# Deep Convolutional Neural Networks for Anomaly Event Classification on Distributed Systems

Jiechao Cheng, Rui Rend

[jetrobert19@gmail.com](mailto:jetrobert19@gmail.com), [renrui@ict.ac.cn](mailto:renrui@ict.ac.cn)

# Abstract

The increasing popularity of servers usage has brought a plenty anomaly operation 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 detection and classification of anomaly events, and gaining critical insights from system log records, we propose a series of approaches with the state-of-the-art deep learning techniques in this paper. We employ typical CNN (Convolutional Neural Network) architectures and deepen them to build novel models for automated classification of anomalous system events detected from the distributed system logs, then measure our diverse array of classification algorithms with standard evaluation metrics. The results of our study reveals the advantages and potential capabilities of our proposed deep CNN models for classification tasks of anomaly events on real-world systems. Furthermore, our approach reaches at least 94% classification accuracy and highest accuracy is up to 98%.

**Index Terms**—Anomaly Detection, Event Classification, Log Analysis, Deep Learning, Convolutional neural network, Distributed System

# 1 Introduction

Logs automatically produced by modern heterogeneous operating systems record the application-specific operational states and temporal events at every period of time. Whilst lacking of structure, mining these logs usually facilitates root-cause analysis and troubleshooting for systems, and harnessing the massive amount of valuable log information is much beneficial for business strategies, even detecting and grouping abnormal operations on large-scale systems is favorable for further decision making. Abnormal behavior or pattern of log data often indicates the presence of the error in execution of the system. The ever-growing number of system services during the last decade have stimulated a flood of interest in log anomaly detection and classification. Anomaly detection related problems are addressed in a great deal of practical applications, including intrusion detection, fraud detection, aw well as system health monitoring. 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.

Machine learning has been widely applied in a diverse array of anomaly detection and recognition techniques, include but not limited to density-based methods [3,4,5,23], correlation-based anomaly detection [6], cluster analysis-based abnormal detection [7,8], ensemble techniques [9,10] etc. However, traditional data mining methods at least to some degree has been overwhelmed by the huge volume of log data. Motivated by the recent success of deep learning largely in field of image classification, natural language processing, speech recognition, etc., specifically, high accuracy performance exhibited by CNN in complex detection and classification tasks, consequently, we decide to explore a solution with deep learning algorithm to handle such complex scenarios for gaining relevant optimum results. Deep learning has been chosen due to its more promising effectiveness in discriminative feature learning process for anomaly detection and categorization tasks. Alternatively, labeled data set is taken into consideration here for anomaly classification and fed into the state-of-the-art deep CNN models.

Our contributions Include but not limited to: we propose a simple yet efficient deep CNN approach enables to obtain favorable performances in anomaly classification. What’s more, we extend the anomaly classification research to deep learning, which applying complex architecture with non-linear spatial temporal transformations. Our deep CNN model is scalable to classify abnormal situations or categorize unpredictable events on heterogeneous systems at large scale. Last but not least, by applying deep learning algorithms to our studies, we’ve gained key insights and achieved potentially high classification accuracy.

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

# 2 Related Work

**Anomaly Event Detection and Classification.** There has been considerable amount of researches about anomaly event detection and classification of heterogeneous systems latterly. 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 detection and classification, some of them are widely considered in the current anomaly classification researches. P. Fiadino et al. [19] has introduced and implemented statistical detection approaches and diagnosis of anomaly events. Lazarevic et al. [14] compared LOF (Local Outlier Factor), k-NN (k-Nearest Neighbors), PCA (Principal Component Analysis) and unsupervised SVM (Support Vector Machine) algorithms for anomaly intrusion detection. Ding et al. [15] have researched SVDD (Support Vector Data Description), k-NN classifier, k-means and GMM (Gaussian Mixture Model) for anomaly detection. Amer et al. [16] proposed One-class Support Vector Machines to conduct anomaly detection, they also applied LDCOF (Local Density Cluster-based Outlier Factor) [18] algorithm to carry out anomaly detection by estimating the densities of total clusters based on a spherical distribution hypothesis across all the cluster members. Surprisingly, M. Gupta et al. [22] applied anomaly event identification on the temporal data. Moreover, methods like sub-space clustering [20,27] have also been utilized in anomaly detection and classification. Outlier ensembles [17] enable to boost their joint detection performance by combining with multiple anomaly recognition algorithms.

