Deep Convolutional Neural Networks for Anomaly Event Classification on Distributed Systems

Jiechao Cheng*
International School of Software
Wuhan University
Wuhan, China
$jetrobert19@gmail.com$

Rui Ren*, Lei Wang and Jianfeng Zhan
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
$\{renrui, wanglei\_2011, zhanjianfeng\}$ @$ict.ac.cn$

arXiv:1710.09052v1 [cs.DC] 25 Oct 2017

***Abstract*—The increasing popularity of server usage has brought a plenty of anomaly log events, which have threatened a vast collection of machines. Recognizing and categorizing the anomalous events thereby is a much salient work for our systems, especially the ones generate the massive amount of data and harness it for technology value creation and business development. To assist in focusing on the classification and the prediction of anomaly events, and gaining critical insights from system event records, we propose a novel log preprocessing method which is very effective to filter abundant information and retain critical characteristics. Additionally, a competitive approach for automated classification of anomalous events detected from the distributed system logs with the state-of-the-art deep (Convolutional Neural Network) architectures is proposed in this paper. We measure a series of deep CNN algorithms with varied hyper-parameter combinations by using standard evaluation metrics, the results of our study reveals the advantages and potential capabilities of the proposed deep CNN models for anomaly event classification tasks on real-world systems. The optimal classification precision of our approach is 98.14%, which surpasses the popular traditional machine learning methods.**

***Keywords*-anomaly event classification; deep learning; convolutional neural network; log preprocessing; distributed system**

I. INTRODUNCTION

Event logs automatically produced by modern heterogeneous operating systems have unique properties, they play an important role in facilitating troubleshooting and anomaly failure prediction [9]. Due to the fact that such event logs usually comprise a considerable amount of redundant information and usually lacking of structure, they are unable to directly being applied in anomaly detection and data diagnosis. Therefore, it is of supreme importance to preprocess and clarify these raw events carefully before applying them to further decision making [16, 18]. Log preprocessing, a process used in the raw logs, not only filters useless data but also maintains crucial information for refined anomaly classification. To distill system logs for promoting anomaly classification on the distributed systems, a novel and effective log preprocessing technique is proposed here to handle 200,000 anomaly events coming from more than 1 million events of the distributed systems over a period of a year. The main parts of the proposed preprocessing method consists of three steps: First, we employ several regular expressions for coarse-grained categorization based on event contents, then we classify these events with fine-grained manually labeling. Second, we propose an efficient filtering approach to clarify the temporal and spatial redundant information of event messages via related stop-words and punctuations. Finally, to address the problem of being unsuitable for numerical calculation with the original unstructured event logs in our models, we build 14 different semantic dictionary libraries for numerical format conversion of filtered anomaly events. Indeed, our preprocessing approach has already proved its potentiality of working well with the state-of-the-art machine learning models on raw event logs and providing key insights for anomaly event prediction in the experiment.

The primary objective of our study is to classify anomaly events, which often indicate the presence of the error execution in systems. Classifying the anomaly event behavior of large scale heterogeneous systems is helpful for system administrators to constantly observe the health of the system and locate the root failure for anomaly diagnosis, task scheduling, and performance optimizations [39]. Machine learning such as SVM(support vector machines), as well as boosting has been widely applied in a diverse array of anomaly classification and recognition techniques, include but not limited to density-based methods [4,5], correlation-based anomaly detection [6], cluster analysis-based abnormal detection [8], ensemble techniques [10] etc. However, without necessary time-consuming custom manually engineered features set from a larger amount of data, traditional machine learning is not up to the complex classification tasks. Nevertheless, these specific features extracted by current machine learning methods are unable to be widely applied. Moreover, machine learning methods to some degree would be overwhelmed by the increasing huge

*Both authors contributed eqully to this work.

volume of log data. Surprisingly, deep learning approaches, due to their more promising effectiveness in discriminative feature learning process, are capable of handling complex scenarios of huge data with a high dimensionality and automatically implement dimension reduction to produce the optimal features set for anomaly detection and categorization tasks.

To achieve our research goal, we present an efficient approach based on deep CNN architectures in this paper to categorize anomaly events on large scale systems. The whole procedure of log preprocessing and anomaly event classification is shown in Figure 1. Our contributions Include but not limited to:

- We first propose a novel and effective log preprocessing approach which contains event labeling, event filtering and format conversion, the raw unstructured event logs then can be transformed to semantic-based numerical feature set which is proper for numerical computation in our deep CNN models via proposed preprocessing method.
- We implement a new classification method based on deep learning networks non-linear spatial temporal transformations for anomaly event categorization. In addition, to discover a simple yet efficient deep CNN approach enables to achieve outstanding performances in the anomaly classification tasks, we design and implement a series of CNN-based models combining various hyper-parameters settings.
- And last but not least, to demonstrate the effectiveness of our proposed deep CNN models, we have tested several popular machine learning classification methods, including AdaBoost, decision tree, MLP (multilayer perceptron), Nave Bayes, random forest and SVM (support vector machine), on our preprocessed anomaly event logs. The results show that our novel approach can classify anomaly event automatically and effectively, obtain a competitive precision 98.14%, which is outperforms the result of decision tree (96.92%), random forest (95.59%), MLP (94.99%), SVM with linear kernel (94.79%), Nave Bayes (87.48%), SVM with RBF kernel (86.15%), AdaBoost (64.59%) and SVM with poly kernel (56.92%).

The reminder of this article is organized as follows: section 2 presents a brief description of CNN architecture and reveals our deep CNN models utilized in the rest of the paper. In section 3, we provide the details of our experiments including data preprocessing and format conversion of event logs employed in classification training, parameters settings, evaluation metrics and experimental results and analysis. Section 4 briefly reviews the related works. Finally, we briefly summarizes our findings in the conclusion section.

