



DEEP ANOMALY DETECTION SYSTEM USING SYSTEM LOGS

MR. PURIMPAT CHEANSUNAN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF ENGINEERING
(COMPUTER ENGINEERING)
FACULTY OF ENGINEERING
KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI


2019


Deep Anomaly Detection System Using System Logs


Mr. Purimpat Cheansunan B. Eng. (Computer Engineering)


A Thesis Submitted in Partial Fulfillment
of the Requirements for
the Degree of Master of Engineering (Computer Engineering)
Faculty of Engineering
King Mongkut's University of Technology Thonburi
2019


Thesis Committee


.....Chairman of Thesis Committee
(Assoc. Prof. Anan Banharnsakun, Ph.D.)


.....Member and Thesis Advisor
(Asst. Prof. Phond Phunchongharn, Ph.D.)


.....Member
(Asst. Prof. Khajonpong Akkarajitsakul, Ph.D.)


.....Member
(Asst. Prof. Santitham Prom-on, Ph.D.)


.....Member
(Lect. Warasinee Chaisangmongkon, Ph.D.)

Thesis Title	Deep Anomaly Detection System Using System Logs
Thesis Credits	12
Candidate	Mr. Purimpat Cheansunan
Thesis Advisor	Asst. Prof. Phond Phunchongharn, Ph.D.
Program	Master of Engineering
Field of Study	Computer Engineering
Department	Computer Engineering
Faculty	Engineering
Academic Year	2019

Abstract

System logs contain the complete information of service operations which is usually written by developers. In recent years, anomaly detection on log messages is popular. Several researchers applied the machine learning techniques to the system logs to detect the anomalous behaviors of the system. At the same time, CERN organization planned to establish a new data center in 2020 to improve the performance of the system. According to the new datacenter, ALICE intends to build a monitoring system for the new system. However, ALICE has been deploying the conventional monitoring system but requires the intervention of administrators. In this work, a novel anomaly detection framework is proposed by using Convolutional Neural Network (CNN) along with Transformer which is popular in machine translation tasks. The proposed framework automatically monitors the service operation logs and detects the anomalous events based on the log messages. The dataset used in the work is HDFS operation logs. To evaluate the model, the measurement metrics include Precision, Recall, F-measure, Miss detection rate, and False alarm rate. Finally, we design the experiments to compare the performance against the existing works which use LSTM (Long-Short Term Memory)-based and CNN-based approaches.

Keywords: ALICE/ Anomaly detection/ CERN/ Convolutional Neural Network/ Self-Attention/ Transformer

หัวข้อวิทยานิพนธ์	ระบบตรวจจับความผิดปกติด้วยการเรียนรู้ของเครื่องเชิงลึกบนล็อกของระบบ
หน่วยกิต	12
ผู้เขียน	นายปรีมพัฒน์ เจียรสุนันท์
อาจารย์ที่ปรึกษา	ผศ.ดร.พร พันธุ์งาญ
หลักสูตร	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
คณะ	วิศวกรรมศาสตร์
ปีการศึกษา	2562

บทคัดย่อ

ล็อกของระบบเป็นแหล่งรวมข้อมูลการทำงานของระบบทั้งหมด ซึ่งตามปกติแล้วจะเป็นสิ่งที่ผู้พัฒนาระบบเขียนเอาไว้ ในช่วงหลายปีมานี้ ระบบตรวจจับสิ่งผิดปกติบนข้อความล็อกกำลังได้รับความนิยม มีนักวิจัยหลายคนนำเทคนิคการเรียนรู้ของเครื่องมาใช้กับล็อกของระบบ เพื่อตรวจจับความผิดปกติที่เกิดขึ้นภายในระบบ ในขณะเดียวกันองค์กร CERN ได้วางแผนที่จะก่อตั้งศูนย์ข้อมูลแห่งใหม่ เพื่อพัฒนาปรับปรุงประสิทธิภาพของระบบ เนื่องจากการก่อตั้งศูนย์ข้อมูลแห่งใหม่ หน่วย ALICE จึงตั้งใจว่าจะพัฒนาระบบเฝ้าระวังขึ้นมาใหม่สำหรับศูนย์ข้อมูลใหม่นี้ อย่างไรก็ตาม ALICE มีการติดตั้งระบบเฝ้าระวังแบบเดิมอยู่แล้ว แต่เป็นระบบที่จำเป็นจะต้องมีผู้ดูแลระบบเข้าช่วยเหลือในการตั้งค่าและเฝ้าระวังต่าง ๆ ในงานนี้เราจึงนำเสนอระบบตรวจจับความผิดปกติในรูปแบบใหม่ที่สามารถทำงานได้โดยอัตโนมัติ ซึ่งใช้เทคนิคของการคอนโวลูชัน (CNN) และทรานสฟอร์มเมอร์ ที่ได้รับความนิยมในงานแปลด้วยเครื่อง นอกจากนี้ ข้อมูลที่เราใช้ในงานนี้ คือ ข้อมูล HDFS ซึ่งเราได้ประเมินประสิทธิภาพของแบบจำลอง ด้วยการใช้ตัวชี้วัด ได้แก่ ความแม่นยำ ความอ่อนไหว การวัดค่า F ความผิดพลาดในการตรวจพบ และ อัตราการแจ้งเตือนผิดพลาด และสุดท้ายนี้ เราออกแบบการทดลองเพื่อเปรียบเทียบประสิทธิภาพของแบบจำลองเรากับงานก่อนหน้านี้ซึ่งใช้ LSTM (Long-Short Term Memory) และ CNN

คำสำคัญ: การตรวจจับความผิดปกติ/ การสนใจตนเอง/ โครงข่ายประสาทแบบคอนโวลูชัน/
ทรานสฟอร์มเมอร์/ ALICE/ CERN

ACKNOWLEDGMENTS

I am taking this opportunity to express my appreciation to everyone who always helps me during the period of my study and thesis. First of all, I would like to greatly acknowledge Asst. Prof. Phond Phunchongharn, a lecturer and my advisor at the Computer Engineering Department, King Mongkut's University of Technology Thonburi, who always gives me marvelous advice, contributions, and other help at all time of research and this thesis.

Besides my thesis advisor, I especially thank the members of my thesis committee: Assoc. Prof. Anan Banharnsakun, Asst. Prof. Khajonpong Akkarajitsakul, Asst. Prof. Santitham Prom-on, and Dr. Warasinee Chaisangmongkon for their excellent suggestions and contributions to criticism.

Moreover, I would like to express my sincere appreciation to all the people in the CERN ALICE O2 team, in which I had been an intern for a summer internship period. Especially, I would like to greatly thank Vasco Chibante Barroso and Adam Wegrzynek for giving me inspirations and suggestions.

Last but not least, I am grateful to lecturers who have taught me the knowledge and all officers who have always been willing to offer help to me. Finally, a special thank goes to my family and my friends who always give me encouragement and support. All of these things have motivated me to complete this thesis successfully.

CONTENTS

	PAGE
ENGLISH ABSTRACT	ii
THAI ABSTRACT	iii
ACKNOWLEDGMENTS	iv
CONTENTS	v
LIST OF TABLES	vii
LIST OF FIGURES	viii
 CHAPTER	
1. INTRODUCTION	1
1.1 Statement of Problem	2
1.2 Objectives	2
1.3 Scopes	2
1.4 Expected Benefit	2
 2. LITERATURE REVIEW AND THEORY	3
2.1 What is Anomaly	3
2.2 Anomaly Detection Approaches	4
2.3 Other Approaches	14
2.4 Literature Review Conclusion and Discussion	15
 3. METHODOLOGY	18
3.1 Dataset	18
3.2 Methodology	19
3.3 Experimental Design	25
 4. EXPERIMENTAL RESULTS	30
4.1 Experiments for Hyperparameter Adjustment	30
4.2 Experiments for Anomaly Detection Performance Comparison	35
4.3 Methodology Limitation Discussion	40

CONTENTS (Cont'd)

	PAGE
5. CONCLUSION	41
REFERENCES	42
CURRICULUM VITAE	47

LIST OF TABLES

TABLE	PAGE
2.1 Literature survey summary	16
3.1 Number of records in each dataset	18
3.2 Number of log key occurrences in dataset	19
3.3 Confusion matrix	27
3.4 Hardware Specification	29
4.1 Experiments on hyperparameter adjustment of CNN-based model	31
4.2 Effect of window size in anomaly detection	32
4.3 Experiments on hyperparameter adjustment of Attention-based model	34
4.4 Detection efficiency comparison	35
4.5 Analysis of variance of recall over 5 models	38

LIST OF FIGURES

FIGURE	PAGE
2.1 Example of point anomaly and collective anomaly in credit card fraud.....	3
2.2 Example of service logs event which contextual anomaly can occur.....	4
2.3 General framework architecture of anomaly detection on logs event.....	4
2.4 Example of log keys and parameters	5
2.5 The simplest version of recurrent neural network.....	8
2.6 Inside the Long Short-Term Memory Block.....	8
2.7 LeNet-5 Network Architecture	9
2.8 Attaching attention (α) into Neural Machine Translation model.....	9
2.9 Architecture of dot product attention	10
2.10 Scaled Dot-Product Attention Mechanism.....	11
2.11 Architecture of stacked LSTM network.....	12
2.12 Normal LSTM (Black bordered nodes) and Bidirectional LSTM (all nodes).....	13
2.13 Normal LSTM attached with attention layer	13
2.14 Overview construction of anomaly detection framework using CNN.....	14
2.15 The architecture of Transformer	15
3.1 Density plot of session length	20
3.2 Example of sequence preparation	20
3.3 Number of pattern occurrences	21
3.4 Overview architecture of our methodology	22
3.5 Convolutional-based network architecture	23
3.6 Attention-based network architecture	24
4.1 Hyperparameter tuning of convolutional-based model architecture.....	32
4.2 Hyperparameter tuning of attention-based model architecture.....	34
4.3 Heatmap of cosine similarity between log keys.....	36
4.4 Example plot of attention weights	37
4.5 Dunnett post hoc test for determining pair with different means.....	38
4.6 Training time comparison	39
4.7 Detection productivity comparison.....	40

CHAPTER 1 INTRODUCTION

CERN (The European Organization for Nuclear Research) is the biggest organization that performs the experiments to research in the field of particle physics. There are 4 particle detectors on the LHC (Large Hadron Collider) which is the particle collider that accelerates the proton and other ions to close to the speed of light. The accelerated ions will collide with the other one at the particle detectors that perform the large amount of collision data which might contain the facts that can discover new matters. Thus, the detector of the LHC is important for acquiring, processing, and storing the data that occur in the particle collision experiment.

ALICE (A Large Ion Collider Experiment) is one of the LHC's particle detectors in which the collisions happen. For a large amount of collision data, the ALICE designs the system to gather all experiment data which are called ALICE O2 system (Online-Offline) as stated in Wegrzynek, et al. (2018). After the success in Run 1 (2010-2013) and Run 2 (2015-2018) of the ALICE experiments, in Run 3 (2020), the new O2 system will consist of thousands of computing machines with the higher capability. However, the system still requires high availability and low probability of failure. Therefore, in the new O2 system, the monitoring system is crucial for preventing the failure of the system and increasing the reliability of the system.

Although the ALICE O2 IT team have developed the monitoring system based on the performance metrics, they still require the monitoring system using the service logs since it is another way to acquire the information from the system while runtime. Unfortunately, the log messages are unstructured and written by many developers without the writing standard. Consequently, a system administrator must have the domain knowledge of that system or applications to interpret and understand what the logs mean. In the real world, it is impractical to always watch the monitoring screen to determine the abnormal behaviors that might cause the failure of the system. Thus, autonomous anomaly detection is the solution to the problem.

Anomaly detection is a method in the field of predictive maintenance which detects abnormal patterns and automatically alarms the system administrators to resolve problems before the failure occurs. There are challenges in our research. Firstly, the system log data are unstructured data. Secondly, anomaly detection is a time-series problem that relates to the order of log generation. Both challenges allow us to propose a deep learning model with a log parsing tool to build an anomaly detection system. The learned model will detect abnormal patterns of logs and automatically notify the system administrators to solve the problems before the system fails.

The remainders of this thesis are organized as follows: Chapter 2 presents the literature review, related researches and previous works of the anomaly detection approach. Chapter 3 presents our proposed work and describes the design and overview of the framework. The performance of the proposed system is evaluated in Chapter 4. Finally, Chapter 5 concludes this thesis.

1.1 Statement of Problem

In this research, there are 2 main problem statements related to the anomaly detection on system logs:

1. How to design an autonomous anomaly detection framework using operation log messages which are unstructured data to provide an alarm when anomalous behaviors are detected.
2. How to implement the anomaly detection based on the sequence of log keys and variables which are parsed from the system log using deep learning techniques.

1.2 Objectives

According to the problem statements, this thesis aims:

1. To propose and develop an autonomous log-based anomaly detection framework by using deep learning techniques.
2. To evaluate the performance of our anomaly detection model with the existing work.

This thesis is a part of the European Organization for Nuclear Research, CERN.

1.3 Scopes

1. The log parser in our framework is adapted from the opensource python algorithm proposed by He, et al. (2018).
2. The dataset used in this work is collected from the Hadoop Distributed File System (HDFS).
3. The model is developed with Python programming.

1.4 Expected Benefit

1. Our anomaly detection model can detect the majority of the abnormal behaviors of the HDFS logs.
2. Our proposed method can outperform the existing anomaly detection models.

CHAPTER 2 LITERATURE REVIEW AND THEORY

In this chapter, the theoretical concepts and approaches of anomaly detection are presented. Firstly, the overview of the anomaly detection is introduced, and then the content will drill down to the context of the log-based anomaly detection. There are three main kinds of the approaches based on the methods that they used. Finally, the conclusion of literature review is presented at the end of this chapter.

2.1 What is Anomaly

In the real world, most of the problems usually occur without any prevention which will impact the system performance. Normally, before the appearance of the problem, there are some behaviors that are rarely present and dissimilar from the normal behavior of the system, which are called anomalies. According to the survey of Chalapathy, et al. (2019), the types of anomalies are separated into 3 types including point anomaly, contextual anomaly, and collective anomaly.

2.1.1 Point Anomaly

Firstly, anomalous points or behaviors in the data randomly happen in the real world. Unfortunately, the appearance of a point anomaly is hard to guess when it will happen. The credit card fraud problem is the simplest example of the point anomaly. According to the top of Figure 2.1, the suspicious transaction of the credit card is the point anomaly.

May-22	1:14 pm	FOOD	Monaco Café	\$ 1,127.80	Point Anomaly
May-22	2:14 pm	WINE	Wine Bistro	\$ 28.00	
...					
Jun-14	2:14 pm	MISC	Mobil Mart	\$ 75.00	Collective Anomaly
Jun-14	2:05 pm	MISC	Mobil Mart	\$ 75.00	
Jun-15	2:06 pm	MISC	Mobil Mart	\$ 75.00	
Jun-15	11:49 pm	MISC	Mobil Mart	\$ 75.00	
May-28	6:14 pm	WINE	Acton shop	\$ 31.00	
May-29	8:39 pm	FOOD	Crossroads	\$ 128.00	
Jun-16	11:14 am	MISC	Mobil Mart	\$ 75.00	Collective Anomaly
Jun-16	11:49 am	MISC	Mobil Mart	\$ 75.00	

Figure 2.1 Example of point anomaly and collective anomaly in credit card fraud
(Source: Chalapathy, et al., 2019)

2.1.2 Collective Anomaly

A collective anomaly is a contrast of the point anomaly, which represents in the collection of small points of values. The collective anomaly can be detected only when focusing on the group of transactions or points. For example, Figure 2.1 shows the collective anomaly in the green lines. As the transaction is shown, if a researcher considers individual transactions, these anomaly transactions will be classified as normal. On the other hand, considering the group of the transactions will help us to accurately classify them.

2.1.3 Contextual Anomaly

The last type of anomaly is a contextual anomaly related to behaviors and time. For example, the generation of system log which is the scope of this thesis is the time-series data. Normally, the generation of system log is based on the template that a developer has written as shown in Figure 2.2. Thus, our objective is to detect the contextual anomaly in

log events by observing a pattern of log appearance, if it is mismatched from the normal behavior in the template, then flagging it as an anomaly.