**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. Up to now, deep learning has been proved to achieve substantially good results in diverse fields, such as computer vision [32] and audio recognition [33]. In this article, we apply deep CNN approaches in the case of anomaly event classification of large-scale systems.

# 3 Methodology

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

## 3.1 Convolutional Neural Network

From the perspective of means of processing data, the most suitable neural network architecture for classification tasks is CNN, if properly built, it allows you to model the most sophisticated pattern dependencies. As a typical DNN (Deep Neural Network), CNN [28] consists 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 automatically selects the most discriminating features, while the pooling layer preserves the invariant feature patterns. 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. This hierarchical 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.

By contrast, during the backpropagation operation of the network, the network’s decision about the input features is compared to the expected results, and the difference between the network’s predictions and real ground-truth is utilized to modify the activation thresholds repeatedly until converging to the expected output results.

Assuming that the input vector, the hidden vector and the output vector denoted by X, H and Y respectively. Given that X = (x1,x2,..., xn). ..

Where w is weight matrix, b is a bias vector, …activation function, optimization algorithm, and cost function..

The activation function 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. 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 stochastic gradient descent. 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.

## 3.2 Deep Convolutional Neural Network

The one key difference between deep CNN and baseline CNN is that deep CNN owns numerous layers in the networks while basic CNN contains three-layer networks at most. Baseline CNN consists of 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 with deep stack of neuron 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. 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 proposed deep CNN models is depicted in figure 1, the value transformation steps are described in the following equations.

In order to discover the optimal architecture, we established various CNN models with varied convolutional layers, different sizes of hidden layer, diverse learning rate and dropout probability.

## 3.3 Model Implementation

The batch size and epoch are 1000, 400 respectively in our CNN architecture. We use ReLU activation in the fully connected layer and Softmax activation in the output layer, Adam gradient decent for the optimizer. The loss function is MSE (mean squared error). SGD (Stochastic Gradient Descent) gives us the direction of less error, and the learning rate determines how big of a step is taken in that direction. If the learning rate is too high, you may overshoot the error minimum; if it is too low, your training will take forever. This is a hyper-parameter you may need to adjust.

When deep learning models are compared across huge dataset and 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.

To find the optimal hyper parameter values for our models, we do implement a serials of deep CNN models with different combination of parameter settings. For the learning rate we pick a value from the set {0.0001, 0.001, 0.01, 0.1}. The possible values for the hidden layer size are {16, 32, 64, 128}.

The input vector consists of 140 features and output vector is comprised of 10 anomaly classifications. As a result, the dimension of input and output is 140 and 10, respectively.

# 4 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.

## 4.1 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 we decide to take the half of the original dataset to carry out the experiments. 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 1. 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 Fig. 1.

Table 1. 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 (can’t 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 DATA NOW | Disk | 11 |
| Error: selected processor does not support ‘strexb’ | Processor | 12 |
| Buffer I/O error on device dev sda | IO | 8 |

**Preprocessing.** 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 Stopword List1 and punctuations comes from all commonly used symbols in the English language.

**Feature Generation.** Previous work [25] proposed to discover the regular pattern of system log messages by tracking the source code, but it is not an effective way for handling with sophisticated distributed system logs. The original gathered dataset here is the unstructured log text, which is unable to be processed in the neural networks, 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.

Feature Generation

Semantic Similarity

Feature Vector

Normalization

Training Dataset

Preprocessing

Filtering Stop-Words

Filtering Symbols

Raw Log Events

Fig. 1 Flowchart of Data Preprocessing and Feature Generation

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:

1. <http://www.ranks.nl/stopwords/>

(1)

where refers to Levenshtein distance [34], which is also referred to as edit distance [35], a string metric for measuring the difference between sequence A and sequence B (Here, A refers to log message sequence, B refers to keyword sequence of dictionary libraries). Note that the first element in the Equation 1 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.

