Spring 2018

Anomaly Detection for Application Log Data

Aarish Grover
San Jose State University

Anomaly Detection for Application Log Data

A Thesis

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements of the Class

CS298

By

Aarish Grover

May 2018

The Designated Thesis Committee Approves the Thesis Titled

Anomaly Detection For Application Log Data

by

Aarish Grover

APPROVED FOR THE DEPARTMENT OF COMPUTER

SCIENCE SAN JOSE STATE UNIVERSITY

May 2018

Dr. Robert Chun Department of Computer Science

Dr. Katerina Potika Department of Computer Science

Mr. Uma Panda NetApp, Inc.

ABSTRACT

In software development, there is an absolute requirement to ensure that a system once developed, functions at its best throughout its lifetime. Application log data is critical to maintaining application performance and thus techniques to parse, understand and detect anomalies in application log data are critical to ensuring efficiency in software development. While initially hampered by limited hardware and lack of quality datasets, anomaly detection techniques have recently received a surge of interest with advancements in machine learning technology and especially neural networks. In this paper, we explore anomaly detection, historical techniques to detect anomalies and recent advancements in neural networks, which promise to revolutionize anomaly detection in application log data. Further, we analyze the most promising anomaly detection techniques and propose a hybrid model combining LSTM Neural Network and Auto Encoder which improves upon existing techniques.

Index Terms – **Anomaly Detection, Application logs, Machine Learning, Neural Networks**

ACKNOWLEDGEMENTS

I would like to thank Dr. Robert Chun for his continued support and providing me the guidance necessary to work on this project. I would like to thank my advisor Dr. Robert Chun and committee members Dr. Katerina Potika and Mr. Uma Panda for teaching me core skills needed to succeed and reviewing my project. And finally, I would like to thank my parents for their patience and advice they gave me throughout my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

Applications generate massive amounts of log data, which automatically produced timestamped data that represents every single system and user event for all users of the application [1]. This data is generated throughout the lifetime of an application and tends to be time-series, incredibly unstructured, textual, poorly formatted and is generated at an incredible rate as the application scales and adds more users [2]. Walmart has generated 2.5 Million Petabytes of data for their flagship cloud computing application [1].

Log data also contains anomalies which represent potential system faults and are thus critical to debugging application performance and errors. The details and timestamps of the anomaly offer a starting point for discovering when, how and where errors in the application occurred [12]. Any debugging process requires a developer or support engineer to parse this data manually, reading through line by line until an anomaly is located. Log data also changes completely from application to application as the exact formatting, level of detail and verbosity are defined by the application developer's development style which varies widely across the industry [3]. Because of growing demands, companies are forced to deploy an increasing number of engineers to manage log data, and even then, a vast number of anomalies such as error messages, warning notifications and network intrusion attempts are not detected. One possible solution for addressing application log anomaly detection is to autonomously detect anomalies in application log data [12].

In this paper, we explore anomalies in log data and existing anomaly detection techniques [3]. We then seek to provide insight into the questions:

- What are the types of anomalies?

- Which methods exist for anomaly detection?
- Are Neural Networks appropriate for application log data anomaly detection?
- Which Anomaly Detection method gives the best results and can we improve upon them?

We have analyzed scientific research fundamental to Anomaly Detection which is represented as a conceptual map in Fig. 1.

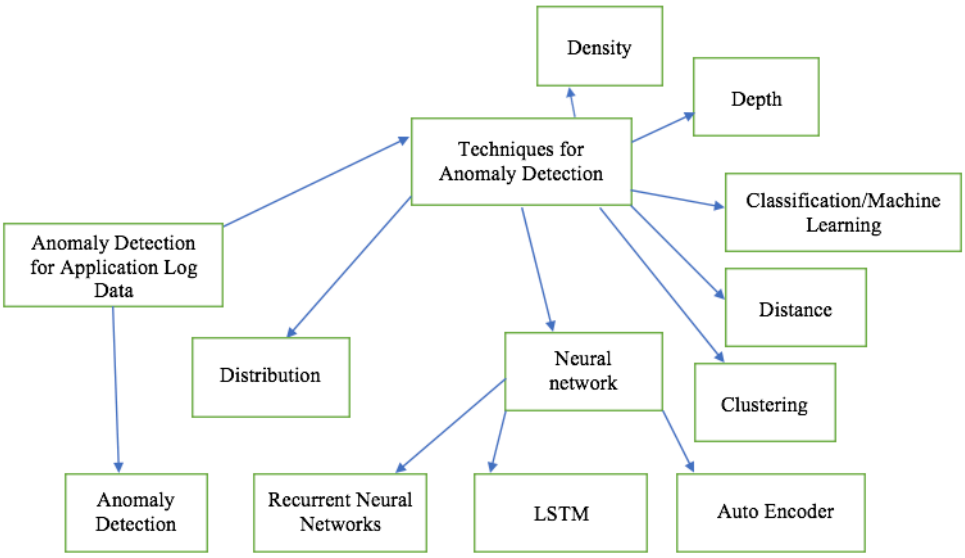

Figure 1. Conceptual map of literature review

II. ANOMALIES IN LOG DATA

Anomalies in application log data are considered to be patterns or characteristics which do not follow the average or normal behavior during perfect operation[4]. As described by Grubbs [13] 'an outlying observation, or "outlier", is one that appears to deviate markedly from other members of the sample in which it occurs.' Such anomalies can be initiated through malicious actors, system level bugs or incorrect user operation and are often symptoms of imminent system failure or breach. Anomaly detection for application log data is particularly challenging whether automated or done manually for the following reasons.

A. Unstructured Plain Text

B. Redundant Runtime Information

C. Large Unbalanced Data

A. Unstructured Plain text and Variation

An application log as shown in Fig. 2 is unstructured and stored as plain text. This lack of structure complicates data analysis, which is further exacerbated by logging formats which vary completely between applications. Fig. 2 shows Azure Application log data which varies completely both in content as well as structure when compared to Fig. 3 HDFS Application log data.

```
2017-02-15 13:54:26 AZUREOVERVIEW__522C GET / X-ARR-LOG-ID=8b90ffae-54b1-43c7-abaa-af0fcb759fac 80 - 141.101.76.74 Mozill
2017-02-15 13:54:26 AZUREOVERVIEW__522C GET /css/site.min.css X-ARR-LOG-ID=3a3cd496-76fe-42a4-be99-7e34fb69aa63 80 - 141.
2017-02-15 13:54:26 AZUREOVERVIEW__522C GET /lib/bootstrap/dist/css/bootstrap.css X-ARR-LOG-ID=6b92e7d2-45a8-4c5b-a3fc-66
2017-02-15 13:54:26 AZUREOVERVIEW__522C GET /css/site.css X-ARR-LOG-ID=7bc19a0b-6d75-489c-8256-9725652e71e5 80 - 141.101.
2017-02-15 13:54:26 AZUREOVERVIEW__522C GET /lib/bootstrap/dist/js/bootstrap.js X-ARR-LOG-ID=22f94e44-c590-4f47-be5b-f981
```

Figure 2. Azure Application Log Data

```
081109 204106 329 INFO dfs.DataNode$PacketResponder: PacketResponder 2 for block blk_-6670958622368987959 terminating
081109 204132 26 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.43.115:50010 is added
081109 204324 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.203.80:50010 is added
081109 204453 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.11.85:50010 is added t
```

Figure 3. HDFS Application Log Data

B. Redundant Runtime Information

As can be seen in Fig. 3 Application logs contain runtime information such as IP Address of servers. This data changes during execution and varies from server to server and is hence redundant for the purpose of anomaly detection. Also as shown in Fig. 3, application log data contains domain-specific data such as "blockMap updated" for HDFS logs, which combined with redundant runtime information increases the complexity of anomaly detection.

C. Large Unbalanced Data

Application log data is designed to record all changes to an application and hence contains data that is heavily unbalanced in favor of non-anomalous execution. The data generated by HDFS in [26] contains only 16,838 anomalies (only 1.5%) out of over 11,175,629 log events. The size and unbalanced nature of log data thus complicate the anomaly detection process.

III. TYPES OF ANOMALIES

A. Point Anomaly

A point anomaly is data which deviates significantly from the average or normal distribution of the rest of the data [14]. Such data is often system generated, and the significant deviation is restricted to specific data points and shares little context with the rest of average or normal data [4].

Point anomalies are the simplest to detect and multiple techniques exist to automate point anomaly detection [14]. Point anomalies can be quickly discovered and fixed, and thus rarely have a significantly detrimental effect on applications [4,14]. Fig. 4 and Fig. 5 are examples of point anomalies.

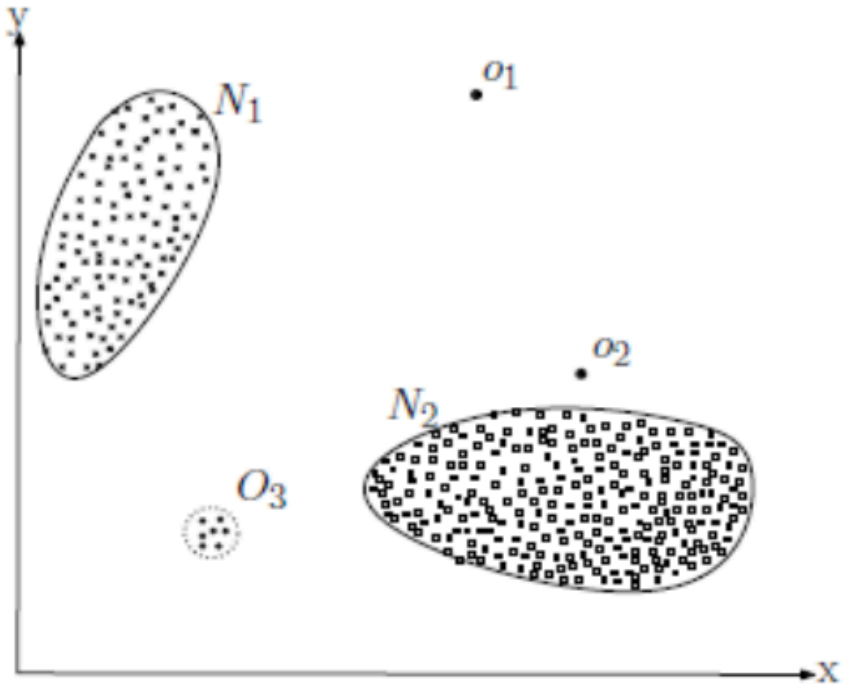

Figure 4. Point Anomalies [27]

```
WARN dfs.DataNode$DataXceiver: 10.251.30.85:50010:Got exception while serving blk_-2918118818249673980
```