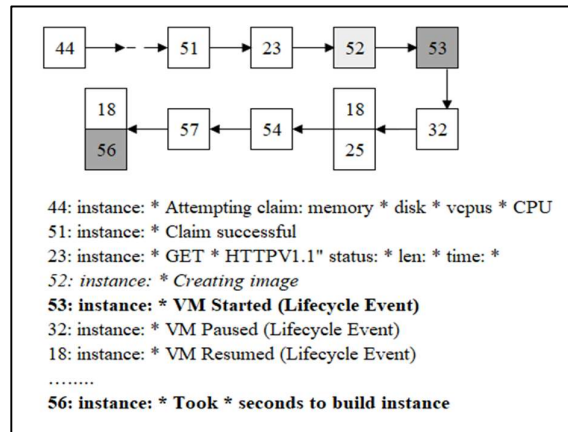


Figure 2.2 Example of service logs event which contextual anomaly can occur
 (Source: Du, et al., 2017)

2.2 Anomaly Detection Approaches

The rise of the anomaly detection approach can help to improve system performance and reduce the number of system failure occurrences. Generally, the term of anomaly detection, sometimes called outlier detection, is often used to call the methodology that attempts to find the anomalous patterns which deviate from the normal patterns. More specifically, the anomaly detection is widely used in many fields such as network intrusion, suspicious failure of the machine, and unexpected execution event of the system. The beginning of the anomaly detection research is the statistical method presented by Edgeworth (1887) to detect the outlier since the early of the nineteenth century. After that, there are a lot of researches that attempt to build an anomaly detection system by using various techniques such as machine learning, deep learning.

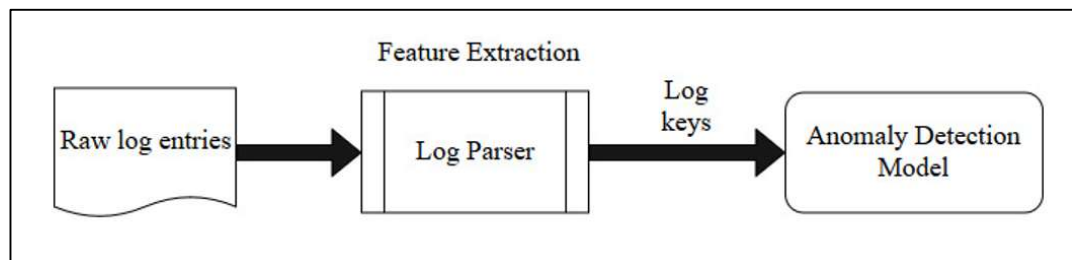


Figure 2.3 General framework architecture of anomaly detection on logs event

In the context of log events, researchers usually focus to detect anomalous log generating patterns which could be the cause of system failure. Several researchers attempt to invent an automated anomaly detection system. In the following sections, literature surveys in the anomaly detection area are presented. As shown in Figure 2.3, the general architecture of the existing frameworks consists of two primary components including Log Parser, and Anomaly Detection Model. First, the authors begin with log parser. Further, the statistical approach which is the beginning era of the autonomous anomaly detection system will be introduced, then forwarding to the machine learning era, and then moving

deeper to the modern era which the deep learning is applied to the anomaly detection system. The following approaches are the anomaly detection based on the service logs event which is the objective of this thesis. Finally, the authors conclude the literature survey at the end of this chapter.

2.2.1 Log Parser

In machine learning research, there is the crucial stage that is the preprocessing stage. Especially in logs dataset, raw logs are in an unstructured form which is not compatible with some machine learning techniques. Therefore, log parsing is an interesting topic for the log analysis field. Surprisingly, during the survey on this topic, a lot of research papers propose the log parsing techniques which have the similar objective that is to convert the log messages which are in an unstructured form into the structured form called Log key or Message type. For example, Nagappan and Vouk (2010) presented the clustering method which could group similar log messages into the same group. Moreover, the clustering method was applied to the log parsing by Hamooni, et al. (2016), which was called LogMine. The objective of LogMine was to create the templates of the logs as the log keys. Recently, the online log parsing was proposed by He, et al. (2018) in order to respond to the demand for online monitoring. They applied the directed acyclic graph technique to parse the log events.

log message (log key underlined)	log key	parameter value vector
$t(1)$ <u>Deletion of file1 complete</u>	$k1$	$[t(1) - t(0), \text{file1Id}]$
$t(2)$ <u>Took 0.61 seconds to deallocate network ...</u>	$k2$	$[t(2) - t(1), 0.61]$
$t(3)$ <u>VM Stopped (Lifecycle Event)</u>	$k3$	$[t(3) - t(2)]$
...

Figure 2.4 Example of log keys and parameters
(Source: Du, et al., 2017)

2.2.2 Statistical Approaches

In the early days of anomaly detection, the traditional approaches such as rule-based method have been used to capture the deviating log patterns. Unfortunately, the rapid increase of generated logs makes the traditional way impractical because there are a lot of patterns to be detected. Hence, the statistical approach has been introduced to perform the autonomous anomaly detection system.

Before the machine learning becomes popular, the statistical-based method is widely applied to solve the various tasks as well as anomaly detection. The interesting research papers which were the fundamentals of logs-based anomaly detection were proposed by Xu, et al. (2009a, 2009b). During that time, this research presented the novel anomaly detection technique using Principal Component Analysis (PCA) which is the statistical method. They implemented the PCA model on the count of message vectors of HDFS logs. Moreover, they visualized the detection results in the form of a decision tree which is more interpretable for a human operator. Interestingly, the detection model performed great efficiency on the features extracted from the logs. However, the statistical model is usually an offline process that must aggregate the log entries into the session group, and then the statistical value has been computed. Therefore, the statistical approach is not suitable for the online setting that is in the scope of our thesis.

2.2.3 Machine Learning Approaches

During the era of machine learning, a lot of researchers applied the machine learning techniques in their works as well as in the anomaly detection field. The types of problems in the machine learning could be categorized into supervised and unsupervised learning based on the datasets that the researchers used. In the context of anomaly detection on log events, there are both supervised and unsupervised learning researches.

Firstly, the authors survey the supervised learning method which is applied to the classification problem. For the supervised learning method, the labeled dataset is required to be used as an input of the machine learning model. For example, Support Vector Machine (SVM) has been used to perform the failure prediction presented in Fulp, et al. (2008) as well as Featherstun and Fulp (2010). Both research papers applied SVM with the spectral kernel on the Linux system logs to forecast the failure of the hard disk in the future. According to these researches, considering the sequence of logs, their methods performed better prediction results. As described above, the limitation of the supervised learning method is the requirement of the labeled dataset which sometimes is not enough for building the model. Thus, the authors continue to survey the unsupervised learning method.

The unsupervised learning method is the method that forms a model without the requirement of the labeled dataset. This is the main benefit of the unsupervised learning method for a researcher who does not have the labeled dataset to make a prediction. This method is so popular in the context of anomaly detection on log events. Du and Cao (2015) proposed the hierarchical clustering operation to preprocess log messages and then assign an anomaly score to each log. Furthermore, Vaarandi, et al. (2018) introduced the data mining-based approach to detect the anomalous messages in the system logs. They attempted to create a model that automatically created a filtering rule for the anomaly message with Simple Event Correlator (SEC) to filter the matched log messages.

Besides the supervised and unsupervised learning, there is a state machine method which builds the state model to detect the anomaly in the system. For example, Finite State Automaton (FSA) was selected to build a workflow model of the normal execution system in Fu, et al. (2009) to detect the anomalous system performance. They built the workflow model and applied it to detect the new incoming log sequence. Moreover, the FSA was also used in Yu, et al. (2016) to create the workflow monitoring system which could detect the execution problems. Recently, Debnath, et al. (2018) proposed the real-time log analysis system which was called LogLens, using the state machine method. They introduced the two-state models including stateless model which performed log parsing and stateful model which was the FSA that created a state sequence from the output of the stateless model.

2.2.4 Deep Learning Approaches

In the present, the technology comes so far and so fast, deep learning becomes popular in this era with its amazing performance on various tasks such as computer vision, speech recognition, and anomaly detection. Before going further to the detail of the existing works of anomaly detection tasks, the authors would like to briefly explain the technical concept of deep learning and the relevant techniques of the anomaly detection model.

2.2.4.1 What is Deep Learning

Deep learning is a subset of the machine learning universe that is much deeper and more powerful to solve more complicated tasks. With the higher capability of the model, Krizhevsky, et al. (2012) performed the amazing classification performance with dramatical error reduction and won the competition which established the era of deep learning and leveraged the interest of researchers in the world. After that, there are a lot of researches applying deep learning to their task, which perform incredible improvement on the results as well as the anomaly detection field.

2.2.4.2 Recurrent Neural Network

An anomaly detection task is usually judged as a sequential problem because the anomalous event occurs after some abnormal patterns are shown in the system. Thus, Recurrent Neural Network (RNN) is always the first network architecture choice for handling sequential tasks. According to Figure 2.5, the simplest version of the recurrent neural network called Elman network has only 3 layers including input layer, hidden layer (context), and output layer. During the training stage, inputs are separately computed by timestep t which is usually represented as $x(t)$, and then fed into the hidden layer. The hidden layer obtains 2 ways of inputs including $x(t)$ and the context from the previous timestep $s(t-1)$. Because of the repeated process in the hidden layer, this neural network is called recurrent and can remember and consider the surrounding context of the input which is the interesting concept to handle sequential tasks. However, the RNN can consider only 5 - 10 contexts of the previous timestep because of the vanishing gradient problem as stated by Staudemeyer RC and Morris ER (2019). The problem occurs during the back-propagation process, when the sequence is longer, the error signal can grow or shrink in every timestep. Therefore, the solution is Long Short-Term Memory neural network (LSTM).

Long Short-Term Memory neural network is one kind of the RNN that considers the context of the surrounding timestep. Firstly, the LSTM was proposed by Hochreiter S. and Schmidhuber J. (1997) to resolve the vanishing gradient problem that usually occurred in very long sequence data as stated in the paper. To resolve the problem, the LSTM came with memory cells and many gates to handle the vanishing problem. Additionally, the core feature of the LSTM that provided the capability to store short-term memory for longer period was Constant Error Carousel (CEC) as explained in Staudemeyer RC and Morris ER (2019). Specifically, the CEC is the error signal like a feedback from the hidden layer to both self-layer and other layers. Therefore, gate units in the LSTM became a handler to manage the CEC signal that could be conflicted during the model training. According to Figure 2.6, there are 3 important gates including input gate, output gate, and forget gate. Firstly, the input gate responds to the signals to prevent irrelevant signals that attempt to disturb the content. Next, the output gate controls the permission to access the memory cell and protect other memory cells from the irrelevant signals which originate from the network unit. During the training in process, the memory cell continuously preserves every cell state over time until the memory cell is full, every cell state will be gone, and the LSTM will become the simple RNN immediately. Therefore, the forget gate becomes important to resolve the problem by learning to reset the internal state of the memory cell which is no longer needed.

Moreover, there was the advanced architecture of the LSTM which considered two directions of inputs including forward (left-to-right), and backward (right-to-left), called Bi-directional LSTM as known as BLSTM. The model with bidirectional computing was

applied to classify phonemes in Graves A and Schmidhuber J. (2005). This advanced architecture provided the power to retrieve more information and more context features from the input sequence surrounding the target which made the BLSTM an outstanding performance in some sequential tasks, i.e., Named Entity Recognition (NER) in Shao, et al. (2016).

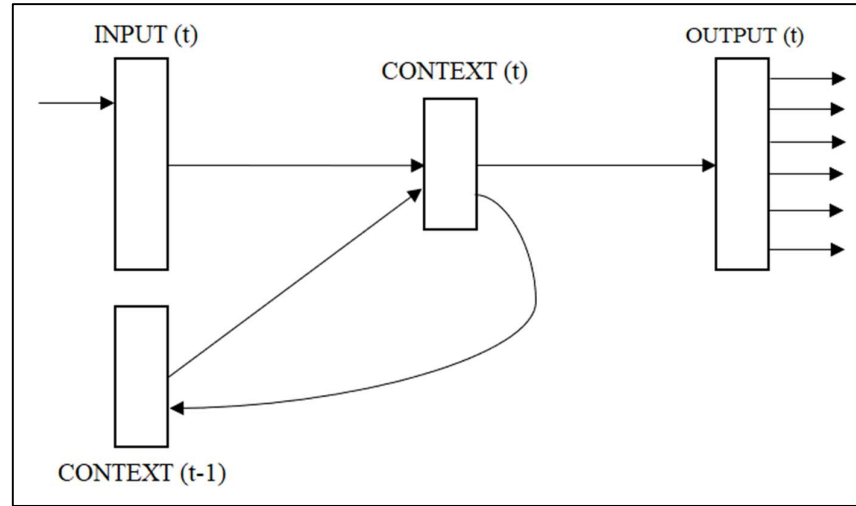


Figure 2.5 The simplest version of recurrent neural network
(Source: Mikolov, et al., 2010)

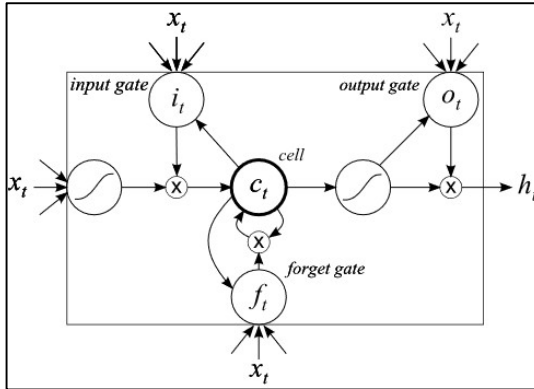


Figure 2.6 Inside the Long Short-Term Memory Block
(Source: Brueckner R. and Schuler B., 2014)

2.2.4.3 Convolutional Neural Network

Convolutional Neural Network, also known as CNN, is a well-known deep learning technique that was firstly invented by Fukushima K. (1980), but CNN became popular in 1998 by the publication of Yann, et al. (1998) who applied CNNs in their deep network called “LeNet-5”. In Yann, et al. (1998), the architecture of the deep convolutional neural network was presented for the first time for handwriting recognition in an image which was called “MNIST” dataset. According to Figure 2.7 LeNet-5 Network Architecture, LeNet-5 consists of convolutional kernels and sampling components.

CNN is not only applied to pattern recognition tasks in the field of an image which is represented in two-dimensional data, but also used in the field of Natural Language Processing (NLP). In NLP, there is a difference which is a dimension of input. Specifically, the input of the NLP is usually presented in a sequential form, which is one-dimensional data, referred as 1D data. Therefore, the CNN that moves the kernel in the 1D data is called 1D-Convolutional Neural Network (1D-CNN).

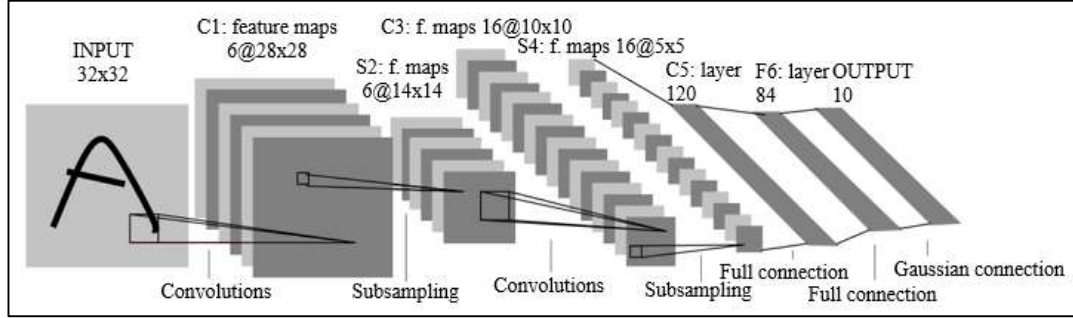


Figure 2.7 LeNet-5 Network Architecture
(Source: LeCun, et al., 1998)

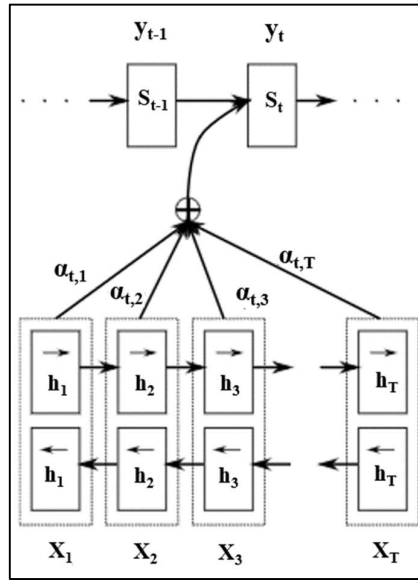


Figure 2.8 Attaching attention (α) into Neural Machine Translation model
(Source: Bahdanau, et al., 2014)

2.2.4.4 Attention Mechanism

Attention mechanism was invented to improve the machine translation model performance, which was usually built as Encoder-Decoder network architecture. Bahdanau, et al. (2014) proposed the attention mechanism and apply it to neural machine translation for the first time. Interestingly, the improvement of translation efficiency was significant since the attention weights provided the decoder focus points while translating the target, which was called Additive attention as shown in Figure 2.8. This is the beginning of many innovation researches that were applied in their works such as image captioning by Xu, et al. (2015). To compute an alignment score, there are several methods

including additive attention, dot-product attention, and scaled dot-product attention. The equations for computing the alignment score are presented in the rest of this section.