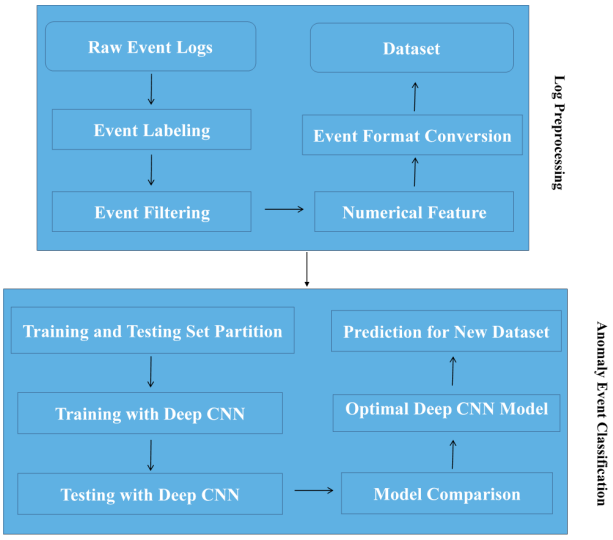

Figure 1. Architecture of event preprocessing and anomaly classification.

II. METHODOLOGY

In this section, we first briefly introduce baseline CNN architecture and its operating principle, then we introduce deep CNN networks. Finally, we describe our models implementation combined diverse convolutional layers and fully connected layers to address anomaly classification problems for large scale systems.

A. Convolutional Neural Network

From the perspective of means of processing data, the most suitable neural network architecture for classification tasks is CNN [28], if properly built, it allows you to model the most sophisticated pattern dependencies. As a class of feed-forward artificial neural networks, CNN architectures consist of several pairs of convolution and pooling layers and it seems like an appealing black-box solution for particular classification works, efficient but very challenging to understand the detailed working mechanism. During the feedforward phase, the lowest layer of the CNN models is responsible for the collection of raw data such as images and videos, each single neuron of the lower layer stores the information and pass the information further to the next layer. The convolutional layer captures the small parts of the input with a group of local filters and produces a 2-dimensional activation map of that filter, which automatically selects the most discriminating features. And the pooling layer preserves the invariant feature patterns. The activation function after the convolutional operation determines whether and to what extent a signal should be sent to connected nodes. A frequently used activation is just a basic step function that is 0 if its input is less than some threshold and 1 if its input is greater than the threshold. These steps can be utilized repeatedly as many times as desired, a new convolutional layer can be added on

a pooling layer, then followed by another pooling layer, over and over again. Finally, top fully connected layers combine all fed features to do the classification job. In a nutshell, the data information in the feedforward process of a CNN architecture is transferred from lowest layer to highest layer and more abstracted feature patterns will be collected after every iteration.

By contrast, during the backpropagation operation of the network, the networks decision about the input features is compared to the expected results, and the difference between the networks predictions and real ground-truth is utilized by an optimization algorithm to modify the activation thresholds repeatedly until converging to the expected output results. The optimization algorithm determines how the network learns, and more accurately how weights are modified after determining the error. The most common optimization algorithm used is SGD (Stochastic Gradient Descent). SGD gives us the direction of less error, and the learning rate determines how big of a step is taken in that direction. A cost function is a measure of error, which evaluates how well the neural network performed when making decisions about a given training sample, compared to the expected results. This hierarchical architecture organization about CNN network is well-known for modeling spatial matrix data and generating good results in image processing [29, 30] and speech recognition [31] tasks. Figure 2 depicts a typical CNN architecture.

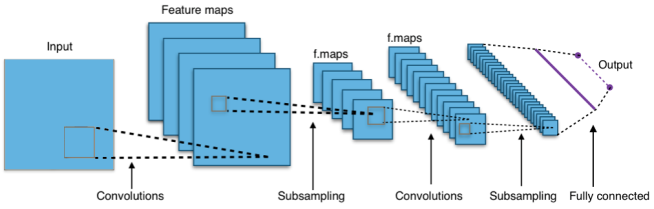

Figure 2. Typical CNN architecture.

B. Deep Convolutional Neural Network

Deep learning model is widely employed lately as it is very efficient in a large number of scenarios, especially for huge amount of high-dimensional datasets. The one key difference between deep CNN and baseline CNN is that deep CNN owns numerous layers in the networks while baseline CNN consists of only one or two hidden layers and this type of structure is not well suitable for the computation of large scale dataset. Therefore, we developed a set of deep CNN models comprises deep stack of neuron layers, such as convolutional and fully connected layers, automatically extracting the features required for the classification problems. Each hidden layer of deep CNN is responsible for training the unique set of features based on the output of the previous layer. The activation of each unit is computed via feature maps from the convolutional layer by using following function:

$$a^{l+1} = \sigma(W^l a^l + b^l) \tag{1}$$

where a^l denotes the activation for the units in layer l, W^l is the weight matrix in layer l, b^l represents the bias in the layer l, σ is the non-linear activation function.

Additionally, the activation function in the fully connected layers are also ReLUs, i.e. $f(x) = max(0, x)$. The output of the model after the softmax layer yields the probability distribution of label classes. The fundamental components of our deep CNN models contain several convolutional layers, pooling layers, fully connected layers and softmax classifier layers. As the number of hidden layers increases, the complexity and abstraction of feature patterns also increase. Ren et al. [26] have reported that adding both relatively more convolutional and connected layers to pre-trained networks enable to improve performance of classification. The overall architecture of one of proposed deep CNN models is depicted in Figure 3, layer 1 is the input layer, from layer 2 to 5 are convolutional layers, fully connected layers include layer 6 and 7, the last layer is softmax classifier layer.