Figure 5. Point Anomaly in HDFS Log Data

B. Contextual Anomaly

A contextual anomaly is identified as anomalous behavior restricted to a specific context, and normal according to other contexts [4, 14]. This type of anomaly also referred to as conditional anomaly, is often difficult to detect as it requires deep domain knowledge to understand the context within which the anomaly arises [4]. Fig. 6 shows a contextual anomaly in HDFS log data. Here, a ‘received block’ log would not be an anomaly on its own, but in the context of occurring between 60 deletions, it is treated as an anomaly.

```
INFO dfs.FSDataset: Deleting block blk_-7106479503467535906 file /mnt/hadoop/dfs/data/current,
INFO dfs.FSDataset: Deleting block blk_-7411858598798393933 file /mnt/hadoop/dfs/data/current,
INFO dfs.FSDataset: Deleting block blk_-5429479049793046826 file /mnt/hadoop/dfs/data/current,
INFO dfs.FSDataset: Deleting block blk_-6431101765137189231 file /mnt/hadoop/dfs/data/current,
INFO dfs.FSDataset: Deleting block blk_-2923662094689783995 file /mnt/hadoop/dfs/data/current,
583 INFO dfs.DataNode$DataXceiver: 10.250.14.143:50010 Served block blk_-664656559337730574 to
788 INFO dfs.DataNode$PacketResponder: Received block blk_3224393364314749254 of size 67108864
INFO dfs.FSDataset: Deleting block blk_1483582953997932733 file /mnt/hadoop/dfs/data/current/
INFO dfs.FSDataset: Deleting block blk_-3607708283707030582 file /mnt/hadoop/dfs/data/current,
INFO dfs.FSDataset: Deleting block blk_4365203784873840210 file /mnt/hadoop/dfs/data/current/
INFO dfs.FSDataset: Deleting block blk_1920931690498309324 file /mnt/hadoop/dfs/data/current/
```

Figure 6. Contextual Anomaly in HDFS Log Data

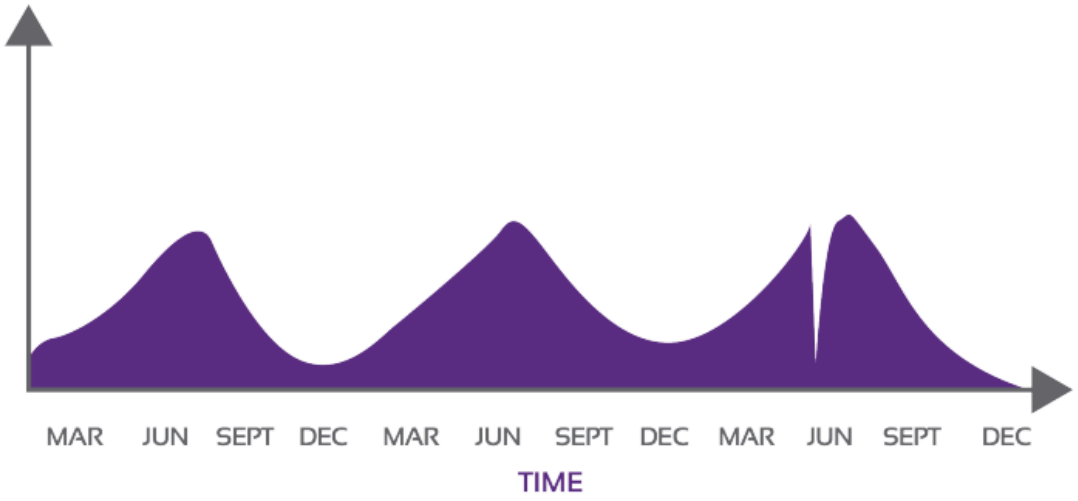

Figure 7. Contextual Anomaly [27]

C. Collective Anomaly

Unlike contextual and point anomalies, collective anomalies appear as a group of anomalous values in data [4,15]. Collective anomalies are anomalous behavior of a collection of data instances with respect to the complete dataset. Individual data instances might not represent an anomaly, however, presence in the collective anomaly data instances is an indicator of anomalous behavior [15]. It must be noted however that on its own, a data instance does not represent collective anomalies and must appear in a collection of data to be collectively anomalous [4]. Fig. 9 shows a collective anomaly in HDFS log data. While adding to an ‘invalid

set' may not be an anomaly on its own, several similar consecutive additions to an ‘invalid set' represent a collective anomaly in HDFS log data.

Figure 8. Collective Anomalies [27]

```
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-4438918035940270891 is added to invalidSet of 10.251.71.193:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_5616920288053661280 is added to invalidSet of 10.251.42.9:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_1815340037591016775 is added to invalidSet of 10.250.9.207:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_2154193600525549460 is added to invalidSet of 10.251.43.147:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_5580254744888391593 is added to invalidSet of 10.251.201.204:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-1641081065757890722 is added to invalidSet of 10.251.107.242:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_5505696082121135363 is added to invalidSet of 10.251.214.130:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_5689009809369418749 is added to invalidSet of 10.251.91.84:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_8155177385081669900 is added to invalidSet of 10.251.197.161:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_247452006316846355 is added to invalidSet of 10.250.6.191:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_2657997043417110888 is added to invalidSet of 10.250.13.240:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-323838382557775734 is added to invalidSet of 10.251.202.209:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-719470845056640928 is added to invalidSet of 10.251.90.64:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-7999375661905777727 is added to invalidSet of 10.251.106.37:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-2451769929123196359 is added to invalidSet of 10.251.199.159:50010
```

Figure 9. Collective Anomaly in HDFS Log Data

IV. EVOLUTION OF ANOMALY DETECTION TECHNIQUES

Anomaly detection techniques have evolved with the advent of big data and machine learning. Initially approached using statistical techniques, anomaly detection quickly evolved into a field of its own encompassing statistical, depth, density, clustering, distance, machine learning and neural network based approaches.

A. Statistical/Distribution Based Anomaly Detection

In order to leverage Statistical Anomaly Detection, the log dataset is organized in terms of its overall statistic distribution and the data points which stand out or do not conform to this distribution are removed or examined [5]. These approaches are simple to implement but are

complicated by ever-changing definitions for anomalies in different domains [5, 16]. A transaction of $1 Million would be anomalous for personal finance applications but not for Investment banking applications. Thus, such approaches require prior knowledge of the dataset without which detecting contextual or collective anomalies can be incredibly challenging, especially for application log data which varies from application to application [5, 16].

B. Depth-based Anomaly Detection

The depth-based approach works around the requirement to organize data by its statistical distribution and instead leverages convex hulls and flag objects to compute anomalies in the outermost layers [5]. This, however, requires heavy computation, is unable to detect contextual anomalies and is not suitable for high volume, high-velocity datasets such as application logs [5].

C. Clustering Based Anomaly Detection (Unsupervised Machine Learning)

Clustering, considered to be the Swiss army knife of statistical modeling generates clusters out of similarities in datasets thus removing data points which don't conform to these clusters as anomalies [5]. K Means is the most popular technique in use and can be effective at detecting contextual and collective anomalies.

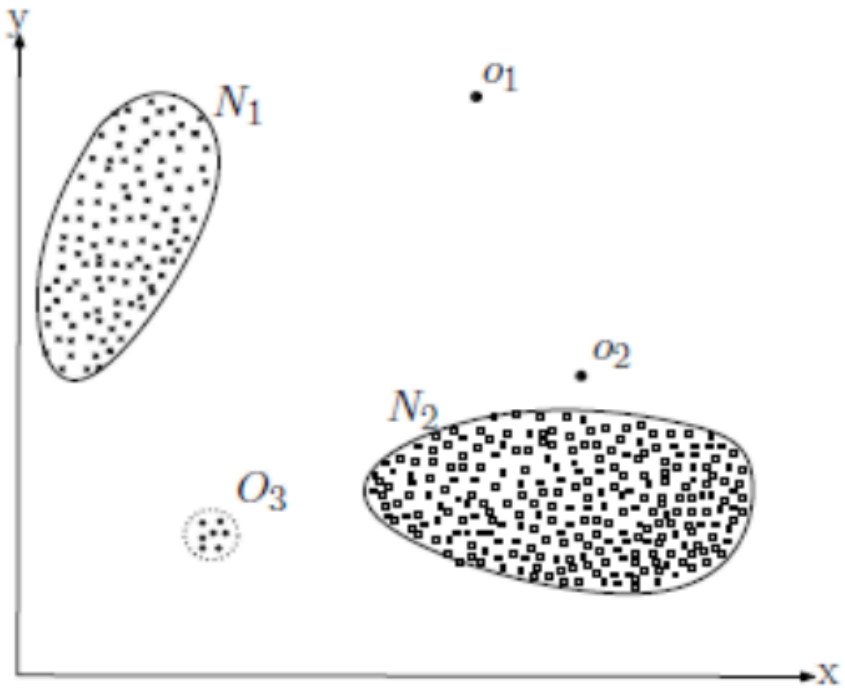

Figure 10. Clustering based Anomaly Detection [27]

D. Distance-Based Anomaly Detection

This category of anomaly detection method detects the distance of an element from a subset closest to it. Although this method, works well in many situations, it fails when applied to datasets with an unpredictable distribution with both sparse and dense regions. This is also referred to as the multi-density problem, which rules out detecting collective anomalies [5].

E. Density Based Anomaly Detection (Unsupervised Machine Learning)

Density-based anomaly detection techniques are explicitly designed to get around the multi-density problems that distance-based methods suffer from. Density-based methods leverage the local outlier factor (LOF). The LOF is a quantification of how much each dataset lies outside the normal behaviors, which itself, depends on the local density of its neighborhood. As Density-based methods include density alongside distance, density based methods can work

much better with unpredictable distributions of sparse and dense regions.