Additive attention mechanism is the weighted sum of the encoder hidden state which is used as a context vector in the decoder model. According to equation 2.1, additive attention (c_i) is a summation of the products of the weighted sums of the hidden state and final hidden state (h_j) when the model decodes the output y_i . In addition, e_{ij} is an alignment score that considers the hidden state j and previous state s_{t-1} (equation 2.3).

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \quad (2.1)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \quad (2.2)$$

$$e_{ij} = a(s_{t-1}, h_j) \quad (2.3)$$

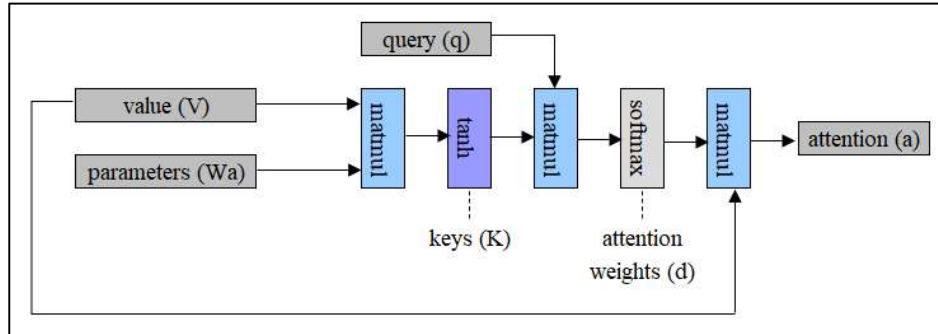


Figure 2.9 Architecture of dot product attention
(Source: Brown, et al., 2018)

Dot-product attention mechanism comprises 3 components including query (q), key (k), and value (v). Query is a vector that determines values to focus on by multiplying them with key and applies a softmax function to get the attention of each value as presented in Figure 2.9. The output of the attention layer (a) is computed by the following equations.

$$K = \tanh(VW^a) \quad (2.4)$$

$$d = \text{softmax}(qK^T) \quad (2.5)$$

$$a = dV \quad (2.6)$$

The variable \mathbf{K} is the key which is the function of the hyperbolic tangent of value \mathbf{V} multiplied by parameter \mathbf{W}^a (equation 2.4). After that, the key is multiplied by query

vector \mathbf{q} and fed into the softmax function to produce the vector \mathbf{d} (equation 2.5). Finally, the output \mathbf{a} is computed by the multiplication of \mathbf{d} and \mathbf{V} (equation 2.6).

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.7)$$

Furthermore, Vaswani, et al. (2017) introduced the scaled version of dot-product attention mechanism which attended to fewer number of key-value pairs applied to their Neural Machine Translation (NMT) called Transformer. Although the scaled dot-product attention attended to few entries of key-value pairs, the author applied the scaled dot-product attention multiple times on the inputs instead. To explain the scaled dot-product attention mechanism equation, Figure 2.10 represents the calculation process to compute the context vector. According to Figure 2.10, the calculation process differs from the normal dot-product attention mechanism in only the scale on the key inputs. Therefore, the equation of the scaled dot-product attention is stated as equation 2.7 where the parameter d_k represents the dimension of the key input.

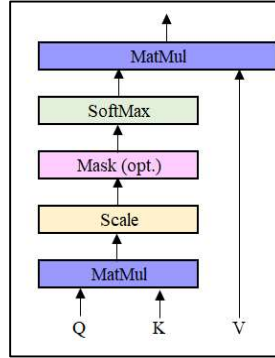


Figure 2.10 Scaled Dot-Product Attention Mechanism
(Source: Vaswani, et al., 2017)

2.2.4.5 Existing works

Anomaly detection is the task that attempts to capture patterns that deviate from the normal ones. In recent years, there are a lot of papers which apply the deep learning technique to anomaly detection problems in various ways. Mostly, the related sequential model is selected to handle the task because the anomaly detection is usually considered in the sequential context. Du, et al. (2017) is one of the papers that proposed the anomaly detection framework based on the sequential model which was called DeepLog. To detect the anomalies, the authors developed the language model of normal execution patterns based on the stacked LSTM which is a kind of RNN architectures as shown in Figure 2.11.

The framework consists of one log parser and three models including the log key anomaly detection model, parameter value anomaly detection model, and the workflow model for diagnosing detected anomalies. Interestingly, the detection level is on each log entry, so the detection mode can immediately capture the abnormality on the detection phase. To evaluate the model performance, the AUC-ROC which is the evaluation method for classification problem is used to measure the model performance. The DeepLog outperformed the traditional anomaly detection methods (e.g. PCA) in various ways

including the detection level and the model performance. However, the DeepLog performed the detection on the log key, thus the framework required the algorithm or template that could accurately map the log messages into the log keys.

Moreover, in the cybersecurity field, Tuor, et al. (2017) proposed the insider threat detection framework which assigned the anomaly score to each network authentication log. The authors handled the problem as the anomaly detection problem, so they applied the deep learning techniques to build the normal behavior model to capture an anomalous network activity. For performance comparison, they applied two kinds of the neural network models including DNN and RNN, and compared them to the traditional approaches (e.g. PCA, Isolation forest, and SVMs). The experimental results demonstrated that both deep learning techniques significantly outperformed the traditional models. However, this framework used the aggregated features as the inputs of the model which were computed based on each time window, for example, an hour, a day, or a week. Hence, the anomalous activity could be misdetected.

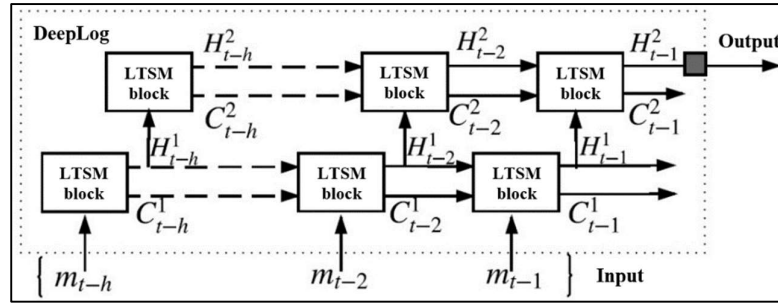


Figure 2.11 Architecture of stacked LSTM network
(Source: Du, et al., 2017)

Furthermore, Tuor, et al. (2018) came up with the novel anomaly detection on the same problem as stated in Tuor, et al. (2017). In this paper, the authors directly applied the RNN based language model on the raw system log, and compared it among different model constructions including normal LSTM model, Bidirectional LSTM model, and Tiered models. Additionally, the authors also developed each model in both the word-level model and character-level model to determine the suitable model for various situations. The objective function or cost function of the model was computed as the cross-entropy loss presented in equation 2.4:

$$Loss = \frac{1}{n} \sum_{t=1}^n H(x_{(t)}, p_{(t)}) \quad (2.8)$$

The variable n is the number of tokens that is fed to the LSTM network while $x_{(t)}$ is the input token at timestep t and $p_{(t)}$ is the predicted token at timestep t . According to Figure 2.12, at timestep t equal to 1, the cross-entropy loss is computed by using $x_{(1)}$ which is the actual value token and $p_{(1)}$ which is the prediction distribution of the previous timestep that is t equal to zero in this case. Then, the cross-entropy loss is used to update the parameter of the model and used as the anomaly score of each log line.

As reported in the paper, the character-level is the best for the immediate detection model while the word-level model is proper for the model with aggregated daily features at the end of the day. Unfortunately, the tiered models which have more complex architecture do not perform a significant difference compared to the normal LSTM and bidirectional LSTM. Subsequently, Brown, et al. (2018) introduced the upgraded version of Tuor, et al. (2018) which applied the attention mechanism to the models in their previous paper to improve the detection performance and produce more interpretable results. The model architecture is presented in Figure 2.13.

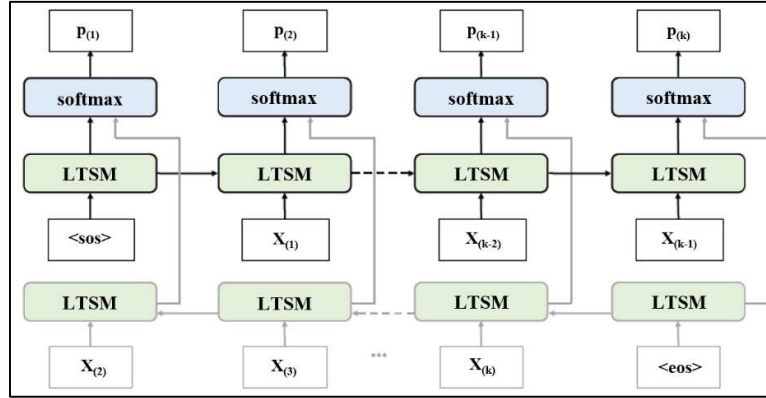


Figure 2.12 Normal LSTM (Black bordered nodes) and Bidirectional LSTM (all nodes) (Source: Tuor, et al., 2018)

In this paper, the authors experimented with a variety of query vectors using the dot product attention method which performed different detection performance. The equations and architecture of the dot product attention method were presented in 2.2.4.4. Moreover, the authors designed the experiment with five different query vectors. Surprisingly, the attention mechanisms could boost the model performance of the normal LSTM to be equal to the Bidirectional LSTM model in the word-level model. However, a different type of the query vector did not impact the model performance as all types performed the similar performance. Unfortunately, the attention did not outperform the previous model performance in the character-level models.

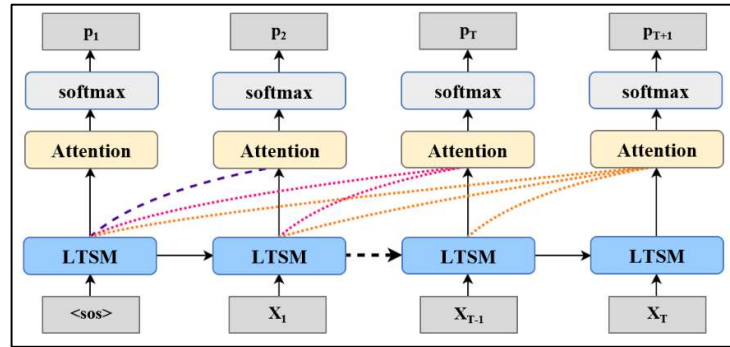


Figure 2.13 Normal LSTM attached with attention layer (Source: Brown, et al., 2018)

Besides the recurrent network, Convolutional Neural Network (CNN) is also chosen for analyzing system logs. Lu, et al. (2018) proposed the novel framework for detecting anomalies in HDFS dataset using CNN. Figure 2.14 represents the overall process of the proposed framework. Firstly, the raw log messages were parsed into the log key using the log parser and then fed into the embedding layer called Logkey2vec which was inspired by the word2vec proposed by Mikolov, et al. (2013). Interestingly, Logkey2vec is a trainable layer that can be updated during the model training session by gradient descent. Then, the convolutional layers received the embedded matrix to determine the hidden relationship between the logs. Finally, the feedforward neural network layer with the SoftMax function produced the output that classified the logs as either normal or abnormal.

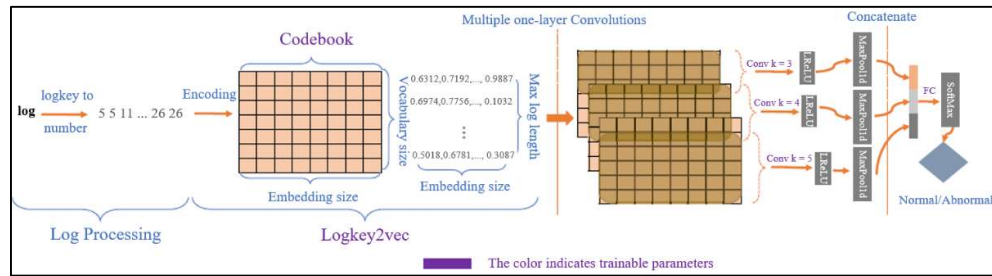


Figure 2.14 Overview construction of anomaly detection framework using CNN
(Source: Lu, et al., 2018)

The authors compared the CNN model with the existing LSTM model (Du, et al. (2017)) and the MLP model on the HDFS logs. Moreover, they also designed the experiment to study the impact of the embedding layer, thus they developed the CNN and MLP with and without the embedding layer. As stated in the paper, the CNN performed the outstanding model performance which reached 99 percent of accuracy and significantly beat all other baseline models. Furthermore, the model with the embedding layer outperformed the non-embedding model. In addition, the authors provided the reason why the CNN outperformed the LSTM network. Although the nature of the LSTM is suitable for sequential data, it does not actually perform best on the short sequence (e.g. system logs). Therefore, the CNN with the logkey2vec outperformed the LSTM approach with a reasonable difference.

2.3 Other Approaches

Detecting anomalies in the system logs can be treated as the language model as aforementioned, which is like the machine translation problem. Therefore, the authors surveyed the techniques used to solve the machine translation problem to be adapted to our anomaly detection problem. Firstly, Transformer, the interesting technique, was presented by Vaswani, et al. (2017). The authors came up with the novel machine translation framework using entirely the self-attention mechanism in the model without any recurrent network. Normally, the architecture of the machine translation consists of encoder and decoder to encode a sentence from one language into another language. More specifically, the encoder consists of sublayers including the attention component and position-wise feed-forward component. For the decoder, the alignment of components is the same as the encoder except the inclusion of the attention component which performs on the output of the encoder layer. Moreover, both the encoder and decoder use the embedded vectors as inputs. Interestingly, the impact of no recurrent layer can be solved

by the position encoding layer which can provide the relative information of token in the sequence. The overall architecture of the Transformer is demonstrated in Figure 2.15 which represents the encoder on the left-half and decoder on the right-half.

Besides the Transformer, Elbayad, et al. (2018) proposed the outperforming machine translation technique using a 2D convolutional neural network. The convolutional architecture was based on DenseNet proposed by Huang, et al. (2017) which provided the great performance on image classification. The authors demonstrated the improvement of performance against the Transformer and the other recurrent techniques. As reported in the paper, the 2D CNNs outperformed the other techniques on the small length of sentence. For the longer length, the 2D CNNs had slightly lower performance against the Transformer. However, focusing on the parameter of the model, the 2D CNNs had the performance close to the Transformer with a much smaller number of parameters than the Transformer, about 3 times. Therefore, the 2D CNNs technique is an interesting method that has a small number of the parameters, but produces high performance.