C. Model Implementation and Training

When deep learning models are compared across huge dataset instead of hyper parameters, different architectures have little systematic advantages over another in the measurement of performance [11]. Since hyper parameters are crucial for model initialization, unsuitable hyper parameter settings have an adverse effect on model final results. Greff et al. [21] have reported that the learning rate and the size of hidden layers play an important role in the model performance. The deep CNN models proposed in our experiments for anomaly classification problems are implemented in TensorFlow framework. These models are trained in a completely supervised way with the gradient backpropagating, and the network parameters are optimized by cross-entropy minimization in loss function with mini-batch gradient descent. We use ReLU activation in the fully connected layer and softmax activation in the output layer, adam gradient decent is applied to the optimizer, the loss function is MSE (mean squared error). The number and size of the parameters in the networks varies according to which type of layers it belongs to, and different size or number of parameters have discriminative impact in ultimate anomaly event classification.

More often than not, more parameters generally facilitate robustness of the model, and yet massive parameters usage increases the dimensionality of the computation space, and might bring in undesirable effects like sparsity, and some redundant or irrelevant features are likely to diminish performance of models in classification, we here aim at discovering the optimal hyper-parameter settings for CNN

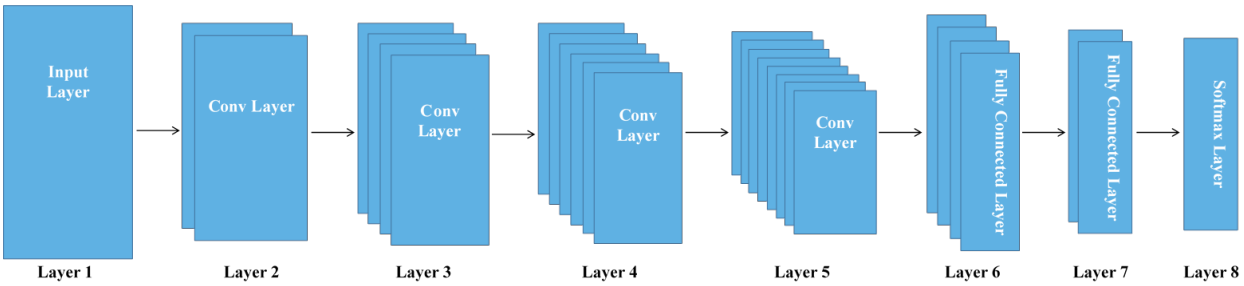

Figure 3. Architecture of the deep CNN framework.

architecture. To assess the sensitivity of varying hyper-parameters to the performance of our classification models with respect to anomaly event log, we employ tuning starting from 1) the learning rate; 2) the number of the convolutional filter; 3) the number of the convolutional and fully connected layer combinations (e.g., 2+2, 5+3, 7+3) to 4) the dropout probability value. For the learning rate we pick a value from the set 0.0001, 0.001, 0.01, 0.1, and the possible values for the dropout probability are 0.25, 0.5, 0.75, 1.0. We varied the number of the convolutional filter from 16, 32, 64 to 128. The overall detailed number and size combinations of diverse parameters are presented in Table 1.

To achieve the goal of training and testing efficiently, our dataset are split into mini-batches of a size of 1,000 during per iteration. Weights are initialized randomly, dropout operations are included in every model. The input of all our models consists of 140 vector features and output vector is comprised of 10 anomaly labels classes. As a result, the dimension of input and output is 140 and 10, respectively. Furthermore, we train and test our models in 10 different random train/test dataset combinations to decrease the influence of the bias in all evaluations.

III. EXPERIMENTS

We have conducted an extensive set of experiments to assess the effectiveness of our models with different architectures and hyper parameter settings by using standard evaluation metrics.

A. Dataset

There are totally 200,000 anomaly event records of our distributed systems gathered spanning a period of the whole year from early May 2016 to May 2017 are available here for our experimental studies. Using a larger set of data as the input for a deep learning model is not always the best choice, as this would increase the dimension of the parameters, and also introducing sparsity issues. As a result, it may negatively impact the performance of classification algorithms, hence in the experiments we decide to take the half of the original data as experimental dataset, 80% of which for training the classifier and the rest for testing.

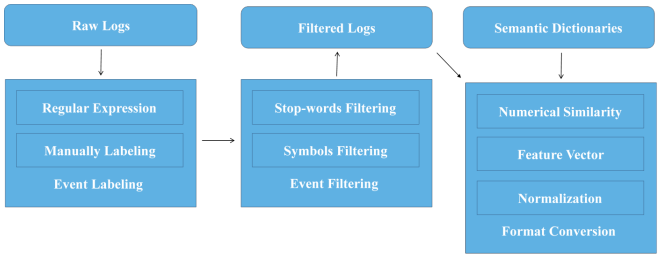

Figure 4. Flowchart of event logs preprocessing.

1) Event Labeling: To validate our results effectively and precisely, we have labeled and checked all dataset manually and assigning each of them a specified class. The examples of contents of log message, anomaly types and classification labels are shown in Table 2. To achieve more accurate classification performance, data preprocessing, feature generation namely data conversion and normalization need be taken into account before the formal training of models. The detailed procedures of log text preprocessing and numerical feature generation modules are depicted in Figure 4.

2) Event Filtering: Irrelevant or redundant features normally bring much noise to the feature extraction tasks, and resulting in inferior classification performances, in order to generate appropriate features for models to handle with, a task like practical anomaly classification often requires a data preprocessing, which removes the noise from the raw data and leads to statistically significant increase in classification accuracy [2]. The step of the preprocessing procedure consists of filtering out redundant information through stop-words and punctuations. Our stop-words reference Long Stop-word List[1] and punctuations comes from all commonly used symbols in the English language.

3) Event Format Conversion: Previous work [7] proposed to discover the regular pattern of system log messages by tracking the source code, but it is not an effective way for handling sophisticated distributed system logs. The original gathered dataset here is the unstructured log text, which is unable to be processed in the neural networks,

[1]http://www.ranks.nl/stopwords/

Table I
NUMBER AND SIZE COMBINATIONS OF PARAMETERS FOR DEEP CNN MODEL.