## 4.2 Settings

Training the neural net is the step that will take the most time and hardware. Running training on GPUs will lead to a significant decrease in training time. We describe our detailed experiments settings in this section, including optimal hyper-parameter values for our deep CNN models to obtain best performance. Our 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

## 4.3 Evaluation Metrics

Training data is used to recognize the anomaly types within the system. Now, evaluate the performance of the model using test dataset.

This is the process of supervised learning i.e. log data patterns can be defined in advance.

To make a fair comparison between the anomaly performances of classification to determine which model performs best, we specifically extract a confusion matrix with the true positive (TP), false positive (FP), true negative (TN), and false negative (FN) counts

from the results of a model. Then, we can computer four standard evaluation metrics (best accuracy, precision, recall and f1-score) shown in equations 1, 2, 3 and 4 respectively.

Accuracy denotes.., Precision signifies..., Recall ..,F1-score ... Equations of the metrics based on confusion matrix are presented as follow:

## 4.4 Results and Discussion

**Classification Accuracy.**

To observe and assess the effects of varying the values of these hyper-parameters to the performance of the network with respect to HAR sensor data, we incorporated greedy-wise tuning starting from the number of layers (one-layer, ; two-layer, ; and three-layer, ), the number of feature maps , the size of the convolutional filter , and the pooling size . We varied the number of layers from 1 to 3, the number of feature maps from 10 to 200 in intervals of 10 (the same number for all layers [13]), the filter size from 1x3 to 1x15, and pooling size from 1x2 to 1x15.

**Sensitivity of Parameters.** We evaluate the sensitivity of varies the size of hidden layer, the number of layers, learning rate and dropout. In the following, we vary the width of hidden layer, the number of layers, learning rate and dropout respectively while keeping the other parameters as the best settings.

It is obvious that the learning rate is by far the most important hyper-parameter, always accounting for more than two thirds of the variance. The next most important hyper-parameter is the hidden layer size, followed by the input noise,

We characterised the effect of the number of convolutional layers employed to automatically learn feature representations. Figure 8 shows that increasing the number of convolutional layers tends to increase the performance for the dataset, improving by 1% when a new layer is added.

1. Hidden Layer Size. Not surprisingly the hidden layer size is an important hyper-parameter affecting the LSTM network performance. As expected, larger networks perform better, and the required training time increases with the network size.

In the following, we evaluate the effect of different hidden layer sizes of the CNN configuration. Assume CNN is composed of a convolution layer with the filter size of 10 units, a max-pooling layer with a sub-sampling factor of 3, and two top fully connected hidden layer having 1024 nodes and 18 node respectively. And one additional the softmax top layer to generate state posterior probabilities. We have tested CNN with the size of hidden layer from 16 to 128.

The recognition results are shown in Fig 7. The results with 128 hidden layers are better than that with 32 layers because ..... The best results are consistently achieved when setting the size of hidden layer to 3.

2) Learning rate. We evaluate the sensitivity of the learning rate for the weight values {0.0001, 0.001, 0.01,0.1}. The general trend show that, the accuracy of CNN steadily improves from [0.0001, 0.001]. Then it decreases even the weight continues increasing. It shows that this small amount of learning rate is important for the model to learn.

Learning rate is the most important hyper-parameter, therefore it is very important to understand how to set it correctly in order to achieve good performance. Figure 4 shows how setting the learning rate value affects the predicted average performance on the test set.

3) Dropout. Dropout has a tune-able hyper-parameter p, which represents the probability of retaining a hidden unit in the network. We explain the effect of varying this hyper-parameter. The comparison is done when the number of hidden units is held constant. This means all the nets have the same architecture at test time but they are trained with different amount of dropout. Fig 10 shows the test accuracy obtained as a function of p. It can be observed that the performance is insensitive to the value of p if 0.5 ≤ p ≤ 0.75, but drops sharply for small value of p. This is to be expected because for the same number of hidden units, having small p means very few units will turn on during training phase.

During several months of trials, we successfully group anomalies with high accuracy. We aim at modeling...

Using more features increases the dimensionality of the feature space, usually bring in undesirable effects like sparsity, and some redundant or irrelevant features may diminish performance of models in classification.