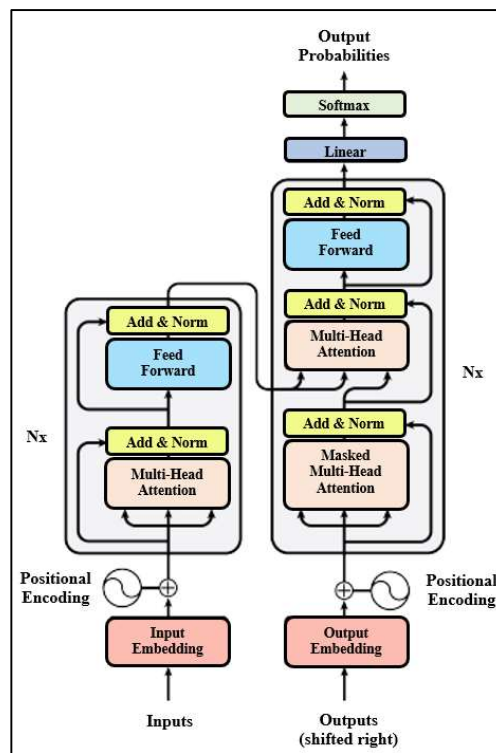


Figure 2.15 The architecture of Transformer
(Source: Vaswani et al., 2017)

2.4 Literature Review Conclusion and Discussion

In this section, the benefit and drawback of each approach will be discussed. Moreover, the objective of each work is described in Table 2.1. Firstly, the literature review can be clustered by the approach. There are three main approaches, namely, statistical approach, machine learning approach, and deep learning approach.

The statistical approach is the traditional approach to handle an anomaly detection task. In our review, PCA was applied to detect the outlier patterns in Xu, et al. (2009a, 2009b).

Although the detection performance is quite high on the HDFS and OpenStack dataset, this method is an offline method which requires the completed session of logs for processing. Therefore, this is the serious drawback of the statistical approach because it is unable to practically use in the real world.

The machine learning approach is the beginning of the novel anomaly detection approaches. Firstly, the machine learning was presented in the failure prediction area by Featherstun and Fulp (2010) using SVM with spectral kernel to handle sequence data. However, SVM is one of the classification techniques which needs the failure of the system dataset to train the model. Unfortunately, the failure is an infrequent occurrence, so the lack of data problem makes this approach impossible for the healthy server which never or rarely fails. Moreover, the clustering approach is more interesting for the anomaly detection task which does not require the labeled dataset to train the model. Du, et al. (2015) presented the hierarchical clustering to group the similar logs in the same cluster and then an anomaly score would be computed. Although the method is interesting, the detection efficiency is lower than the traditional way because the features that they used are not effective enough. Furthermore, the inspiration of the recent researches is the state machine which learns the patterns of logs and then creates the workflow model to represent the normal behavior. There are several algorithms such as Markov chain model presented in Haque, et al. (2017) and Automata models proposed by Fu, et al. (2009), Yu, et al. (2016), and Debnath, et al. (2018).

Table 2.1 Literature survey summary

Methodology	Benefit	Drawback
PCA	Capturing the log generating pattern and detecting the anomalous patterns.	Processing on session-level and Training offline.
SVM	Presenting the method that handles sequence logs with SVM kernel.	Requiring labeled dataset.
State Machine	Representing the state model that is simple to understand.	Detecting data based on some features of log entry which are not generalized for other systems.
LSTM	Operating on log-level and automatically updating the model.	Training recurrent network consumes so much time.
CNN	Extracting the hidden relationships between logs.	Requiring labeled dataset.
Transformer	Using self-attention instead of convolution and recurrent components. Performing great result in Machine Translation Task.	Never been used in anomaly detection task.

In recent years, the deep learning technique is popular in the anomaly detection area with amazing capability. Firstly, Du, et al. (2017) proposed the anomaly detection framework that could operate the detection on log-level called DeepLog. The learner of the DeepLog

is LSTM which learns the normal behavior of the log to build a language model. Surprisingly, the DeepLog outperformed the traditional methods as well as providing the model to be automatically updated for the unseen log entries. However, the DeepLog is based on the LSTM algorithm which is the recurrent network that is unable to parallelize the data in the training process, so the DeepLog spent so much time to build the model. After that, Lu, et al. (2018) came up with the novel anomaly detection framework with a convolutional neural network. The authors made use of the CNN that could extract the hidden relationship in the logs by the 2-dimension operation. Moreover, the reason that the CNN outperformed the LSTM was the characteristic of the log sequence which was not long enough to leverage the maximum performance of the LSTM. Therefore, the CNN is more suitable for detecting anomalies on the log sequence than the LSTM. Furthermore, the authors surveyed the cybersecurity area and found the interesting series of research including Tuor, et al (2017, 2018) and Brown, et al. (2018) which presented the solution for detecting insider threats of computer networks. The authors defined the problem as an anomaly detection problem, so this is an inspiration to be applied in this work. In the series, they proposed several frameworks including DNN, LSTM with various architectures. The most interesting approach was the model of the recurrent network attached with attention mechanism because it improved the model performance as well as representing the interpretable model while making a prediction. This work (Brown, et al. (2018)) inspired us to research on the attention mechanism which is applied in our work.

Normally, the attention mechanism is used to solve a machine translation task which is a sequence-to-sequence problem. There are two interesting papers including Vaswani, et al. (2017) and Elbayad, et al. (2018). Firstly, the first mentioned paper presented the entire self-attention architecture without any convolution or recurrent component. Secondly, the attention-like algorithm operated by the convolutional network performed the slightly high performance compared to the first paper. Moreover, the computation time was lower than the first paper because of a lower number of parameters to be adjusted while training. However, both approaches had never been applied to the anomaly detection task. Hence, the authors decided to apply the Vaswani et al. (2017) to the proposed framework because it consists only of the self-attention component.

In this research, the authors proposed the novel anomaly detection framework which was an end-to-end framework that comprised the Log Parser and the detection model, which was implemented on the deep learning technique. More specifically, the authors attempted to research on the single dimension Convolutional Neural Network (1D-CNN) as well as the Attention-based model called Transformer mentioned above. Furthermore, this research is the first research in the anomaly detection field that applied the Transformer model to the anomaly detection task. Finally, the comparison of our proposed framework with the detection models including the CNN and Transformer against the existing frameworks including the DeepLog which uses the LSTM and CNN-based framework will be contributed to prove that the self-attention outperformed the convolutional and recurrent components.

CHAPTER 3 METHODOLOGY

In this chapter, the explanation of the proposed methodology which is novel anomaly detection frameworks applied to the log events is presented. The authors compare the proposed approaches to the existing methods. Moreover, the authors experiment on the different model parameters to improve the model performance. The chapter is separated into three sections including dataset, methodology, and experimental design.

3.1 Dataset

In this work, the log dataset is obtained from DeepLog (Du, et al., 2017) which contains operation log sequences of HDFS. Fortunately, log messages are parsed into log keys which are ready to be pushed into the model. The dataset consists of normal and abnormal sessions of HDFS as presented in Table 3.1. Since the abnormal cases are rare, the dataset contains the normal operation cases in much larger volume than the abnormal ones, thus imbalance problems appeared. Therefore, this research will use only the normal cases of 1 percent of volume (4,855 sessions) to train the anomaly detection model.

Table 3.1 Number of records in each dataset

Dataset	Number of sessions			Number of unique log keys
	Normal	Anomaly	Total	
Training	3,884	-	4,855	14
Validating	971	-		
Testing	553,366	16,838	570,204	28
Total	558,221	16,838	575,059	28

As stated in Table 3.1, the dataset contains both normal and anomaly sessions that have different numbers of unique log keys. To show the distribution of log keys,

Table 3.2 represents the number of log key occurrences divided into three parts including Training-Normal, Testing-Normal, and Testing-Anomaly. Interestingly, there is a group of log keys that appears only in the testing set and some of them occur only in the anomaly session. However, almost the log keys that occur in the training set always occur in the testing set, which means they also include the anomaly set. Therefore, the distribution of log key occurrences represents the complexity of dataset which the normal and anomaly sessions are similar.

Moreover, the authors visualize the distribution of session length in Figure 3.1 to explain the characteristics of the normal and anomaly sessions. According to Figure 3.1 a), the density plot of the normal session length in the training set is shown with the average line at 19.59 log key per session. More specifically, the distribution of both train normal in Figure 3.1 a) and test normal in Figure 3.1 b) is similar. On the other hand, the distribution of the anomaly session length is different with lower mean of session length and higher deviation of session length. As shown in Figure 3.1 c), there are two peaks including peak point near zero and peak point at twenty. Finally, Figure 3.1 d) represents the overlapping area between the normal and anomaly session lengths, which shows the complexity of the anomaly detection which the normal and anomaly sessions could have the same length of session.

Table 3.2 Number of log key occurrences in dataset

Log key	Training	Testing		Total
	Normal	Normal	Anomaly	
1	-	-	10	10
2	1,070	115,836	3,129	120,035
3	3,735	413,148	11,842	428,725
4	3,213	344,733	8,259	356,205
5	14,595	1,662,687	45,945	1,723,227
6	29	2,588	4,480	7,097
7	-	-	90	90
8	-	-	49	49
9	14,565	1,660,098	31,845	1,706,508
10	-	-	89	89
11	14,565	1,660,098	32,010	1,706,673
12	-	-	34	34
13	-	-	1,464	1,464
14	-	-	77	77
15	-	-	65	65
16	29	2,513	4,395	6,937
17	-	-	47	47
18	29	2,514	4,459	7,002
19	-	-	5	5
20	-	222	5,323	5,545
21	11,908	1,354,660	35,473	1,402,041
22	4,856	553,366	16,838	575,060
23	11,880	1,352,365	31,923	1,396,168
24	-	-	4	4
25	29	2,514	4,459	7,002
26	14,622	1,664,857	40,256	1,719,735
27	-	3	972	975
28	-	12	1,276	1,288
Total	95,125	10,792,214	284,818	11,172,157

3.2 Methodology

The objective of this project aims to build the anomaly detection framework on the service logs dataset as presented in Figure 3.4. Thus, this research proposed a methodology which consists of three crucial parts including data preparation, deep learning model, and evaluation method. Next, the details of each part will be described.

3.2.1 Proposed Framework

To achieve our objective, the proposed framework architecture is designed to contain three primary parts as shown in Figure 3.4. Firstly, the data preparation part consists of three components including Log parser, Preprocessing, and Data partitioning, which manipulate the log entries to be a suitable form for building the model. Secondly, the modeling part is the most important part of our framework which the deep learning

techniques were applied to precisely detect the anomalous behaviors in the system. Lastly, the evaluation part aims to measure the model performance for detecting anomalies.

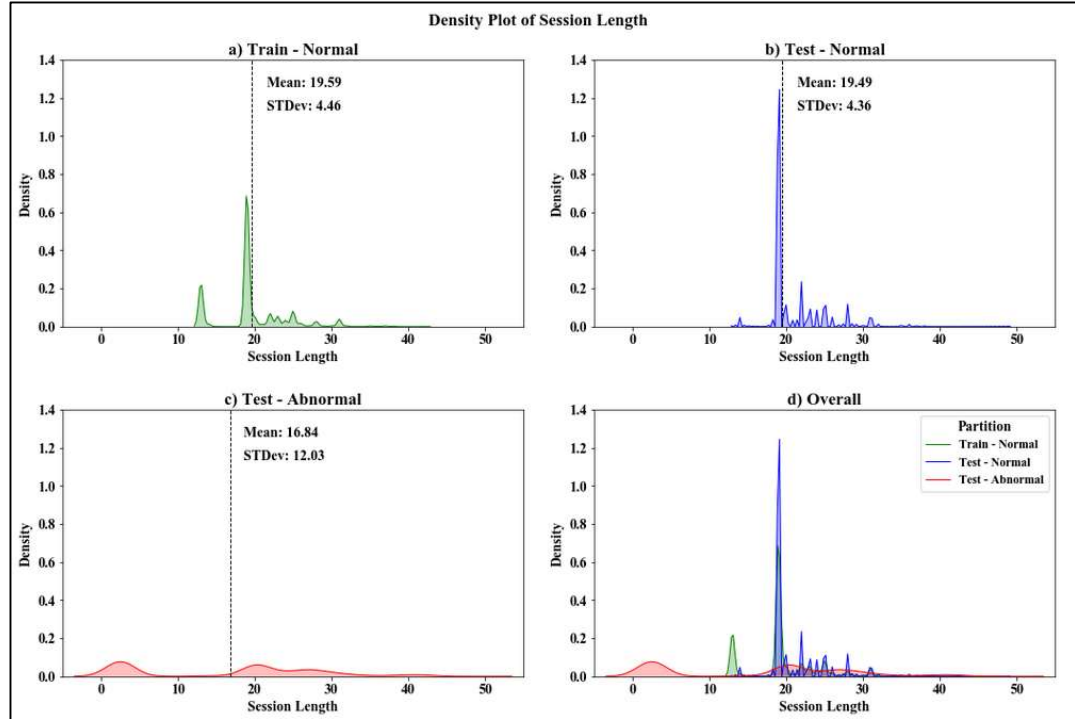


Figure 3.1 Density plot of session length

3.2.2 Data Preparation

In this work, the experiments are done on the HDFS operating system which is already preprocessed in the form of log key, so applying log parser is not required. However, in the real world, the operating log messages are coming in the form of unstructured text, so the authors must apply the log parsing process to them before using them to build the model. This is the reason why the log parser component in the data preparation step is injected before fetching the data to the modeling step.

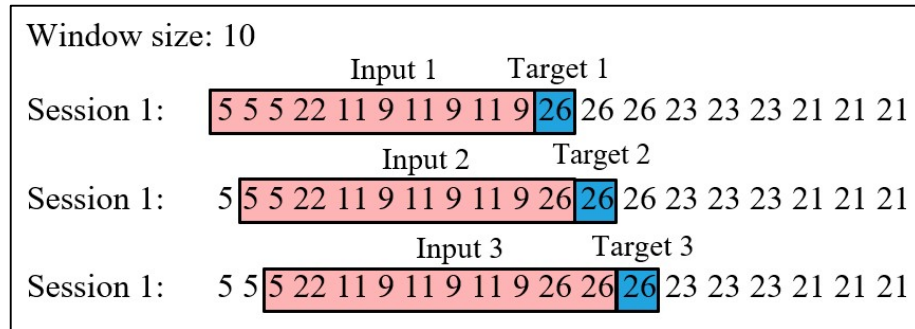


Figure 3.2 Example of sequence preparation

3.2.2.1 Log Parser

According to section 2.2.1, the existing log parsing techniques which performed the different performance on the different datasets are presented. For HDFS, there is the recent technique that can accurately parse the log messages into the log keys, which is called Drain as presented in He, et al. (2018). Fortunately, the Drain is published as an open-source python application. Therefore, the Drain is selected as our log parser. After the parsing step, the parsed logs are unordered, so the rearrangement based on the timestamp to group them into a group of the sessions by the identity field, which is called *blk_id* in HDFS, is applied. Moreover, with the limitation of the deep learning model, the input that is fetched to the model must be in the same length. Thus, the ordered session data are sliced into the input sequence and target log key by a sliding window as displayed by the red rectangle and blue rectangle respectively in Figure 3.2. Furthermore, the authors called the pair of the input and target as a sequence pattern. Interestingly, the authors found the variation of the sequence patterns in the data that over 33,000 kinds of the sequence patterns occurred in the normal set and there were nearly 20,000 unique patterns of the anomaly set. The bar chart in Figure 3.3 represents the number of the unique sequence patterns in both the normal and anomaly sets. With the variation of the unique patterns, the authors decide to handle the anomaly detection task with the deep learning techniques. Finally, the input-target pairs are passed to the next step, data partitioning.

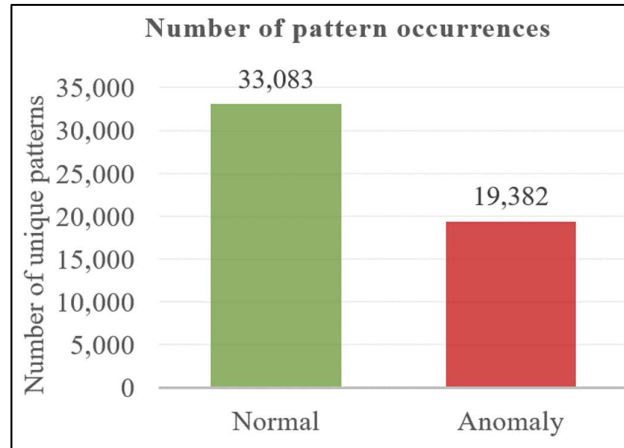


Figure 3.3 Number of pattern occurrences

3.2.2.2 Data Partitioning

Before being fed to the model, the dataset is separated into three sets including training set, validating set, and testing set. First, the training set entirely contains normal execution data which are used by the model to learn the patterns of normal behaviors. Secondly, the validating set also contains only normal execution data which are used to validate the learned model to adjust the parameters in each iteration. Lastly, the testing set contains a mix of the normal and abnormal behaviors which is used to evaluate the detecting performance of the model. Specifically, the separation is done on the session level which is the group of the log keys.