Number of Layer	Fully-Connect Size	Learning Rate	Dropout Value
(Conv + Full)	(Conv5+Full2)	(Conv2+Full2)	(Conv5+Full3)
{2+2, 2+3, 2+4, 3+2, 3+3, 3+4, 5+2, 5+3, 5+4, 7+2, 7+3, 7+4}	{16, 32, 64, 128}	{0.0001, 0.001, 0.01, 0.1}	{0.25, 0.5, 0.75,1.0}

*Conv: the convolutional layer; Full: the fully connected layer.

Table II
CATEGORIZATION AND LABELING OF ANOMALY EVENTS.

Content of Log	Category	Label
Could not find keytab file: /etc/libvirt/krb5	File	0
No DHCPOFFERS received	Network	1
Wake up new task 0xd3/0x120	Service	2
Error dropping database (cant rmdir testdb)	Database	3
WARNING: Unable to determine session	Communication	4
Read-error on swap-device (253:1.16968112)	Memory	5
No CSI structure available	Driver	6
Respawning too fast: disabled for 5 minutes	System	7
Security real capable no audit	Security	10
FAILED SMART self-check. Back up now	Disk	11
Selected processor does not support 'strexb'	Processor	12
Buffer I/O error on device dev sda	I/O	8

consequently, construction of numerical event-wise feature sets representing the semantics of each log message field on real distributed systems for facilitating the calculation in the deep learning models has been conceived with the objective of fundamentally maintaining the underlying structural properties of the temporal log events as much as possible. Such formatted numerical feature layout allows us to learn and analyze the temporal events efficiently for anomaly categorization tasks, additionally, it protects the sensitive information with respect to system operations at the same time.

In reality, the structured numeric feature generation from an unstructured log message text requires a solid background knowledge of the original information, given that projected features are so different from the original patterns from a data view [12]. Therefore, we have built up a number of dictionary libraries involving total 14 topic types for numerical data transformation of log message events and enable them to extract important pattern features to implement numeric computation in our model architectures. Every topic type of dictionary libraries contains tens of keywords to represent the characteristics of different anomalous log events, which then would be labeled properly with specific numbers. For every message sequence of anomaly log events, the semantic similarity value between them and all keywords in the dictionary libraries is regarded as a generated feature vector. The numeric features gained in this way not only retain certain textual structures of the original log events, but also maintain the differentiation among a variety of anomaly types. The semantic similarity of anomaly events and dictionary keywords is defined by:

$$S_{x,y} = \frac{1}{lev_{x,y}} \times 10 \tag{2}$$

where lev refers to Levenshtein distance [9], which is also referred to as edit distance [23], a string metric for measuring the difference between sequence x and sequence y (Here, x refers to log message sequence, y refers to keyword sequence of dictionary libraries). Note that the first element in the Equation (2) corresponds to the primitive semantic similarity equation, for ease of calculation, it is multiplied by 10.

Considering the time-consuming nature of the feature construction and feature projection in deep CNN training tasks, we merely select top 10 semantic similarity values between each single log event and all topic types in dictionary libraries, then combine them into a feature vector as the representative of this anomaly log event. As a consequence, each event record contains 140 numerical event-wise attributes. Furthermore, normalization, to the best of our knowledge, is the final step before the anomaly classification algorithm being executed. The min-max normalization which usually normalizes every feature into a [0, 1] interval is commonly used in practical applications, it is also employed in our experiments. Ultimately, normalized numerical features and matching labels are utilized as dataset to evaluate our proposed models.

B. Settings

Training the neural network usually takes much time and requires advanced hardware settings. The training and classification tasks of the proposed models are run on GPUs, which leads to a significant decrease in training time. Our detained experiment run environment configuration is listed as below:

CPU: Intel Xeon E5-2630, 2.4 GHz; GPU: Nvidia Tesla M40; RAM: 64GB; OS: Ubuntu 16.04.

C. Evaluation Metrics

To make a fair comparison between our anomaly classification models to evaluate which model performs best, we specifically utilize the confusion matrix with the precision, recall, F1-score and the best accuracy metrics for our classification models. The accuracy, recall, precision and F1-score are defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5}$$

$$BestAccuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

where TP (True Positives) is the number of true positives, TN (True Negatives) is the number of true negatives, FP (False Positives) is the number of false positives, and FN (False Negatives) is the number of false negatives.

D. Results and Discussion

In this section, we review and compare the sensitivity of varied hyper-parameters to the performance of a set of proposed models in terms of best accuracy rate, recall, precision and F1-score.

1) Learning rate: Learning rate is one of the most important hyper-parameters, therefore it is very important to understand how to set it correctly for achieving good performance. If the learning rate is too high, you may overshoot the error minimum, if it is too low, your training will take forever. In our experiments, we evaluate the sensitivity of the learning rate with the weight values setting to 0.0001, 0.001, 0.01, 0.1, the number of the convolutional layer and fully connected layer are both 2 while the dropout is 0.75. Table 3 depicts the trends of precision, recall and f1-score when the learning rate is increased. The average precision, recall, F1-socre and the best accuracy of proposed deep CNN models all steadily increase when the learning rate weight decreasing from 0.1 to 0.0001. We can get best precision when learning rate equals to 0.0001, the recall also obtains best value when we set learning rate to 0.0001. The best accuracies of all models are 100% except the model with the learning rate at 0.1.

2) Convolutional Filter Number: Not surprisingly the number of the convolutional filter is an important hyper-parameter affecting the deep CNN network performance. We test our deep CNN models with varied numbers of the convolutional filter from 16, 32, 64 to 128, given that all proposed models are composed of five convolution layers and two top fully connected hidden layers having the learning rate identical to 0.0001 with one additional softmax layer employed to generate posterior probabilities. Still, we set the initial dropout value to 0.75. The classification results in Table 4 shows that increasing the number of the convolutional filter tends to promote the performance of the models on the test dataset. However, when the number goes up to 128, the performance goes bad, even worse than the outcome of the model with 16 convolutional filters. No doubt, the best results are achieved when setting the number of the convolutional filter to 64, which will be implemented in all our next models.