In this section we describe the proposed anomaly classification approach based on deep learning, focusing on the principal features as input. Deep learning is widely employed lately as it is very efficient in a large number of scenarios, especially for huge amount of high-dimensional datasets.

Besides convolutional neural network approach, we consider deep learning approach in our work. In addition, ... It is clear that the selection of features for classification tasks plays a major role in its empirical performance. CNN approach generally consider the temporal analysis of certain features, it employs a powerful ... to build appealing ...

Convolutional neural network is composed of multiple layers of neurons, each of them generally represented by a non-linear function [], every neural employs an activation function that maps the weighted inputs to the output that is passed to the following layer. The weights, originally set to random values, are iteratively adjusted during the training phase.

Global accuracy, recall and precision, F1-score are consider here to evaluate the performance of our deep learning classification models. Global accuracy Ai indicates the… Recall Ri means, … Precision Pi is… These four standard metrics are widely used for performance evaluation in classification tasks. Accuracy measures…, precision measures…, recall measures…, whereas F1-score measures..

Figure 2 depicts the performance comparison of the 2 classifiers in the classification of all 14 anomalous types. To decrease the influence the bias might bring in to all the evaluations, we employ 8-fold cross-validation with different random splits of the dataset, which indicates that we train and test our models in 8 different train/test dataset combinations.

There are no particular bias for both classifiers. The CNN classifier shows a slightly higher variance in the results, which might suggests that the model is slightly less robust and prone to leading to over-fitting problems.

CNN models provide great insights about

The data proportion of database anomaly instances in whole anomalous events is bigger than others as presented in Fig. 4, thus recognizing anomaly database events is pretty easy, and it will be unfair to the overall performance of model training and evaluation.

Figure 5 depicts the trends of accuracy, precision, recall and f1-score when the learning rate is increased. We can get best accuracy at learning rate is 0.0001. The recall obtains best recall when we set learning rate 0.001.

Figure 6 shows the effects of hidden layer size on model classification performance. With the size growing, accuracy.., precision..,

After optimizing both models, initial results showed CNN had much greater accuracy. The confusion matrices below allows visualization of performance for the CNN (fig 4)

models by showing percentage of total test data’s predicted label as compared to its ground-truth label.

Using the same CNN network with tuned regularization parameter and learning rate, the greatest accuracy of 98% was achieved for network classification. However, all 6 classes data on the same model yielded average 92.1% accuracy.

We evaluate our proposed deep learning approaches in this section by comparing anomalous instances classification performances achieved by all methods.

In the testing phase, we test the 20,000 events and repeat 10 times to calculate the average performance of CNN classifiers.

Not only the occurrence event is categorized in our experiment, but also the whole duration is classified here. Note that CNN is meant to be applied in the temporal dataset.

There are big differences between the models shown in Table 5. Based on those results, we can see that the model E has the best performance overall, Another interesting trend we can observe is the fact that

Some of the models provide more insights about the performance of models if we look at the more detailed results in table 6. Model B’s F1-score actually comes close to the F1-score of the model C

CNN achieves almost perfect classification performance in both cases, even slightly surpassing the CNN classifier. Figure 2 presents the classification results achieved by CNN on all features. We can conclude that CNN offers an accuracy comparable or even slightly better that that achieved by CNN in all anomalous types.

The classification performance obtained of type database anomalies is slightly worse than that of type file.

It should also be taken into account that there is a certain level of subjectiveness to the manual classification process, which may influence the results. This is caused by the fact that there are no strict

rules that determine whether something is an anomaly. Model E seems to give the best performance of the all models.

The classification performance for each class is also shown in terms of recall and precision measures, with the network anomaly instance having lowest recall equal to 88%. In particular, there is a noticeable misclassification overlap between this anomaly event and file anomaly event attributed to the physical location of the device and its difficulty to categorize them: future works will have to investigate the necessary steps in order to improve the discrimination of these anomaly events.

**Fig. 3.** Performance of different convnet architectures with increasing number of layers and feature maps on (a) validation data and (b) test data

# 5 Conclusion

In this paper we have proposed a deep convolutional neural network approach for anomaly event classification on distributed systems, offering a very powerful and straightforward technique to categorize anomalous instances. We generated dataset by extracting semantic information from large scale system log text. We take a series studies to find the proper learning rate and hidden layer size. We achieved an overall accuracy of 98.01%, which shows the potentiality of deep CNN for anomaly classification tasks on distributed systems.