3.2.3 Model Training

In this work, there are two proposed deep learning architectures to handle the anomaly detection tasks, which are adapted and inspired from the recent existing works that were

mentioned in chapter 2. More specifically, this research studies two interesting models including CNN-based and Attention-based models. Therefore, the model architecture and parameter adjustment are separated for each model and presented in the following sub-sections. Moreover, the existing works including LSTM (Du, et al., 2017) model and CNN (Lu, et al., 2018) are implemented to be compared with the proposed work.

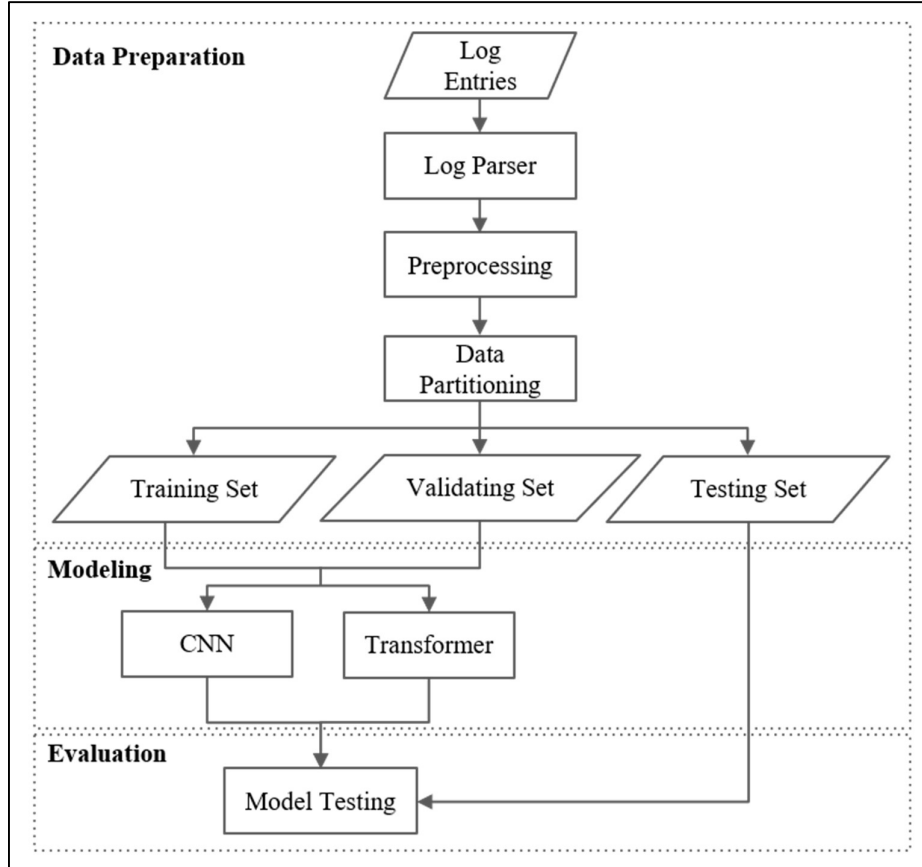


Figure 3.4 Overview architecture of our methodology

3.2.3.1 Convolutional-based Model (CNN)

Firstly, the authors would like to introduce the convolutional-based architecture which is adapted from the network proposed by Lu, et al. (2018). To make it more practical in the real world, the model is trained on the normal execution log events inspired by Du, et al. (2017) that aimed to build the language model. According to Figure 3.5, the network architecture is composed of five layers including:

1. Input Layer

Mainly, an input layer is responsible for handling an incoming log key sequence with the specific sequence length. Furthermore, the sequence of log keys is fetched through this layer to the embedding layer which is next to the input layer to be converted into the matrix.

2. Embedding Layer

Normally, a computer machine manipulates everything in a numerical way, but the input sequence of log keys is judged as a word in a text form. Thus, applying the embedding

process to an input vector is required before going further. In this layer, the vector of the log key sequence is encoded into a 2-dimensional matrix which is a suitable form for using in the convolution layer. Thus, the output shape from this layer is (Sequence length, Embedding size).

3. Convolution Layer

In our work, the convolution layers are composed of stacked 1-Dimensional Convolutional layers (Conv1D) which are designed to work as parallelized, Leaky Rectified Linear Unit (LReLU), and Max pooling layers as shown in Figure 3.5. Each of the convolution pairs has its own size of kernel and differs from the other because of the context reason and effect of different lengths of log key sequences (i.e. 1st Conv1D with kernel size 3 considers three log keys locating close to each other). Moreover, the outputs of the convolution components are activated by the Leaky Rectified Linear Unit (LReLU) to increase the training speed as well as solving the dying ReLU problem. Lastly, the Max pooling is applied to gain only the maximum value of each output from the LReLU to prevent the occurrence of an overfitting problem.

4. Concatenation Layer

Because of parallelized computing, the outputs of the convolution components must be merged into a single feature matrix in the concatenation layer. Furthermore, the concatenated output will be flattened into the vector which is the suitable form for being fetched to the classifier layer.

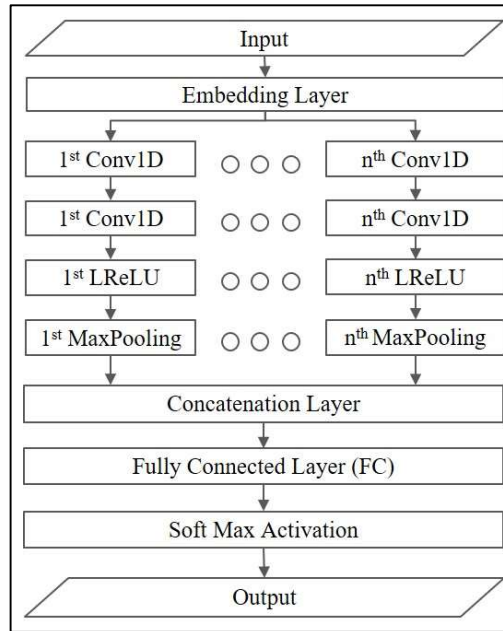


Figure 3.5 Convolutional-based network architecture

5. Fully Connected Layer (FC)

In this layer, dropout regularizer is applied to the flattened output vector to prevent the overfitting problem. After that, the dense layer with the SoftMax activation produces the output vector. Finally, the output of the model is the vector of the probability of a log key to be the next log key in the sequence.

3.2.3.2 Attention-based Model (Transformer)

The second model is adapted from Transformer's architecture (Vaswani, et al. (2017)), which is proper for the machine translation task that outputs the target sequence from the incoming sequence into the anomaly detection task and predicts the most possible token from the incoming sequence. More specifically, Figure 3.6 represents the attention-based model architecture which consists of five layers including input, embedding, encoder, decoder, and output layer.

1. Input Layer

The primary role of this layer is to carry the input and then transfer it to the next layer. According to the framework in Figure 3.6, the inputs of the encoder side and decoder side are different. For the encoder, the input is the sequence of the incoming log keys used to predict the next sequence. On the other hand, the sequence of the previous outputs is fed to the decoder used for calculating the self-attention score.

2. Embedding Layer

Normally, the machine learning techniques are learning the vector of the numerical value, so the required process before building the model is embedded. More specifically, embedding is a tool that maps word tokens from a vocabulary to the vector of the numeric which is useful for the language modeling task. Moreover, the embedding also reduces the dimension of data by using the contextual similarity.

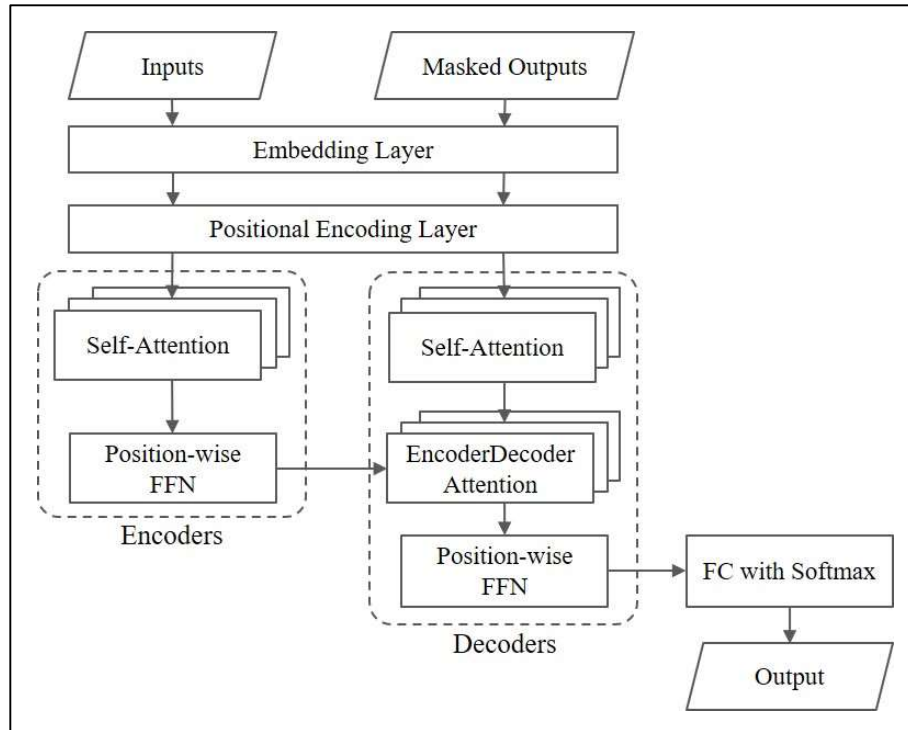


Figure 3.6 Attention-based network architecture

3. Encoder Layer

In the encoder, there are three sub-layers including the multi-headed self-attention, the feed-forward neural network, and the connection layer which perform the addition and normalization on the outputs of both previous sub-layers. More specifically, the multi-

headed self-attention consists of many attention heads which the number of heads can be adjusted. The output of the multi-headed attention is an attention score computed by the scaled dot product and SoftMax operation as shown in equation 3.1.

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3.1)$$

The variables Q, K, and V are the query, key, and value matrices respectively. Moreover, the d_k is the dimension of the key matrix used to scale the score before applying the SoftMax. Then, the normalization layer performs the addition and normalization operations on the attention scores before sending them to the feed-forward neural network.

4. Decoder Layer

In the decoder, the additional sub-layer is an encoder-decoder attention layer which receives the key (K) and value (V) matrices from the output of the last encoder layer and the query matrix (Q) from the decoder self-attention. This layer is important for predicting the next token because it helps the model to refer to the relevant token in the input sequence. For the remaining processes, the decoder performs the same thing as the encoder does such as calculating attention scores, adding and normalizing outputs between sub-layers.

5. Output Layer

The output layer consists of two layers including a linear fully connected neural network and a SoftMax layer. The fully connected layer converts the decoder output into the vocabulary vector which contains all possible log keys. Further, a SoftMax function is applied to the vector from the previous layer to produce the probability distribution. Finally, the maximum value is selected as the output of the model.

3.3 Experimental Design

In this section, the design of the experiment used in this work is presented. Firstly, the authors describe the details of the existing approaches which are used to be compared with our approach. Then, the description of each experiment will be presented. Finally, the evaluation method is presented at the end of the section.

3.3.1 Experimental Setup

Firstly, the experiment is separated into two parts including performance comparison against the existing approaches and experiment on parameter tuning of our methods. According to the survey in chapter 2, this research focuses on the anomalous behavior detection in service logs, so the authors choose the similar works, namely, DeepLog (Du, et al., 2017), CNN based anomaly detection (Lu, et al., 2018).

1. DeepLog is the anomaly detection framework which applies the stacked LSTM to learn the sequence patterns and build the language model.
2. CNN based anomaly detection is the novel framework which applies CNN to determine the hidden relationships among the tokens of sequences and then produce the possible next log key.

Moreover, the authors experiment on the parameter tuning to improve our model capability and observe the effect of each parameter adjustment. The regularization techniques such as dropout will be applied to our framework to prevent the overfitting problem.

3.3.2 Design of Experiment

As described in the previous section, there are two parts of the experiment to evaluate our approach capability against other approaches. Thus, the detail of each experiment will be described in the remaining of this section.

3.3.2.1 Hyperparameter Tuning

In this experiment, the authors focus on the hyperparameter adjustment of our proposed models and the CNN-based and Attention-based models, because there are a lot of hyperparameters that can be tuned to improve the model capability.

Objectives

1. To observe the performance impact of each hyperparameter adjustment.
2. To improve the detection performance of our model.

Dataset

To achieve the objective, the detection performance is evaluated on a dataset which is always used in the evaluation of the existing anomaly detection frameworks, that is, HDFS dataset. Therefore, the HDFS dataset is used to prove the performance of the model implemented according to the papers as well as to evaluate our model performance.

Output

The output of the experiment is the detection performance metric of each hyperparameter adjustment. Then, the best parameter setting is chosen from the best detection performance.

3.3.2.2 Comparison against Existing Approaches

According to the survey, there are similar researches which focus on the same objective as this work, so the experiment is designed to compare the existing approaches against our proposed approaches. There are two main objectives for this experiment.

Objectives

1. To prove the performance result that is stated in the papers.
2. To evaluate the performance of our method compared to the other approaches.

Dataset

The dataset in this experiment is also the same as the previous one, that is, HDFS dataset.

Output

The output of the experiment is the detection performance metric that will be described in section 3.3.3. Finally, the authors compare the detection performance of our method with other existing methods.

3.3.3 Evaluation

To evaluate the model performance, the authors analyze the performance of anomaly detection by using measurement metrics. In this work, the detection task is treated as one

of classification tasks which consists of two classes including anomaly and normal, so the evaluation metric for classification efficiency can be used in this work. Generally, there are four metrics which are usually used in a classification task including precision, recall, and f-measure. To calculate the metrics, the matrix that contains relevant terms called a confusion matrix is shown in

Table 3.3. Hence, the authors will introduce the confusion matrix at the beginning, and then the calculation of the metrics is presented.

According to

Table 3.3, the confusion matrix consists of four terms including True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN). In this work, the authors focus on detecting the anomalous behavior, so the positive class in the matrix is the anomaly class. On the other hand, the negative class represents the remaining class which is the normal class.

Table 3.3 Confusion matrix

		Predicted	
		Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

True Positive (TP) is the number of the predicted outputs that the model correctly predicts as the positive class. For example, the predicted result is an anomaly and the actual value is also an anomaly, then TP will be added by one.

False Negative (FN) is the number of the predicted outputs that the model incorrectly predicts in the actual positive class. For example, the predicted result is normal, but the actual value is an anomaly, then FN will be added by one.

False Positive (FP) is the opposite of FN which counts the number of the predicted outputs that the model incorrectly predicts in the actual negative class. For example, the predicted result is an anomaly behavior, but the actual value is a normal behavior, then FP will be added by one.

True Negative (TN) is the number of the predicted outputs that the model correctly predicts as the negative class. For example, the predicted result is a normal behavior and the actual value is a normal behavior as well, then TN will be added by one.

In this work, the authors aim to detect the anomalous behavior which is the rare occurrence that causes the imbalance class problem, so the accuracy of the model cannot be focused. Thus, the authors decided to use other metrics to evaluate our proposed methods including True Positive Rate, False Positive Rate, Precision, and F-Measure that will be described below.