3) Convolutional and Fully Connected Layer: Deeper networks generally are well suitable for the computation of huge dataset, and that is one of the main reasons why we build deep CNN models, which consists of deep stack of convolutional neural layers and fully connected layers. Of course, the obvious drawback of deeper networks architecture required is that they requires more training time. We evaluate the impact of different number of the convolutional layer and fully connected layer to the ultimate performance of models with 64 convolutional filters while setting the learning rate to 0.0001 and dropout identical to 0.75. Table 5 provides more insights about the specific classification performance of a series of proposed models. Of which, the combination with five convolutional layers and three fully connected layers offers an precision comparable even slightly surpassing the one equipped with five convolutional layers and two fully connected layers. When the fully connected layers keep constant, increasing the number of the convolutional layer from 2 to 5 improves the performance of proposed models, and networks that having too many layers (here denotes 7 layers) obtain poor performance. Moreover, the best accuracy of all models is 100%. More intuitive results are shown in Figure 5.

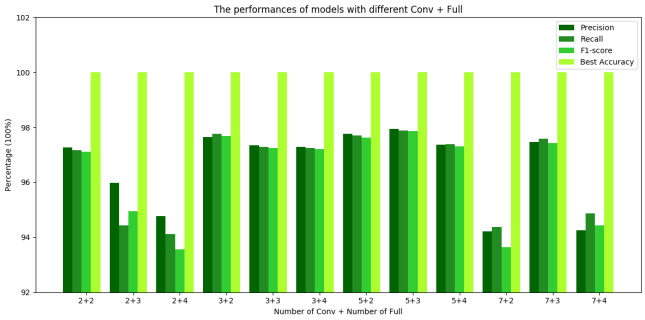

Figure 5. Evaluation Results of different models with varied numbers of the convolutional layer and the fully connected layer. Conv: the convolutional layer; Full: the fully connected layer.

4) Dropout: Dropout represents the probability (P) of retaining a previous hidden unit in the network [17]. We assess the effects of varying the size of dropout to our models, which have 5 convolutional layers and 3 fully connected layers with learning rate at 0.0001, while the other hyper-parameters are held constant. Table 6 demonstrates the precision, recall, F1-score and the best accuracy obtained by models with different dropout values. It can be observed that the performance of models drops smoothly when 0.5 dropout 1.0. This can be expected because having big dropout value means too many units will turn on during training phase. The model with dropout value identical to 0.5 gains almost perfect performance, which is better than any other models mentioned above. This indicates that this parameter setting is the optimal hyper-parameter combination for proposed models to handle with the anomaly event

Table III
THE PRECISION, RECALL, F1-SCORE AND THE BEST ACCURACY OF MODELS (CONV2+FULL2) WITH DIFFERENT LEARNING RATE ON TESTING DATA.

Learning Rate	Precision (%)	Recall (%)	F1-score (%)	Best Accuracy (%)
0.0001	**97.23**	**97.07**	**96.99**	100
0.001	96.95	96.87	96.77	100
0.01	83.52	88.34	84.13	100
0.1	52.53	72.48	60.91	83.0

*Conv2+Full2: two-layer convolutional operation and two-layer fully connected operation.

Table IV
THE PRECISION, RECALL, F1-SCORE AND THE BEST ACCURACY OF MODELS (CONV5+FULL2) WITH DIFFERENT NUMBERS OF THE CONVOLUTIONAL FILTER ON TESTING DATA.

Number of Conv Filter	Precision (%)	Recall (%)	F1-score (%)	Best Accuracy (%)
16	97.77	97.71	97.63	100
32	97.83	97.90	97.83	100
64	**98.02**	**98.00**	**97.99**	100
128	97.47	97.56	97.50	100

* Conv5+Full2: five-layer convolutional operation and three-layer fully connected operation.

Table V
THE PRECISION, RECALL, F1-SCORE AND THE BEST ACCURACY OF MODELS WITH DIFFERENT LAYER COMBINATIONS (CONV+FULL) ON TESTING DATA.

Number of Conv+Full	Precision (%)	Recall (%)	F1-score (%)	Best Accuracy (%)
2+2	97.26	97.17	97.10	100
2+3	95.98	94.44	94.94	100
2+4	94.77	94.11	93.56	100
3+2	97.64	97.77	97.68	100
3+3	97.34	97.28	97.24	100
3+4	97.29	97.24	97.20	100
5+2	97.77	97.71	97.63	100
5+3	**97.95**	**97.89**	**97.86**	100
5+4	97.36	97.38	97.31	100
7+2	94.22	94.37	93.64	100
7+3	97.46	97.58	97.43	100
7+4	94.26	94.87	94.44	100

* Conv+Full: the convolutional layer and the fully connected layer.

classification.

The iteration progress of the best model (Conv5+Full3, learning rate = 0.0001, dropout = 0.5) is depicted in Figure 6, and normalized confusion matrix heat map is presented in Figure 7. The confusion matrix heat map indicates that quite a few existing prediction errors are due to the confusion between security events and others, disk events and others, communication events and driver events, I/O events and service events. This is because the fact that the data proportion of these anomaly events are relatively small, thus recognizing them is pretty difficult, and it is unfair to the overall evaluation of the model. Observing the detailed proportion results of the entire testing datas predicted label as compared to its ground-truth label for each specific anomaly event classification in Table 7, we can conclude that the performance of the best proposed model on testing data is excellent, particularly in the case of processor, database, memory and system events, all of which gain 100% precision, recall and F1-score.

