IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 11, 2021, accepted December 28, 2021, date of publication January 7, 2022, date of current version January 18, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3141161

AnomalyAdapters: Parameter-Efficient Multi-Anomaly Task Detection

UĞUR ÜNAL AND HASAN DAĞ, (Member, IEEE)
Management Information Systems, Kadir Has University, 34083 Istanbul, Turkey
Corresponding author: Uğur Ünal (ugur.unal@khas.edu.tr)

This work was supported in part by The Scientific and Technological Research Council of Turkey (TUBITAK) under Grant 120E487.

ABSTRACT The emergence of technological innovations brings sophisticated threats. Cyberattacks are increasing day by day aligned with these innovations and entails rapid solutions for defense mechanisms. These attacks may hinder enterprise operations or more importantly, interrupt critical infrastructure systems, that are essential to safety, security, and well-being of a society. Anomaly detection, as a protection step, is significant for ensuring a system security. Logs, which are accepted sources universally, are utilized in system health monitoring and intrusion detection systems. Recent developments in Natural Language Processing (NLP) studies show that contextual information decreases false-positives yield in detecting anomalous behaviors. Transformers and their adaptations to various language understanding tasks exemplify the enhanced ability to extract this information. Deep network based anomaly detection solutions use generally feature-based transfer learning methods. This type of learning presents a new set of weights for each log type. It is unfeasible and a redundant way considering various log sources. Also, a vague representation of model decisions prevents learning from threat data and improving model capability. In this paper, we propose AnomalyAdapters (AAs) which is an extensible multi-anomaly task detection model. It uses pretrained transformers' variant to encode a log sequences and utilizes adapters to learn a log structure and anomaly types. Adapter-based approach collects contextual information, eliminates information loss in learning, and learns anomaly detection tasks from different log sources without overuse of parameters. Lastly, our work elucidates the decision making process of the proposed model on different log datasets to emphasize extraction of threat data via explainability experiments.

INDEX TERMS Anomaly detection, adapters, cyber threat intelligence, explainability, log, transfer learning.

I. INTRODUCTION

System security poses a big step for enterprises, governments, and safety critical systems. Adaptation of Industry 4.0 and IoT concepts open up more vulnerabilities, because the systems become more interconnected. In large-scale systems misidentifying an action can obstruct operations and negatively affect the maintenance of their services. Monitoring and analyzing threats is crucial as the state of technology grows rapidly. The more complex a system becomes, the harder it is to detect threats' behavior. Thus, scalable and flexible security solutions are required for an organization [1]. Anomaly detection systems are a part of Intrusion Detection or Prevention Systems (IDS/IPS), which are connected to different sources. A common practice is to use rule-based applications with the help of system administrators that are responsible for investigating events based on the threat intelligence. These types of approaches tend to fail, due to joined sources in a system yielding excessive data. Identifying anomalous behavior differs in sources and also is challenging considering streaming data in an online setting. Therefore, detecting anomalous events accurately and timely is crucial [18]. Log is accepted as an universal indicator of events for debugging and analysis purposes. They are designed to deliver information about an action and its related variables of a system. System logs are the main source of monitoring cyber incidents in real-time [6]. Continuous expansion of configurations of logs with each update to a system complicates sustaining the stability of defense mechanism.

Anomaly detection is the process of revealing undefined and abnormal actions in the system according to movements that are usually detrimental, predefined, or determined by an

The associate editor coordinating the review of this manuscript and approving it for publication was Shadi Alawneh.

observation [2]. This is a data-driven technique for investigating unexpected behaviors [6]. A log is a unstructured text that is designed for debugging and monitoring. It is stored in a text form for readability and convenience. Creating logs for readability produces an excessive number of instances and increases the difficulty of automation [8]. Moreover, it makes detecting anomalies harder with a combination of many sources [7].

Log mining, parsing, and anomaly detection techniques must evolve to capture a decisive intelligence. Anomaly detection studies can be divided into two categories: log key-based and semantic-based, according to how they use log data. As key-based methods, earlier works focused on static indicators or kill-chain analysis methods utilizing logs such as; PCA [9], invariant mining [10], and workflow monitoring [11]. DeepLog [12] approaches logs as an unstructured text and adopts text processing techniques to extract log templates(or keys) with a parsing tool [24]. It uses long short-term memory (LSTM) to predict next log keys via learning the current normal log event sequence from antecedent events. Furthermore, advances in deep networks started to lead anomaly detection studies. More recent studies oriented toward NLP techniques are able to extract contextual information. These are also called log semantic-based methods. LogAnomaly [16] and LogRobust [17] both utilize semantic information of log sequences with combination of their templates. Transformer [5] architecture brings promising results in various domains' problems and tasks, especially in text data. Thus, it suitable to be experimented in anomaly detection studies. HitAnomaly [15] indicates the instability of log parsing tools and combines semantic information with log's parameter values. Another recent work, Logsy [18] removes the need of log parsing tools to prevent information loss in yielding templates and uses transformer model with a multi-head attention mechanism. To achieve that, these semantic-based works utilize a pretrained embedding to transfer knowledge into anomaly detection task. Transfer learning methods are not signified between anomaly task domains; however the method of implementation can improve tasks in the existing environment. To that extent, we believe that anomaly detection studies based on log data can be improved via semantic information, which is enabled by transformer architecture. Besides, it can be optimized and adapted for applications in which multiple models need to be trained for anomaly tasks in an online setting.

In this study, we approach anomaly detection as a data-driven application and propose a task-based anomaly detection method considering a central system that manages multiple sources. To achieve that, we utilize an adapter-based learning in the detection model. Adapters were first introduced as transfer learning method for detecting visual representation [48], and later introduced for language processing for transformers [45]. Additionally, as discussed in [2] and [3], we study the types of anomalies in three categories: *point*, *conditional*, and *collective*. We are motivated by the advantages and versatility of transformer-based language models and propose a model for host-based anomaly detection systems. Considering each log as a sentence and system-calls as a language; our aim is to gain semantic information through adapters to distinguish anomalies. Using the nature of language models, we aim to use a multi-purpose approach, which is expandable to new sources without loss of information and overuse of parameters.

Our contributions can be summarized as follows:

- We utilize ROBERTa [31] English language model as a knowledge base, which is a robust version of the BERT architecture. In contrast to related studies, we use Byte-Pair Encoding [13] instead of WordPiece [14] in tokenization.
- Instead of a fully fine-tuning model, we have designed language and anomaly adapters for system logs to transfer knowledge without loss of information.
- We experimented on widening the applicability of anomaly detection in the systems. We designed multi-anomaly task detection using a combination of multiple adapters.
- We also presented explainability on our evaluation through gradient-based algorithms and visualized model decisions for investigation of cyber threat data.