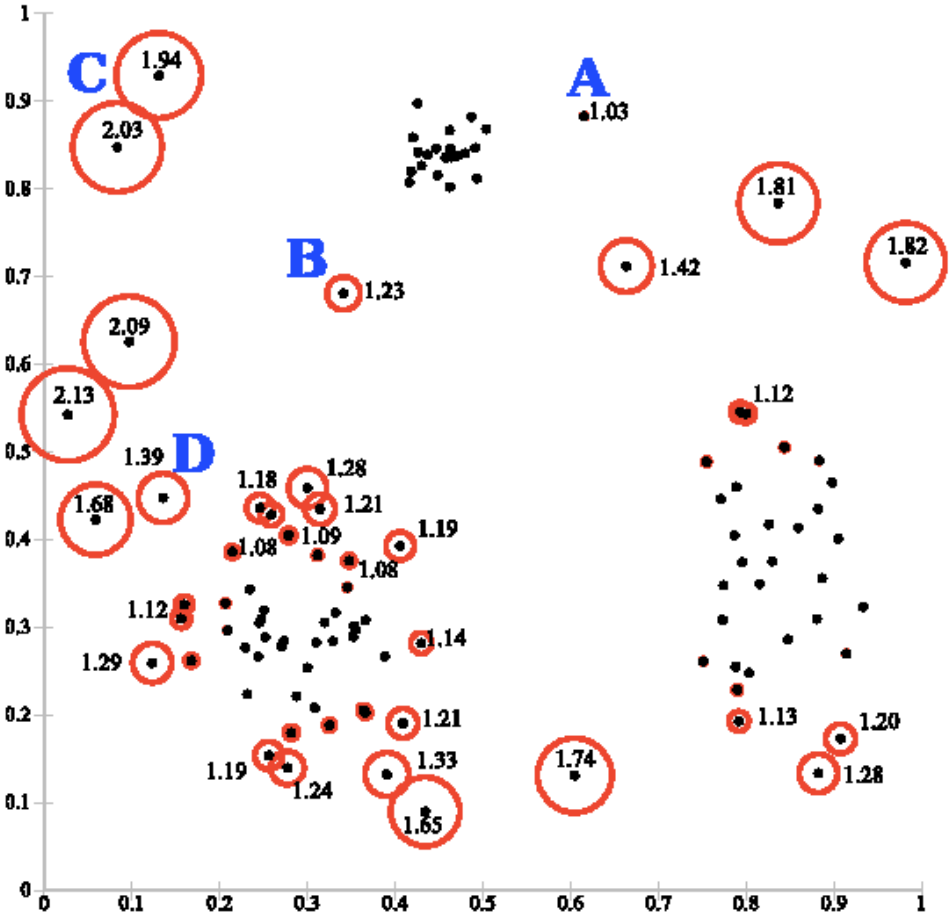

Figure 11. Density Based Anomaly Detection [28]

F. Spectral Decomposition

Spectral decomposition is a mathematical technique to artificially lower the dimensionality of the dataset. Spectral decomposition techniques based on Principal Component Analysis(PCA) [24] work by splitting the dataset space into normal, noise and anomaly subspaces allowing for simpler and more effective anomaly detection. [25].

G. Supervised Machine Learning Based Anomaly Detection

In order to solve by classification, the problem is redesigned as an identification problem where the entire dataset is classified into anomalous or non-anomalous data [21]. This process

happens in two parts, starting with training a model on a subset of the data and leveraging this trained model to test the rest of the data [5]. As log data is incredibly verbose and highly imbalanced, classification learning overfits the model. Unless datasets are well balanced, classification learning struggles to generalize and accurately classify anomalies [21].

H. Classic Neural Network Based Anomaly Detection

Classic Neural Network anomaly detection techniques behave in a similar way to Machine learning approaches and thus mandate well-balanced datasets [23]. Recent advances in Neural Networks related to Recurrent Neural Networks, Long Short-Term Memory Neural Networks, and Auto-Encoders have been used extensively to solve a myriad array of problems related to anomaly detection such as Network intrusion detection, sensor data anomaly analysis, ECG time series anomaly detection as well multiple other domains [23]. These more recent advances are able to handle contextual and collective anomalies particularly well due to hidden layer based memory mechanisms which makes a good fit for accurate and generalizable anomaly detection, even with imbalanced datasets [22].

V. NEURAL NETWORK BASED MODELS FOR ANOMALY DETECTION

Recurrent Neural Networks (RNN) alter standard neural networks by allowing output from hidden layers of neurons to serve as inputs to the next layer [18, 23]. This allows for incorporation of feedback at every stage and thus the network is able to leverage history to make classification decisions [6, 18]. This approach makes RNNs desirable for Anomaly detection as retaining context in log events is critical to discovering contextual anomalies.

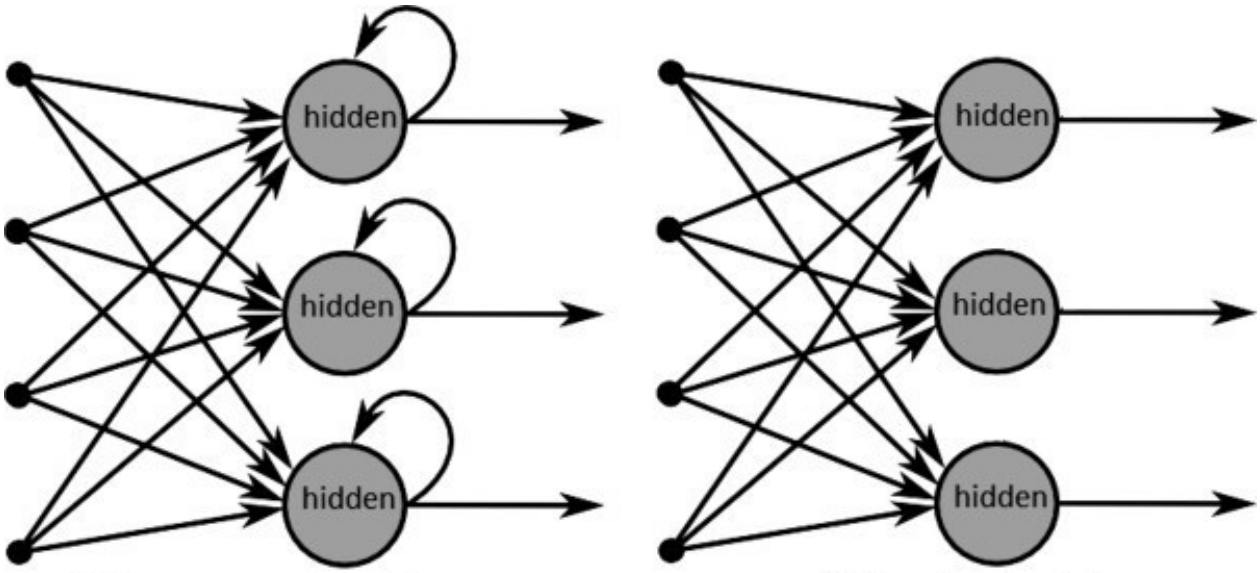

Figure 12: Recurrent neural network architecture comparison with forward neural network [29]

RNNs are extensively used for Anomaly detection with approaches varying from clustering, classification to reconstruction [18]. Recent Neural Network based anomaly detection research has revolved around Long Short-Term Memory Neural Networks and Auto-Encoders, both of which often leverage recurrent Neural Networks [18].

A. Anomaly Detection with LSTM Neural Network

Long Short-Term Memory (LSTM) Neural networks are a subcategory of Recurrent Neural Networks especially suited towards learning long-term dependencies between the input data. LSTM architecture is represented by memory blocks which themselves are essentially, recurrently connected structures [6].

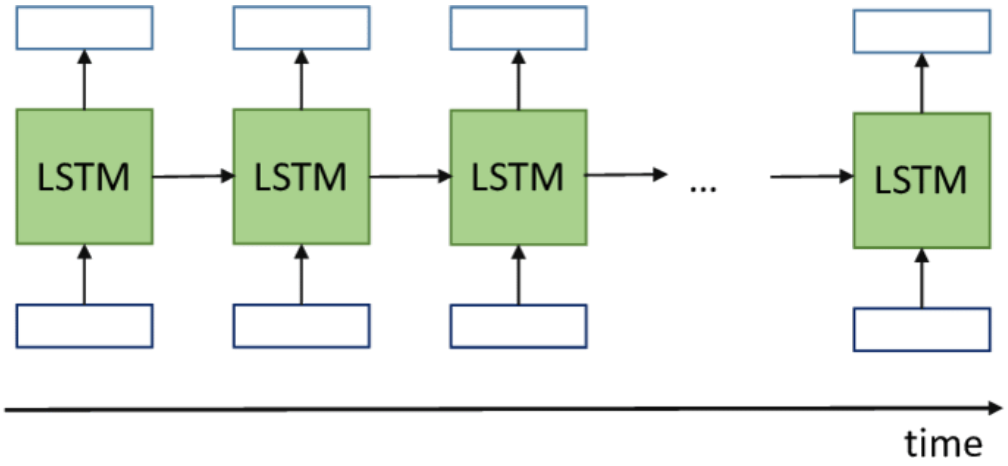

Figure 13. LSTM Architecture [32]

B. Anomaly Detection with Auto Encoder

Auto-Encoders are artificial neural networks designed to induce a representation for datasets by learning approximations of the dataset's identity function [19]. They are generally paired with a Decoder which is used to recreate the initial dataset using the representation defined by autoencoders. In terms of architecture, an autoencoder can be as simple as a feed-forward Neural Network which may or may not be recurrent [19].

The input and output layers are connected to each other with hidden layers. The basic approach to using Auto-Encoders revolves around using an Encoder to map an approximation of the dataset, which is then reconstructed by the decoder [19]. This reconstructed representation contains the most important features, at which point a reconstruction error is calculated to decipher anomalies.

