Network Alarm Log Data Enhancement Method Based on GAN Model

Yiding Liu, Yuting Li, Yang Yang(✉), Zhipeng Gao, and Lanlan Rui

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, People's Republic of China
yyang@bupt.edu.cn

Abstract. With the rapid development of information technology, the scale of network equipment has increased immeasurably, generating massive amount of alarm log data. It will cause huge losses if alarm log cannot be monitored properly and handled in time. Nevertheless, most alarm log data are unbalanced and difficult to be collected. Thus, the result of training of fault diagnosis method based on deep learning, which usually requires a large amount of training data, are not ideal, and the effect of fault diagnosis method is limited. To address the issues of insufficient and unbalanced alarm log data, this paper proposes an alarm log data enhancement method based on LeakGAN. This model introduces self-attention mechanism to discriminator and batch normalization to generator. In adversarial training, parameters of the association algorithm are introduced to rescale the rewards. Finally, this paper conducts comparative experiments with dataset from OpenStack cloud server, which proofs the proposed model can improve the quality of generated alarm log data and accelerate the convergence of the model. In addition, the alarm log data generated can effectively improve the accuracy of the fault diagnosis algorithm.

Keywords: Alarm log · Fault diagnosis · Data enhancement · Generative adversarial nets · Attention · Batch normalization · Association algorithm

1 Introduction

At present, with the rapid development of information technology, computers have been widely applied in the field of network service industries. The amount of network alarm log data generated increased immeasurably as the scale of network access devices increased. As network structure becomes increasingly complex, it will cause huge losses including information loss and increasing of the burden of network maintenance if alarm logs cannot be monitored in time and handled effectively.

Fault diagnosis method completes fault reasoning by analyzing characteristics and association of alarm logs. The main tasks include fault detection, fault localization, and testing. Alarm log data have problems including unbalanced distribution and strongly association with time, existence of many redundant alarm logs, and the sparsity of abnormal alarm logs. Thus, datasets collected often cause the poor training and effect of the fault diagnosis methods based on deep learning.

Q. Liu et al. (Eds.): CENet 2022, LNEE 961, pp. 125–134, 2022.
https://doi.org/10.1007/978-981-19-6901-0_14

To address the issues of unbalanced and insufficient alarm log data, text data enhancement algorithms can be introduced. Generative adversarial nets (GAN) [1] cannot be applied directly to the training of discrete alarm log data, because the gradients cannot be updated by back propagation. As the research progressed, people combined with the ideas of adversarial training, reinforcement learning of policy gradients and proposed text generation models such as SeqGAN [2], LeakGAN [3]. LeakGAN solves the problems including reward sparsity of SeqGAN and can be competent for the long text generation environment of alarm logs. However, few problems of LeakGAN including the lack of filtering of feature vectors and poor convergence rate have gradually been revealed.

Thus, this paper proposes a LeakGAN-based alarm log data enhancement model. Several structural improvements are proposed and few parameters of the association algorithm are introduced in the reward rescale process in adversarial training. The alarm log data enhancement algorithm proposed can converge fast and effectively improve the quality of generated alarm log texts, which improves the performance of fault diagnosis.

2 Related Works

Traditional text enhancement techniques mainly includes rule-based and template generation methods [4]. The essence of these methods is generating text through pre-designed language rules or templates, which can effectively generate texts with merely simple rules. However, the diversity and fluency of generated alarm logs are not ideal.

Goodfellow et al. proposed Generative Adversarial Nets (GAN) [1]. GAN consists of a generator G and a discriminator D, which corresponds to a minimax two-player game. D attempts to judge the authenticity of the generated data from G, while G uses generated data to cheat D until they reach a Nash Equilibrium state. In 2017, Kevin et al. proposed RankGAN [5], a ranking model which better solves the problem of output discretion. Later, Gulrajani et al. proposed WGAN [6], in which the Wasserstein distance is proposed to measure the distribution gap between generated and real data, which effectively addressed the difficulty with training. In 2017, Yu et al. proposed SeqGAN [2] based on the idea of reinforcement learning, solving the scoring issue by Monte Carlo Tree Search. However, SeqGAN is subject to pattern collapse caused by one side being too weak. Guo et al. proposed LeakGAN in 2018 [3], which solves the problem of rewards and pattern collapse in SeqGAN by hierarchical reinforcement learning.

