Log Message Anomaly Detection and Classification Using Auto-B/LSTM and Auto-GRU

Amir Farzad[1][0000−0003−2499−7696] and T. Aaron Gulliver[2][0000−0001−9919−0323]

[1] Department of Electrical and Computer Engineering, University of Victoria,
`amirfarzad@uvic.ca`

[2] Department of Electrical and Computer Engineering, University of Victoria, PO Box 1700, STN CSC, Victoria, BC Canada V8W 2Y2 `agullive@ece.uvic.ca`

Abstract. Log messages are now widely used in software systems. They are important for classification as millions of logs are generated each day. Most logs are unstructured which makes classification a challenge. In this paper, Deep Learning (DL) methods called Auto-LSTM, Auto-BLSTM and Auto-GRU are developed for anomaly detection and log classification. These models are used to convert unstructured log data to trained features which is suitable for classification algorithms. They are evaluated using four data sets, namely BGL, Openstack, Thunderbird and IMDB. The first three are popular log data sets while the fourth is a movie review data set which is used for sentiment classification and is used here to show that the models can be generalized to other text classification tasks. The results obtained show that Auto-LSTM, Auto-BLSTM and Auto-GRU perform better than other well-known algorithms.

Keywords: LSTM · BLSTM · GRU · Autoencoder · Deep Learning · Neural Network · Log messages · Anomaly detection · Classification

1   Introduction

Software frameworks and systems such as web search engines and cloud computing servers are now prevalent in everyday life. Availability is expected 24/7 and access failures can cause significant hardship to both organizations and clients. In a framework, each logging statement produces log messages corresponding to a particular task. Unstructured log messages contain runtime data including verbosity level, timestamp (event time), and raw message content which is a text description of the framework activity. The structure of these messages can vary significantly between frameworks which makes anomaly detection in unstructured logs difficult [43].

Logs which record runtime data are used for many tasks including anomaly detection [21], [30], [40], [6], [7], [18], program verification [3], fault diagnosis [48] and performance monitoring [27]. Some techniques use rules for detecting anomalies in log messages but these require explicit domain knowledge [41]. Most consider just a single component of a log message such as the verbosity level which restricts the anomalies that can be identified [42], [45], [35]. These tasks can be done manually on a small scale, but such anomaly detection depends on extensive manual review of logs and a keyword search cannot be used for detecting suspicious log messages for large scale frameworks [21]. Modern frameworks produce huge amounts of log data, e.g. at a rate of over 50 Gigabytes (millions of lines) per hour [25]. Thus, automated log analysis strategies for anomaly detection and classification are necessary.

An Autoencoder [36] is a feed forward Artificial Neural Network (ANN) that can can learn features from data and the data struc-

ture [38], [33]. Autoencoders have been applied to many different tasks such as probabilistic and generative modeling [19], [32], representation learning [14], and interpolation [24], [34]. A Long Short-Term Memory (LSTM) network [15] is a Recurrent Neural Network (RNN) that uses a cell to retain sequence information and remember long-term dependencies. LSTMs have been successfully used for tasks such as language modeling [44], translation [22], analysis of audio and video data [23], [4], phoneme classification [9], online mode-detection [29], emotion recognition [39] and acoustic modeling [37]. An LSTM only utilizes past context but a Bidirectional Long-Short Term Memory (BLSTM) [10] can utilize both the past and future contexts by processing the input data in both the forward and backward directions. BLSTMs have have been used for numerous tasks such as text recognition and classification [1], [46]. A Gated Recurrent Unit (GRU) [2] is similar to an LSTM network and the performance is comparable. However, it has been shown to provide better results than an LSTM in tasks such as speech recognition [17].

Log messages can be considered as sequences so an LSTM or GRU network with an Autoencoder is a suitable structure. Thus in this paper, models are proposed called Auto-LSTM, Auto-BLSTM and Auto-GRU that first extract features from log messages using an Autoencoder and then use the resulting trained features in an LSTM, BLSTM or GRU network for anomaly detection and classification. The models are evaluated based on the accuracy, precision, recall and F-measure using three labeled log message data sets, namely Blue-

Gene/L (BGL)[3], Openstack[4] and Thunderbird[5]. The IMDB movie review data set[6] is also considered for sentiment classification to illustrate model generalization to other text classification tasks. It is shown that good results are obtained with the proposed models for all four data sets with the same configuration.