II. BACKGROUND AND RELATED WORK

Anomaly detection is the activity to distinguish unmatched, peculiar, or unknown examples from the data [2]. This type of detection techniques are used in different applications such as; fraud detection in finance, intrusion detection in cyber security, fault detection in safety critical systems, and access control models [20] in critical infrastructures. These defense applications have a system-wide priority, since it is crucial to maintain their services. Analyzing system logs is also a way to understand runtime behavior. As an example, a peculiar network traffic flow at a workstation points out a port scan attack, which is an investigation attack by hackers to find open ways or check the state of security of an organization. In addition, a vast number of logs are created by complex systems constrain analyzes manually [21]. System operators usually investigate state of a system, but large number of attributes included in logs generate complexity prohibiting the understanding contextual information. Most solutions for anomaly detection are for a specific domain or problem, because the availability of the data for stating anomalous behavior is a problem [4]. As in the definition, detection of anomalies are simple; however, in application domain, it is very challenging. Key components of anomaly detection are detection techniques, problem characteristics, and the application source [19].

There are several categorization of the existing anomaly detection techniques, but one can confine them into; log template or key based, log semantic-based under the hood of supervised, and unsupervised methods [12], [15]–[18]. Key-based methods use log parsing tools to overcome free text problem and identify structured versions of logs as a template. There are two parsers that have been tested in

recent works. Spell is an unsupervised parsing method which operates based on longest common sub-sequence. Drain, named Drain3 with Python3 compatibility update,[1] is an online tree based parser with specific written rules [25]. Several setbacks appear in utilizing parser: requiring manual configurations and controlling rules become complexier, wrong parsed logs create false alarms due to the inability in capturing parameter values or actions [15], and acquired templates can cause loss of information [18]. Recent studies have mainly focused on capturing semantics from logs using pretrained embeddings to overcome these problems. It also means less processing requirement before a preparing detection model.

Considering anomaly detection as an NLP task, using pretrained word or sub-word embeddings greatly increases the accuracy instead of a sparse definition such as a one-hot representation. Word2vec [26] and fastText [27] are shallow deep network based language models used in the area. There are two types of usages in anomaly detection: pretrained embeddings for encoding directly or utilizing related algorithms to create a variant from scratch. Word Embeddings for Anomaly Classification (WEAC) method [29] extracts features from event logs through word embeddings, which indicate abnormal behaviors. Skip-gram and Continuous Bag of Words are used in training from scratch. So, Word2vec algorithm was used to gather vector representation of words. On the contrary, WEAC does not discard infrequent words, because it is important not to omit those for anomaly detection. LogAnomaly [16] presents template2vec algorithm which is based on the distributional lexical-contrast embedding (dLCE)'s method [28] to define word representation based on log sources from scratch. Produced vector representations are the inputs fed into LSTM model to detect anomalies. LogRobust [17] uses pretrained fastText embeddings, which is already trained on the Wikipedia dump.[2] It attempts to capture semantic information of log events and eliminates more parsing errors, due to provide better similarity in embedding space.

Natural language understanding methods have improved with the introduction of transformer-based LMs. BERT [30] is a pioneer language representation model trained on English Wikipedia and BooksCorpus in the pretraining stage. It is a masked language model that efficiently provides bidirectional semantics. It is greatly contributed in various NLP tasks, due to its fine-tuning ability to adapt downstream tasks. BioBERT [32], SciBERT [33] and NeuroBERT [34] are examples of variants of transferring knowledge in different domains. In anomaly detection studies, HitAnomaly [15] uses BERT for gathering word vector representations to build log sequence embeddings, then uses the information to distinguish anomalies within hierarchical transformer blocks. Logsy [18] uses its own tokenization method and creates a log vector token that is similar to '[CLS]' token presented in BERT paper. It represents a summary of a log event and identifies anomalous behavior with a transformer model.

There are two examples of transfer learning methods: feature-based and fine-tuning. The anomaly detection methods, we investigated, utilize feature-based transfer learning. They profit from pretrained embeddings to define log sequences' representations and are adapted into proposed deep learning architectures (LSTM, Bi-LSTM and Transformer). In procuring security of a complex system, central log monitoring tools are responsible for analyzing sequences from multiple and nonidentical log sources. Proposed deep networks need to adapt each different source, which relates to different tasks based on the source. In this process, both feature-based and fine-tuning present new updated weights for each task. This is an inefficient way considering transferred model's degree of sharing parameters. if we are up to create new models for each source or update learned weights, the processes cause loss of information also known as catastrophic forgetting [39]. In an online setting, streaming vast amount of log sources create a necessity to train new model for a new source sequentially without retraining shared models.

In our work, we focus on log semantic-based methods and improve anomaly detection as a downstream task. We utilize pretrained ROBERTa language model. In contrast to its predecessor (BERT), it uses a dynamic changing masking pattern, is able to support longer sequences and discards next sentences prediction task in pretraining [31]. By this way, the model indicates enhanced performance in post-training methods and downstream tasks in experiments [31]. To learn datasets and anomalies, we deploy adapter-based [45] transfer learning to create scalable and parameter-efficient model which is applicable to various log sources at once. We aimed to build a compact model, considering stream of log sequences as an input.

III. EXPERIMENTS

The proposed model is constructed as a pipelined flow. First, log events are gathered from system logs and prepared for language model training, then we prepare log language adapters for learning synthetic structure. Second, we prepare data structure of log sequences according to definition of anomalies, then we build structured logs for anomaly adapters. Third, we combine anomaly adapters(AAs) for multi-anomaly task objective. Lastly, we evaluate our experiments with related metrics and compare with recent studies, but *importantly* we test single-source and multi-source pipelines with explainability methods to understand model decisions and acquire feedback on treat data.

Our experiments are performed using a local AI-powered machine. We used a Volta-type architecture GPU with 16GB memory (3xNVIDIA RTX A4000-16GB) and Intel(R) Core(TM) i9-10900X CPU @ 3.70GHz. Volta architecture allows mixed precision ability in execution and enables faster iterations in our experiments. 'O1' option -Mixed Precision- is used (NVIDIA-Automatic Mixed Precision library [42]).

[1] https://github.com/IBM/Drain3
[2] https://dumps.wikimedia.org/

It means tensor-type calculations is made on FP16(fixed precision, 16) which are called white-listed operations. Moreover, black-listed operations are executed in FP32(fixed precision, 32) such as softmax. By this way, large-scale of logs trained and adapted to tasks more efficiently and timely [23].

Source code of experiments can be found on the github page.[3]

A. DATASETS

1) FIREWALL LOGS

The firewall dataset consists of 14,277,447 logs. Three days activity in a corporate network are simulated. We have used all log sequence except for the first day, which includes a Denial of Service (DoS) attack. We have extracted %0.01 of the abnormal event. Most of the data in first day is predominated by DoS attack, which we omitted and edited data without changing timeline of log events, since attack focuses on only several workstations in the network. 172,135 number of normal logs and 16,902 number of anomalous logs, which consist of DoS, Port scanning, worms and unknown machine connections. This dataset was also mentioned in finding a DoS attack at [12]. This dataset is particularly simulated for IEEE Visual Analytics Science and Technology (VAST) 2011 MiniChallenge-2.