True Positive Rate (TPR), also called Recall, is the percentage of the positive class that the model correctly detects. As explained above, the positive class in this work is the rare occurrence, so the TPR will be low when the model entirely predicts the majority class which is the normal class, then the awareness that our model is inefficient will be presented. The authors determine the TPR from equation 3.2.

$$TPR = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad (3.2)$$

False Positive Rate (FPR), commonly called false alarm rate, represents the misdetection rate that the normal class (Negative) is predicted as abnormal which is the positive class. Therefore, this will bother a system administrator who is alarmed by the model. The calculation of the false alarm rate is shown in equation 3.3.

$$FPR = \frac{\text{False Positive}}{\text{True Negative} + \text{False Positive}} \quad (3.3)$$

False Negative Rate (FNR) is the rate of the wrongly predicted actual positive class which means the positive class is identified as negative by the model. This is also called false alarm rate. In this work, this error seriously affects the administrators who maintain the system because the anomalous behavior is identified as normal, so it might cause unexpected failure in the server. The calculation of the FNR is presented in equation 3.4.

$$FNR = \frac{\text{False Negative}}{\text{True Positive} + \text{False Negative}} \quad (3.4)$$

Precision is the ratio between the number of the correctly predicted positive classes and the total of the predicted positive classes. The precision represents the bias of the model, for example, the entire predicted results are in the positive class, so the precision will be fifty percent in case of the balanced data calculated by equation 3.5.

$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (3.5)$$

F-measure is the measurement metric computed from the precision and recall values as shown in equation 3.6.

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3.6)$$

3.3.4 Hardware and Software Specification

Our experiments are conducted on Linux server machine with hardware specification displayed in Table 3.4. Additionally, the proposed methods are developed by Python programming and open-source python libraries including Keras, Drain, and NumPy.

Table 3.4 Hardware Specification

Component	Description
Operating System	Ubuntu 16.04 LTS
CPU	Intel Broadwell 8 cores
Ram	32 GB
Hard disk	100 GB

CHAPTER 4 EXPERIMENTAL RESULTS

In this chapter, results and analytical discussions of our experiments on our proposed approaches are demonstrated. This chapter is divided into 2 parts based on the experiment objective as mentioned in section 3.2.2.

Firstly, the hyperparameter tuning experiment is presented to determine the best set of parameters that performs the best detection efficiency and detection productivity as well. In this part, the authors demonstrate the effects across the variation of parameters in both CNN-based and Attention-based models. Additionally, the authors consider only a single parameter a time in order to make the results more reliable and better comparable.

Secondly, the authors aim to present our detection performance on the anomaly detection task. Therefore, the first part of chapter explains the outstanding performance of our proposed approach against the existing anomaly detection approaches using the general service logs dataset obtained from Hadoop Distributed File System (HDFS). In addition, the comparison in terms of detection efficiency and productivity between the deep learning architectures including Convolutional Neural Network (CNN), Long-Shot Term Memory (LSTM) and Attention-based model (Transformer) is presented in this part.

To handle with the variation of the experimental results, the authors evaluated five times for the best parameter setting of each method to determine the averaged measurement results, which made a performance comparison experiment more confidential. Due to the limitation of computational time, the hyperparameter tuning experiments are validated only a single time to find the best parameter setting on each model. Finally, all results represented in this chapter are the best results that this research has so far.

4.1 Experiments for Hyperparameter Adjustment

In this work, there are two network architectures which have multiple choices of parameter adjustment. To achieve the best set of model parameters, the authors designed the experiments on both proposed models to explore and study the impact of each model parameter. In this section, the authors demonstrate the experimental results and analytical discussions on both CNN-based model and Attention-based model.

4.1.1 Experiments on Convolutional-based Model

The authors examined various CNN architecture designs and parameter setups in this experiment. Based on the model architecture presented in Figure 3.5, the authors attempted to insert or remove some layers in order to determine the best architecture in anomaly detection. Our experiments were designed to determine the best set of the hyperparameter adjustment which is presented in Table 4.1. More specifically, four interesting parameters are defined for the CNN-based network architecture including window size (w), number of convolution layers (cl), number of convolution components in parallel (n), and number of candidate predictions (c). In Table 4.1, the parameter adjustment and model performance of each experiment is presented. Moreover, the authors defined the parameter values as a default set in the first record in Table 4.1. Note that the number of the kernel size values and pooling size values depend on the number of the convolution components because each component has its own kernel size differing from each other as described in 3.2.3.1.

Table 4.1 Experiments on hyperparameter adjustment of CNN-based model

No.	Parameter				Precision (%)	Recall (%)	F1 (%)	FPR (%)	FNR (%)
	<i>w</i>	<i>cl</i>	<i>n</i>	<i>c</i>					
Default	10	2	3	9	94.76	99.53	97.09	0.17	0.48
1	Variation of window size (<i>w</i>)								
1.1	6				89.61	86.25	87.90	2.26	13.75
1.2	8				93.51	91.14	92.31	1.41	8.86
1.3	12				93.51	96.00	94.74	1.39	4.00
2	Variation of convolution layer (<i>cl</i>)								
2.1	1				96.97	89.24	92.95	0.08	10.76
2.2	3				96.10	85.06	90.24	0.86	14.94
3	Variation of parallelized convolution component (<i>n</i>)								
3.1	1				95.76	90.49	93.05	0.12	9.51
3.2	2				96.33	91.31	93.75	0.11	8.69
3.3	4				94.81	97.33	96.05	1.11	2.67
4	Variation of number of candidate predictions (<i>c</i>)								
4.1	3				80.28	99.88	89.01	0.75	0.12
4.2	5				85.07	99.80	91.85	0.53	0.20
4.3	7				90.02	99.29	94.43	0.33	0.71

Firstly, the experiment was designed to study the impact of context information by varying the size of a sliding window (*w*). This study provided the effect of the context information when the model received different sizes of information to make a prediction. As stated in Table 4.1, the detection performance was improved in both precision and recall, and reached the highest point at $w = 10$. Moreover, the reduction of the misdetection rate (FNR) and false alarm rate (FPR) obviously represented the impact of the context information. However, the higher size of the window did not always perform the better performance, for example, the size of the window more than 10 was worse than the window size equal to 10 which was the peak point. To understand the effect of the window size, the authors explored the data with the different sizes of the window to study the impact of the window size on the anomaly detection task. Table 4.2 consists of the variation of the window size, single output pattern, and total unique patterns. More specifically, the single output pattern was a sequence pattern that had only one target as described in Figure 3.2 in the sequence preparation section. According to Table 4.2, the increment of the window size not only rose the portion of pattern that produced only a single output target, but also increased the total number of patterns that provided more complicated tasks for the model to solve. Thus, the optimal point according to the experimental result was the window size equal to 10 that covered quite the high amount of the single output pattern portions.

According to Figure 3.5, the authors defined the number of the consecutive convolution layers as two ($cl=2$) which could be called the stacked convolution layer. To demonstrate the impact of the convolution layers, the authors set up the experiment to vary the number of layers. The model architecture with the stacked layer performed significantly better precision than both single layer ($cl=1$) and triplet layers ($cl=3$), but slightly lower in recall as displayed in Figure 4.1 b). On the other hand, the FPRs of the single layer and stacked layer were slightly better, but the value of the FNR was significantly different, which

meant the stacked layer provided more capability to correctly capture the anomalous event. Finally, the stacked convolution layer was the most suitable anomaly detection model from the experimental results.

Table 4.2 Effect of window size in anomaly detection

Window size	Single output pattern (%)	Total Patterns
6	68.36%	9,178
8	76.05%	20,905
10	81.90%	36,373
12	84.74%	53,036

Furthermore, the number of the parallelized components (n) was considered to study the effect of kernel size as previously described. Normally, the convolutional neural network handles the context information with the size of kernel. Hence, the variation size of kernel could be a factor that provides the better performance. This research worked on 4 sub-experiments with 1,2,3, and 4 parallelized components which applied different kernel sizes to each component. Additionally, there were 4 sizes of kernel that were applied in the experiments including 2, 3, 4, and 5. According to Table 4.1, the experimental results explained the impact of the context information via the improvement of the model performance when the number of the parallelized components increased. Moreover, the results represented the combination effect of the extracted features which was from the different sizes of the kernel. However, the excessive information could not improve the model as displayed in Figure 4.1 c) when the number of the components was more than 3.

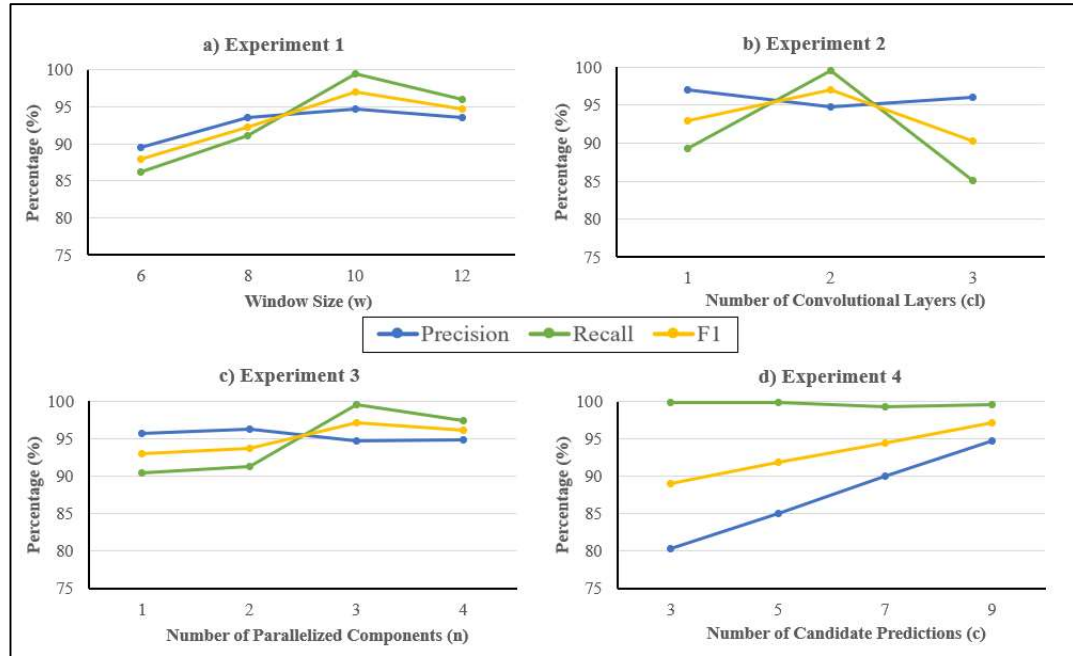


Figure 4.1 Hyperparameter tuning of convolutional-based model architecture

Since this work was considered as language modeling, the number of prediction candidates (c) was interesting to study that some sequential patterns could provide different prediction targets. Moreover, the authors found that over 86 percent of the patterns had only one output target which was the next event during training the dataset exploration to determine the appropriate value for the number of the prediction candidates. Interestingly, the rest of the sequence patterns covering about 16 percent provided more than 1 output target, and this would be the false alarm if the authors applied the top-1 prediction candidate. Therefore, the authors drilled down on the multiple output target set and found that the maximum number of the different output targets in the training dataset was 7 which meant one sequence pattern could perform seven different events. Nevertheless, the training dataset was only 1 percent of the overall dataset which could not be a representation of the overall data, so the authors evaluated the model on the different numbers of the prediction candidates. Surprisingly, the experimental results demonstrated in Table 4.1 refused the fact that was found during the exploration. The model with the top-7 prediction candidates could perform only 90 percent of precision. Hence, the authors increased the number of the prediction candidates to be the top-9 prediction that performed 94.76 percent of precision, and the model could retain the recall at 99.53 percent as shown in Figure 4.1 d). Although the increment of the prediction candidates provided better precision, maintaining the recall was necessary as well, so the top-9 prediction was the most suitable value to keep both precision and recall at an acceptable level.

4.1.2 Experiments on Attention-based Model

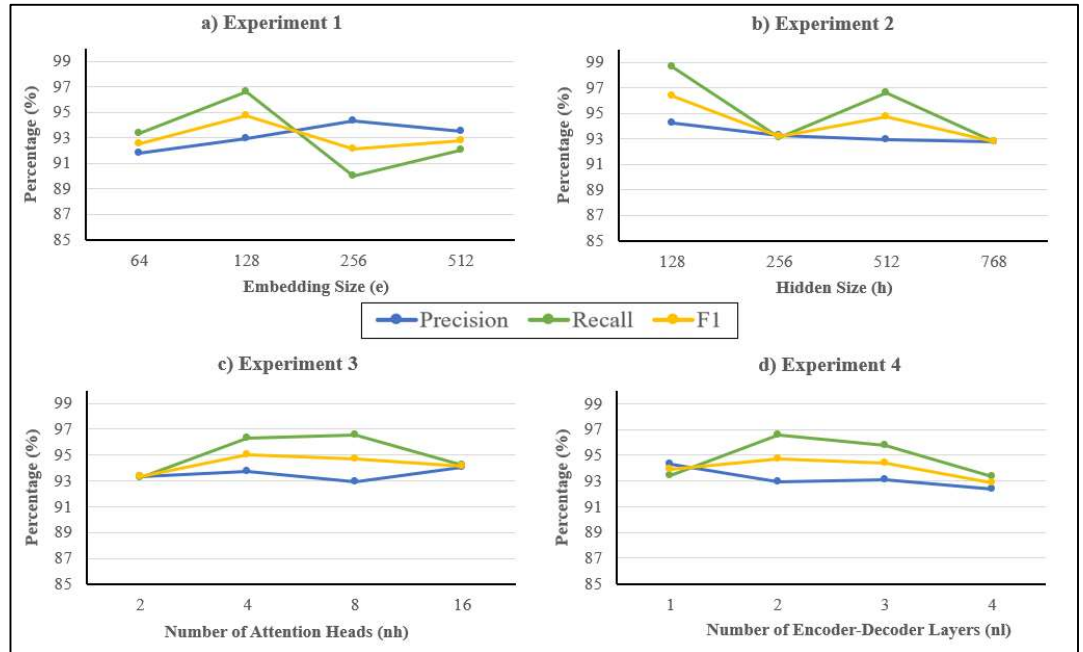
In this experiment, the authors explored the impact of parameter variation on the Attention-based network. According to the network architecture in Figure 3.6, the authors focused on 4 parameter adjustment which potentially affected the model efficiency including embedding size (e), hidden layer size (h), number of attention heads (nh), and number of encoder and decoder layers (nl). The details of the experiments and measurement of model performance are presented in Table 4.3. Additionally, the authors defined the parameter values as the default set in the first record in Table 4.3.

The authors designed the experiments for the attention-based model after all CNN-based experiments were done. Thus, in this experiment, the authors used knowledge and experiences from the previous experiment including the number of prediction candidates. Moreover, the authors defined the default parameter set based on the original Transformer model in Vaswani, et al. (2017) and modified some parameters to be suitable for our situation including the number of encoder-decoder layers.

Firstly, the authors studied the embedding size (e) which comprised positional embedding and word embedding vectors. According to the experimental results, the model with the lower size of the embedding vector had lower efficiency of the precision metric. On the other hand, the recall was significantly improved when increasing the size of the vector, however, after the peak with the embedding size equal to 128, the recall performance dramatically decreased as presented in Figure 4.2 a). From this experiment, the authors concluded that the size of the embedding vector should be proper to the size of the vocabulary because our vocabulary was not large, so the 128-dimension features were enough.

Table 4.3 Experiments on hyperparameter adjustment of Attention-based model

No.	Parameter				Precision (%)	Recall (%)	F1 (%)	FPR (%)	FNR (%)
	<i>e</i>	<i>h</i>	<i>nh</i>	<i>nl</i>					
Default	128	512	8	2	92.93	96.56	94.71	0.22	3.44
1	Variation of embedding size (<i>e</i>)								
1.1	64				91.78	93.33	92.55	0.25	6.67
1.2	256				94.27	89.96	92.07	0.17	10.04
1.3	512				93.51	92.03	92.77	0.19	7.97
2	Variation of hidden layer size (<i>h</i>)								
2.1	128				94.19	98.65	96.37	0.19	1.35
2.2	256				93.28	93.10	93.19	0.20	6.90
2.3	768				92.72	92.80	92.76	0.22	7.20
3	Variation of attention heads (<i>nh</i>)								
3.1		2			93.31	93.29	93.30	0.20	6.71
3.2		4			93.74	96.34	95.02	0.20	3.66
3.3		16			94.06	94.18	94.12	0.18	5.82
4	Variation of number of encoder-decoder layers (<i>nl</i>)								
4.1			1		94.32	93.44	93.88	0.17	6.56
4.2			3		93.10	95.76	94.41	0.22	4.24
4.3			4		92.36	93.31	92.83	0.23	6.69

**Figure 4.2** Hyperparameter tuning of attention-based model architecture

the hidden size did not perform the improvement of the model performance, but slightly dropped from one to one in each experiment. Therefore, the authors realized that the larger hidden size could make the model confused with some noisy features that did not provide important information for the model.