Given all that, the learning rate is obviously the most

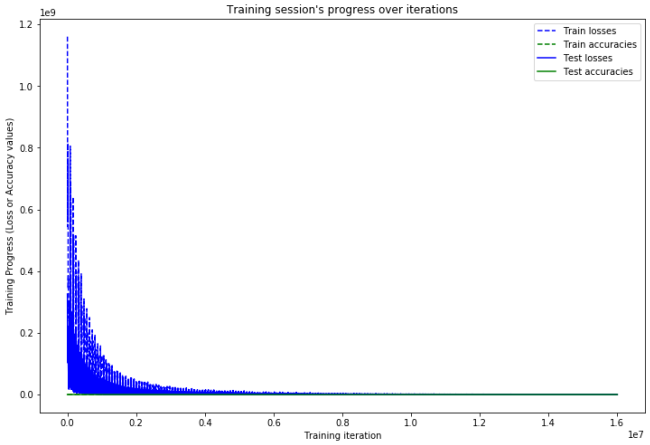

Figure 6. Iterations progress of the best deep CNN model (Conv5+Full3) on training and testing data.

important hyper-parameter by far. The next most important hyper-parameter is the dropout probability, followed by the

Table VI
THE PRECISION, RECALL, F1-SCORE AND THE BEST ACCURACY OF MODELS WITH DIFFERENT LAYER COMBINATIONS (CONV+FULL) ON TESTING DATA.

Dropout Value	Precision (%)	Recall (%)	F1-score (%)	Best Accuracy (%)
0.25	93.27	93.18	92.13	99.80
0.5	**98.14**	**98.14**	**98.11**	100
0.75	97.95	97.89	97.86	100
1.0	93.27	92.73	91.62	100

* Conv5+Full3: five-layer convolutional operation and three-layer fully connected operation.

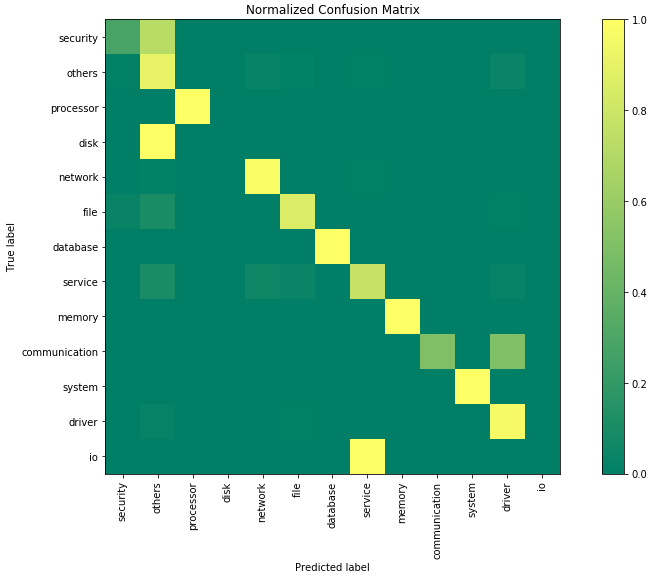

Figure 7. Normalized confusion matrix heat map of the best deep CNN model (Conv5+Full3) on testing data.

number of the convolutional layer and fully connected layer, the number of the convolutional filter.

Table VII
THE PRECISION, RECALL AND F1-SCORE OF THE BEST MODEL (CONV5+FULL3) FOR ANOMALY EVENT CLASSIFICATION.

Anomaly Event	Precision	Recall	F1-score	Support
Security	0.23	0.28	0.25	18
Others	0.84	0.91	0.87	911
Processor	**1.0**	**1.0**	**1.0**	213
Disk	0.0	0.0	0.0	2
Network	0.98	0.98	0.98	2891
File	0.89	0.86	0.87	348
Database	**1.0**	**1.0**	**1.0**	14,495
Service	0.91	0.77	0.84	535
Memory	**1.0**	**1.0**	**1.0**	132
Communication	**1.0**	0.50	0.67	4
System	**1.0**	**1.0**	**1.0**	8
Driver	0.88	0.96	0.92	433
I/O	0.0	0.0	0.0	10
Avg/Total	0.98	0.98	0.98	20,000

* Conv5+Full3: 5 convolutional layers and 3 fully connected layers.

5) Comparison with Other Methods: To show the potentiality of our deep CNN models for complex classification issues, we test our preprocessed event logs on several popular machine learning models for anomaly classification implementation. Table 8 depicts the details of precision, recall and F1-socre results of different classification fashions. There is no doubt that our proposed deep CNN (5 conv + 3 fully) gains the highest precision (98.14%), followed by decision tree (96.92%) and random forest (95.59%), the next is MLP (94.99%), then followed by SVM with linear kernel (94.79%), Nave Bayes (87.48%), SVM with RBF kernel (86.15%) and AdaBoost (64.59%). The worst model is SVM with poly kernel, which only gets 56.92% precision.

Table VIII
RESULTS COMPARISON OF DIFFERENT MODELS FOR ANOMALY EVENT CLASSIFICATION.