We chose to introduce this dataset because of the explainability motivation aligned with existing Use of Policy Rules in documentation. Additionally, the dataset presents new type of anomalous events different than HDFS dataset, which also fits the expected scenario.

2) HDFS

Hadoop Distributed File Systems (HDFS) dataset was first presented in Xu *et al.*'s work [9]. It consist of 11,175,629 logs gathered from Amazon EC2 nodes. A total of 10,887,379 logs are tagged normal and 288,250 logs are tagged abnormal. Dataset can be found in LogHub, which is collection of system log datasets for AI-based analytics [40].

Both datasets include ground truth information about anomalous and normal behaviors. HDFS dataset includes labeled block IDs indicating which block's log sequence is anomalous. Firewall dataset can be found in challenge called Computer Network Operations at All Freight Corporation.[4] Reviewer documents and Use of Policy Rules for All Freight Corporation provide ground truth related to attacks in Firewall and other log files(such as; PCAP and IDS logs).

B. CLEANING DATA

Log sources are for controlling and analyzing system events. Those are prepared by system developers in nature of free text for readability concerns [18]. It is crucial to clean duplicated terms and augment symbolic information in the text without losing information. This process helps build a better knowledge base for the anomaly detection model.

In the firewall dataset, *message_codes* are inserted into log events for identification, as an index. Some event logs include source and destination IPs. They are written in parenthesis. Also, hex coded information can be found included in brackets. This represents duplication of information. We removed redundant text content and kept semantics intact. Symbolic presentation of event actions, e.g., '$-\ >$', is converted to 'to' in verbally describable form. In the HDFS dataset, event logs consist of headers which its content also is included in readable form. '*INFO dfs.FSNamesystem: BLOCK*..*' and '*WARN dfs. PendingReplicationBlocksPendingReplicationMonitor:..*' are some examples which are removed to prevent duplication. In this dataset, block information scripted in different forms, we merged block identifiers *'blk_-'* and *'blk_'* to *'blk'* for text regularization. These domain specific cleaning steps are applied to sources before building log vector representations.

C. PROCESSING

Logs can be considered unstructured or semi-structured type of text. We aim to gather much broader contextual information. To achieve that, processing data in our setup is two-folds; First, we prepare data for a log language model. Second, we prepare data for a log sequence anomaly detection model.

In log language model, we maintained *line by line* arrangement of the log events in firewall and HDFS datasets and applied cleaning steps. In this manner, we can learn contextual structure of an event log.

In anomaly detection, datasets' timeline and order of logs need to keep intact during preprocessing, since log order has a huge impact on defining anomalous events. In our definitions, see Figure-1, timeline is used to point out order, not specifically time that log occurs. Anomalous events differs in their data structure. In section *a)*, log events formed as $T = [t_1, t_2, .., t_N]$ such that, t_{n-x} is an event consists of semantic features. On timeline $n - x$, a log event has a abnormal token or token groups or whole log event. In section *b)*, log events are structured as $T = [t_1, t_2, .., t_N]$, such that t_{n-x+1} describes an event in the context of t_{n-x} and t_{n-x+2}. On the timeline, event flow should not step on t_{n-x+1} unless it is abnormal. In section *c)*, log events are structured as $T = [t_1, t_2, .., t_N]$ such that, log events collectively create unwanted behavior for system health between t_{n-x} and t_{n-y}. Contextual signs reveal anomalous behavior which spread through log sequences in point, conditional and collective anomalies.

From the point of language processing, each log line is processed in a distinct context based on anomaly type. In a simpler context, each line in log data set as $L = [f_1, f_2, .., f_N]$ such that $f_i, i \in [i, .., N]$. N is number tokens created by Byte-Pair Encoding (BPE) [13] and has similarities to WordPiece algorithm used in original BERT paper. Original BPE algorithm was used for compressing bytes. In this version of the algorithm, it combines most frequent characters to form

[3]https://github.com/uunal/anomaly-adapters

[4]http://vacommunity.org/Computer+Networking+Operations+at+All+Freight+Corporation

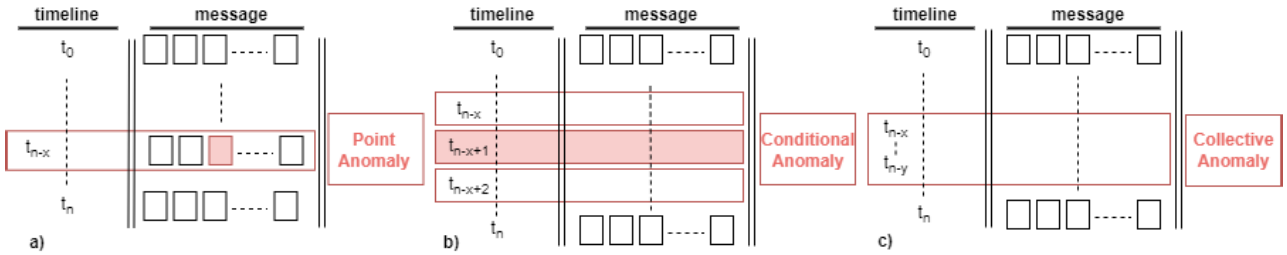


FIGURE 1. Processing log sources by anomaly types.

n-grams till whole words. Using vocabulary of ROBERTa language model, we prepared chunks of 512 tokens (maximum) via BPE for corresponding log sequence. Sequence of tokens is defined by behavior of log event. In HDFS dataset, this is determined using block ID. In firewall dataset, this is determined by normal and various anomalous events. For example, if f_x describes port scan attack, all continuation logs included in the chunk until max tokens are reached without splitting a log event.

IV. ANOMALY DETECTION MODEL

Anomaly detection system is a part of intrusion detection or security event monitoring (SIEM) tools [43]. Also, anomalous events are not predefined or not expected patterns in the normal activity [2]. The detection system analyzes log events within diverse range of sources and indicate anomalous patterns. To detect these patterns, we can explicate the problem as binary classification [36], [37].

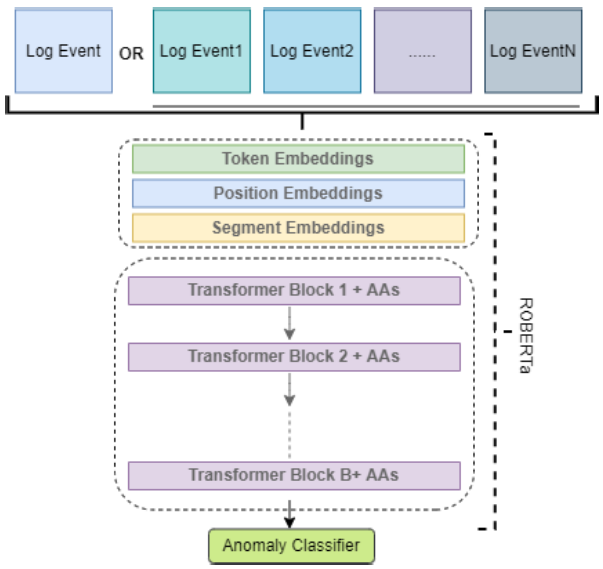


FIGURE 2. Overview of anomaly detection model.

