TRACK: Optimizing Artificial Neural Networks for Anomaly Detection in Spark Streaming Systems

Ahmad S Alnafessah
a.alnafessah17@imperial.ac.uk
Imperial College London
London, UK

Giuliano Casale
g.casale@imperial.ac.uk
Imperial College London
London, UK

ABSTRACT

Due to the growth of Big Data processing technologies and cloud computing services, it is common to have multiple tenants share the same computing resources, which may cause performance anomalies. There is an urgent need for an effective performance anomaly detection method that can be used within the production environment to avoid any late detection of unexpected system failures. To address this challenge, we introduce, TRACK, a new black-box *training workload configuration* optimization with a neural network driven methodology to identify anomalous performance in an in-memory Big Data Spark streaming platform. The proposed methodology revolves around using Bayesian optimization to find the optimal training dataset size and configuration parameters to train the model efficiently. TRACK is validated on a real Apache Spark streaming system and the results show that the TRACK achieves the highest performance (95% for F-score) and reduces the training time by 80% to efficiently train the proposed anomaly detection model in the in-memory streaming platform.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Machine learning**; • **Security and privacy** → *Intrusion/anomaly detection and malware mitigation.*

KEYWORDS

Performance Anomalies, Apache Spark, Artificial Intelligence, Neural Network, Big Data, Machine Learning

ACM Reference Format:
Ahmad S Alnafessah and Giuliano Casale. 2020. TRACK: Optimizing Artificial Neural Networks for Anomaly Detection in Spark Streaming Systems. In *13th EAI International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS '20), May 18–20, 2020, Tsukuba, Japan.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3388831.3388860

1 INTRODUCTION

There are various options for open-source Big Data technologies for data-intensive applications. Among various Big Data technologies, in-memory processing technology, such as Apache Spark, has become widely adopted by industries because of its speed, generality, ease of use, and compatibility with other Big Data systems. We here consider Spark based streaming workloads.

With the growing complexity of Big Data and cloud systems, failure management requires significantly higher levels of automation and attention. An anomaly is defined as an abnormal behavior during the execution of a program. While some studies address the challenges of performance anomaly detection for batch processing [8], there is a lack of effective automated performance anomaly detection solutions specifically built for Apache Spark streaming systems. There is a need for a technique that can be used to efficiently train a machine learning model to detect performance anomalies within streaming workloads in production environments.

This paper contributes to address the challenge of anomaly identification by investigating new hybrid learning techniques for in-memory Big Data systems. We developed and optimized artificial neural neTwoRks based methodology for Anomaly deteCtion in sparK streaming systems (TRACK), which has a tuning method capable of training a machine learning model with a limited budget and number of experiments. TRACK revolves around using neural networks with Bayesian Optimization (BO) to find the optimal training dataset size and configuration parameters to efficiently train the model to achieve the highest accuracy (95% F-score) and reduces the training time by 80%. A validation based on real datasets from Apache Spark streaming system is provided to demonstrate that the proposed methodology identifies performance anomalies, the ideal configuration parameter, and the optimal training dataset size while reducing the number of experiments by 75%.

2 RELATED WORK

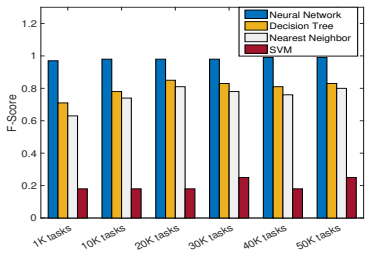
Performance anomaly detection techniques are highly needed by Big Data and large scale systems. Several studies illustrate that most of the root causes of bottleneck and anomalous performance are machine resources such as computer processing units (CPUs) [2].

Fulp et al. [4] use a machine learning approach to detect and predict the likelihood of system failures using an SVM based on information of Linux system log files. Although SVM models are effective at making decisions from well-behaved feature vectors, they can be more expensive for modeling the variation in large datasets and high-dimensional input features [3]. Qi et al. [8] propose a white-box model that uses classification and regression trees to analyze straggler root causes.