Deep CNN model B is the most successful with 92.6% accuracy

In a nutshell, we believe this appealing approach is capable of providing high insights for understanding system server operations without disclosing any business sensitive information. By depending on deep learning techniques, we have shown the classification performance of the labeled anomalies in an efficient fashion. In general, CNN approach outperforms the CNN method. We will explore better deep learning model for anomaly classification in the future work.

This area is still on-going research, and it requires a lot of work to build the model for the time series. Should you succeed, you may achieve outstanding performance results in terms of accuracy.

In this paper, we have made a comparison between different models to anomaly event classification. We evaluated the performance of these models when applied to an anomaly event task for application log data analysis. It turned out that the relatively model E gave the best performance. However,

The obvious but useful extension to this work would be to extend the experiments to more algorithms to see if there are algorithms available that work even better. With more data could also be useful since that could add the possibility of researching changing trends in the data and how the different algorithms cope with these changes.

However it also faces many challenges.

There are still a few things that need to be done to get this automatic classification running in a production environment.

The novelty of our work lies in the automatic anomaly classification of system events and states from streaming operational logs. Through the evaluation of ..;

Experiments with larger datasets are needed to further study the robustness of the proposed technique. Further improvements may be achieved by using unsupervised pretraining and repeating pooling operations in multiple layers of the CNN model.

# Acknowledgments

This work was supported in part by Institute of Computing Technology, Chinese Academy of Sciences.

# References

[1] Victoria J. Hodge, Jim Austin. A Survey of Outlier Detection Methodologies. Artificial Intelligence Review. 2004; 22(2):85-126. doi:10.1007/s10462-004-4304-y.

[2] Smith M. R., Martinez T. Improving classification accuracy by identifying and removing instances that should be misclassified. The 2011 International Joint Conference on Neural Networks. 2011; p. 2690. ISBN 978-1-4244-9635-8. doi:10.1109/IJCNN.2011.6033571.

[3] Knorr E. M., Ng R. T., Tucakov, V. Distance-based outliers: Algorithms and applications. The VLDB Journal the International Journal on Very Large Data Bases. 2000; 8(3-4):237-253. doi:10.1007/s007780050006.

[4] Breunig M. M., Kriegel H.-P., Ng R. T., Sander, J. LOF: Identifying Density-based Local Outliers. Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. SIGMOD. 2000; pp. 93–104. ISBN 1-58113-217-4. doi:10.1145/335191.335388.

[5] Schubert E., Zimek A., Kriegel H. P. Local outlier detection reconsidered: A generalized view on locality with applications to spatial, video, and network outlier detection. Data Mining and Knowledge Discovery. 2012; 28: 190-237. doi:10.1007/s10618-012-0300-z.

[6] Kriegel H. P., Kroger P., Schubert E., Zimek A. Outlier Detection in Arbitrarily Oriented Subspaces. 2012 IEEE 12th International Conference on Data Mining. 2012; p.379. ISBN 978-1-4673-4649-8. doi:10.1109/ICDM.2012.21.

[7] He Z., Xu X., Deng S. Discovering cluster-based local outliers. Pattern Recognition Letters. 2003; 24(9-10):1641-1650. doi:10.1016/S0167-8655(03)00003-5.

[8] Campello R. J. G. B., Moulavi D., Zimek A., Sander J. Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. ACM Transactions on Knowledge Discovery from Data. 2015; 10(1):5:1-51. doi:10.1145/2733381.

[9] Zimek A., Campello R. J. G. B., Sander J. R. Ensembles for unsupervised outlier detection. ACM SIGKDD Explorations Newsletter. 2014; 15:11–22. doi:10.1145/2594473.2594476.

[10] Zimek A., Campello R. J. G. B., Sander J. R. Data perturbation for outlier detection ensembles. Proceedings of the 26th International Conference on Scientific and Statistical Database Management-SSDBM. 2014; p.1. ISBN 978-1-4503-2722-0. doi:10.1145/2618243.2618257.