Earlier log semantic-based approaches utilize mainly a feature-based transfer learning. Transformer-based variants' are good at learning from huge chunks of data and produce millions of parameters. Considering explosion of logs and nature of analysis, detection models need to adapt different (ab)normal behavior without retraining for each source. By this way, we prevent creating new parameters and forgetting information of the latter for each task [46].

Adapter fine-tuning is introduced for transformers architecture as explained in [45], aims to create a bottleneck in transformer block to restrain created parameters and ease sharing. We utilize ROBERTa as base model which has Θ parameters. This will be our shared parameters across learning log sources and anomaly detection tasks. Each task adapter introduces new parameters Φ and attached to corresponding transformer block n such that $n \in \{1, 2, .., T\}$, T is the number of transformer block used pretrained model (in our case, T = 12). To formulate, Φ is trained with loss function as L and used source data as D for each task, see (1). By this way, each task presents new set of parameters which contains %1-3.4 of the base model [45]. For task t = 0:

$$\Phi_0 \leftarrow \arg\min_{\Phi} L_0(D_0; \Theta, \Phi) \tag{1}$$

As in described in processing step, there two types of log data structure is created. First, we kept each log event separately in order to capture syntax in log language modeling. This process is only implemented in training language adapters for further composition with log anomaly adapters. Second, we formalise log sequences according to defined anomaly types, see Figure 1. Streamed log sequences are encoded with BPE tokenizer and fed into detection model.

In this work, we propose AnomalyAdapters which is a flexible, modular and parameter-efficient transformer-based model which provides transferring knowledge without losing learned parameters and sharing among tasks with adapter-tuning [45]. Our anomaly detection approach is two folds for a log source: log source language learning, anomaly task learning. Lastly, we propose multi-anomaly task detection with AdapterFusion [46] method to analyze multiple sources simultaneously.

A. LOG LANGUAGE ADAPTERS

Language modeling is required to comprehend distribution of a log source [38]. Masked Language Model(MLM) training improves base model to represent syntactic structure of a downstream task [41]. Therefore, building log source's language model expedites comprehending semantics of log events. In MLM objective, randomly selected tokens in the

log event. From that selected tokens, %80 of them replaced with *[MASK]* special token, %10 of them unchanged and %10 of them changed with token in the vocabulary [31]. In MLM training, cross-entropy loss function is used for optimization of the model. In (2), it aims to learn q distribution from inputed log event to true distribution of p in log source. D_{KL} denotes Kullback–Leibler (KL) divergence from p to q, and training attempts to minimize divergence [38].

$$\begin{aligned} H(p,q) &= -\sum_{x} P(x) log P(x) - \sum_{x} P(x) log \frac{q(x)}{p(x)} \\ &= H(p) + D_{KL}(p|q) \end{aligned} \tag{2}$$

In log language adapter (LLA) training, we kept original ROBERTa model implementation from Huggingface [44] and add adapter modules into transformer blocks using Adapters' library [47]. We are using language adapter which introduced in [49]. It is able to learn language specific transformations, and we utilizing to adapt various log types. Adapter modules are optimized and actual weights of base model are frozen during training. This way we efficiently create less parameters in tuning. In Figure 3, we have shown how log language adapter module is added into transformer block. We aim to transfer the information into distinguishing anomalous activities.

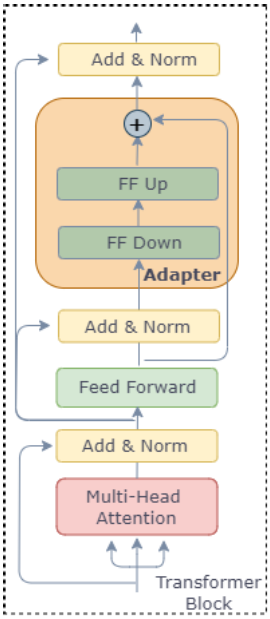


FIGURE 3. Log source's language adapter inside transformer block [46].

B. LOG ANOMALY DETECTION

In this section, we provide an architectural addition to adapt anomaly detection in log sequence representations. Adapters are able to create composition blocks in order to share information at ease, see Figure 4. Language adapters are intended to capture source specific knowledge. Furthermore, task adapters aim to learn downstream task. In our setup, anomaly detection is the second-order downstream task which adapting behavior of log sequences [49]. Anomaly adapters learns these behaviors in a binary classification setup. In this step of training, only log anomaly adapter(LAA) is activated and optimized. Thus, Log LA and transformer weights are kept frozen.

In (3), *LLA* includes a down-projection to *hxd* where h is the hidden size of the model and d is the adapter's dimension with a ReLU activation afterwards. Finally an up-projection to *dxh* is applied. The output of the log LA is fed into a down projection again with following a swish activation function. Then, up-projection is applied again to match dimensions with h layers. In addition, r indicates residual value from transformer block's feed forward layer. Each value represents adapter components in corresponding transformer block b.

$$\begin{aligned} LLA_b(h_b, r_b) &= U_b(ReLU(D_b(h_b))) + r_b \\ LAA_b(h_b, r_b) &= U_b(swish(D_b(LLA_b))) + r_b \end{aligned} \tag{3}$$

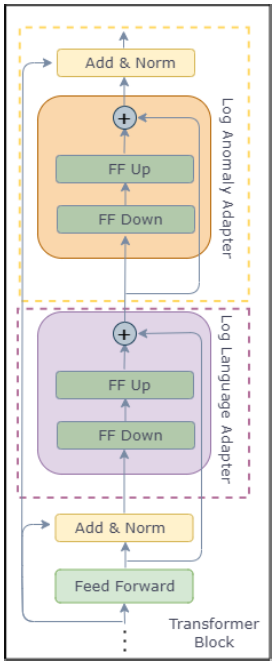


FIGURE 4. Log sequence's anomaly task adapter inside transformer block.

C. MULTI-ANOMALY TASK DETECTION

In real-life log monitoring and analysis tools, log instances are gathered from various machines in a system. To extend the applicability of the approach, we propose multi-anomaly task detection with creating composition of different LLA and LAA stacks. We introduce a new ψ number of parameters to learn how to cooperate stacks together on solving multiple anomalies from different sources. In (4), for combined task t we learn Ψ_t parameters for n different task such that $n \in \{1, 2, \ldots, N\}$.

$$\Psi_t \leftarrow \arg\min_{\Phi} L_t(D_t; \Theta, \phi_1, \ldots, \phi_N, \Phi) \tag{4}$$

In this approach, presented ψ parameters consist of Query(Q_b), Key(K_b) and Value(V_b) that b indicates corresponding transformer block. In each block, output of feed forward layer fed into Q_b and adapter's output use as input for K_b and V_b. In this way, we utilize attention-based learning to decide which stack should be responsible for incoming log sequence.