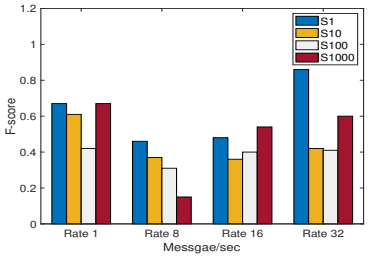
In this paper, we utilize the performance of Bayesian optimization hyperparameter tuning and the efficiency of neural networks to accurately detect anomalous behavior in Big Data systems. Some performance anomaly identification studies have been conducted

VALUETOOLS '20, May 18–20, 2020, Tsukuba, Japan

ACM ISBN 978-1-4503-7646-4/20/05…$15.00
https://doi.org/10.1145/3388831.3388860



(a) Comparison and sensitivity analysis of the impact of Spark workloads size (number of Spark tasks) on neural networks, decision tree, nearest neighbor, and SVM for CPU anomalies detection in Spark streaming workloads.



(b) F-score for anomaly detection using neural network with Rate 1, 8, 16, and 32. Size (1, 10 ,100, and 1000)

Figure 1: Motivation examples for TRACK.

in the literature for different purposes. However, currently there is still a shortage in studies that offer efficient automated anomaly detection, especially for in-memory Big Data stream processing technologies, such as Apache Spark streaming.

3 MOTIVATING EXAMPLE

Here we demonstrate that the neural networks is more accurate than other well-known algorithms. For our experiments, the workloads are exponentially generated as messages to be sent to the data stream processing system with some predefined characteristics, such as the rate of (*message/sec*) and the size of messages in lines. A detailed comparison is shown in Figure 1(a) to examine the impact of Spark workload size (number of tasks) on the neural networks model and compares it with the nearest neighbor, decision tree, and SVM. Six Spark streaming workload sizes with the same configurations are examined for sensitivity analysis, which are 1k, 10k, 20k, 30k, 40k, and 50k tasks. From Figure 1(a), it is evident that the neural networks model outperforms all the other algorithms.

We examine a baseline experiment to prove that there is an important need for a solution to find the optimal dataset size and configuration parameters of stream workload to train the anomaly detection model to generalize the model to detect anomalous behaviors in in-memory Big Data systems. Figure 1(b) shows some design factors and response variables (F-score) for different streaming workload configurations where the proposed neural network is trained with a single combination of configurations parameters (e.g., rate r and size s) and test it against all the other workloads stream configurations, which include rates (1,8,16, and 32) and sizes (1,10,100, and 1000). As can be seen from Figure 1(b), it is not apparent which set of workload configurations that can be used to efficiently train the machine learning model to achieve the highest accuracy with less time consuming to train the model and detect the anomalous performance in the Spark streaming system. With WordCount Spark streaming application (only two parameters), it is also challenging to find the ideal dataset size to efficiently train the anomaly detection model to comprehensively cover all the seen types of anomalies.

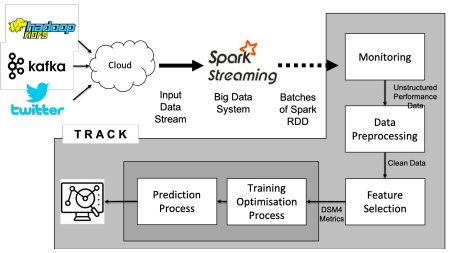


Figure 2: TRACK Methodology for anomaly detection)

4 METHODOLOGY

In this section, we introduce TRACK, which is Bayesian Optimization (BO) and neural network driven methodology to train and detect performance anomalies in Apache Spark streaming systems. Figure 2 shows the TRACK processes of anomaly detection for the proposed method. The following subsections give a brief information about BO, neural network, training, testing, and feature selection.