In the attention-based model, there was the number of attention head parameters (nh) to consider because the information from the different numbers of heads could cause the different levels of efficiency of the model. The number of attention heads was increased by the power of two including 2, 4, 8, and 16 because each attention head had to have the same size of the input vector. From the experimental result, the larger number of the attention heads provided higher precision performance. On the other hand, the extreme small and large number of the attention heads performed worse on recall than the moderate number of the attention heads i.e., 4 and 8 heads.

Finally, the number of encoder and decoder layers (nl) was considered to study and determine the suitable value for our work. In this experiment, the number of layers was like the previous experiment that the authors attempted to stack the convolution layer. Thus, the authors varied the number of layers from a single layer until four layers to demonstrate their impact. According to Table 4.3, the authors found the trend of the decrement of precision while increasing the number of the layers. However, the recall had different trends that significantly increased until reaching a peak at two layers and slightly dropped at three layers, then dramatically decreased at four layers. Hence, the authors decided to choose the stacked encoder-decoder layer to maintain the precision and recall.

Table 4.4 Detection efficiency comparison

No.	Approach	Precision (%)	Recall (%)	F1 (%)	FPR (%)	FNR (%)
1	LSTM + No Embedding	95.92	95.25	95.56	0.12	4.75
2	Bi-LSTM + No Embedding	95.74	91.34	93.40	0.12	8.66
3	Stacked CNN + No Embedding	88.06	94.77	91.28	0.39	5.23
4	Transformer + No Embedding	87.33	99.16	92.87	0.44	0.83
5	LSTM + Embedding (Du, et al., 2017)	96.34	97.05	96.68	0.11	2.95
6	Bi-LSTM + Embedding	94.69	97.10	95.88	0.17	2.90
7	CNN + Embedding (Lu, et al., 2018)	96.71	87.02	91.60	0.09	12.98
8	Stacked CNN + Embedding (Our proposed)	94.52	99.17	96.79	0.17	0.83
9	Transformer (Our proposed)	93.96	98.61	96.23	0.20	1.39

4.2 Experiments for Anomaly Detection Performance Comparison

According to the objective stated in 3.3.2.1, the authors attempted to prove the experimental results of the recently published papers and compare them to our proposed methodology. In this experiment, the term of performance was defined as 2 terms including term of detection efficiency and term of detection productivity. Firstly, in section 4.2.1, the detection efficiency of the anomaly detection frameworks is demonstrated. In addition, the analytical discussions about our experimental results are presented in this section. In the following section, the authors compared the algorithm detection rate of our proposed methodology against the existing detection algorithm in order to show the outstanding productivity of our proposed methodology.

4.2.1 Detection Efficiency Comparison

Generally, the key measurement metric to demonstrate the performance of the model is the efficiency of the detection model. In the anomaly detection field, the authors focus on the detection efficiency as well. Hence, the experiment was designed to compare our proposed models' efficiency against the existing works including LSTM, Bi-LSTM, and CNN.

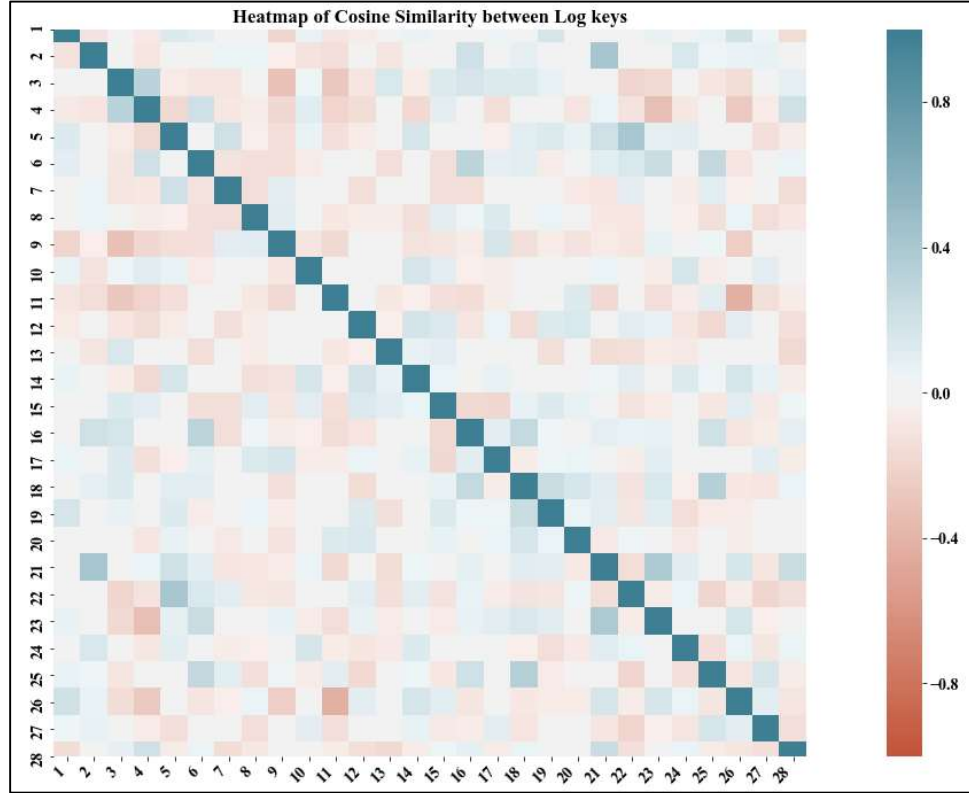


Figure 4.3 Heatmap of cosine similarity between log keys

For the experimental results as shown in Table 4.4, the authors separated the experiment into two ways including the models with and without the embedding layer to study the impact of the embedding layer in the feature extraction task. Firstly, the authors began the experiment with four models without the embedding layer presented in the first four rows in Table 4.4. As the results show, the LSTM could outperform the other models in terms of recall and F1 without the embedding layer. After that, the embedding layer was applied to four prior models to study the difference. Interestingly, the embedding layer could improve the recall significantly and obviously reduced the misdetection rate and false alarm rate in every model except the Transformer. On the other hand, there was no significant improvement of the precision in both the LSTM and Bi-LSTM models while the CNN and Transformer could be remarkably improved about 8 percent of precision. More specifically, the embedding layer provided the relation information among the log keys for the model via the latent features extracted from the log key that improved the predicting efficiency. As presented in Figure 4.3, the relationship between the log keys was represented through a heatmap of cosine similarity between the log keys. Moreover, the stacked CNN with embedding performed the higher recall and F1 compared to the LSTM and Bi-LSTM models with the embedding layer because the CNN was better to

find a local pattern in the sequence. Thus, applying the embedding layer to our proposed network architectures is the best idea.

According to Figure 4.3, the relationship among the log keys of the HDFS dataset was presented in the form of cosine similarity. As visualized in the heatmap, the amount of similarity represented the level of the relationship between two log keys. For example, the log key number 5 was mostly similar to the log key number 22 represented by dark blue color in the heatmap. Moreover, there was a negative side of similarity that meant they were similar in the opposite way i.e., the log key number 26 and log key number 11. With this relationship, the detection efficiency of the CNN model and attention model was improved.

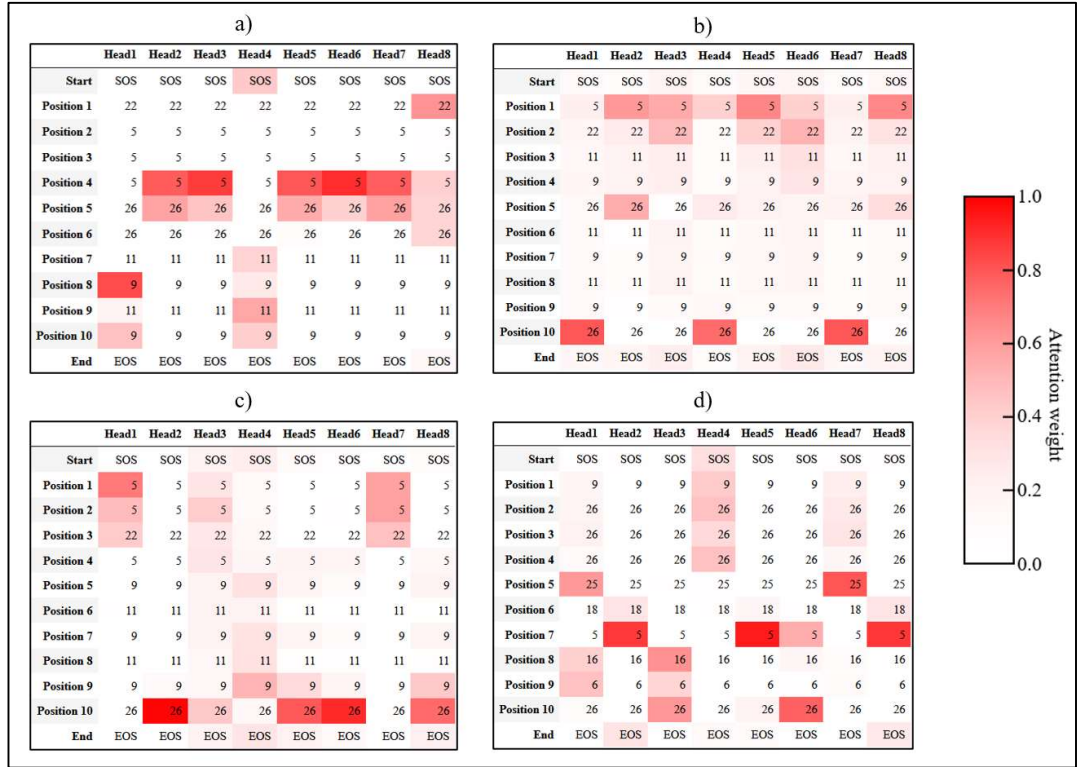


Figure 4.4 Example plot of attention weights

Since the Transformer model was based on the attention mechanism, the authors decided to briefly discuss the attention map which represented the interaction and relationship between the input and output during the prediction step. The example of the attention map shown in Figure 4.4 represented the attention weight of each input log key in the sequence while the model predicted the log key “26” as the most possible to be the next event. Note that the attention weight lied between 0 and 1 which meant how much the model paid attention to that input. Moreover, the attention map consisted of 8 columns because of the multi-head attention component that meant each column contained the attention weight of each attention head. As shown in Figure 4.4 a), almost attention heads paid attention to the log key number 5 at position 4 as well as the log key number 26 at position 5 while predicting the log key number 26 which could imply that there was some relationships between the log keys 5 and 26 in those positions. Moreover, the figure showed that the model made use of the position feature while making a prediction as well. To visualize

the variation of the pattern and attention map, the remaining figure in Figure 4.4 displayed some samples of other patterns which the model prediction result was the log key number 26 as well.

Table 4.5 Analysis of variance of recall over 5 models

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	4	0.048463	0.012116	62.18	0.000
Error	20	0.003897	0.000195		
Total	24	0.05236			

Finally, the authors also came up with the statistical evidence including the analysis of variance (ANOVA) table and Dunnett test. According to the ANOVA table presented in Table 4.5, the analysis of the baseline models including the LSTM, Bi-LSTM, and CNN models and the proposed models including the Stacked-CNN and Attention models was done on the recall metric. Interestingly, the p-value stated in

Table 4.5 was less than 0.1 which meant there was a pair of models that had different means of recall at 90% confidence. To determine which pair was different, the Dunnett test was applied to show the confidence interval of each pair. As shown in Figure 4.5, the confidence interval of the pair of the baseline and stacked-CNN models did not contain zero that meant the average recall of the stacked-CNN and baseline models was significantly different at 90% confidence.

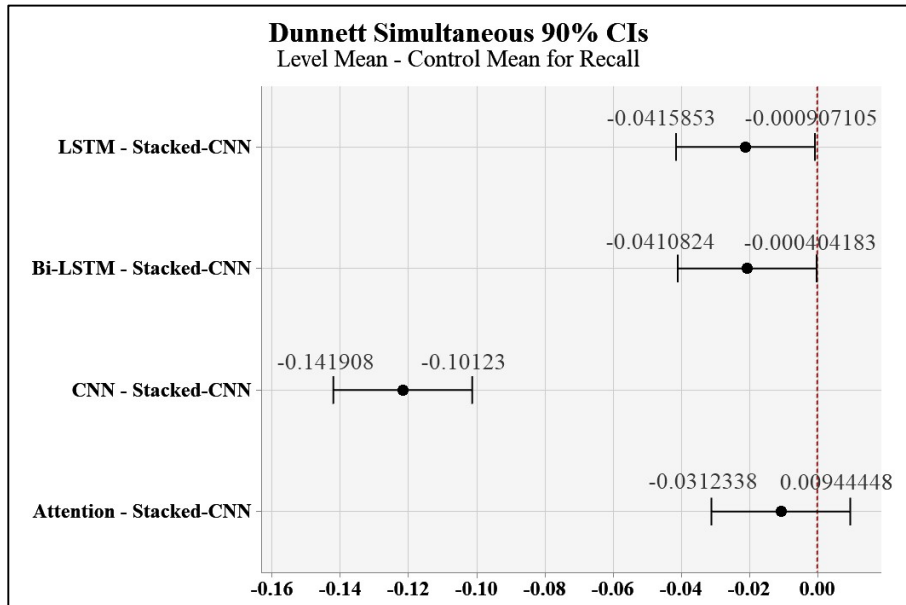


Figure 4.5 Dunnett post hoc test for determining pair with different means

Therefore, the proposed methods with the best hyperparameter setting significantly outperformed all existing anomaly detection methods with very high recall and very low misdetection rate and false alarm rate. Moreover, the authors have demonstrated that the attention-based network (Transformer) could perform close to the CNN efficiency as presented in Figure 4.5, and could significantly beat the LSTM-based network performance.

4.2.2 Detection Productivity Comparison

Although the detection efficiency of the model is the first thing that should be considered to determine the best model, the detection productivity is as important as the detection efficiency in the anomaly detection field because detecting anomalous events should be done as a near real-time system. For example, the model performs very high detection efficiency, but consumes very much time on detection as well. Finally, the model is delayed to capture the abnormal event that might cause the impact on the system. Therefore, to make our proposed framework more practical, the authors decided to consider the performance in terms of time consuming such as training time as well as the prediction productivity of the models.

Firstly, training time is the time spent during modeling the detection model (i.e., modeling CNN). Time consuming in the training process among the LSTM, Bi-LSTM and our proposed models (CNN-based and Attention-based) is displayed in Figure 4.6. The figure represents the training time of all models both without and with the embedding layer. As you can see, the family of the recurrent networks consumed a lot of training time since they normally did the computation in a sequential way while the convolutional-based and attention-based networks could do it in parallel. Hence, our proposed methods were much faster to build the detection from the same size of the dataset. Moreover, the scalability of our proposed methods is higher because of the lower training time usage.

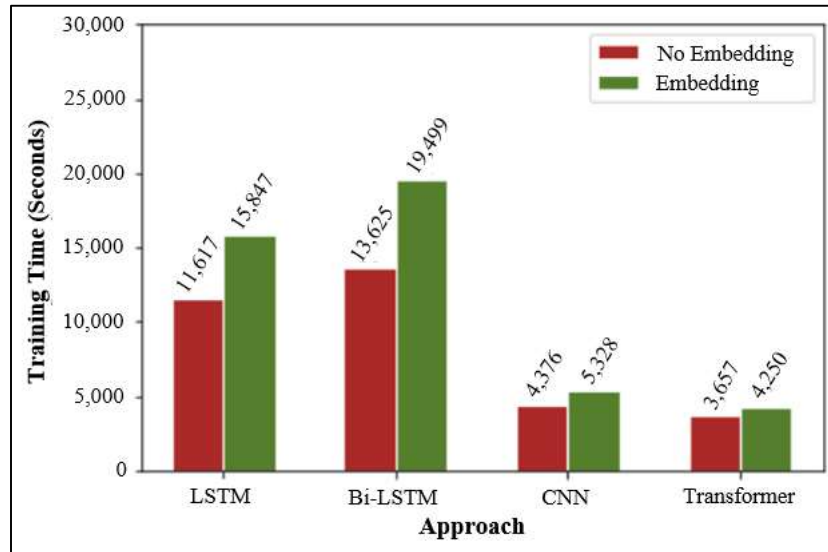


Figure 4.6 Training time comparison

Furthermore, the authors demonstrated the prediction productivity in terms of the number of the log events that the model could predict per second. As shown in Figure 4.7, our proposed models provided the higher productivity than both the LSTM and Bi-LSTM about 2.5 times. As described above, the CNN-based and attention-based models performed the parallelized computation, therefore, they could make a prediction faster than the RNNs family which operated in a sequential way. Moreover, this research found that the convolutional-based network had higher prediction productivity compared to the Transformer model.