We calculate output of values from each adapters and transformer block:

$$\begin{aligned} z'_{b,n} &= z^T_{b,n} V_b \\ Z'_b &= [z'_{b,0}, \ldots, z'_{b,N}] \end{aligned} \tag{5}$$

Key and query values are input into a softmax function to learn which log AA is suitable for that log sequence. Then, it is multiplied with AAs values create output.

$$\begin{aligned} s_b &= softmax(h^T_b Q_b \otimes z^T_{b,n} K_b) \\ o_b &= s^T_b Z'_b \end{aligned} \tag{6}$$

In this multi-anomaly task training, we combined Firewall Log AA and HDFS Log AA under the fusion module explained above. Combination of AAs in fusion structure is shown in Figure 5 that represents each transformer block in the base model.

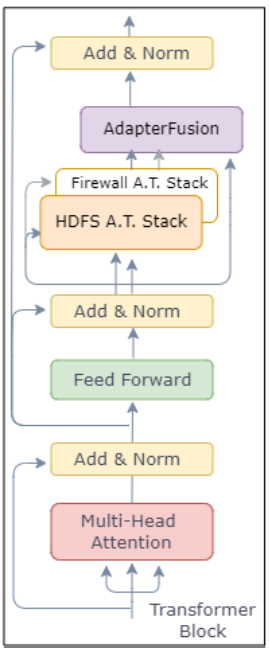


FIGURE 5. Multi-anomaly task detection block in each transformer block.

V. EVALUATION

In the experiments we applied the processing steps required for both Firewall and HDFS datasets in Section III. First we prepared log sources for language adapter training. Then, we selected half of the datasets for language modeling. In this selection we kept distribution of normal and abnormal log events. In firewall dataset, type of events are found via attacks that cause anomalies. In HDFS dataset, it is determined by the distribution of normal and abnormal blockIDs. Stratified sampling was used in the process of the data splitting. In log sequence anomaly adapter training, log events are transformed into normal and anomaly definitions as described in 1. In both processing, normal events structured collectively. Additionally, we have used %80 of data for training and %20 of data for testing in each training phase. For additional training hyperparameters, see Appendix B.

A. EVALUATION METRICS

Anomaly detection is a binary classification problem. False Positive (FP) rates indicates wrongfully detected anomalies and False Negative (FN) shows missed anomaly ratio in detection from existing anomalous log events. To maximize the performance, FP and FN rates should be minimized. For this reason, we utilize Precision, Recall and F1-score measures in evaluation.

$$\begin{aligned} \texttt{Precision} &= \frac{TP}{TP+FP} \texttt{Recall} = \frac{TP}{TP+FN} \\ \texttt{F1-Score} &= \frac{2 \cdot \texttt{Precision} \cdot \texttt{Recall}}{\texttt{Precision} + \texttt{Recall}} \end{aligned} \tag{7}$$

		Precision	Recall	F1-score
Key-based	PCA	0.98	0.67	0.79
	DeepLog	0.9	0.96	0.93
Semantic-based	LogRobust	0.98	0.93	0.95
	LogAnomaly	0.96	0.94	0.95
	HitAnomaly	0.94	0.95	0.95
	Single AAs	0.97	0.94	**0.95**
	Multi AAs	0.96	0.93	0.945

FIGURE 6. Evaluation on HDFS dataset. Evaluation metrics for Single AAs for Firewall datasets are: ***Precision***:0.99, ***Recall***:0.98, ***F1-score***:0.98.

In both training process and dataset, we used pre-trained ROBERTa language model, as a transformer variant, to encode and adapt defined anomaly types through adding a bottleneck element. We have used several baselines to compare log key-based and log semantic-based anomaly detection methods. In log key-based approaches, we compared with two studies, PCA [9] which analyzes log representation as count vectors, DeepLog [12] which uses LSTM model to predict next log key in workflow. In log semantic-based approaches, LogAnomaly [16] creates feature-based learning via dLCE log vector representation in LSTM model. LogRobust [17] is another solution which initiates log representation with shallow deep embeddings and facilitates from Bi-LSTM model in detection. HitAnomaly [15] uses BERT-based log and parameter embeddings with hierarchical transformer architecture. As a counterpart, AnomalyAdapters is a novel way to train on various log sources with an efficiency. And we are able build composable and scalable anomaly detection model. As a result, we have selected HDFS dataset as a

common comparator and utilized firewall logs to establish diversity in sources.

Additionally, we investigated the amount of newly introduced parameters for log language and anomaly adapters. ROBERTa model has 120M parameters which we share among different anomaly tasks and sources. Single AAs solution presents; %1.47 in LLA,%2.66 in LAA of the base model's parameters in Firewall logs, %1.47 in LLA,%3.38 in LAA of the base model's parameters in HDFS logs. Multi AAs fusion solutions presents additional %30 of base model's parameters for detecting anomalies from multiple sources. In comparison to methods used in log semantic-based anomaly detection models, we generated %2-4 base model parameters for the anomaly detection model on a single log source instead of creating %100 or more task specific parameters.

In overall, we achieved on-par results with recent studies with less parameters in Single AAs model for the HDFS dataset. For the Firewall dataset, we achieved acceptably high scores, especially in F1-score (0.98) in Single AAs model. In combination of both log datasets, multi-anomaly task detection model achieves considerably high F1-score (0.945) with highly shared parameters without compromising contextual information. This approach also establishes competitive advantage on building extensible models for anomaly detection in an online setting.

VI. EXPLAINABILITY OF MODEL DECISION AND THREAT DATA

In recent years, understanding deep neural network becomes necessity with acquiring good results. Complex models can create precise decision making on trained tasks, but lack of comprehending how. Yet, not showing importance of model functionality in domain applications impedes further advancements in deep networks [50]. There are many domains that need an explainability of a model decision such as; health, education and security [51]. In cyber security domain, using algorithms to test a model function is beneficial in perspective of CTI life cycle [52]. These algorithms builds comprehensive visuals to unbox decision making by deep networks. Doshi-Velez states that lack of problem formulation creates 'incompleteness' [51]. We believe that rapidly changing technological advancements obstruct adaptability of model function to a problem in cyber domain, in consequence of incompleteness.

Transformer architecture and its applications to different domain problems are considered as complex or black-box model [50]. In cyber security, DNN-based solutions to anomaly or intrusion detection have lack of presenting a way to explain inference results. In general, experiments are based on trusting a model decision via only evaluation metrics. Using attributing techniques can reveal the affect of input features on decision making and more importantly enlightens cyber threat data. By this means, it can be used to improve proposed solutions.