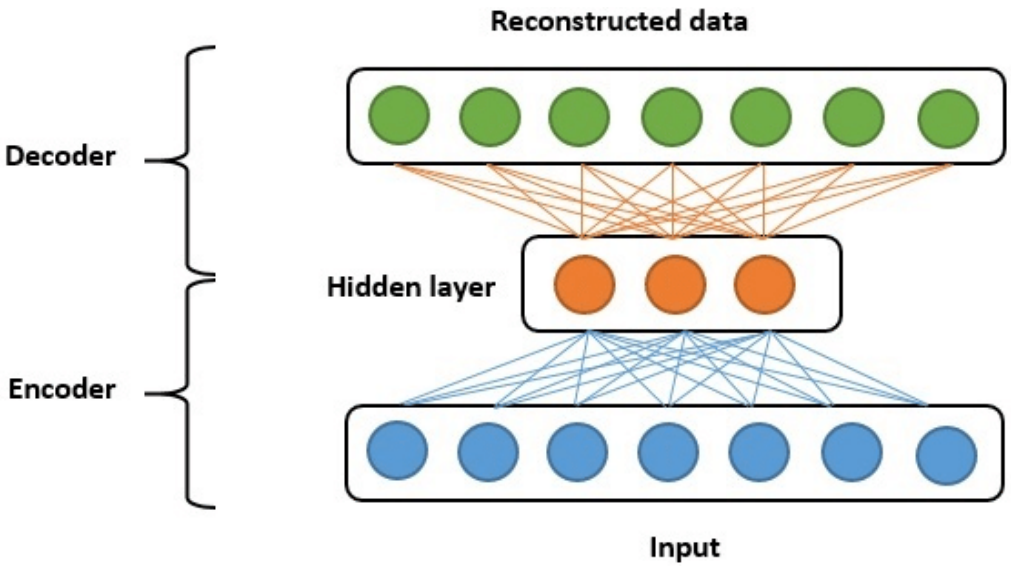

Figure 14. Auto Encoder Learning [33]

VI. SUMMARY OF THE CURRENT STATE-OF-ART

Machine learning, LSTM Neural Networks, and Auto-Encoders are designed to be used with data that is unbalanced, unstructured, unlabeled and domain specific. The heavy research in these areas combined with the ubiquity of commodity hardware has made these techniques accessible, powerful and very effective for anomaly detection [19].

Kumari et al leverage unsupervised K-Means Clustering on network traffic data to successfully detect anomalies with much improved accuracy over existing approaches [34].

Olsson et al. [8] developed an unsupervised learning approach to detect collective anomalies by deriving an “anomalous score” for each anomaly. They evaluate their model using an artificial dataset as well as two industrial datasets and successfully detect anomalies in moving crane data as well as fuel consumption data over time [8].

In [9], Malhotra et al. leverage LSTM Neural Networks for solving time series anomaly detection. Stacked LSTM were trained on non-anomalous data and this trained model was used to predict the next n number of time steps. This was compared with actual data for the next n

time steps to successfully derive anomalies.

Marchi et al. [10,11] combined non-linear predictive denoising autoencoders (DA) with LSTM cells to successfully identify anomalous signals in audio data. They leveraged previous frames to predict auditory spectral features for the next short-term frame. This combined model was tested using acoustic recorders with non-anomalous signals showing up average reconstruction error and anomalous signals showing up higher reconstructive errors. An anomaly score was then created using reconstruction error and a threshold was identified to indicate anomalies above it. Using public datasets, their model outperformed competing, existing models. [10,11]

Sakurada et al. [20] use auto encoders as dimensionality reduction tools to detect anomalies in both real world and artificial datasets. The datasets used are time series based and results from their Auto Encoder architecture are compared in detail with legacy spectral decomposition techniques such as PCA. Sakurada et al. [20] also compared their results with denoising autoencoder and compared results with both the standard Auto Encoder and PCA techniques. In their experiments, Auto-Encoders successfully detected subtle anomalies which were missed out by PCA. Further, denoising Auto-Encoders provided greater accuracy as well as successful detection of subtle anomalies [20]. The authors also noted that as opposed to kernel PCA techniques, Auto-Encoders require fewer computational cycles [20].

As our literature review shows, Unsupervised machine learning, LSTM Neural Networks, and Auto-Encoders are well suited to detecting point, collective and contextual anomalies in application log data. Further, these state of the art techniques are being used independently and across domains to successfully detect anomalies in datasets that share characteristics with application log data.

VII. HYPOTHESIS AND CONTRIBUTION

Combining the reconstruction properties of Auto-Encoders with the contextual efficacy of LSTM Neural Networks can lead to improvements in application log data anomaly detection. Our contribution is to develop a combined LSTM-AE architecture for application log anomaly detection, that can improve on existing state of the art in terms of accuracy and generalization.

The LSTM-AE (LSTM Auto Encoder) must be very accurate in order to avert system failure or slowdown. It is also imperative that the combined LSTM-AE generalize well over application log data for different applications. The experiments will thus measure accuracy and generalization of our LSTM-EA and benchmark against existing techniques such as unsupervised machine learning and LSTM Neural networks.

VIII. EXPERIMENT SETUP

A. Datasets

As our work is related to application log data, it becomes critical to ensure high-quality production datasets for conducting experiments. However, live application Log Data can be challenging to obtain due to a multitude of privacy and security concerns. By perusing the literature on the subset, we were able to identify public Log Datasets. We have selected two distinct datasets for our experiments in order to measure generalization as well as accuracy.

Application	Range	Dataset Size	Log Message Count	Identified Anomalies
HDFS	38.7 Hours	1.55 G	11,175,629	16,838
BGL	7 Months	708 M	4,747,963	348,460

Table 1. Dataset Details [25]

The HDFS Dataset has been obtained from a production Amazon EC2 system and is well suited for anomaly detection. Not only does it contain 11,175,629 log messages with 16,838 anomalies, the dataset is also pre-labeled by domain experts, which will be useful for identifying correctness of algorithms [25]. The dataset varies significantly from BGL log dataset both in terms of total size of the dataset as well as the proportion of anomalies.

The BGL Dataset has also been obtained from a production system and is inherently well suited for anomaly detection. Not only does it contain 4,747,964 log messages with 348,460 anomalies, the dataset is also pre-labeled by domain experts, which is critical for evaluation. This dataset is more balanced than the HDFS dataset which will aid our analysis and allow testing for generalization. It is also smaller in size to the HDFS dataset which positions it perfectly to test generalization with a lower quantity of data.

B. Implementation Details

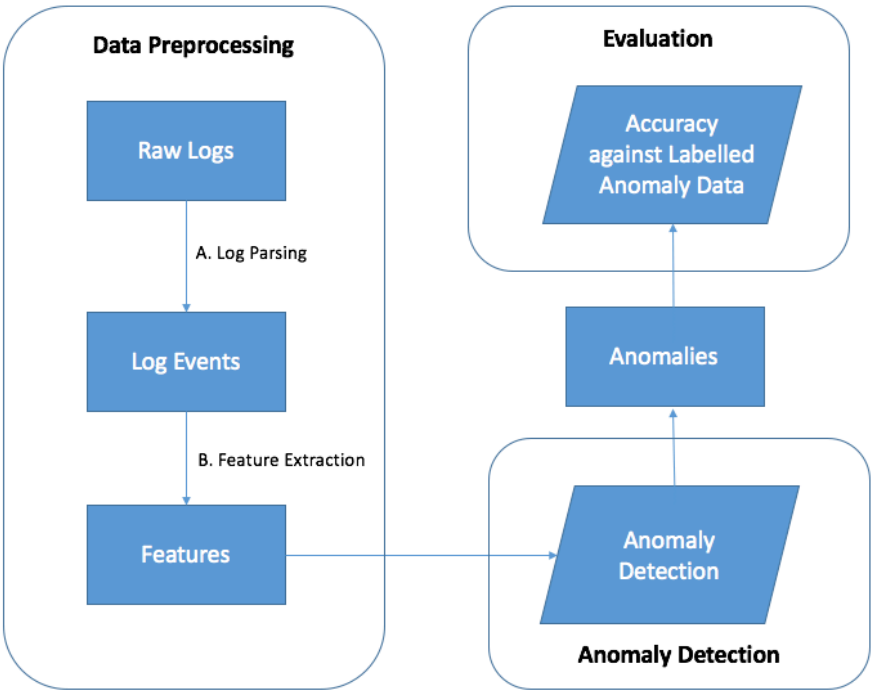

Figure 15. Experiment Implementation Flow

The first step of our experiment implementation is Data preprocessing, where the raw log data is transformed into Features that can be ingested by our Anomaly detection algorithms. Raw data is used as input to the Log Parsing phase. Log parsing removes extraneous or execution specific detail from the raw data and the output is used as input for the Feature extraction phase. At this stage, parsed log data is converted to numerical features which are used as input into our anomaly detection algorithm, which then identifies anomalies based on the technique in use [25]. These anomalies are then cross-validated with the domain expert labeled list of anomalies for each dataset to identify false positives, false negatives, true positives and true negatives in order to derive precision, recall, and F1 score metrics.

C. Data Preprocessing

As can be seen below, the initial Dataset contains unstructured time series data, which need to consolidate into single events based on common criterion (block number for HDFS and timestamp for BGL). This is done as follows:

1) Raw Log Data is used as input for Log Parsing.

081109 204015 308 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_8229193803249955061 terminating

081109 205019 308 INFO dfs.DataNode$DataXceiver: Received block blk_8229193803249955061 from /10.252.194.69

081109 206017 308 INFO dfs.DataBlockScanner: Verification succeeded for block blk_8229193803249955061

The log parsing stage removes runtime details such as IP address, log type(INFO) and thread ID (308). The Logs now contain only constant data, no runtime variables and are ready to be used as input to the Feature extraction phase of data preprocessing as follows.

081109 204015: PacketResponder * for block * terminating
081109 205019: Received block * from *
081109 206017: Verification succeeded for block *

2) Log events created during parsing are grouped together and used as input to Feature extraction.

081109 204015: PacketResponder * for block * terminating
081109 205019: Received block * from *
081109 206017: Verification succeeded for block *

Once we have isolated all log events for a particular block, we count the frequency of these events to create an event count vector.

3) Numerical Features are created using frequency of events

For block 8229193803249955061 the event vector is [3,2,2,0,0,0,0,0,0,0…].