The feature extraction layer of LeakGAN discriminator is a convolutional neural network (CNN), which is responsible for obtaining high-dimensional feature vectors and leaking them to generator. The generator of LeakGAN consists of a Manager and a Worker module, respectively. Manager, a Long Short-Term Memory network (LSTM), is responsible for receiving the leaked information generated by the discriminator and output the target embedding vector. While worker is responsible for outputting the action embedding vector. Two vectors then are combined and sampled, generating the next word. Subsequently, the current sequence is filled by a Monte Carlo tree search and the result is passed to discriminator to score. The score, or rewards, will be used to update parameters of generator. However, the alarm log generated by LeakGAN is unable to

significantly improve the accuracy of fault diagnosis due to the lack of filter and fusion of high-dimensional feature in discriminator. Meanwhile, the existence of covariate transformation in generator makes it hard for model to converge in training.

3 Proposed Method

Firstly, the proposed model introduces self-attention mechanism [7] in discriminator model to improve the quality of generated alarm log. Secondly, it introduces a batch normalization method [8] in the discriminator model to improve the model performance. Finally, it introduces a reward rescale according to the minimum confident parameter of the association algorithm FP-Growth [9] in adversarial training process to further improve the quality of the alarm log text.

4 The Discriminator Model Based on the Self-attention Mechanism

The feature extraction part of the LeakGAN model consists of an input layer, a convolutional layer, and a pooling layer, which are responsible for extracting a high-dimensional feature vector of the input data and passing it to the generator. The quality of the generated text depends on the quality of this feature vector to a great extent. However, the lack of filtering and fusion of the generated high-dimensional features results in the generated alarm log text not being effective in improving the diagnostic effect of the fault diagnosis model. To address this problem, this paper adds a Scaled Dot-Product Attention before and after the convolutional layer of CNN respectively.

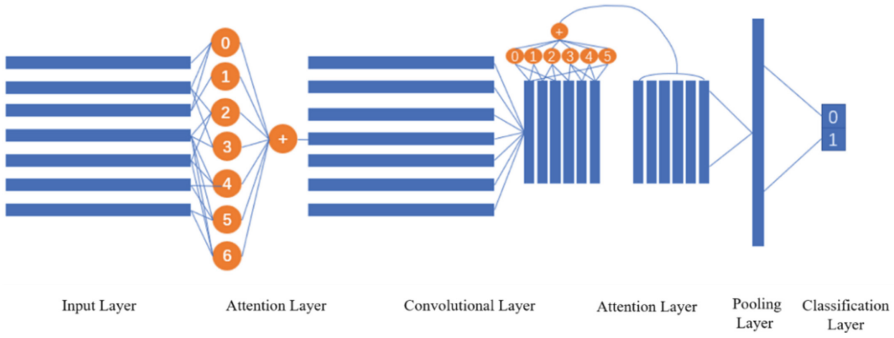

Fig. 1. The discriminator based on the self-attention mechanism

As Fig. 1 shown above, the discriminator consists of an input layer, an attention layer, a convolution layer, an attention layer, and a pooling layer. The input layer is responsible for vectorizing the input sequence of length n into an n*h feature matrix. The scaling parameter of the attention layer before the convolution equals to the input matrix dimension h. The convolution kernel size is c*h. The attention layer after convolution applies self-attention mechanism to the feature vector after convolution of each class of convolution kernels, and the scaling parameter of this layer is equal to the number of

convolution kernels to ensure that the feature vector size is constant. The dropout layer is responsible for using the feature vectors as leakage information to guide the generator to generate the alarm log text. The classification layer will perform the classification job and pass the output reward to generator.