In our experiments we have tested three gradient-based algorithms to explain inference results in our evaluation. Integrated Gradients (IG) [53] method tries to understand inference of a deep network with its input features. Gradients are, simply, the coefficients learned by DNN. It can create cause-effect relationship on the model inference stage. Acquiring IG is to accumulate gradients along with a path considering input x and x'. In Eq. (8), we can see calculation of integrated gradient for i^{th} dimension for x considering F is the model function.

$$\mathrm{IG}_i(x) = (x_i - x_i') \times \int_{a=0}^{1} \frac{\partial F(x' + \alpha \times (x - x')}{\partial x_i} d\alpha \quad (8)$$

Smooth Gradient(SG) method yields gradients and acts on them as *saliency* or *sensitivity maps*. This method brings noise into gradient calculation and can be combine with other gradient map techniques. By label(or class), it is known that sensitivity maps correlates with decision boundaries [54]. Especially, it is working with image classification very well and comprehensible by human perception. Expert knowledge and experience are needed to interpret a specialized domain such as; cyber security domain and anomaly detection task. In (9), SG_c calculates the effect of minimum change on class decision.

$$\hat{\mathrm{SG}}_c(x) = \sum_{1}^{n} \mathrm{SG}_c(x + \eta(0, \sigma^2)) \quad (9)$$

Lastly, Input Reduction (IR) is different way to analyze interpretation. In contrast to saliency interpreters we discussed before, it examines the importance via counterfactual way [55]. Importance is defined by difference in confidence change after altering input values. In (10) shows the calculation of importance on input perturbation. This gradient-based methodology also enlightens the pathological behavior of a model. In the reduction process, we may see one or two tokens to be selected at the end and the method protects the original result. By this way it also reveals adversarial examples for a model.

$$\mathrm{IR}(x_i|x) = f(y|x) - f(y|x_{-i}) \quad (10)$$

In Figure 7 and 8, we have presented an example sequence from HDFS and firewall logs. We chose a log sequence which alarms the detection model as anomaly. For brevity, we omitted part of log sequence, as methods indicate less importance or lower gradient-based value on the model decision. In Figure 7, we are looking at an anomalous behavior of a block in HDFS logs. IG method focuses context between sequences and shows the most impacting phrase as 'not belong' (event action) and its context. SG method slightly differs from others and focuses to create boundary on a starting point of the action such as; 'request received' or 'added invalid-set'. Subwords that are highlighted in grey show omitted inputs without changing model decision. IR method focuses on the same phrase again as in IG to decide anomalous behaviour. The result also depicts adversarial example for the log sequence. In Figure 8, we investigated a

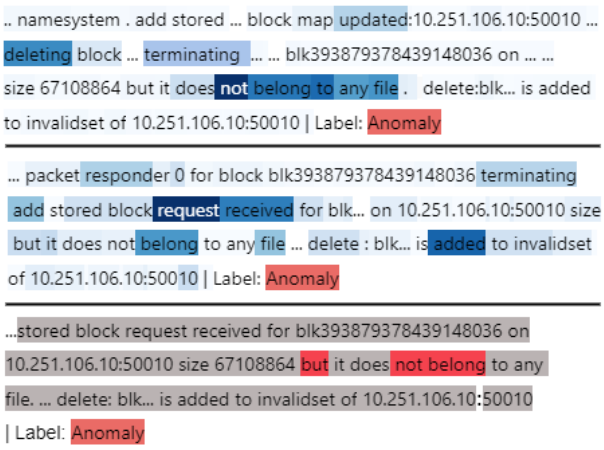


FIGURE 7. Model decision on HDFS logs by integrated/smooth gradients and input reduction methods.

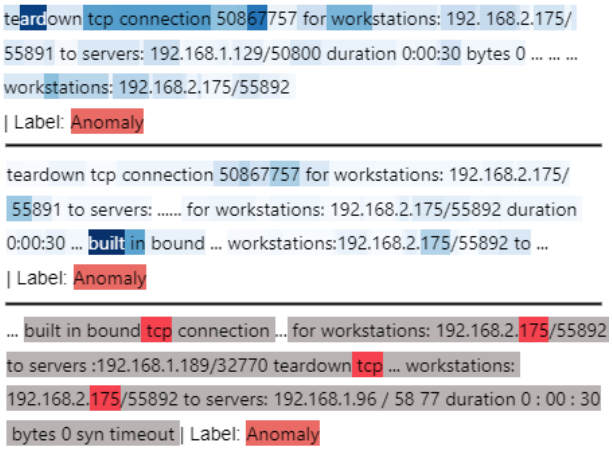


FIGURE 8. Model decision on firewall logs by integrated/smooth gradients and input reduction methods.

port scan activity on workstations. IG method emphasises overall context of a log sequence, but indicates 'tcp connection' for creating an abnormal event on the workstation. SG method again focuses on the action word 'built' of an event boundary, but also points out IP range (.175) defined in the network. IR method singled out 'tcp' and '.175' which is a good example of pathological behaviour of a model, but we can comprehend that connection type and source IP are the indicators of an anomaly. To sum up, overall results are logical, methods focus on workstations which are infected and port scanning other systems in their sub-network. Additionally, *.175* is not in the range of defined IPs in the Use of Policy Rules for the tested network and sequences conditionally point out port scan attack.

Overall in our explanation tests, we used proposed models for Single and Multi AAs (see Appendix C) and examine model decision without providing any context information, policy rules for the network or configuration file of a log type prior to training a model. Comparing facts from HDFS and Firewall dataset, our proposed model understand the reasoning behind an anomaly and can match useful threat data. Also, models exposes their pathological behaviors to us that some tokens in context have high importance in decision making. This also leaves a gap for improving the current stage.

VII. CONCLUSION

Security applications are a necessity for systems in different domains, such as enterprises and critical infrastructures. Anomaly detection is the crucial part of these systems for ensuring security of the continuous activities. Logs are the first source to consult when analyzing events in a system. By this means, system administrators and security professional put log monitoring systems into center of security operations centers. In addition to that, SIEM tools are the preferred implementation space for security enhancements.

Log events are recorded in free form or unstructured text. System developers prefers to build readable log events in exchange to ease manual monitoring [35]. It also opens up a problem when considering the complex nature of systems. Manual labor can not match in existing problem space, hence there are many suggested solutions based on automating log analysis in anomaly detection systems. There are different categorization of presented solutions. If we simplified solution proposal under security domain, we can divide them into two: log key-based and semantic-based anomaly detection methods. Semantic-based methods mainly elaborates contextual knowledge of logs from pretrained deep or shallow networks. These findings also reveal the need of researching learning methods considering applicability to the domain needs.

Under this hood, we proposed AnomalyAdapters, which provides an extensible and modular approach for anomaly detection. It brings a competitive advantage on yielded parameters and simultaneous adaptability to different log sources. Addition to that, adapter's bottleneck architecture improves sharing information without catastrophic forgetting issues. In our experiments, we have compared our work with other recent studies in the field and also tested model decisions to get feedback in a readable form. Explainablity is a known issue for black-box models, thus it also enables threat intelligence actively in the log semantic-based learning which opens a new direction for enhancing solution of anomaly detection problem.

Future directions of this work is to focus on collaborating with algorithms in learning which interprets semantic-based anomaly detection models. By this way, we may create intelligible decisions, which can be acted efficiently and timely. Enhancing quality of the decisions, not numeric evaluations only, consolidates into consistent decision making on identifying anomalous behaviors.

APPENDIX A
UTILIZED ADAPTER ARCHITECTURES

The base adapter structure includes a residual connection, a reduction factor (2,8,16,64) which is the bottleneck that makes able to down and up projections and a non-linearity layer (ReLU, LeakyReLU, Swish) [46], see 9. This form of a base adapter is used in both LLA and LAA setups. Adapter structure variations and possible implementation presented in Pfeiffer's work [46].