The main contributions are as follows.

1. The Auto-LSTM, Auto-BLSTM and Auto-GRU models are proposed for anomaly detection and classification.
2. The proposed models are evaluated using four data sets and the results are compared with those for other well-known models. This shows that the proposed models provide the best performance.

The rest of the paper is organized as follows. Section 2 presents the Autoencoder, LSTM, BLSTM and GRU architectures followed by the proposed models. The simulation results and discussion are given in Section 3. Finally, Section 4 provides some concluding remarks.

2   System Model

In this section, the Autoencoder, LSTM, BLSTM and GRU architectures employed in the proposed models are described.

2.1   Autoencoder Architecture

An Autoencoder [36] is a feed-forward multilayer neural network with the same number of input and output neurons. The goal is to

---

[3] https://github.com/logpai/loghub/tree/master/BGL
[4] https://github.com/logpai/loghub/tree/master/OpenStack
[5] https://github.com/logpai/loghub/tree/master/Thunderbird
[6] https://ai.stanford.edu/~amaas/data/sentiment

learn a compressed representation while minimizing the error for the input data. Training is done using the backpropagation algorithm according to a loss function. An Autoencoder with more than one hidden layer is called a deep Autoencoder [20]. Having many encoder and decoder layers enables a deep Autoencoder to represent complex input distributions. Fig. 1 shows the architecture of an Autoencoder with an input layer, an output layer, and two hidden layers.
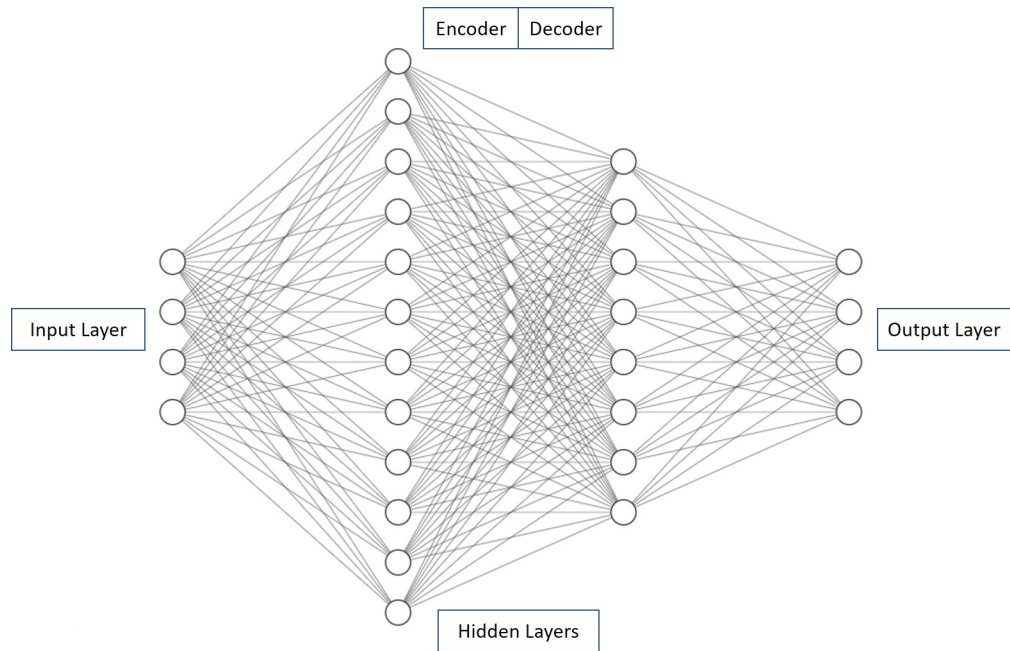

Fig. 1. Autoencoder structure with an input layer, output layer and two hidden layers.

2.2   LSTM Architecture

An LSTM is a recurrent neural network (RNN) [15] which has been successfully used to solve sequential data problems [8]. It has cells to store information in blocks which can be recurrently connected.