Method	Precision%	Recall%	F1-score%
AdaBoost	64.59	73.54	68.25
Decision Tree	96.92	96.90	96.62
MLP	94.99	94.87	94.43
Nave Bayes	87.48	89.07	87.60
Random Forest	95.59	94.37	94.30
SVM-Linear	94.79	94.25	93.18
SVM-RBF	86.15	89.47	87.45
SVM-Poly	56.92	73.70	63.47
Deep CNN	**98.14**	**98.14**	**98.11**

IV. RELATED WORK

A. Anomaly Classification

The ever-growing number of system services during the last decade have stimulated a flood of interest in log anomaly detection and classification. Anomalous events are also referred to as novelties, noise, exceptions etc. [1]. Anomaly discrimination as a common approach of log analysis [24], enables us to discover the suspicious operations on a server or detect an unauthorized access in a system by exploring the anomalous operation records. To cope with a complex scenario like real-time system management and reduce the influence of anomaly events affecting the end clients, it is of vital importance to effectively detect the occurrence of anomalies and classify them specifically for further refinement of system service. Detecting and categorizing anomalies often require varied algorithms, especially addressing sequences and time series data [13]. Dozens of machine learning approaches have been utilized for the sake of anomaly event classification, some of them are widely

considered in the current anomaly classification researches. Kimora et al. [38] apply a supervised machine learning approach to categorize network failures via the log data. P. Fiadino et al. [19] has introduced and implemented statistical detection approaches and diagnosis of anomaly events. Lazarevic et al. [14] compare LOF (Local Outlier Factor), k-NN (k-Nearest Neighbors), PCA (Principal Component Analysis) and unsupervised SVM (Support Vector Machine) algorithms for anomaly intrusion analysis. Ding et al. [15] have researched SVDD (Support Vector Data Description), k-NN classifier, k-means and GMM (Gaussian Mixture Model) for anomaly detection. Methods like subspace clustering [20, 27] have also been utilized in anomaly detection and classification. M. Gupta et al. [22] apply anomaly event identification on the temporal data. Ma et al. [37] utilize the joint failure probability to predict disk failures. Surprisingly, Gurumdimma et al. [3] combining resource usage and anomaly logs to detect errors in the distributed systems. Rare pattern mining is another import method for log analysis [25]. Unlike these studies, our study focuses on providing an effective deep learning method to improve the anomaly classification performance.

B. Deep Learning

Deep learning algorithms normally build data-driven models from millions of labeled data then make predictions based on the data which they are capable of learning from. The more data a deep learning model is trained on, the higher accuracy it will gain. During the execution of a bulk data transition in deep learning architecture, meta data is treated as an input and processed through a number of layers of the non-linear transformation, then the outputs of the previous layer are processed by each current layer, finally following by the classification result. Owing to the capability of automatically grasping the relevant features required for the solution of the tasks, deep learning models are capable of reducing the burden on the engineer to select the features manually. It is widely accepted that deep learning algorithms are well-suited for supervised classification problems than other previous conventional techniques whilst processing the dataset at large scale. Motivated by the recent success of deep learning largely in diverse fields of computer vision [32], audio recognition [33], biology [34] and medicine [35], particularly, high accuracy performance exhibited by CNN in large scale image classification tasks, we decide to embark on the adventures of the labeled log data fed into the state-of-the-art deep CNN models for anomaly classification.

V. Conclusion

In this paper, we have proposed a novel deep convolutional neural network approach for anomaly event classification on distributed systems, offering a very powerful and straightforward technique to categorize anomalous instances. The dataset utilized in the experiment is generated by converting semantic information from large-scale system log messages to numerical features with our effective preprocessing methodology. To find the proper hyper-parameter settings for proposed models, we have taken a series of studies and built a successful deep model (5 convolutional layers, 3 fully connected layers, 64 convolutional filters, learning rate = 0.0001, dropout = 0.5) with a perfect accuracy of 98.14%, which shows the potentiality of deep CNN for anomaly classification tasks on distributed systems, it is much better than other widely used machine learning methods. Another fascinating phenomenon we discover in the experiment is that, there is a noticeable misclassification overlap between the I/O event and the service event attributed to the relatively small number of the event. Future works will investigate the necessary steps to provide more available data to improve the discrimination of these anomaly events and promote the robustness of the proposed model.

In a nutshell, the study has shown the remarkable classification results of proposed deep CNN approaches in an efficient fashion. We believe this appealing deep learning approach is capable of providing high insights for understanding system anomaly events without disclosing any business sensitive information. This area is still on-going research, and it also faces many challenges. The obvious but useful extension to this work would be to extend the experiments out of the box to more reliable algorithms running in a production environment to see which works better. Should we succeed, we may achieve outstanding performance results for future anomaly event classification in terms of various aspects.

Acknowledgment

This work was supported by National High Technology Research and Development Program of China (Grant No. 2015AA015308). The authors want to thank the Institute of Computing Technology (ICT), Chinese Academy of Sciences for the provided computational resources and support, as well as the anonymous reviewers for their insightful comments.

References

[1] V. J. Hodge and J Austin, "A Survey of Outlier Detection Methodologies," Artificial Intelligence Review, vol. 22, no. 2, pp. 85–126, 2004.

[2] M. R. Smith and T. Martinez, "Improving classification accuracy by identifying and removing instances that should be misclassified," The 2011 International Joint Conference on Neural Networks, pp. 2690, 2011.

[3] N. Gurumdimma, A. Jhumka, M. Liakata, E. Chuah and J. Browne. "CRUDE: combining resource usage data and error logs for accurate error detection in large-scale distributed systems," IEEE, 2016.

[4] M. Breunig, H. Kriegel, R. T. Ng and J. Sander, "LOF: Identifying Density-based Local Outliers," Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 93–104, 2000.

[5] E. Schubert, A. Zimek and H. Kriegel, "Local outlier detection reconsidered: A generalized view on locality with applications to spatial, video, and network outlier detection," Data Mining and Knowledge Discovery, Springer, vol. 28, no. 1, pp. 190–237, 2014.

[6] H. Kriegel, P. Kroger, E. Schubert and A. Zimek, "Outlier Detection in Arbitrarily Oriented Subspaces," 2012 IEEE 12th International Conference on Data Mining, pp. 379, 2012.

[7] R. Vaarandi, "A data clustering algorithm for mining patterns from event logs," in In Proceedings of the 2003 IEEE Workshop on IP Operations and Management (IPOM), pp. 118–126, 2013.

[8] R. J. G. B. Campello, D. Moulavi, A. Zimek and J. Sander, "Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection," ACM Transactions on Knowledge Discovery from Data 2015; vol. 10, no. 5, pp. 1–51, 2015.