Pre-training reduces the adversarial training time and accelerates the model convergence. The discriminator model designed in this paper introduces a cross entropy loss function in the pre-training step, which can significantly improve the balance of the training set, and its expression is shown in (1).

$$H(p,q) = -\sum\nolimits_{x}(p(x)logq(x)) \tag{1}$$

where *p(x)* denotes x true labels, *q(x)* denotes x output labels. Cross entropy denotes the closeness of the actual output to the desired output, and the smaller the value means the closer the two are.

In the alternate training process, the probability that the output is true data needs to be as close to 1 as possible in order to achieve the best alarm log text generation effect. This process is reflected in the objective function of the discriminator, as shown in (2).

$$\mathop{max}_{\gamma} E_{\sim \mathrm{pdata}}[logD(x)] + E_{\gamma \sim \mathrm{G}_{\theta}}[\log(1 - D(x)] \tag{2}$$

where G_θ denotes the generator, and $D(x)$ denotes the probability that the input sequence is authentic.

5 The Generator Model Based on Batch Normalization and LSTM

LSTM overcomes problems of long-term dependency of RNN and can apply to generate long alarm log. However, during each iteration of training, small changes in training data distribution of the first few layers of the network accumulate, causing data distribution near the upper and lower bounds of the activation function, or internal covariate transformation. If left untreated, the layer-by-layer transfer will lead to gradient disappearance. Therefore, batch normalization method needs to be introduced to enhance the performance of the model, which shown in Fig. 2.

The Manager and Worker modules are composed of two identical LSTMs, respectively. Among them, the relevant parameters are illustrated as follows: W is the forgetting weight, x_t is input, b is bias term, σ denotes the activation function, usually Sigmoid, h denotes the latest hidden state, c denotes the state of the cell.

The improved LSTM forward calculation process is represented as follows:

$$i_t = \sigma(BN(W_{xi}x_t) + BN(W_{hi}h_{t-1}) + b_i) \tag{3}$$

$$f_t = \sigma(BN(W_{xf}x_t) + BN(W_{hf}h_{t-1}) + b_f) \tag{4}$$

$$o_t = \sigma(BN(W_{xo}x_t) + BN(W_{ho}h_{t-1}) + b_o) \tag{5}$$

$$c_t = f_t * c_{t-1} + i_t * \tanh(BN(W_{xc}x_t) + BN(W_{hc}h_{t-1}) + b_c) \tag{6}$$

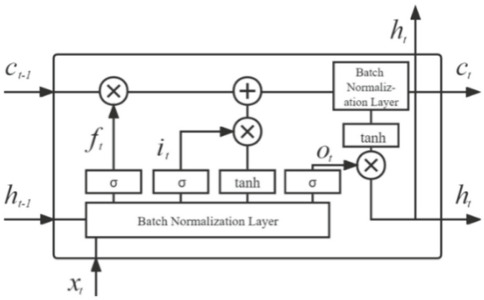

Fig. 2. The generator model based on batch normalization and LSTM

The output h_t of the LSTM at moment t can be expressed as:

$$h_t = o_t * tanh(BN(c_t)) \tag{7}$$

The Manager module outputs the vector a_t by high-dimensional feature vector leaked by the discriminator, and then calculates the target vector g_t by vector unitization. Batch normalization operation is performed on the data before all Sigmoid and tanh activation functions in the LSTM of the generator. The batch normalization solves the problem of gradient disappearance due to layer-by-layer transfer by changing the activation function output from a randomly distributed state to a normal distribution with a fixed variance of 1. Meanwhile, a linear transformation ψ is applied to the target vector g_t to eliminate the data format difference between Manager and Worker. The k-dimensional target embedding vector ω_t is output after the transformation by the weight matrix $W\psi$. The transformation process is:

$$\omega_t = W_\psi(\sum_{i=1}^{c} g_{t-i}) \tag{8}$$

where c denotes the c most recently outputted target vectors. Then, target embedding vector ω_t is passed to the Worker module, and the output result distribution is obtained after the following operations:

$$G_\theta(\cdot|s_t) = softmax(O_t \cdot \omega_t) \tag{9}$$

Then, through a Monte Carlo tree search, the generator generates the output sequence and passes the output to the discriminator. The discriminator generates reward values that are used for further optimization work on the generator parameters. The generator continues to receive the next feature vector and then processes the next word. During adversarial training process, the generator expects the scores of the data generated by the generator to be close to 1 in order to achieve the best alarm log text quality. The general objective function of the generator is as follows:

$$\min_\gamma E_{\sim pdata}[logD(x)] + E_{\gamma \sim G_\theta}[log(1 - D(x)] \tag{10}$$

In the generator training process, the policy gradient descent algorithm is used, but the Manager and Worker module gradient updates are not the same, as described below, respectively. The gradient function of Manager is defined as follows:

$$\nabla_{\theta_m}^{adv} g_t = -Q(s_t, g_t) + \nabla_{\theta_m} d_{cos}(f_{t+c} - f_t, g_t(\theta_m)) \tag{11}$$

where $Q(s_t, g_t)$ denotes the reward value of the discriminator, d_{cos} denotes the cosine similarity, f_{t+c} denotes the feature vector after c time, and g_t is the output target vector of the Manager. The gradient function of Worker is defined as follows:

$$\nabla_{\theta_m}^{adv} g_t = -Q(s_t, g_t) + \nabla_{\theta_m} d_{cos}(f_{t+c} - f_t, g_t(\theta_m)) \tag{12}$$

6 Reward Rescale Mechanism Based on the FP-Growth Confident Parameter

In adversarial training, gradient disappearance often occurs when discriminator is much stronger than generator, triggering pattern collapse problem. As mentioned above, in LeakGAN, the generator generates the output sequence and passes it to the discriminator, which gives the sequence scores, or rewards. Then the reward will be passed to generator to guide the generation of new words. If the reward value is too small or improperly considered, parameters will not be updated in time, leading to the disappearance of the gradient. To address this issue, the reward value needs to be rescaled. In this paper, we introduce the minimum confidence parameter of the association algorithm FP-Growth into the rescale of the reward value in adversarial training. Assuming $R_{\mathrm{B}\times\mathrm{T}}$ denotes the reward matrix, for moment t, the vector R^t of the t^{th} column of the reward matrix is rescaled as follows:

$$R_i^t = \sigma(100m \cdot (0.5 - \frac{rank(i)}{B})) \tag{13}$$

where Rank(i) denotes the element from the highest to the i^{th} element in the vector R^t, $\sigma(\cdot)$ is the activation function, which is used to reproject the distribution based on ranking to a better equivalent distribution, usually a Sigmoid. The parameter m denotes a hyper-parameter, which is the minimum confidence parameter of the association algorithm. The minimum confidence parameter of the FP-Growth algorithm can properly represent the ratio of abnormal alarm logs in the alarm log dataset after artificial adjustment. Its introduction into the reward rescale after proper integration can effectively prevent the reward rescale process from being too aggressive and prevent the generation of redundant alarm log texts. The introduction of the confidence parameter of the association algorithm can solve the problem that training caused by the training parameters is not updated in time. In addition, it reduces the number of invalid alarm log texts generated, thus improves the quality of the generated texts and the reliability of the model.

7 Simulation

7.1 Design of Comparison Experiments

The comparison experiments are conducted by the following two models trained separately, such as Standard LeakGAN model(LeakGAN) and Improved LeakGAN for Alarm Log Data Enhancement(LeakGAN-ALDE). In addition, the results are obtained by comparing a series of Metrics including Negative Log Likelihood, BLEU, and accuracy of fault diagnosis before and after alarm log enhancement.