This is a numerical representation of three Packetresponder events, two received block events and two verification succeeded events for block 8229193803249955061. The zeroes represent events which did not take place for block 8229193803249955061. This process is repeated for the entire dataset and event vectors are created for input to the anomaly detection algorithm.

D. Evaluation Metrics

Our primary focus for this work will be on accuracy and generalization. Keeping this in mind, three metrics have been chosen to best evaluate performance on both these parameters. The metrics are as follows

- Precision: precision gives us information about a model's performance with respect to false positives. Our purpose is to perform anomaly detection to capture the right anomalies and as few false positives as possible. Precision is designed to measure this and is a domain specific metric.

$$precision = \frac{|\{relevant\ documents\} \bigcap \{retrieved\ documents\}|}{|\{retrieved\ documents\}|}$$

$$recall = \frac{|\{relevant\ documents\} \bigcap \{retrieved\ documents\}|}{|\{relevant\ documents\}|}$$

Figure 16. Precision and Recall [31]

- Recall: Recall gives us information about a model's performance with respect to false Negatives. Our purpose is to perform anomaly detection to ensure we capture most or all anomalies and as few false Negatives as possible. Recall is designed to measure this and is a domain specific metric.
- F1 Score Indicates the Harmonic mean of Precision and recall and represents a conservative average less affected by extreme values. The F1 Score is a combination of Precision and recall and represents the tradeoff between precision and recall values, thus providing an effective, domain-specific metric for our evaluation. The harmonic mean is calculated as follows.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figure 17. Harmonic mean (F1 Score) Formula [31]

Precision, recall and F1 Score suit the best for our experiments due to the unbalanced nature of our dataset and are also specific to the domain of classification of which anomaly detection is a subset

IX. EXPERIMENTS AND ANALYSIS

A. Experiment 1: Unsupervised Machine Learning (K-Means Clustering)

This algorithm starts K-Means clustering with K equal to one. The algorithm then iteratively increases K (clusters) until the desired K is reached. At each iteration, we compare anomalies identified with actual anomalies in the dataset. The number of clusters is increased until the accuracy metrics peak, hence the chosen K value represents the cluster size at which accuracy metrics are highest and start dropping as cluster size is increased further. Once complete, the identified anomalies are compared with total anomalies to derive a confusion matrix from which the following results follow.

a) Results

K	Precision	Recall	F1
1	0.55	0.60	0.57
2	0.60	0.63	0.61
3	0.69	0.69	0.69
4	0.69	0.69	0.69
5	**0.75**	**0.70**	**0.72**
6	0.74	0.69	0.71
7	0.74	0.68	0.71

Table 2. K-Means HDFS Dataset Results

K	Precision	Recall	F1
1	0.65	0.70	0.67
2	0.70	0.73	0.71
3	0.75	0.76	0.75
4	**0.78**	**0.79**	**0.78**
5	0.78	0.77	0.77

Table 3. K-Means BGL Dataset Results

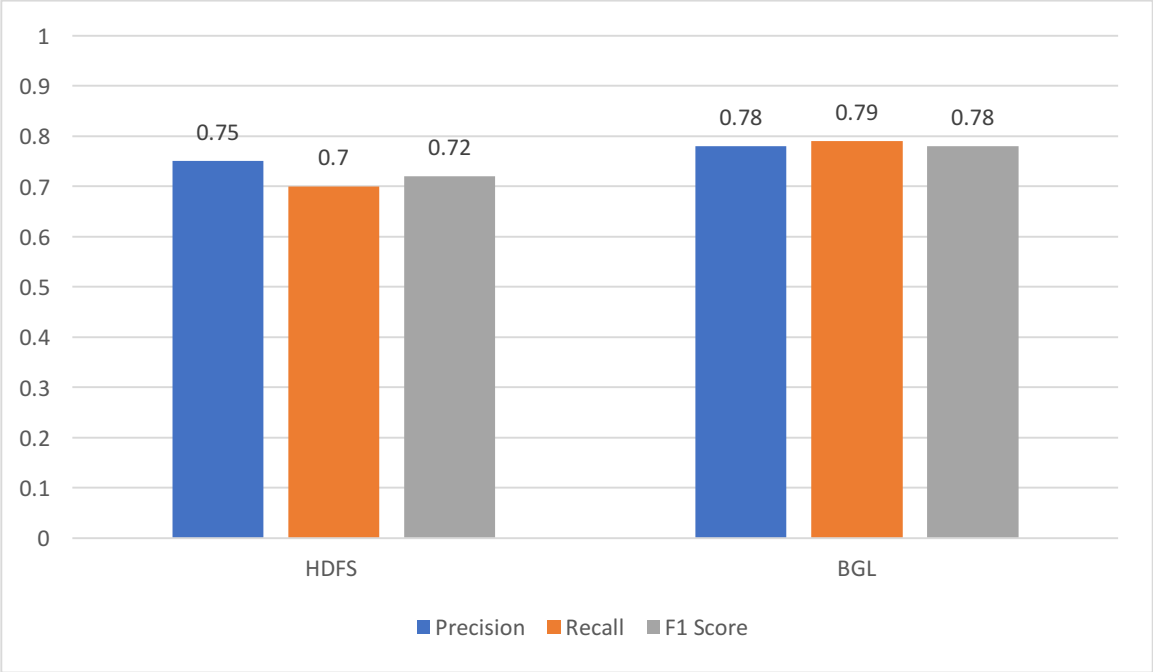

Figure 18. Results for K-Means Algorithm with HDFS and BGL datasets

b) Analysis

The accuracy metrics for the K-Means algorithm are highest for HDFS Dataset with F1 score of 0.72 and for BGL Dataset with F1 score of 0.78. This acts as a baseline accuracy for further experiments to improve upon. As can be seen from Fig 18, the K-Means algorithm does not generalize well across datasets. The F1 score is 8.3% lower for the HDFS dataset than the F1 Score for the BGL dataset which means that the algorithm's effectiveness varies with the dataset and is less effective for the HDFS dataset. Also, the F1 Score is highest at cluster size 5 for HDFS dataset and highest at cluster size 4 for BGL dataset which also means that the algorithm needs to be customized for each dataset and hence does not generalize well.

B. Experiment 2: Unsupervised Machine Learning (K-NN Global Density based)

In the absence of Labelled data for training, we cannot determine an appropriate K value through experimentation. Instead, we perform the experiment for values of K from 10 to 50 and combine all unique anomalies identified with each K value. We limit the K value to 50 as any

further increase does not result in improved results as per Table 4, Table 5. These combined anomalies are compared with total anomalies to derive the following results.

a) Results

K	Precision	Recall	F1
10	0.66	0.7	0.68
20	0.7	0.72	0.71
30	0.71	0.73	0.72
40	0.74	0.77	0.75
50	0.74	0.77	0.75
Avg (10=< K <=50)	**0.71**	**0.74**	**0.72**

Table 4. K-NNs HDFS Dataset Results

K	Precision	Recall	F1
10	0.61	0.66	0.63
20	0.68	0.71	0.68
30	0.7	0.74	0.70
40	0.75	0.76	0.75
50	0.75	0.76	0.75
Avg (10=< K <=50)	**0.70**	**0.73**	**0.71**

Table 5. K-NNs BGL Dataset Results

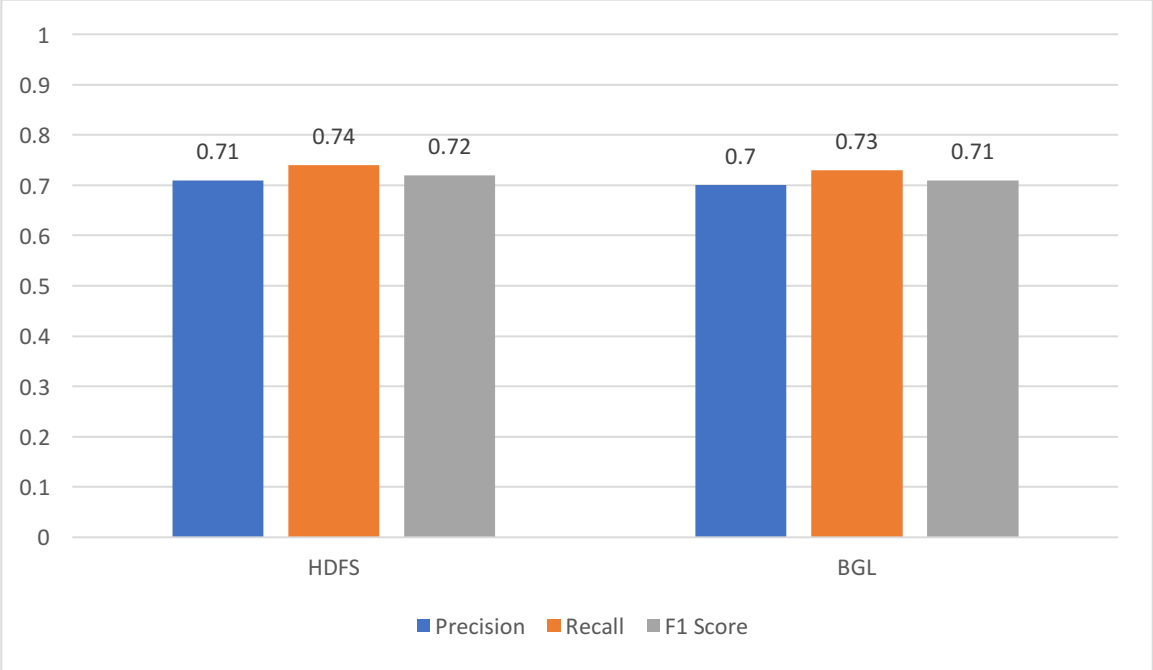

Figure 19: Results for K-NN Algorithm with HDFS and BGL datasets

b) Analysis

As we can see from Fig. 19, the F1 score for HDFS dataset of 0.72 is the same as that achieved with K-Means and that the F1 score for BGL dataset of 0.71 is lower than that achieved by K-Means by 10%. Thus KNN results in lower accuracy metrics as compared to K-Means.

As we can also see from Fig. 19 the KNN algorithm improves generalization over the K-Means algorithm. The F1 score is 1.4% lower for the HDFS dataset than the F1 Score for the BGL dataset which means that the KNN algorithm's effectiveness varies little across datasets compared to the K-Means algorithm. As a part of the KNN Algorithm, results across multiple K values are averaged together, which also means that the algorithm does not require customization compared with K-Means.

