the input is X_t: multi-dimensional telemetry value, and the output is μ_{xt}, σ_{xt}, the reconstructed mean and the variance of X_t. Under this setup, the approach allows for monitoring of tens of thousands of telemetry streams while dimensionality reduction is performed only for visualization and explainability (e.g., "network anomaly score"). However, the Anomaly Detection model is still analyzing all of the data streams without leaving any blind spots.

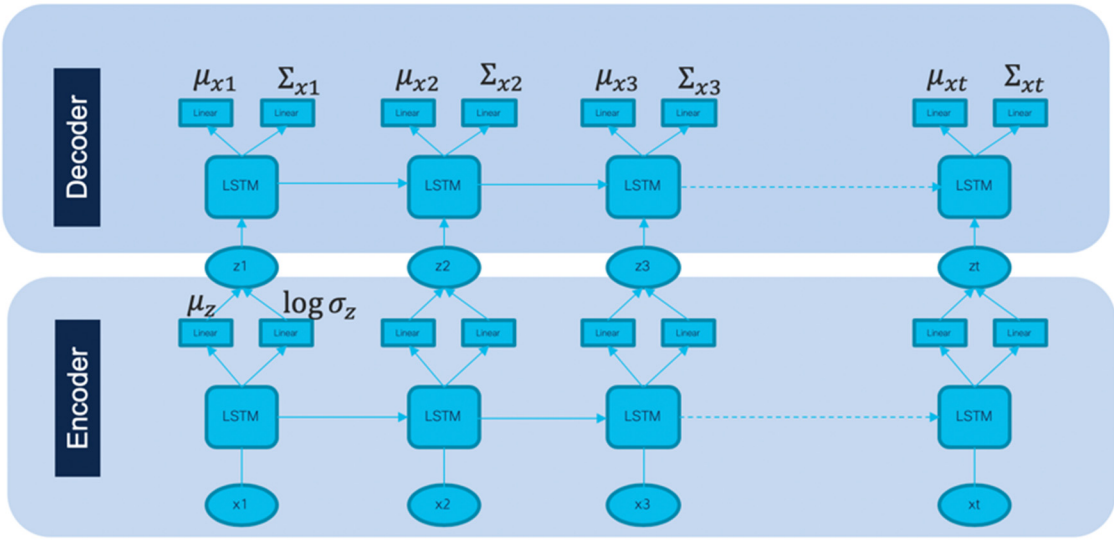

Fig. 3 - Generative Time Series Model

This deep learning solution includes the following procedures:

2.1 Telemetry ingestion

The system collects telemetry data for monitoring and the data is sampled at a suitable frequency. For example, if interested in tracking CPU and memory information, for CPU, for each time period, there are n features: $c_{1t}, c_{2t}, \ldots, c_{nt}$. For memory, there are altogether k features: $m_{1t}, m_{2t}, \ldots, m_{kt}$. All together, $x_t = (c_{1t}, c_{2t}, \ldots, c_{nt,}, m_{1t}, m_{2t}, \ldots, m_{kt})$. The number of times steps (T) for the model usually takes a value from 10 to 100. For example, T=24 is chosen to capture the time correlation for the entire day if the sampling rate is 1 hour and T=96 if the sampling interval is 15 minutes.

2.2 Design structure for LSTM-VAE

The latent variable Z for each time t usually has a low number of dimensions (usually <5 dimensions). In most applications, the latent variable is assumed to be Gaussian noise, where $Z_t \sim N(0, 1)$. For this application, it is proposed to vary the center of normal distribution to be $z_t \sim (\mu_{p(t)}, \Sigma_{p(t)})$. In this case, $p(t)$ is modeled as a function of t, which means each time t can have different mean. This allows introduction of temporal dependency of the time series of data. This adjustment can make the model perform better with time series data.

2.3 Training and testing

The LSTM-VAE model takes the telemetry inputs and provides the learned distribution of the inputs for each time step. During training, all normal inputs are provided without anomalies. During inference time, the real time telemetry input is compared with learned distribution.