4.1 Neural Network Model

The proposed neural networks model in [1] with backpropagation and conjugate gradient are used to train the neural networks to update values of weights and biases in networks. The scaled conjugate is used because it is usually faster than other gradient algorithms [7], especially for time-dependent applications such as real time stream processing. The neural networks model uses a *Sigmoid* transfer function ($\sigma(x) = 1/(1 + e^{-x})$) as an activation function, and *Softmax* transfer function is used in the output layer to handle classification problems with multiple classes. For cost function, *cross-entropy* is used to evaluate the performance of neural networks. *Cross-entropy* is used because it has significant, practical advantages over squared-error cost function [5].

The proposed neural networks contain three layers. The first layer is the input layer that includes a number of neurons equal to the number of input features. The second layer is the hidden layer, which has a number of neurons that is determined by using a *trial and error* method, choosing a number between the sizes of input features n_i and output classes n_o [9]. The hidden layer size between n_i and n_o satisfies our goal in achieving accurate results. The output layer contains a number of neurons equal to the number of target classes (types of anomalies), where each neuron generates boolean values, which are 0 for normal behavior or 1 for anomalous behavior.

4.2 Bayesian Optimization

The proposed methodology revolves around using BO to find the optimal dataset size and configuration parameters to train the neural networks to generalize the model to detect the anomalous behaviors in the Spark streaming system.

When doing Bayesian optimization, there are two main choices to make, which are prior over functions and acquisition function.

Expected-Improvement function in [6] is used as an acquisition function to evaluate the expected amount of performance improvement in the neural network detection model $f(x)$ and ignore any values that cause an increase in the error rate of the model. In other words, x_{best} is the location of the smallest posterior mean (optimal workload configuration), and $\mu_Q(x_{best})$ is the smallest value of the posterior mean. The expected improvement can be described as follows $EI(x) = E_Q[max(0, \mu_Q(x_{best}) - f(x))]$. E_Q indicates the expectation taken under the posterior distribution given evaluations of f at $x_1, x_2, ..., x_n$. The time to assess the objective function may vary over some range of points depending on the region [6].

4.3 Model Training and Testing

The Streaming workload configurations consist of all possible combinations of configuration parameters of $Rate_n$ and $Size_m$, which in total will be $n * m$ combinations ($n * m$ *DSTrain*). The training part of the dataset (*DSTrain*) is divided into 10 equal subsets to find the ideal size dataset. For example, the dataset *DSTrain* workload configuration with rate r_i and size s_j is divided into 10 subsets according to $DSTrain_{r_i,s_j} = DSTrain_{r_i s_j,1} + ... + DSTrain_{r_i s_j,10}$.

The total number of all possible data subsets is $n*m*10$, which is hard and time consuming to find the optimal configuration combinations parameters and dataset size to train the model. More details information is presented in Algorithm 1. To assess the proposed model, we use a well-known standard classification performance metric, which is F-score (F).

4.4 Feature Selection

In this work, we extend our previous proposed method called dataset method four (DSM4), which is built upon a list of Spark performance metrics that are presented in [1]. DSM4 examines the comprehensive internal Spark architecture and Directed Acyclic Graph Spark application by relying on information from the Apache Spark systems, such as Spark executors, shuffle read, shuffle write, memory spill, java garbage collection, tasks, stages, jobs, applications, and execution timestamps for Spark resilient distributed datasets (RDDs). The collected Spark performance metrics are in time series and manually labeled either as normal or anomalous, before passing them as inputs to the proposed model.

5 EVALUATION

This section provides an evaluation of the proposed methodology. We use a random search (RD) algorithm as a baseline on the same datasets, which are generated from Apache Spark Streaming system.

To evaluate the accuracy of the proposed anomaly detection methodology, we developed *Network WordCountExp* benchmark, which is a customized benchmark for stream processing Big Data systems to generate our dataset for datasets generation and training purposes. *Wordcount* is a conventional CPU-intensive benchmark and is widely accepted as a standard micro-benchmark for big data plat forms. The workloads are exponentially generated (with exponential distribution). More than 570 experiments and 135 GB of data have been collected from the Spark streaming system, which we used them to evaluate the proposed work. To inject different types of anomalies, an open source tools (*stress-ng*) have been used to evaluate the proposed methodology with Spark streaming system. A list of performance anomalies is used to generate CPU stress, cache thrashing stress, and context switching stress.