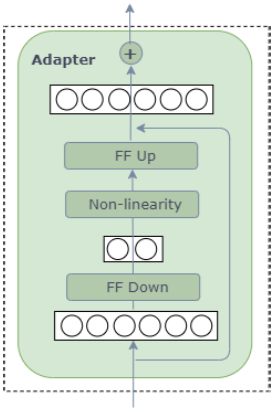


FIGURE 9. Base adapter structure [45].

We presented LLA and LAA stack for complete view of anomaly detection infrastructure inside the transformer block. The type of an adapter structure, implemented for LAA, is shown in Figure 10. In this architecture, the base adapter is added twice for each transformer block of ROBERTa model. One adapter is after multi-head attention and other adapter is added after feed-forward layer [45]. For simplicity, we omitted the lower stack on LAA implementation in Section IV.

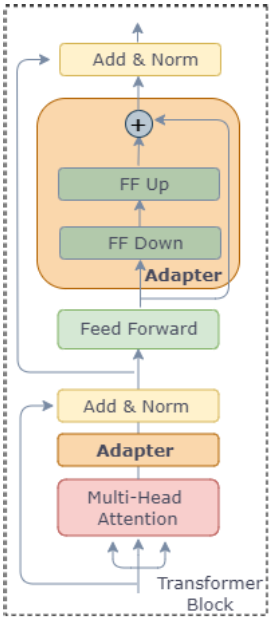


FIGURE 10. Log anomaly adapter detailed implementation inside transformer block [45].

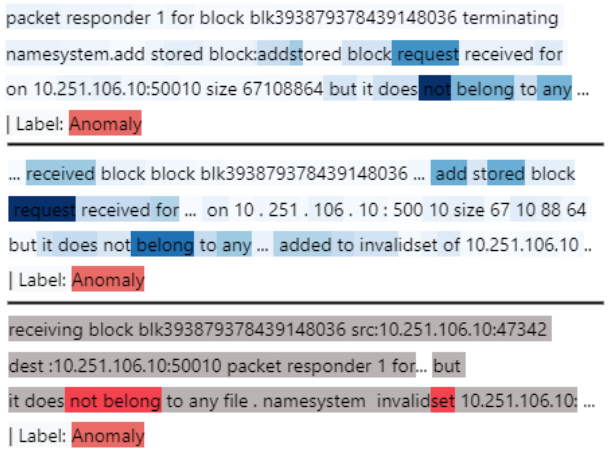


FIGURE 11. Multi AA decision on HDFS logs by integrated/smooth gradients and input reduction methods.

APPENDIX B
TRAINING CONFIGURATIONS

In training, ROBERTa pretrained language model is selected as a base which is transferred during adaptations. The model architecture's configuration is 12 transformer blocks, a hidden size of 768 and a vocabulary size of 50264 subword tokens. It generates approximately 120M parameters at start of the learning process and also, those are shared among adapter-tuning.

For the LLA training, we used the setup in Figure 3 with a reduction factor of 16 and ReLU as a non-linearity function. We have trained 3 epochs in MLM training objective. Same procedure applied for both Firewall and HDFS datasets. For the LAA training, we combined language and anomaly adapters as explained in Section IV. To achieve that, we used the setup in Figure 10 with a reduction factor 16 and a non-linearity using Swish function. Differently, LAA does not have layer norm at the bottom. We have trained 3 epochs in binary classification objective. Same procedure is applied for both Firewall and HDFS datasets. For multi-anomaly task detection's training, we only optimized attention-based adapter selection module for one epoch using combination of Firewall and HDFS dataset.

In all training phases, we implemented an early stopping criteria for controlling degradation in the F1-score and evaluated models in step-wise to prevent overfitting.

APPENDIX C
EXPLAINABILITY: MULTI-ANOMALY TASK DETECTION

Multi-anomaly task detection model fuses various AAs' architectures together. In Figure 11 and 12, we can interpret that different model decision mechanism is protected overall.

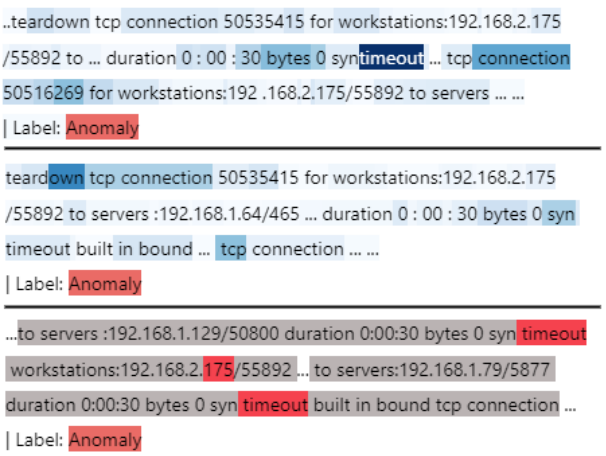


FIGURE 12. Multi AA decision making on firewall logs by integrated/smooth gradients and input reduction methods.

We observe that the base model can be adapted to respond finding anomalies from different sources.

REFERENCES

[1] (2021). *Gartner Top Security and Risk Trends for 2021*. [Online]. Available: https://www.gartner.com/smarterwithgartner/gartner-top-security-and-risk-trends-for-2021

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[3] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019, *arXiv:1901.03407*.

[4] V. Jyothsna, V. V. R. Prasad, and K. M. Prasad, "A review of anomaly based intrusion detection systems," *Int. J. Comput. Appl.*, vol. 28, no. 7, pp. 26–35, Aug. 2011.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[6] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *Proc. 38th Int. Conf. Softw. Eng. Companion*, May 2016, pp. 102–111.

[7] D. Yuan, H. Mai, W. Xiong, L. Tan, Y. Zhou, and S. Pasupathy, "SherLog: Error diagnosis by connecting clues from run-time logs," in *Proc. 15th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2010, pp. 143–154.

[8] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "An evaluation study on log parsing and its use in log mining," in *Proc. 46th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2016, pp. 654–661.

[9] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proc. ACM SIGOPS 22nd Symp. Operating Syst. Princ. (SOSP)*, Oct. 2009, pp. 117–132.

[10] J.-G. Lou, Q. Fu, S. Yang, J. Li, and B. Wu, "Mining program workflow from interleaved traces," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Jul. 2010, pp. 613–622.

[11] X. Yu, P. Joshi, J. Xu, G. Jin, H. Zhang, and G. Jiang, "CloudSeer: Workflow monitoring of cloud infrastructures via interleaved logs," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 2, pp. 489–502, May 2016.

[12] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1285–1298.

[13] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," 2015, *arXiv:1508.07909*.

[14] M. Schuster and K. Nakajima, "Japanese and Korean voice search," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 5149–5152.

[15] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, and Z. Luan, "HitAnomaly: Hierarchical transformers for anomaly detection in system log," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 2064–2076, Dec. 2020.