According to another novel aspect of the described system, by combining the results from aforementioned techniques, the system produces a closed form health score for each device and one risk score for the entire network that is easy to interpret and visualize. Once the health score is low, the network manager can be notified. In addition, the system can provide visual analysis on telemetry(s) which behave abnormally. Moreover, by tracking

anomalies across different devices, the network manager can be notified about how an anomaly propagates through the network.

During system deployment, following KPIs (health scores) are introduced. The score for each device is modeled as f_t = Prob(x_t, μ_t, Σ_t), where x_t is the multi-dimensional telemetry input (note that this is the raw telemetry input, not the reduced dimension) at time t, and μ_t, Σ_t are the learned mean and covariance of the telemetry input at time t. The observation is determined to be out of bound if $f_t < 0.05$. Essentially, this means the probability of seeing such input is small. Monitoring of f_t continues over time. If f_t is out of bounds for longer than some chosen time periods, a flag is raised. The risk of a device i at time t is defined to be: $h_{t,i}$ = -log($f_{t,i}$), which is the negative log probability for device i at time t. A high score means high chance of an anomaly. The risk score of the entire network (for each time t) is $\theta_t = \Sigma_i w_i h_{i,t}$, where w_i is a weight assigned to each device. These sets of weights are given *apriori*. Depending on how critical each device is (e.g., position of the network, vulnerability, etc.), the weights will vary. These sets of weights can be given by a network manager or by customers.

Once the score is low, each of the individual telemetries can be broken down and the reconstruction result can be plotted over time t. That is, if device has k telemetries, there will be k plots of x_t^i, μ_{xt}^i, σ_{xt}^i over t = 1,2, …, T for i in 1...k. The plot shown in Fig. 4 below assumes monitoring of two telemetries of a device. By comparing x_t^i and the reconstructed result μ_{xt}^i, σ_{xt}^i, the telemetry with issues can be pinpointed. In the plots of Fig. 4, the following are showcased: plot 1 and 2 are the individual anomaly monitoring results (breakdowns) for telemetry 1 and 2; and plot 3 is the overall anomaly assessment score which is computed according to the description above. As can be seen from the third plot, the anomaly score f_t drops really low during the second half of the time period due to an issue with telemetry 1 (telemetry 2 is normal). The system is able to detect an anomaly where some telemetries have issues while others do not. This capability is crucial, since it is necessary not only to know whether there is a problem but also which telemetry(s) out of possibly hundreds have issues that need to be addressed.