Algorithm 1: Training and testing methodology for TRACK

Input: Predefined anomaly detection performance $\mathcal{F}$, Configuration space $\mathcal{X}$, and system metrics dataset $\mathcal{D}$

Output: Optimal trained neural network model $\mathcal{M}$, which is able to identify anomalous performance in Spark streaming platform with highest predefined accuracy with less amount of time

```
Configuring streaming workload benchmark
Workload generation with configuration space X
Streaming workload from local network N → Spark system
System profiling to collect dataset of performance metrics
Data cleansing and preprocessing → D
DSTrain = 75% of D ← total training dataset
DSTest = 25% of D ← total testing dataset
F = 0 and DSTrain_c is empty ← current f-score and training data
while ( F ⩽ F ) do
    X_i = EI_PS(X) ← acquisition function finds next configuration
    DSTrain_c = DSTrain_c + DSTrain_{X_i} ← adding current dataset to the previous dataset
    M = TrainNN(DSTrain_c ) ← train neural network model on the new dataset configuration
    F = Fscore(M(DSTest))
    Algorithm
end
```

6 RESULTS

6.1 Finding The Ideal Single Workload Configuration For Model Training

From the previous discussion about motivation example (Section 3), there is a need for a solution to find the ideal single workload configuration (e,g,. Rate r_i and Size s_j) that can be used to train the proposed anomaly detection model to pinpoint the abnormal behavior with highest possible F-score. This will facilitate the using of single workload configuration to be generalized and used to detect anomalies with the other workload configurations. The Spark Streaming workload is used with all possible combinations of *Rate (1, 8, 16, ,and 32)* and *Size (1, 10, 100, and 1000)*, which are 16 combinations in total.

A Bayesian optimization and Neural Networks models (described in Section 4.2 and 4.1) are used to address the need of determining

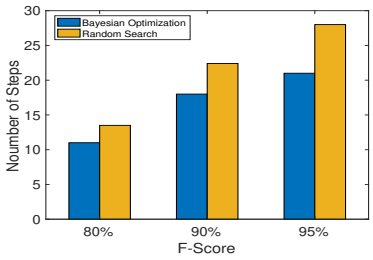


Figure 3: Comparison between Bayesian Optimization and Random Search to reach a predefined F-Score with the most minimum number of steps. Workload has all the possible combination of parameters and CPU anomalies.

the ideal single workload configuration (Rate r_i and Size s_j) with the minimum number of running experiments *n*. To get an accurate result, we repeated the experiments for 50 times then calculate the average of *n*. The result shows that the ideal F-score is reached with a minimum number of running experiments (*n=8*), which is less by 50% of the total number of possible configuration (*n=16*).

6.2 Bayesian Optimization Model to Train Anomaly Detection Technique

A Bayesian Optimization model (discussed in Section 4.2) is used to find the optimal size of the training dataset and the stream workload configurations set to achieve the highest accuracy with less amount of time in training the proposed anomaly detection model. Figure 3 depicts a comparison between Bayesian Optimization and Random Search to reach a predefined F-Score with the most possible minimum number of training steps from the total number of steps, which are 160 steps. The conducted experiments have workloads that contain normal and CPU anomalous behaviors with all the possible combinations of workload configurations. Figure 3 shows the average of 50 experiments that the neural networks train with Bayesian Optimization to achieve the predefined F-score. With Bayesian Optimization, the trained model reaches 95% F-score in 21 steps, whereas 28 steps using a random search (enhanced by 25%). This proves that the proposed model can reduce the time and computation process by 25%.