[16] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, and R. Zhou, "LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, vol. 19, no. 7, pp. 4739–4745.

[17] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, J. Chen, X. He, R. Yao, J.-G. Lou, M. Chintalapati, F. Shen, and D. Zhang, "Robust log-based anomaly detection on unstable log data," in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Aug. 2019, pp. 807–817.

[18] S. Nedelkoski, J. Bogatinovski, A. Acker, J. Cardoso, and O. Kao, "Self-attentive classification-based anomaly detection in unstructured logs," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2020, pp. 1196–1201.

[19] A. W. Colombo, S. Karnouskos, O. Kaynak, Y. Shi, and S. Yin, "Industrial cyberphysical systems: A backbone of the fourth industrial revolution," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 6–16, Mar. 2017.

[20] R. PV and R. Sandhu, "POSTER: Security enhanced administrative role based access control models," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1802–1804.

[21] Q. Fu, J.-G. Lou, Q. Lin, R. Ding, D. Zhang, and T. Xie, "Contextual analysis of program logs for understanding system behaviors," in *Proc. 10th Work. Conf. Mining Softw. Repositories (MSR)*, May 2013, pp. 397–400.

[22] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "Towards automated log parsing for large-scale log data analysis," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 931–944, Nov. 2018.

[23] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *Proc. IEEE 27th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2016, pp. 207–218.

[24] M. Du and F. Li, "Spell: Streaming parsing of system event logs," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 859–864.

[25] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2017, pp. 33–40.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.

[27] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, Dec. 2017.

[28] K. A. Nguyen, S. S. I. Walde, and N. T. Vu, "Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction," 2016, *arXiv:1605.07766*.

[29] A. Pande and V. Ahuja, "WEAC: Word embeddings for anomaly classification from event logs," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 1095–1100.

[30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[31] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.

[32] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "BioBERT: A pre-trained biomedical language representation model for biomedical text mining," 2019, *arXiv:1901.08746*.

[33] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A pretrained language model for scientific text," 2019, *arXiv:1903.10676*.

[34] M. Toneva and L. Wehbe, "Interpreting and improving natural-language processing (in machines) with natural language-processing (in the brain)," 2019, *arXiv:1905.11833*.

[35] C. Bertero, M. Roy, C. Sauvanaud, and G. Tredan, "Experience report: Log mining using natural language processing and application to anomaly detection," in *Proc. IEEE 28th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2017, pp. 351–360.

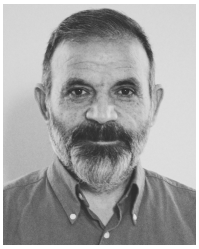
[36] I. Steinwart, D. Hush, and C. Scovel, "A classification framework for anomaly detection," *J. Mach. Learn. Res.*, vol. 6, no. 2, pp. 211–232, Feb. 2005.

[37] R. K. Malaiya, D. Kwon, J. Kim, S. C. Suh, H. Kim, and I. Kim, "An empirical evaluation of deep learning for network anomaly detection," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Mar. 2018, pp. 893–898.

[38] N. Saunshi, S. Malladi, and S. Arora, "A mathematical exploration of why language models help solve downstream tasks," 2020, *arXiv:2010.03648*.

[39] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2017, pp. 3987–3995.
[40] S. He, J. Zhu, P. He, and M. R. Lyu, "Loghub: A large collection of system log datasets towards automated log analytics," 2020, *arXiv:2008.06448*.
[41] K. Sinha, R. Jia, D. Hupkes, J. Pineau, A. Williams, and D. Kiela, "Masked language modeling and the distributional hypothesis: Order word matters pre-training for little," 2021, *arXiv:2104.06644*.
[42] *Training With Mixed Precision: NVIDIA Deep Learning Performance Documentation*. Accessed: Nov. 27, 2021. [Online]. Available: https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html
[43] U. Unal, C. N. Kahya, Y. Kurtlutepe, and H. Dag, "Investigation of cyber situation awareness via SIEM tools: A constructive review," in *Proc. 6th Int. Conf. Comput. Sci. Eng. (UBMK)*, Sep. 2021, pp. 676–681.
[44] T. Wolf, J. Chaumond, L. Debut, V. Sanh, C. Delangue, A. Moi, P. Cistac, M. Funtowicz, J. Davison, S. Shleifer, and R. Louf, "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empirical Methods Natural Lang. Process., Syst. Demonstrations*, Oct. 2020, pp. 38–45.
[45] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, L. Q. De, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in *Proc. Int. Conf. Mach. Learn.*, May 2019, pp. 2790–2799.
[46] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych, "AdapterFusion: Non-destructive task composition for transfer learning," 2020, *arXiv:2005.00247*.
[47] J. Pfeiffer, A. Rücklé, C. Poth, A. Kamath, I. Vulić, S. Ruder, K. Cho, and I. Gurevych, "AdapterHub: A framework for adapting transformers," 2020, *arXiv:2007.07779*.
[48] S. A. Rebuffi, H. Bilen, and A. Vedaldi, "Learning multiple visual domains with residual adapters," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Dec. 2017, pp. 506–516.
[49] J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder, "MAD-X: An adapter-based framework for multi-task cross-lingual transfer," 2020, *arXiv:2005.00052*.
[50] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu, "Explainable AI: A brief survey on history, research areas, approaches and challenges," in *Proc. CCF Int. Conf. Natural Lang. Process. Chin. Comput.* Cham, Switzerland: Springer, Oct. 2019, pp. 563–574.
[51] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," 2017, *arXiv:1702.08608*.
[52] S. Samtani, M. Abate, V. Benjamin, and W. Li, "Cybersecurity as an industry: A cyber threat intelligence perspective," in *The Palgrave Handbook of International Cybercrime and Cyberdeviance*, T. Holt and A. Bossler, Eds. Cham, Switzerland: Palgrave Macmillan, 2019, doi: 10.1007/978-3-319-90307-1_8-1.
[53] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2017, pp. 3319–3328.
[54] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "SmoothGrad: Removing noise by adding noise," 2017, *arXiv:1706.03825*.
[55] S. Feng, E. Wallace, A. Grissom II, M. Iyyer, P. Rodriguez, and J. Boyd-Graber, "Pathologies of neural models make interpretations difficult," 2018, *arXiv:1804.07781*.

UĞUR ÜNAL received the B.Sc. degree in computer engineering from Koç University, İstanbul, Turkey, in 2012, and the M.Sc. degree in advanced computing (business systems) from Brunel University, London, U.K., in 2014. He is currently pursuing the Ph.D. degree in management information systems with Kadir Has University, İstanbul. His current research interests include anomaly detection, and natural language processing and its applications.



HASAN DAĞ (Member, IEEE) received the B.Sc. degree in electrical engineering from Istanbul Technical University, İstanbul, Turkey, in 1987, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Wisconsin–Madison, USA, in 1990 and 1995, respectively. His research interests include power systems, smart grid, cybersecurity, data science and its applications along with high-performance computing.