7.2 Evaluation Metrics

(1) Negative Log Likelihood (NLL)
Negative log likelihood is an accuracy evaluation indicator proposed in SeqGAN [2]. It is defined as (14):

$$NLL_{oracle} = -E_{x\sim G_\theta}[logG_{oracle}(x)] \tag{14}$$

where G_θ denotes the generator, G_{oracle} denotes the human evaluation model, which comes from the meaning of evaluating the generated text based on a priori knowledge, and is a random initialized LSTM. The smaller the value of NLL_{oracle} indicates that the distribution range of the generated data is closer to the real data.

(2) BLEU
BLEU is an auxiliary tool for bilingual evaluation [10]. The purpose of BLEU is to compare the overlap between the generated text and the n-grams in the standard text. The overlap, or score, of BLEU ranges from 0 to 1. The higher the score is, the higher the quality of the generated alarm log text is. The formula is shown in (15).

$$BLEU = BP \cdot \exp(\sum_{n=1}^{N} W_n logP_n) \tag{15}$$

where n of the n-gram is the parameter n in (15), BP denotes penalty factor, which is mainly used to penalize the scenario where the difference between the generated text and the original text is too significant.

(3) Fault diagnosis accuracy
The performance of fault diagnosis is evaluated by accuracy, calculated as (16).

$$accuracy = \frac{\text{Number of valid association rules}}{\text{Number of all association rules}} \tag{16}$$

7.3 Dataset

We collected the log records generated by an OpenStack cloud server as the dataset, with a total of about 200,000 entries. After log parsing and labeling process, the corresponding label templates are generated. The input data consist of integer sequence, where each integer input denotes the index of the corresponding template in label templates. The dataset is divided into three parts representing the test set of abnormal alarm logs, the test set of normal alarm logs, and the training set containing normal and abnormal alarm logs, respectively.

7.4 Simulation Results

Negative Log Likelihood (NLL)
A total of 400 enpochs are trained, and the NLL values of LeakGAN and LeakGAN-ALDE are reported every 5 rounds. Figure 3 (left) shows final NLL training curves.

Table 1. Minimum Negative Log Likelihood Values

LeakGAN	LeakGAN-ALDE
7.291	6.684

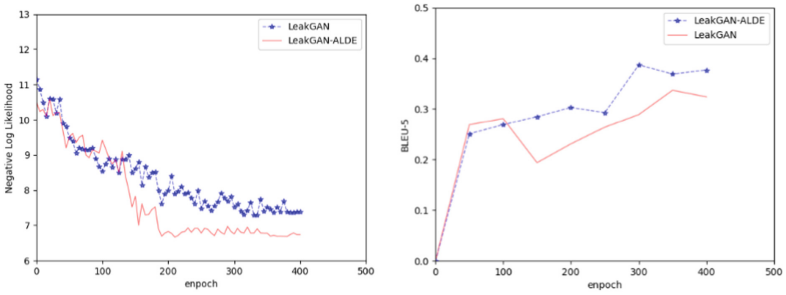

Fig. 3. Negative log likelihood (left) and BLEU-5 Scores (right) curves

As shown in Fig. 3, LeakGAN-ALDE starts to converge rapidly at around 130 enpochs and stabilizes at around 200 enpochs. In contrast, LeakGAN converges more slowly and stabilizes at around 250 enpochs. From Table 1, the minimum NLL of LeakGAN-ALDE reaches 6.684, which is lower compared to value of 7.291 of LeakGAN. Collectively, LeakGAN-ALDE shows a better speed of convergence and minimum NLL value than LeakGAN, which indicates LeakGAN-ALDE outperforms LeakGAN in terms of alarm log generation quality and model performance.