There are other two types of anomalies, which may disrupt the performance of the stream processing system. These two types are cache thrashing and context switching. The proposed model can detect the cache thrashing and context switching anomalies with F-score equal 80% and 95% respectively. The proposed model outperforms the Random Search by more than 25% and can reduce the amount of computations from 160 experiments to 14.

6.3 New Unseen Workload Configurations

The goal of this section is to train the proposed model on predefined workload configurations (*Rate (1, 8, 16, ,and 32)* and *Size (1, 10, 100, and 1000)*) and generalize the model to perform accurately as well with unseen new workload configurations (e,g,. $r_i = 20$ and $s_j = 150$). In this case, the workload is more realistic and reflects the workload characteristics of the stream processing system.

For the training phase, the same Bayesian Optimization and Neural Networks configurations in Section 6.2 are used to train the model on predefined workload configurations (*Rate (1, 8, 16, and 32)* and *Size (1, 10, 100, and 1000)*) to reach 95% F-score for detecting the CPU performance anomalies. For the testing phase, the final model of the training phase is used to detect anomalous behavior but with new unseen workload configurations. Rate can be between 1 to 32 and size can be between 1 to 1000. With the three anomalous workloads (CPU, cache thrashing, and contexts switching), the F-score and standard deviations are 0.93±0.01 for CPU, 0.77±0.02 for cache thrashing, and 0.72±0.04 for context switching. The proposed anomaly detection model has the ability to be trained on 16 workload configurations to be generalized to detect anomalies against 32000 different workload configurations.

7 CONCLUSION

This paper contributes to addresses the challenge of anomalous identification by proposing a new hybrid learning solution, TRACK, for anomaly detection in in-memory Big Data systems. The anomaly detection and tuning method are developed using Bayesian Optimization and neural networks to train the model with a limited budget and computing resources. As can be seen from the experimental results, the proposed model can find the optimal training dataset size and configuration parameters to accurately identify different types of performance anomalies in Big Data systems. The proposed model achieves the highest accuracy (95% F-score) and reduces the execution time by 80%. A validation based on a real dataset for the Apache Spark streaming system is provided to demonstrate that the proposed methodology identifies the performance anomalies, the ideal configuration parameter, and training dataset size with up to 75% fewer experiments. In addition, the proposed model can be easily generalized to cover unforeseen workload configurations.

ACKNOWLEDGMENTS

This research is funded by KACST in Saudi Arabia and the European Union's Horizon 2020 research and innovation program under grant agreement No. 825040 (RADON).

REFERENCES

[1] A. Alnafessah and G. Casale. 2019. Artificial neural networks based techniques for anomaly detection in Apache Spark. *Cluster Computing* (2019), 1–16.
[2] G. Ananthanarayanan, S. Kandula, A. G. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris. 2010. Reining in the Outliers in Map-Reduce Clusters using Mantri. In *Osdi*, Vol. 10. 24.
[3] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie. 2016. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition* 58 (2016), 121–134.
[4] E. W. Fulp, G. A. Fink, and J. N. Haack. 2008. Predicting Computer System Failures Using Support Vector Machines. *WASL* 8 (2008), 5–5.
[5] D. M. Kline and V. L. Berardi. 2005. Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing & Applications* 14, 4 (2005), 310–318.
[6] Mathworks. 2019. Bayesian Optimization Algorithm. https://uk.mathworks.com/help/stats/bayesian-optimization-algorithm.html#bva8tie-1 visited on 1-10-2019.
[7] M. F. Møller. 1993. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks* 6, 4 (1993), 525–533.
[8] W. Qi, Y. Li, H. Zhou, W. Li, and H. Yang. 2017. Data Mining Based Root-Cause Analysis of Performance Bottleneck for Big Data Workload. In *2017 IEEE HPCC Conferencece*. IEEE, 254–261.
[9] K. G. Sheela and S. N. Deepa. 2013. Review on methods to fix number of hidden neurons in neural networks. *Mathematical Problems in Engineering* 2013 (2013).