[11] Campos Guilherme O., Zimek Arthur, Sander Jörg, Campello Ricardo J. G. B., Micenková Barbora, Schubert Erich, Assent Ira, Houle Michael E. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Mining and Knowledge Discovery. 2016; 30(4):891. ISSN 1384-5810. doi:10.1007/s10618-015-0444-8.

[12] Goldstein M. In: Markus Hofmann RK, editor. Anomaly Detection. Data Mining and Knowledge Discovery Series. Chapman and Hall/CRC. 2013; p.367-394.

[13] Aggarwal CC. Outlier Analysis. Springer-Verlag. NewYork. 2013.

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

[15] Ding X, Li Y, Belatreche A, Maguire LP. An Experimental Evaluation of Novelty Detection Methods. Neurocomputing. 2014; 135:313-327. doi:10.1016/j.neucom.2013.12.002

[16] Amer M, Goldstein M, Abdennadher S. Enhancing One-class Support Vector Machines for Unsupervised Anomaly Detection. In: Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description (ODD’13). New York, NY, USA: ACM Press. 2013; p. 8-15.

[17] Zimek A, Campello RJGB, Sander J. Ensembles for Unsupervised Outlier Detection: Challenges and Research Questions a Position Paper. SIGKDD Explor Newsl. 2014; 15(1):11-22. doi: 10.1145/2594473.2594476.

[18] Amer M, Goldstein M. Nearest-Neighbor and Clustering based Anomaly Detection Algorithms for Rapid Miner. In: Simon Fischer IM, editor. Proceedings of the 3rd Rapid Miner Community Meeting and Conference (RCOMM 2012). Shaker Verlag GmbH; 2012. p. 1-12.

[19] P. Fiadino et al. RCA Tool - A Framework for Detecting and Diagnosing Anomalies in Cellular Networks. ITC. 2015.

[20] P. Casas et al. MINETRAC: Mining Flows for Unsupervised Analysis & Semi-Supervised Classification. ITC. 2011.

[21] Greff, Klaus, et al, LSTM: A Search Space Odyssey, arXiv preprint arXiv:1503.04069, 2015.

[22] M. Gupta, J. Gao, C. Aggarwal, and J. Han. Outlier detection for temporal data. Synthesis Lectures on Data Mining and Knowledge Discovery 5.1. Morgan & Claypool Publishers; 2014. pp. 1-129.

[23] E. Schubert, A. Zimek, and H.-P. Kriegel. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. Data Mining and Knowledge Discovery 28.1. Springer; 2014. pp. 190-237.

[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. INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014; 2014. pp. 610-618.

[25] 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), 2013; pp. 118-126.

[26] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. CoRR, abs/1412.3128, 2014.5.

[27] M. Henrion, D. J. Hand, A. Gandy, and D. J. Mortlock. CASOS: A subspace method for anomaly detection in high dimensional astronomical databases. Statistical Analysis and Data Mining, 6(1):53-72, 2013.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278-2324, 1998.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, volume 1, page 4, 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, 35(1):221-231, 2013.

[31] O. Abdel-Hamid, A.-r. 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, pages 4277-4280. IEEE, 2012.

[32] Lee, H.; Grosse, R.; Ranganath, R.; Ng, A.Y. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In Proceedings of the 26th Annual International Conference on Machine Learning (ICML), Montreal, QC, Canada, 14-18 June 2009; pp. 609-616.

[33] Lee, H.; Pham, P.; Largman, Y.; Ng, A. Unsupervised feature learning for audio classification using convolutional deep belief networks. In Proceedings of the 22th Annual Conference on Advances in Neural Information Processing Systems (NIPS), Vancouver, BC, Canada, 8-10 December 2008; pp. 1096-1104.

[34] Levenshtein VI. Binary codes capable of correcting deletions, insertions and reversals. Journal of Soviet Physics Doklady, 1966, 10:707-711.

[35] Navarro, Gonzalo (1 March 2001). A guided tour to approximate string matching. ACM Computing Surveys. 33 (1): 31-88. doi:10.1145/375360.375365. Retrieved 19 March 2015.

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