BLEU Scores
Figure 3 (right) denotes the BLEU-5 scores of LeakGAN and LeakGAN-ALDE in 400 enpochs, reported in every 50 enpochs. Meanwhile, the optimal values of the four metrics including BLEU-2, BLEU-3, BLEU-4, BLEU-5 are reported in this paper. The highest scores of each are shown as Table 2.

Table 2. Highest BLEU Scores

Metrics	LeakGAN	LeakGAN-ALDE
BLEU-2	0.698	0.712
BLEU-3	0.582	0.604
BLEU-4	0.405	0.411
BLEU-5	0.337	0.387

The BLEU-5 scores of LeakGAN-ALDE are basically higher than those of LeakGAN during the training process, and the maximum score difference reached 0.1 points or so. Meanwhile, the enpoch required to reach the maximum score of LeakGAN-ALDE was less than that of LeakGAN. As shown in Table 2, scores of LeakGAN-ALDE are higher than LeakGAN, and even under the highly cumulative model of BLEU-5, it can maintain a score difference of about 0.05, which indicates that LeakGAN-ALDE proposed in this paper is higher than LeakGAN in terms of the generated alarm logs quality.

Fault Diagnosis Accuracy

The following Table 3 shows the fault diagnosis accuracy before and after enhancement of LeakGAN-ALDE. Two sets of FP-Growth parameters are introduced. For the original dataset without data enhancement, parameter set B {min_confidence = 0.05, min_support(count) = 2} is used because the dataset is more unbalanced and it is difficult to mine the fault rules. Relatively, after data enhancement by LeakGAN-ALDE, the parameter set A {min_confidence = 0.25, min_support(count) = 5} is used due to the improvement of the dataset quality.

Table 3. Fault diagnosis accuracy

Accuracy	Before Enhancement	After Enhancement
Set A	73.1%	89.2%
Set B	31.4%	85.9%

As shown in Table 3, the fault diagnosis accuracy with moderate parameters before enhancement is only 31.4% due to unbalance data. Only when the minimum confidence and the minimum support count is adjusted downward to 0.05 and 2, respectively, can it reach the accuracy of 73.1%. After dataset is enhanced, the accuracy of two parameter sets reaches 89.2% and 85.9%, respectively, which is a significant improvement compared with that before the enhancement.

8 Conclusion

In this paper, three improvements are proposed for the model structure and training method of LeakGAN, which are suitable for generating high quality alarm log texts.

The results of simulation show that the alarm log data enhancement model proposed can generate high-quality alarm log texts more stably and quickly compared with traditional model, which will significantly improve the performance of fault diagnosis and solve the problems of insufficient and unbalanced alarm log data.

Acknowledgment. This work is supported by the National Key R&D Program of China (2019YFB2103202).

References

1. Goodfellow, I., et al.: Generative adversarial nets. Advances in neural information processing systems, **27**, 2672–2680 (2014)
2. Yu, L., Zhang, W., Wang, J., Yu, Y.: Seqgan: sequence generative adversarial nets with policy gradient. Proceedings of the AAAI conference on artificial intelligence **31**(1), 2852–2858 (2017)
3. Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., Wang, J.: Long text generation via adversarial training with leaked information. Proceedings of the AAAI Conference on Artificial Intelligence **32**(1), 5141–5148 (2018)
4. Deemter, K.V., Theune, M., Krahmer, E.: Real versus template-based natural language generation: a false opposition. Comput. Linguist. **31**(1), 15–24 (2005)
5. Lin, K., Li, D., He, X., Zhang, Z., Sun, M.T.A.: dversarial ranking for language generation. Adv. Neural. Inf. Process. Syst. **30**, 3155–3165 (2017)
6. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. Adv. Neural. Inf. Process. Syst. **30**, 5767–5777 (2017)
7. Vaswani, A., et al.: Attention is all you need. Advances in neural information processing systems **30**, 2322-2431 (2017)
8. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456, PMLR (2015)
9. Borgelt, C.: An implementation of the FP-growth algorithm. In: Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations, pp. 1–5 (2005)
10. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311–318 (2002)