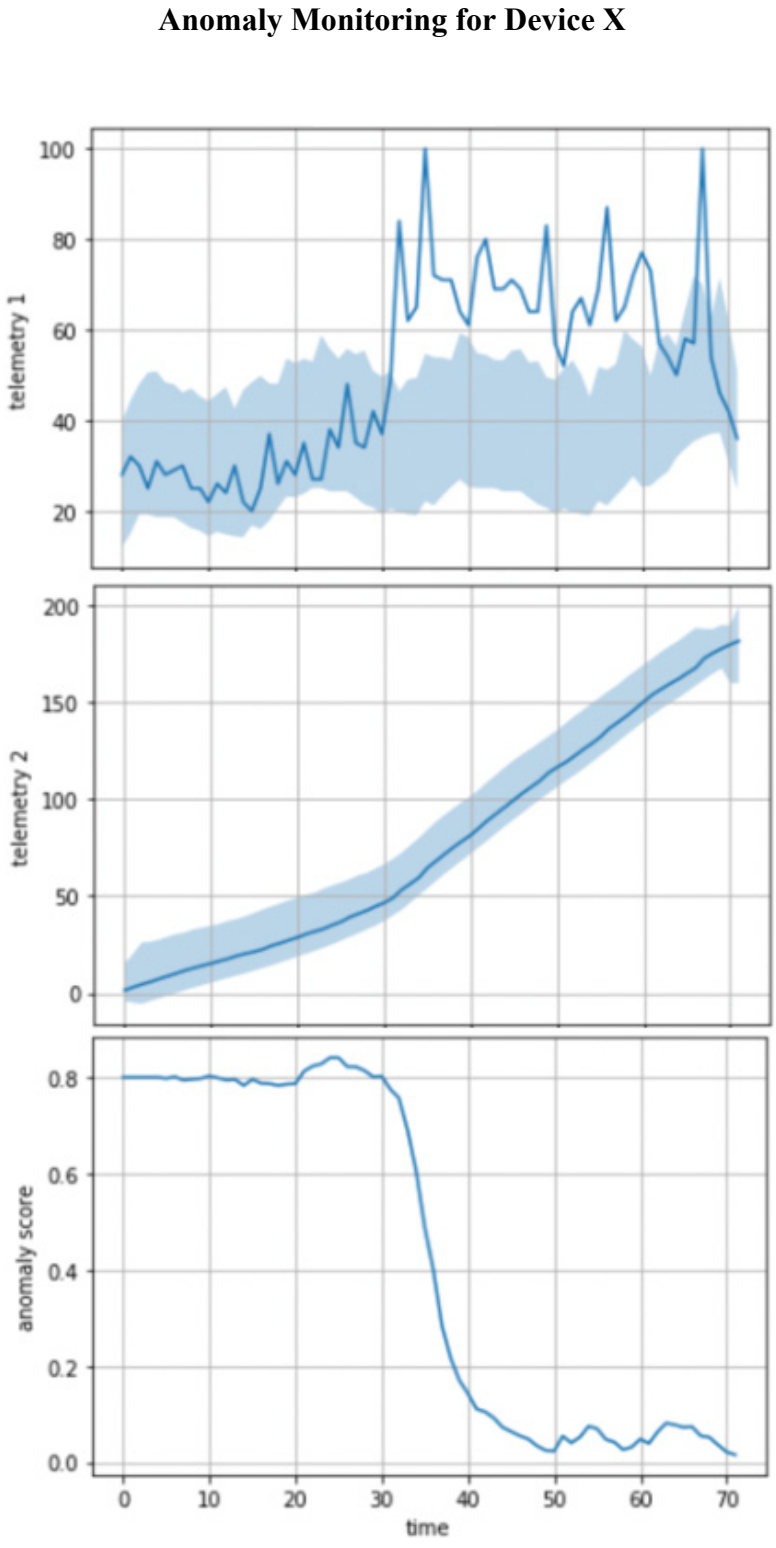

Fig. 4 – Anomaly Monitoring Results of One Device with Two Telemetries

Action items from this approach are given below:

a. Continuously monitor the network base on health score; and

b. If score is low, no action is taken. If score is high, then drill down to each individual telemetry using the observations and model output of the previous *T* times to discover the telemetries with issues.

Yet another novel aspect of the disclosed system relates to resource utilization. Specifically, the anomaly detection system can be used to understand the impact of the network change by comparing old and new baselines. Fig. 5 illustrates one example where the baseline has elevated after the software update. Based on the solution, the device changes groups after an update happens.

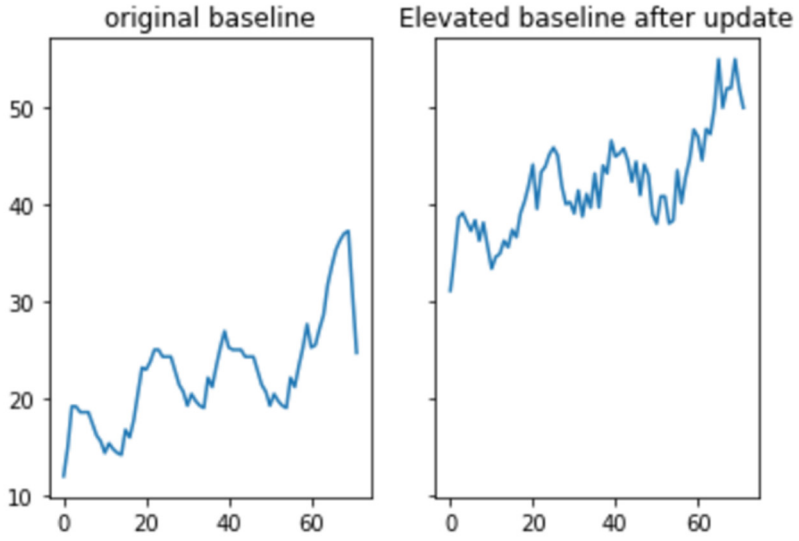

Fig. 5 – Comparison of old and new baselines