C. Experiment 3: LSTM Neural Network

We use LSTM Neural Network for anomaly detection in time series. A two-layer unidirectional LSTM neural network is trained on non-anomalous data and used as a predictor. <Distribution><1.5 funda> The resulting prediction errors are used to assess the likelihood of the anomalous behavior. We also vary Neurons in each of the two layers to determine the combination that achieves the highest accuracy.

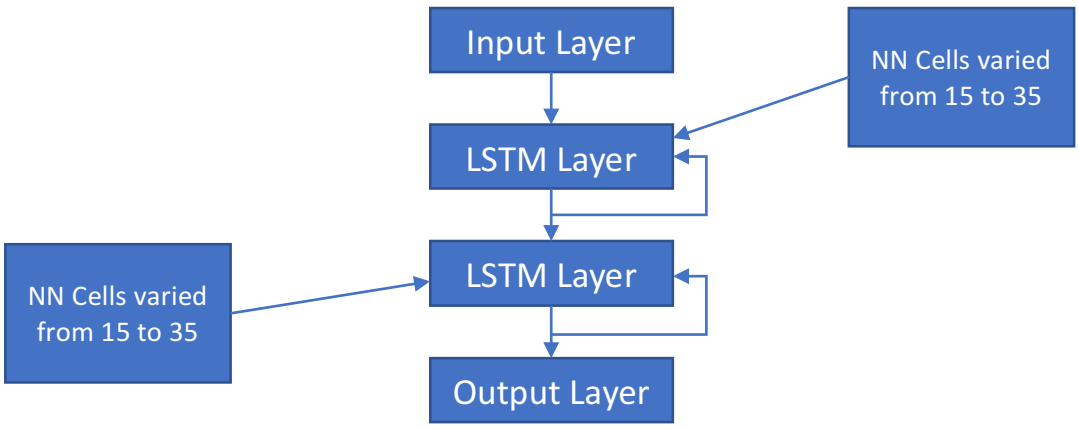

Figure 20. Organization of the two LSTM Layers and Layer Detail

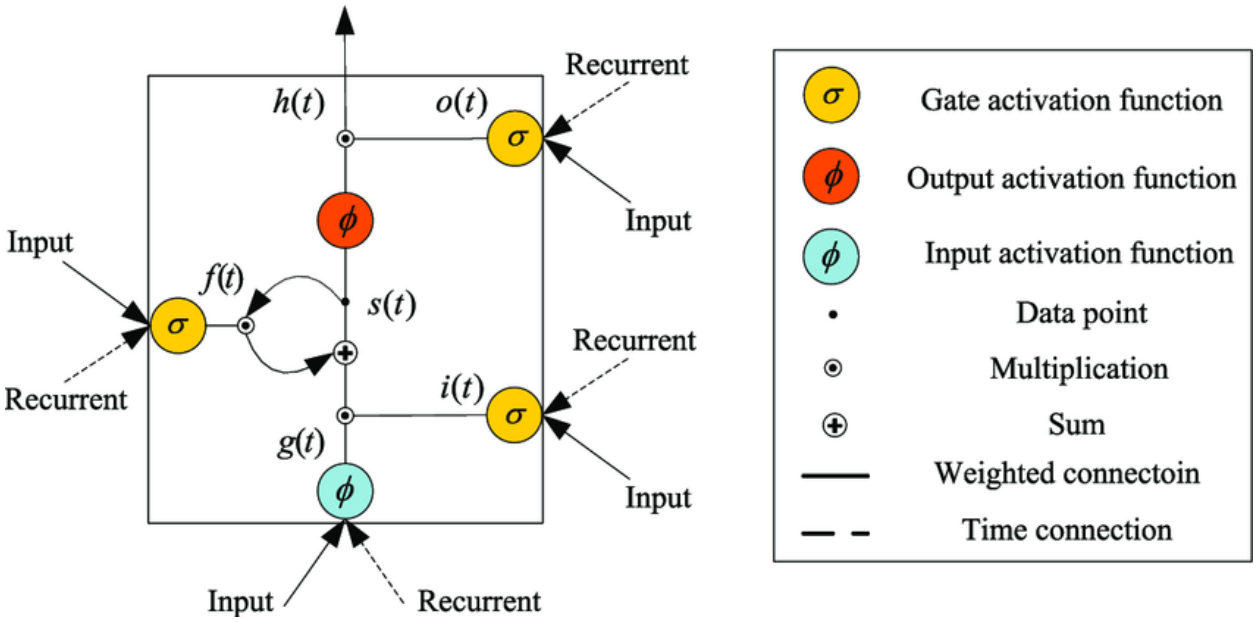

Figure 21. Single LSTM Layer Structure [30]

a) Results

NN Cells (Layer 1) – NN Cells (Layer 2)	Precision	Recall	F1
35-35	0.83	0.83	0.83
35 - 25	**0.85**	**0.83**	**0.84**
25 - 35	0.81	0.79	0.80
15-30	0.82	0.82	0.82
30-15	0.82	0.80	0.81

Table 6. LSTM HDFS Dataset Results

NN Cells (Layer 1) – NN Cells (Layer 2)	Precision	Recall	F1
35-35	0.85	0.84	0.84
35 - 25	**0.89**	**0.86**	**0.87**
25 - 35	0.83	0.82	0.82
15-30	0.82	0.82	0.82
30-15	0.81	0.80	0.80

Table 7. LSTM BGL Dataset Results

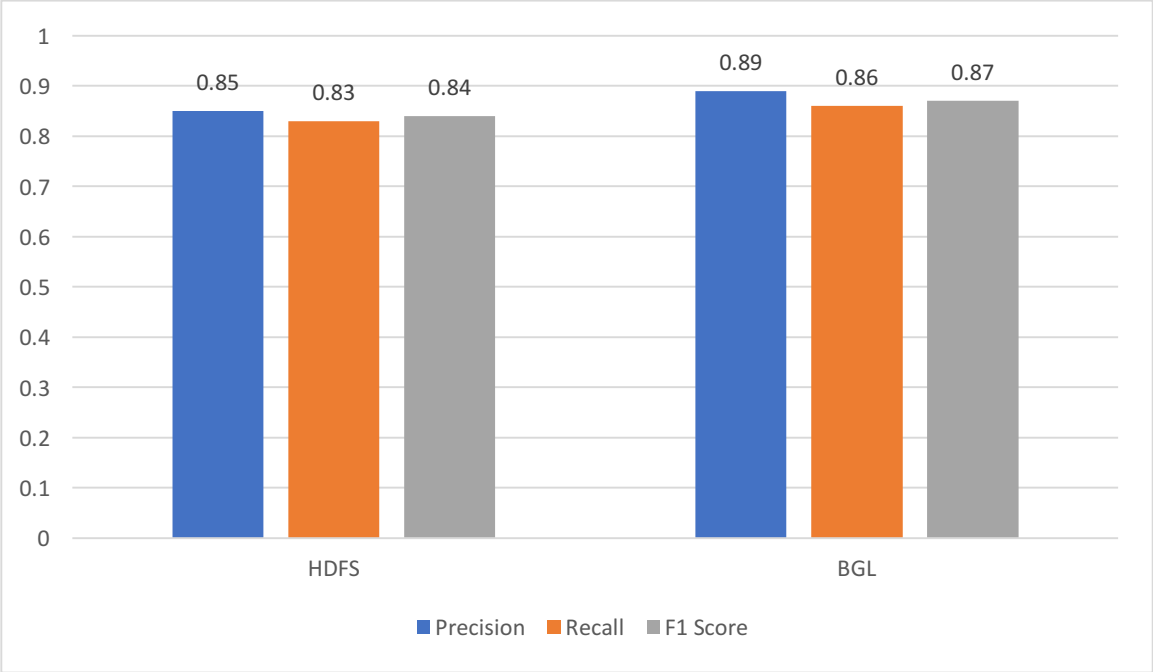

Figure 22. Results for LSTM with HDFS and BGL datasets

b) Analysis

As we can see from Fig. 22, the F1 score for HDFS dataset of 0.84 is the highest we've achieved and improves upon the KNN and K-Means score (0.72) by 16.7%. The same is true for the BGL dataset with an F1 Score of 0.87 representing an increase of 11.5% from K-Means. Also as we can see from Table 6 and Table 7, the highest F1 scores are obtained on the same architecture with 35 – 25 NN Cells, which means that just like KNN, the LSTM Neural Network does not require customization compared with K-Means. As we can also see from Fig. 22 the LSTM Neural Network does not improve generalization over the K-NN algorithm. The F1 score is 3.6% lower for the HDFS dataset than the F1 Score for the BGL dataset which means that the algorithm's effectiveness varies with the dataset and is less effective for the HDFS dataset.

D. Experiment 4: LSTM Auto Encoder (LSTM-AE)

An encoder learns a vector representation of the input features and the decoder uses this to

reconstruct the time-series. The LSTM-based encoder-decoder is trained to reconstruct instances of ‘normal' time series with the target time-series being the input time-series itself. Then, the reconstruction error at any future time instance is used to compute the likelihood of anomaly at that point. In order to accomplish this, we use a two-layer LSTM as used earlier and interface it with LSTM Auto Encoder Decoders. We also vary Neural Network Cells in each of the two layers to determine the combination that achieves the highest accuracy.