[9] Levenshtein VI, "Binary codes capable of correcting deletions, insertions and reversals," Journal of Soviet Physics Doklady, vol. 10, pp. 707–711, 1966.

[10] A. Zimek, R. J. G. B. Campello and J. Sander, "Data perturbation for outlier detection ensembles," Proceedings of the 26th International Conference on Scientific and Statistical Database Management, pp. 1, 2014.

[11] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenkov, E. Schubert, I. Assent and M. E. Houle, "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," Data Mining and Knowledge Discovery, vol. 30, no. 4, pp. 891, 2016.

[12] M. G. Goldstein, "Anomaly Detection," Data Mining and Knowledge Discovery Series, Chapman and Hall/CRC, pp. 367–394, 2013.

[13] A. CC, "Outlier Analysis," Springer-Verlag, NewYork, 2013.

[14] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur and J. Srivastava. "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection," In Proceedings of the Third SIAM International Conference on Data Mining, vol. 3, pp. 25–36, 2003.

[15] X. Ding, Y. Li, A. Belatreche and L. P Maguire, "An Experimental Evaluation of Novelty Detection Methods," Neurocomputing, vol. 135, pp. 313–327, 2014.

[16] M. F. Buckley and D. P. Siewiorek, "Comparative analysis of event tupling schemes," In FTCS-26, Computing Digest of Papers, pp. 294-303, 1996.

[17] N. Srivastava, G. E Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from overfitting," Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] J. Hansen, "Trend Analysis and Modeling of Uni/MultiProcessor Event Logs," PhD thesis, Dept. of Computer Science, Carnegie-Mellon University, 1988.

[19] P. Fiadino, A. Dalconzo, M. Schiavone and P. Casas. "RCA Tool - A Framework for Detecting and Diagnosing Anomalies in Cellular Networks," ITC, pp. 194–202, 2015.

[20] P. Casas, J. Mazel and P. Owezarski. "MINETRAC: Mining Flows for Unsupervised Analysis & Semi-Supervised Classification," ITC, pp. 87–94, 2011.

[21] K. Greff, R. K. Srivastava, J. Koutnik, B. R Steunebrink and J. Schmidhuber, "LSTM: A Search Space Odyssey," IEEE Transactions on Neural Networks, pp. 1–11, 2016.

[22] M. Gupta, J. Gao, C. C Aggarwal and J. Han, "Outlier detection for temporal data," Synthesis Lectures on Data Mining and Knowledge Discovery, vol. 5, no. 1, pp. 1–129, 2014.

[23] G. Navarro, "A guided tour to approximate string matching," ACM Computing Surveys, vol. 33, no. 1, pp. 31–88, 2001.

[24] T. Kimura, K. Ishibashi, T. Mori, H. Sawada, T. Toyono, K. Nishimatsu, A. Watanabe, A. Shimoda, and K. Shiomoto, "Spatio-temporal factorization of log data for understanding network events," 2014 IEEE Conference on Computer Communications, pp. 610–618, 2014.

[25] Y. S. Koh and S. D. Ravana, "Unsupervised Rare Pattern Mining: A Survey," ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 10, no. 4, pp. 45, 2016.

[26] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," IEEE International Conference on Robotics and Automation, pp. 1316–1322, 2015.

[27] M. Henrion, D. J Hand, A. Gandy and D Mortlock, "CASOS: A subspace method for anomaly detection in high dimensional astronomical databases," Statistical Analysis and Data Mining, vol. 6, no. 1, pp. 53–72, 2013.

[28] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[29] A. Krizhevsky, I. Sutskever and G. E Hinton, "Imagenet classification with deep convolutional neural networks," NIPS, pp. 1097–1105, 2012.

[30] S. Ji, W. Xu, M. Yang and K. Yu, "3d convolutional neural networks for human action recognition," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 35, no. 1, pp. 221–231, 2013.

[31] O. Abdelhamid, A. Mohamed, H. Jiang and G. Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition," In Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, pp. 4277–4280, 2012.

[32] H. Lee, R. B Grosse, R. Ranganath and A. Y Ng, "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations," In Proceedings of the 26th Annual International Conference on Machine Learning (ICML), pp. 609–616, 2009.

[33] H. Lee, P. T Pham, Y. Largman and A. Y Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," Neural Information Processing Systems (NIPS), pp. 1096–1104, 2009.

[34] F. Ghasemi, A. Fassihi, H. Perez Sanchez and A. M. Dehnavi, "The role of different sampling methods in improving biological activity prediction using deep belief network," Journal of Computational Chemistry, vol. 38, no. 4, pp. 195–203, 2017.

[35] E. Choi, A. Schuetz, W. F Stewart and J. Sun, "Using recurrent neural network models for early detection of heart failure onset," Journal of the American Medical Informatics Association, vol. 24, pp. 361–370, 2016.

[36] M. Abramowitz and I. A Stegun. "Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables," 9th printing, New York: Dover, p. 928, 1972.

[37] A. Ma, R. Traylor, F. Douglis, M. Chamness, G. Lu, D. Sawyer, S. Chandra and W. Hsu, "Raidshield: characterizing, monitoring, and proactively protecting against disk failures," in Proceedings of the 13th USENIX Conference on File and Storage Technologies, USENIX Association, pp. 241–256, 2015.

[38] T. Kimura, A. Watanabe, T. Toyono and K. Ishibashi, "Proactive failure detection learning generation patterns of largescale network logs," Conference on Network and Service Management, pp. 8–14, 2015.

[39] Y. Liang, Y. Zhang, A. Sivasubramaniam, R. K. Sahoo, J. Moreira and M. Gupta, "Filtering Failure Logs for a BlueGene/L Prototype," Proceedings of the 2005 International Conference on Dependable Systems and Networks, pp. 1, 2005.