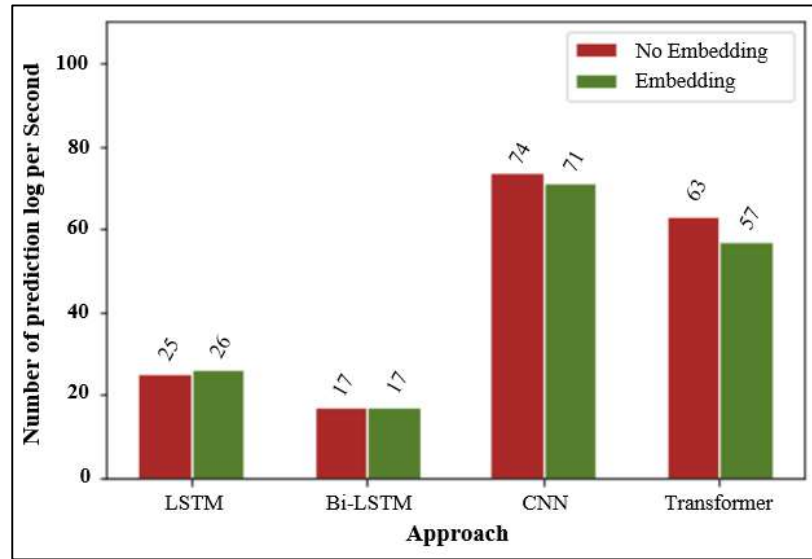


Figure 4.7 Detection productivity comparison

In the practical situation, the anomaly detection must be able to support the real-time system. Hence, the detection model must have low training time and high prediction productivity. Therefore, our proposed methods are more practical to the real-world applications especially the CNN model compared to the LSTM and Bi-LSTM models which are the popular model architectures in the anomaly detection field.

4.3 Methodology Limitation Discussion

After the experimental result demonstration and presentation of the limitations and drawbacks of the existing approaches, this section will be the space to discuss the limitation and drawback of our proposed model. Although our proposed methodologies performed great detection efficiency, there is a limitation of each method that will be discussed in detail.

Since our proposed methodologies applied the deep learning techniques, the fixed size of the model inputs was always required. With the limitation of deep learning, the authors had to prepare the input data as a fixed length of window to capture the information of log sequences to accurately predict the following events. However, in the real world, anomalous events can occur during the gathering phase that means the input is not ready to be fed into the model to predict, so unexpected anomalous events will occur before our proposed methods make a prediction. Consequently, the authors suggest applying the padding to the incomplete sequences that have the operating time of last event more than some threshold values of the normal operation to make it ready to be used by the model to predict the next log event. Although the early detection provides the higher false alarm because of incomplete information of inputs, the administrator will be noticed to check the incomplete sequences, which consumes too much time to perform some tasks that might be the abnormal cases in the future.

CHAPTER 5 CONCLUSION

With the rapid growth of technology in the world, computing cluster is an essential part of almost all technology companies who drive their businesses on modern technologies. Since the modern technologies require the high availability and consistency, computing cluster is always maintained by administrators to serve those tasks. However, the unexpected failure of the machines can occur all the time and seriously affects the computing tasks. Detecting an anomalous event as early as possible can assist the maintenance process in order to avoid an unexpected breakdown. Consequently, anomaly detection is essential for the distributed systems.

Since anomaly detection is a time-series problem, many research papers applied the recurrent network family to capture the anomalous events by treating the problem as a linguistic problem. Therefore, the model was introduced as language modeling to predict the most possible event to occur in the next timestep. Although the detection performance is very well, the drawback of the recurrent network is the wasted time consumed on both training and inference phases because of the sequential computing fashion of the recurrent network. To tackle the recurrent network drawback, the convolutional network was proposed in a single dimension form to solve time-series problems called 1D-CNN. Moreover, attention mechanism, namely Transformer, is the interesting technique that was proposed to handle sequence-to-sequence problems, i.e., machine translation which also solve the drawback of the recurrent network. Thus, this research focused on both 1D-CNN and Transformer models. More specifically, the models were applied to build a language model on the normal execution log events to predict the most possible event (log key) according to the incoming sequence of events. The anomalous event will be detected when the actual log event deviates from the prediction of the model. In the validation phase, the authors evaluated the performance of our proposed framework in terms of precision, recall, and F-measure. Due to the anomaly detection, the authors have focused on the recall performance in order to avoid the miss detection.

In the performance evaluation, we applied the proposed methods with Hadoop Distributed File System (HDFS) operation logs. The experiments were separated into two parts including hyperparameter tuning experiment and performance comparison experiment. According to the experimental results, our proposed frameworks could outperform the existing models in the anomaly detection field including both the LSTM and Bi-LSTM models with higher recall and F-measure. Consequently, the authors have identified the best parameter setting for our proposed CNN model that could achieve the precision = 94.52%, recall = 99.17%, and f-measure = 96.79%. On the other hand, the best parameter setting for the Transformer model was also identified close to the performance against the CNN model which included the precision = 93.96%, recall = 98.61%, and f-measure = 96.23%. Moreover, our proposed frameworks could provide more productive predictions than the LSTM and Bi-LSTM because the CNN and Transformer were operated in fully parallelized mode without any dependency on other time steps that made our proposed models consume lower training and predicting time than the RNN architectures. In the practical situation, the anomaly detection must be able to support the real-time system. Hence, the detection model must have low training time and high prediction productivity. Therefore, our proposed frameworks are more practical to the real-world applications compared to the LSTM and Bi-LSTM models.

REFERENCES

- Bahdanau, D., Cho, K. and Bengio, Y., 2014, **Neural Machine Translation by Jointly Learning to Align and Translate**, [Online], Available: <https://arxiv.org/abs/1409.0473> [2020, March 23].
- Brown, A., Tuor, A., Hutchinson, B. and Nichols, N., 2018, **Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection** [Online], Available: <https://arxiv.org/abs/1803.04967> [2019, April 6].
- Brueckner, R. and Schuler, B., 2014, “Social Signal Classification Using Deep BLSTM Recurrent Neural Networks”, **Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**, 4 May 2014, Florence, Italy, pp. 4823-4827.
- Chalapathy, R. and Chawla, S., 2019, **Deep Learning for Anomaly Detection: A Survey** [Online], Available: <https://arxiv.org/abs/1901.03407> [2019, March 24].
- Chandola, V., Banerjee, A. and Kumar, V., 2009, “Anomaly Detection: A Survey”, **ACM Computing Surveys (CSUR)**, Vol. 41, No. 3, pp. 15.
- Chuvakin, A., 2009, **Public Security Log Sharing** [Online], Available: <http://log-sharing.dreamhosters.com/> [2019, April 8].
- Debnath, B., Solaimani, M., Gulzar, MA., Arora, N., Lumezanu, C., Xu, J., Zong, B., Zhang, H., Jiang, G. and Khan, L., 2018, “LogLens: A Real-time Log Analysis System”, **Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)**, 2 July 2018, Vienna, Austria, pp. 1052-1062.
- Du, M., Li, F., Zheng, G. and Srikumar, V., 2017, “Deeplog: Anomaly Detection and Diagnosis from System Logs through Deep Learning”, **Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security**, 30 Oct 2017, Texas, U.S.A., pp. 1285-1298.
- Du, S. and Cao, J., 2015, “Behavioral Anomaly Detection Approach Based on Log Monitoring”, **Proceedings of the 2015 International Conference on Behavioral, Economic and Socio-cultural Computing (BESC)**, 30 October 2015, Nanjing, China, pp. 188-194.

Edgeworth, F.Y., 1887, "Xli. on discordant observations", **The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science**, Vol. 23, No. 143, pp. 364-375.

Elbayad, M., Besacier, L. and Verbeek, J., 2018, "Pervasive Attention: 2D Convolutional Neural Networks for Sequence-to-Sequence Prediction", **Proceedings of the Conference on Computational Natural Language Learning (CoNLL 2018)**, 31 October 2018, Brussels, Belgium, pp. 1-11.

Featherstun, R. and Fulp, E.W., 2010, "Using Syslog Message Sequences for Predicting Disk Failures". **Proceedings of the 24th International Conference on Large Installation System Administration**, 7 November 2010, California, U.S.A., pp. 1-10.

Fu, Q., Lou, J.G., Wang, Y. and Li, J., 2009, "Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis", **Proceedings of the 2009 9th IEEE International Conference on Data Mining**, 6 December 2009, Florida, U.S.A., pp. 149-158.

Fukushima, K., 1980, "Neocognitron: A Self-Organizing Neural Network Model for A Mechanism of Pattern Recognition Unaffected by Shift in Position", **Biological Cybernetics** 36. 1 April 1980, Vol. 36, No. 4, pp. 193-202.

Fulp, E.W., Fink, G.A. and Haack, J.N., 2008, "Predicting Computer System Failures Using Support Vector Machines", **Proceedings of the 1st USENIX Conference on Analysis of System Logs**, 7 December 2008, California, U.S.A., pp. 5-5.

Hamooni, H., Debnath, B., Xu, J., Zhang, H., Jiang, G. and Mueen, A., 2016, "Logmine: Fast Pattern Recognition for Log Analytics", **Proceedings of the 25th ACM International on Conference on Information and Knowledge Management**, 24 October 2016, Indiana, U.S.A., pp. 1573-1582.

Haque, A., DeLucia, A. and Baseman, E., 2017, "Markov Chain Modeling for Anomaly Detection in High Performance Computing System Logs", **Proceedings of the 4th International Workshop on HPC User Support Tools**, 12 November 2017, Colorado, U.S.A., p. 3.

He, P., Zhu, J., Xu, P., Zheng, Z. and Lyu, M.R., 2018, **A Directed Acyclic Graph Approach to Online Log Parsing** [Online], Available: <https://arxiv.org/abs/1806.04356> [2019, April 4].

Hochreiter, S. and Schmidhuber, J., 1997, "Long Short-Term Memory", **Neural Computation**, 15 November 1997, Vol. 9, No. 8, pp. 1735-1780.

Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, KQ., 2017, "Densely Connected Convolutional Networks", **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017**, 21-26 July 2017, Hawaii, U.S.A., pp. 4700-4708.

Krizhevsky, A. Sutskever, I. and Hinton, G., 2012, "Imagenet Classification with Deep Convolutional Neural Networks", **Proceedings of the 25th International Conference on Neural Information Processing Systems**, 3-8 December 2012, Nevada, U.S.A., pp. 1097-1105.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998, "Gradient-Based Learning Applied to Document Recognition", **Proceedings of the IEEE**, November 1998, Vol. 86, No. 11, pp. 2278-2324.

Lu, S., Wei, X., Li, Y. and Wang, L., 2018, "Detecting Anomaly in Big Data System Logs Using Convolutional Neural Network", **Proceedings of the 2018 IEEE 16th International Conference on Dependable, Autonomic and Secure Computing, 16th IEEE International Conference on Pervasive Intelligence and Computing, 4th International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)**, 12 August 2018, Athens, Greece, pp. 151-158.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J. and Khudanpur, S., 2010, "Recurrent Neural Network Based Language Model", **Proceedings of the 11th Annual Conference of the International Speech Communication Association 2010**, 26-30 September 2010, Chiba, Japan.

Mikolov, T., Sutskever, I., Chen, K., Corrado, GS. and Dean, J., 2013, "Distributed Representations of Words and Phrases and their Compositionality", **Proceedings of the 26th International Conference on Neural Information Processing Systems**, 5-10 December 2013, Nevada, U.S.A., pp. 3111-3119.

Nagappan, M. and Vouk, MA., 2010, "Abstracting Log Lines to Log Event Types for Mining Software System Logs", **Proceedings of the 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)**, 2 May 2010, Cape Town, South Africa, pp. 114-117.

Shao. Y., Hardmeier. C. and Nivre. J., 2016, "Multilingual Named Entity Recognition Using Hybrid Neural Networks", **Proceedings of the 6th Swedish Language Technology Conference (SLTC 2016)**, 17-18 November 2016, Umeå, Sweden.

Staudemeyer, RC. and Morris, ER., 2019, **Understanding LSTM--A Tutorial into Long Short-Term Memory Recurrent Neural Networks** [Online], Available: <https://arxiv.org/abs/1909.09586> [2020, March 20].

Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N. and Robinson, S., 2017, "Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams", **Proceedings of the Workshops at the 31st AAAI Conference on Artificial Intelligence**, 21 March 2017, San Francisco, U.S.A.

Tuor, AR., Baerwolf, R., Knowles, N., Hutchinson, B., Nichols, N. and Jasper, R., 2018, Recurrent Neural Network Language Models for Open Vocabulary Event-Level Cyber Anomaly Detection", **Proceedings of the Workshops at the 32nd AAAI Conference on Artificial Intelligence**, 20 June 2018, New Orleans, U.S.A.

Wegrzynek, A., Chibante Barroso, V. and Vino, G., 2018, "Monitoring the New ALICE Online-Offline Computing System", **Proceedings of the 16th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2017)**, 8-13 October 2017, Barcelona, Spain, pp. 195-200.

Vaarandi, R., Blumbergs, B. and Kont, M., 2018, "An Unsupervised Framework for Detecting Anomalous Messages from Syslog Log Files", **Proceedings of the 16th IEEE/IFIP Network Operations and Management Symposium (NOMS 2018)**, 23 April 2018, Taipei, Taiwan, pp. 1-6.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, AN., Kaiser, Ł. and Polosukhin, I., 2017, "Attention is All You Need", **Proceedings of the 31st International Conference on Neural Information Processing Systems**, 4-9 December 2017, California, U.S.A., pp. 5998-6008.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R. and Bengio, Y., 2015, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", **Proceedings of the International Conference on Machine Learning 2015 (ICML 2015)**, 1 June 2015, Lille, France, pp. 2048-2057.

Xu, W., Huang, L., Fox, A., Patterson, D. and Jordan, MI., 2009a, "Detecting Large-Scale System Problems by Mining Console Logs", **Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles**, 11 October 2009, Montana, U.S.A., pp. 117-132.

Xu, W., Huang, L., Fox, A., Patterson, D. and Jordan, M., 2009b, "Online System Problem Detection by Mining Patterns of Console Logs", **Proceedings of the 2009 9th IEEE International Conference on Data Mining (ICDM 2009)**, 6 December 2009, Florida, U.S.A., pp. 588-597.

Yu, X., Joshi, P., Xu, J., Jin, G., Zhang, H. and Jiang, G., 2016, "Cloudseer: Workflow Monitoring of Cloud Infrastructures via Interleaved Logs", **ACM SIGPLAN Notices**, Vol. 51, No. 4, pp. 489-502.

CURRICULUM VITAE

NAME Mr. Purimpat Cheansunan

DATE OF BIRTH 5 November 1994

EDUCATIONAL RECORD

HIGH SCHOOL High School Graduation
Wat Raja-O-Ros School, 2012

BACHELOR’S DEGREE Bachelor of Engineering (Computer Engineering)
King Mongkut’s University of Technology Thonburi,
2016

MASTER’S DEGREE Master of Engineering (Computer Engineering)
King Mongkut’s University of Technology Thonburi,
2019

PUBLICATION Cheansunan, P. and Phunchongharn, P., 2019,
“Detecting Anomalous Events on Distributed Systems
Using Convolutional Neural Networks”, **Proceedings
of the 2019 IEEE 10th International Conference on
Awareness Science and Technology (iCAST 2019)**,
23 October 2019, Morioka, Japan, pp. 1-5.

**SCHOLARSHIP/
RESEARCH GRANT** Big Data Experience Center (BX) Scholarship