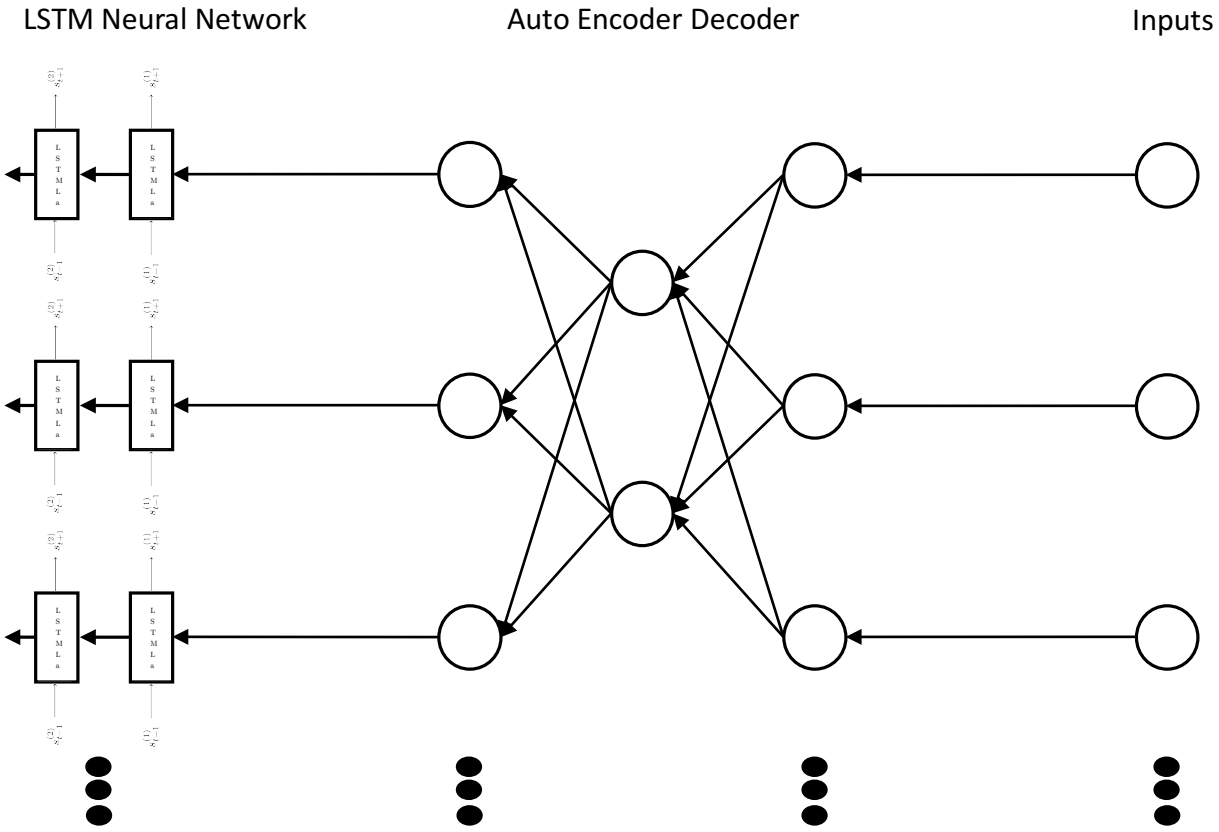

Figure 23: LSTM-AE Abstraction

An encoder-decoder model learned using only the normal sequences can be used for detecting anomalies in multi-sensor time-series: The intuition here is that the encoder-decoder pair would only have seen normal instances during training and learned to reconstruct them. When given an anomalous sequence, it is not able to reconstruct it well, and hence leads to

higher reconstruction errors compared to the reconstruction errors for the normal sequences. Analyzing these reconstruction errors leads to the following accuracy results.

a) Results

NN Cells (Layer 1) – NN Cells (Layer 2)	Precision	Recall	F1
35-35	0.88	0.87	0.88
35 - 25	**0.89**	**0.88**	**0.88**
25 - 35	0.83	0.80	0.81
15-30	0.86	0.83	0.85
30-15	0.85	0.81	0.83

Table 8. LSTM HDFS Dataset Results

NN Cells (Layer 1) – NN Cells (Layer 2)	Precision	Recall	F1
35-35	0.87	0.88	0.87
35 - 25	**0.89**	**0.89**	**0.89**
25 - 35	0.85	0.81	0.83
15-30	0.86	0.82	0.84
30-15	0.85	0.81	0.83

Table 9. LSTM BGL Dataset Results

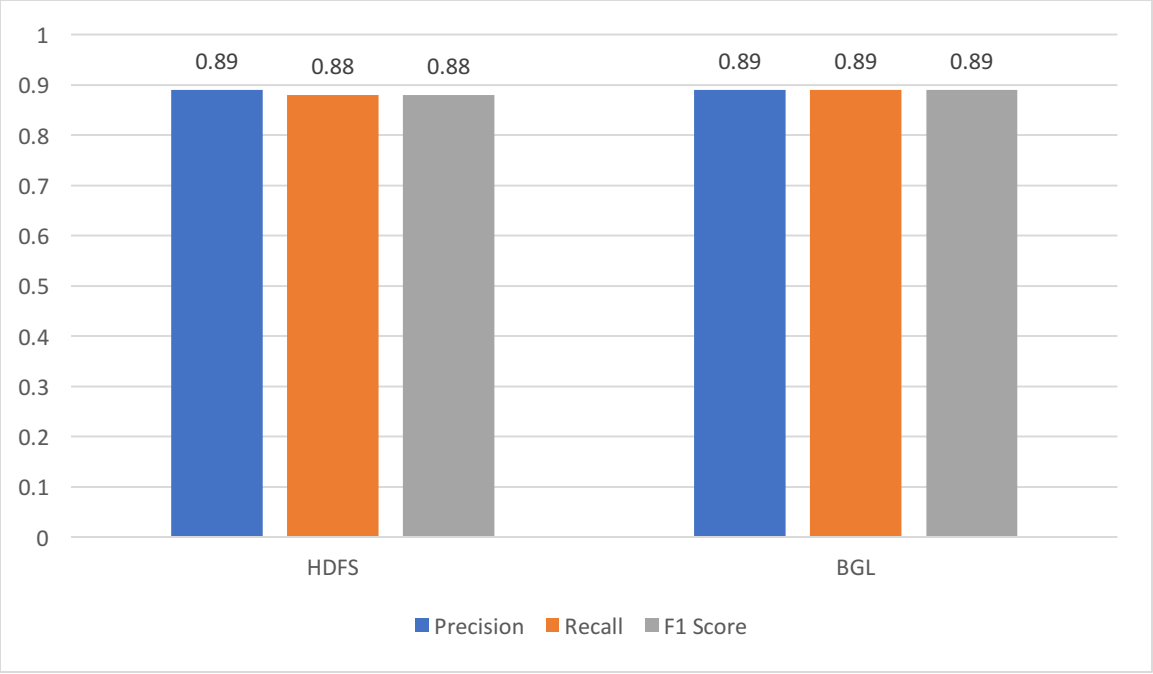

Figure 24. Results for LSTM-AE with HDFS and BGL datasets

b) Analysis

As we can see from Fig. 24, the F1 score for HDFS dataset of 0.88 is the highest we've achieved and improves upon the LSTM score by 4.7%. The same is true for the BGL dataset with an F1 Score of 0.89 representing an increase of 2.2% from LSTM.

Also as we can see from Table 8 and Table 9, the highest F1 scores are obtained on the same architecture with 35 – 25 NN Cells, which means that just like LSTM, the LSTM-AE Neural Network does not require customization, unlike K-Means. As we can also see from Fig. 24 the LSTM Auto Encoder Neural Network improves generalization over the LSTM Neural Network. The F1 score is only 1.1% lower for the HDFS dataset than the F1 Score for the BGL dataset which means that the algorithm's effectiveness varies the least across datasets compared to the rest of our algorithms.

X. SUMMARY OF RESULTS

In our experiments, we compared the accuracy and generalization of four anomaly detection approaches including Unsupervised K-Means, Unsupervised K-NN, LSTM, and LSTM-AE. The LSTM-AE scored the highest accuracy for HDFS (Precision: 0.89, Recall: 0.88, F1: 0.88) and BGL (Precision: 0.89, Recall: 0.89, F1: 0.89). LSTM-AE also generalized the best across our two datasets with only 1.1% variation between results on both datasets.

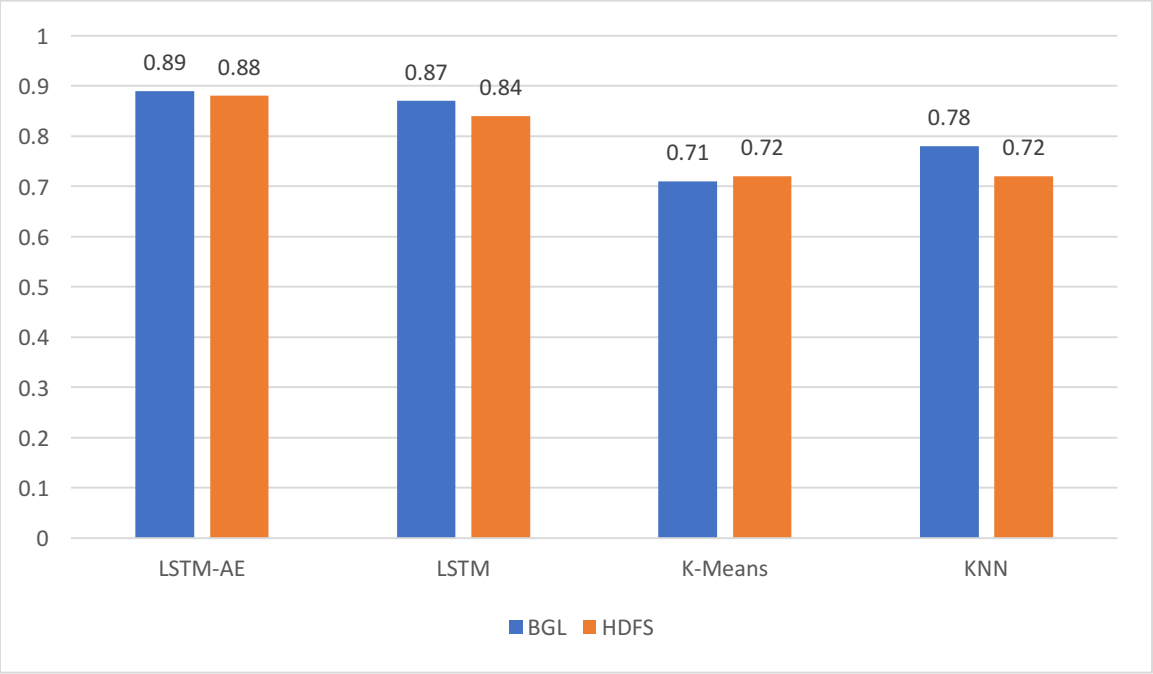

Figure 25. Summary of Results for all experiments

In [25] Shillin et al use Unsupervised Machine Learning, with the same input datasets of HDFS and BGL. Their Invariant Mining approach [25] outperformed our LSTM-AE with respect to accuracy, achieving for HDFS (Precision: 0.88, Recall: 0.95, F1: 0.91) and BGL (Precision: 0.83, Recall: 0.99, F1: 0.91). However, LSTM-AE generalizes better than [25]. LSTM-AE has only 1.1% variation between the Precision and Recall of HDFS and BGL datasets as opposed to a variation of 6% between Precision and recall for the invariant Mining approach used in [25] also using the same HDFS and BGL datasets.

In [26] Du et al use Deep LSTM Neural Networks with online validation and the same input datasets of HDFS and BGL. Their Deep Learning approach [26] outperformed LSTM-AE in accuracy, achieving for HDFS (Precision: 0.95, Recall: 0.96, F1: 0.96) and BGL (Precision: 0.82, Recall: 1.0, F1: 0.90). However, LSTM-AE generalizes better than [26]. LSTM-AE has only 1.1% variation between the Precision and Recall of HDFS and BGL datasets as opposed to a variation of 8% between Precision and recall for the Deep Learning approach used in [26] also

using the same HDFS and BGL datasets.

XI. CONCLUSION

Over the course of this study, we answered our research questions by investigating anomalies and anomaly detection methods. We also explored Neural networks and state of the art techniques to detect anomalies in application log data. Leveraging this research, we hypothesized a hybrid LSTM Auto Encoder model with the aim of achieving improvements in accuracy and generalization. We conducted experiments of existing state of the art K-Means Unsupervised, KNN Unsupervised, LSTM and compared the accuracy and generalization with our contribution of LSTM-AE. Through these experiments and their analysis, we discovered that the LSTM-AE does improve accuracy and generalization. We also compared our results with those obtained by the scientific community using the same input datasets. In this case, we noted that although the LSTM-AE did not improve accuracy, it did deliver improved generalization.

XII. FUTURE WORK

The primary aspect of future work revolves around increasing the accuracy of our LSTM-AE implementation while ensuring that generalization is maintained. This can be achieved by experimenting with Bidirectional LSTM Neural Networks as opposed to our existing Unidirectional implementation, increasing the number of LSTM layers to take advantage of Deep learning, or even incorporating online learning as has been done in [26].

Apart from this, a primary objective of future work would be to catalog and measure computation time for each of our experiments to determine real-world effectiveness alongside accuracy and generalization. The purpose of this research area is to minimize time and resources spent on application log anomaly detection and thus a comprehensive review should be

undertaken of space complexity, runtime complexity, resource utilization and time taken during execution.

Another aspect of future work would be testing our LSTM-AE architecture across multiple datasets to determine if the generalization ability holds across a wider variety of Application log data.

REFERENCES

[1] A. Katal, M. Wazid, and R. H. Goudar. "Big data: issues, challenges, tools and good practices." in Contemporary Computing (IC3), 2013 Sixth International Conference on, pp. 404-409. IEEE, 2013.

[2] IM. Chen, S. Mao, and Y. Liu. "Big data: A survey.", in Mobile Networks and Applications 19, no. 2 (2014): pp. 171-209, 2014.

[3] H. Zawawy, K. Kontogiannis, and J. Mylopoulos. "Log filtering and interpretation for root cause analysis.", in Software Maintenance (ICSM), 2010 IEEE International Conference on, pp. 1-5. IEEE, 2010.

[4] Ahmed, Mohiuddin, Abdun Naser Mahmood, and Md Rafiqul Islam. "A survey of anomaly detection techniques in financial domain." Future Generation Computer Systems 55 (2016): 278-288.

[5] Martí, Luis, Nayat Sanchez-Pi, José Manuel Molina, and Ana Cristina Bicharra Garcia. "Anomaly detection based on sensor data in petroleum industry applications." Sensors 15, no. 2 (2015): 2774-2797.

[6] Nanduri, Anvardh, and Lance Sherry. "Anomaly detection in aircraft data using Recurrent Neural Networks (RNN)." In Integrated Communications Navigation and Surveillance (ICNS), 2016, pp. 5C2-1. IEEE, 2016.

[7] Bontemps, Loïc, James McDermott, and Nhien-An Le-Khac. "Collective Anomaly Detection Based on Long Short-Term Memory Recurrent Neural Networks."
In International Conference on Future Data and Security Engineering, pp. 141-152. Springer International Publishing, 2016.

[8] Olsson, T., Holst, A.: A probabilistic approach to aggregating anomalies for unsupervised anomaly detection with industrial applications. In: FLAIRS Conference. pp. 434–439 (2015)

[9] Malhotra, P., Vig, L., Shroff, G., Agarwal, P.: Long short-term memory networks for anomaly detection in time series. In: Proceedings. p. 89. Presses universitaires de Louvain (2015)

[10] Marchi, E., Vesperini, F., Eyben, F., Squartini, S., Schuller, B.: A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional lstm neural networks. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 1996–2000. IEEE (2015)

[11] Marchi, E., Vesperini, F., Weninger, F., Eyben, F., Squartini, S., Schuller, B.: Non-linear prediction with lstm recurrent neural networks for acoustic novelty detection. In: 2015 International Joint Conference on Neural Networks (IJCNN). pp. 1–7. IEEE (2015)

[12] Gunter, Dan, Brian L. Tierney, Aaron Brown, Martin Swany, John Bresnahan, and Jennifer M. Schopf. "Log summarization and anomaly detection for troubleshooting distributed systems." In Proceedings of the 8th IEEE/ACM International Conference on Grid Computing, pp. 226-234. IEEE Computer Society, 2007.

[13] Grubbs, Frank E. "Procedures for detecting outlying observations in samples." Technometrics 11, no. 1 (1969): 1-21.

[14] Gogoi, Prasanta, D. K. Bhattacharyya, Bhogeswar Borah, and Jugal K. Kalita. "A survey of outlier detection methods in network anomaly identification." The Computer Journal 54, no. 4 (2011): 570-588.

[15] Zheng, Yu, Huichu Zhang, and Yong Yu. "Detecting collective anomalies from multiple spatio-temporal datasets across different domains." In Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, p. 2. ACM, 2015.

[16] Markou, Markos, and Sameer Singh. "Novelty detection: a review—part 1: statistical approaches." Signal processing 83, no. 12 (2003): 2481-2497.

[17] Agyemang, Malik, Ken Barker, and Rada Alhajj. "A comprehensive survey of numeric and symbolic outlier mining techniques." Intelligent Data Analysis 10, no. 6 (2006): 521-538.

[18] Kozma, R., M. Kitamura, M. Sakuma, and Y. Yokoyama. "Anomaly detection by neural network models and statistical time series analysis." In Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on, vol. 5, pp. 3207-3210. IEEE, 1994.

[19] Andrews, Jerone TA, Edward J. Morton, and Lewis D. Griffin. "Detecting anomalous data using auto-encoders." International Journal of Machine Learning and Computing 6, no. 1 (2016): 21.

[20] Sakurada, Mayu, and Takehisa Yairi. "Anomaly detection using autoencoders with nonlinear dimensionality reduction." In Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, p. 4. ACM, 2014.

[21] Steinwart, Ingo, Don Hush, and Clint Scovel. "A classification framework for anomaly detection." Journal of Machine Learning Research 6, no. Feb (2005): 211-232.

[22] Nanduri, Anvardh, and Lance Sherry. "Anomaly detection in aircraft data using Recurrent Neural Networks (RNN)." In Integrated Communications Navigation and Surveillance (ICNS), 2016, pp. 5C2-1. IEEE, 2016.

[23] Ringberg, H.; Soule, A.; Rexford, J.; Diot, C. Sensitivity of PCA for traffic anomaly detection. In ACM SIGMETRICS Performance Evaluation Review; ACM: New York, NY, USA, 2007; Volume 35, pp. 109–120.

[24] Fujimaki, R.; Yairi, T.; Machida, K. An approach to spacecraft anomaly detection problem using kernel feature space. Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, Chicago, IL, USA, 21–24 August

[25] S. He, J. Zhu, P. He and M. R. Lyu, "Experience Report: System Log Analysis for Anomaly Detection," 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), Ottawa, ON, 2016, pp. 207-218. doi: 10.1109/ISSRE.2016.21

[26] Du, Min, F. Li, G. Zheng, and V. Srikumar. "DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning." In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285-1298. ACM, 2017.

[27] Hayes, Michael A., and Miriam AM Capretz. "Contextual anomaly detection framework for big sensor data." Journal of Big Data 2, no. 1 (2015): 2.

[28] Kriegel, Hans-Peter, P. Kröger, E. Schubert and A. Zimek. “LoOP: local outlier probabilities.” CIKM (2009).

[29] D. Mulder, Wim, S. Bethard, and M. Moens. "A survey on the application of recurrent neural networks to statistical language modeling." Computer Speech & Language 30, no. 1 (2015): 61-98.

[30] Guo, Liang, N. Li, F. Jia, Y. Lei, and J. Lin. "A recurrent neural network based health indicator for remaining useful life prediction of bearings." Neurocomputing 240 (2017): 98-109.

[31] Ji, Xiaonan, and Po-Yin Yen. "Using MEDLINE elemental similarity to assist in the article screening process for systematic reviews." JMIR medical informatics 3, no. 3 (2015).

[32] Twinanda, A. P., D. Mutter, J. Marescaux, M. de Mathelin, and N. Padoy. "Single-and Multi-Task Architectures for Surgical Workflow Challenge at M2CAI 2016." arXiv: 1610.08844 (2016).

[33] A. Ben Said, A. Mohamed, T. Elfouly, K. Harras and Z. J. Wang, "Multimodal Deep Learning Approach for Joint EEG-EMG Data Compression and Classification," 2017 IEEE Wireless Communications and Networking Conference (WCNC), CA, 2017, pp. 1-6

[34] Kumari, R., M. K. Singh, R. Jha, and N. K. Singh. "Anomaly detection in network traffic using K-mean clustering." In Recent Advances in Information Technology (RAIT), 2016 3rd International Conference on, pp. 387-393. IEEE, 2016.