These cells solve the vanishing gradient problem. Each LSTM block contains self-connected cells with input, forget, and output gates. These gates are designed to store information longer than feedforward neural networks to improve performance [8]. A block of an LSTM network contains recurrently connected cells as shown in Fig. 2. The cell input at time t is x_t and the corresponding values for the input, forget and output gates are

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{1}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{2}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{3}$$

respectively, where W and U are the weight matrices, b is the bias vector, and σ is the sigmoid activation function. The block input at time t is

$$\hat{C}_t = \tanh(W_C x_t + U_C h_{t-1} + b_C), \tag{4}$$

where W_C and U_C are the weight matrices, b_C is the bias vector, and tanh denotes the hyperbolic tangent activation function. The cell state at time t is

$$C_t = f_t \odot C_{t-1} + i_t \odot \hat{C}_t, \tag{5}$$

where $\odot$ denotes point-wise multiplication. Finally, the block output at time t is given by

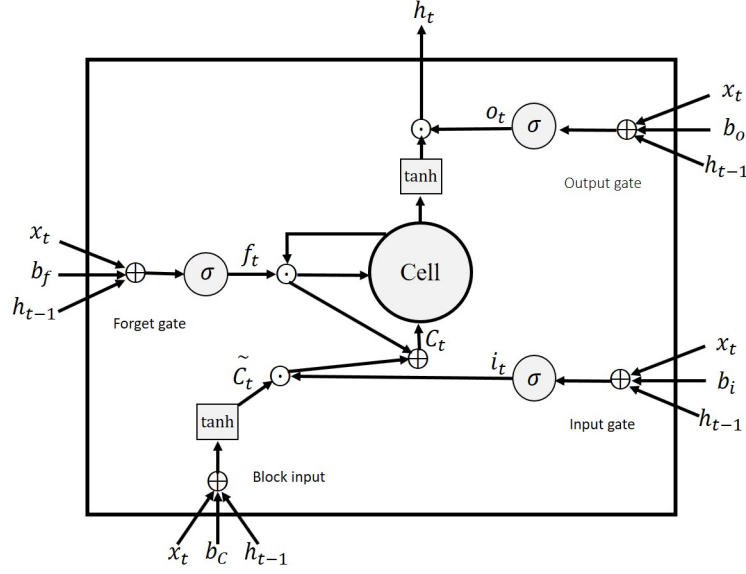$$h_t = o_t \odot \tanh(C_t). \tag{6}$$

Fig. 2. A LSTM block with input gate, output gate, forget gate, block input, and sigmoid and hyperbolic tangent activation functions.

2.3   BLSTM Architecture

Bidirectional Long-Short Term Memory (BLSTM) consists of two LSTM networks. The input data is fed into two LSTM networks forwards and backwards with respect to time t, respectively, and both are connected to the same output layer. BLSTMs have the benefit of using both the past context and future context in a sequence. In a BLSTM, the forward and backward LSTM outputs at time t are concatenated which is expressed by

$$h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}], \tag{7}$$

where $\overrightarrow{h_t}$ is the forward block output and $\overleftarrow{h_t}$ is the backward block output. The final output at time t is

$$y_t = \sigma(W_y h_t + b_y), \tag{8}$$

where W_y is the weight matrix, b_y is the bias vector and σ is the activation function. The BLSTM network architecture with forward and backward LSTM layers is shown in Fig. 3.
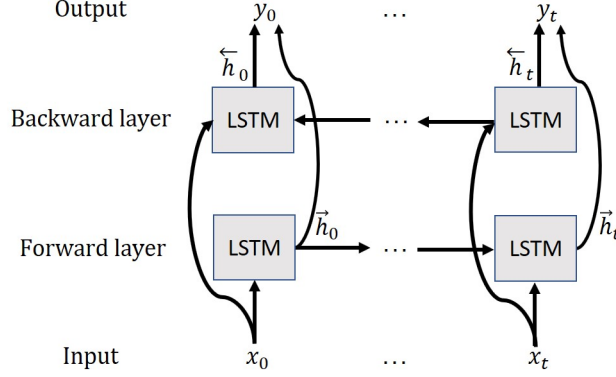

Fig. 3. BLSTM architecture with forward and backward LSTM layers.

2.4 GRU Architecture

A Gated Recurrent Unit (GRU) is a modified LSTM network. The difference is that a GRU has a reset gate and an update gate. A block of a GRU network is shown in Fig. 4. In this block, the reset gate is used to decide how much is forgotten. The reset gate is expressed by

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \tag{9}$$

where W_r and U_r are the weight matrices, and b_r is the bias vector. The update gate is given by

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \tag{10}$$

where b_z is the bias, and W_z and U_z are the weight matrices. The block output at time t is then

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h), \quad (11)$$

where W_h and U_h are the weight matrices, and b_h is the bias vector.
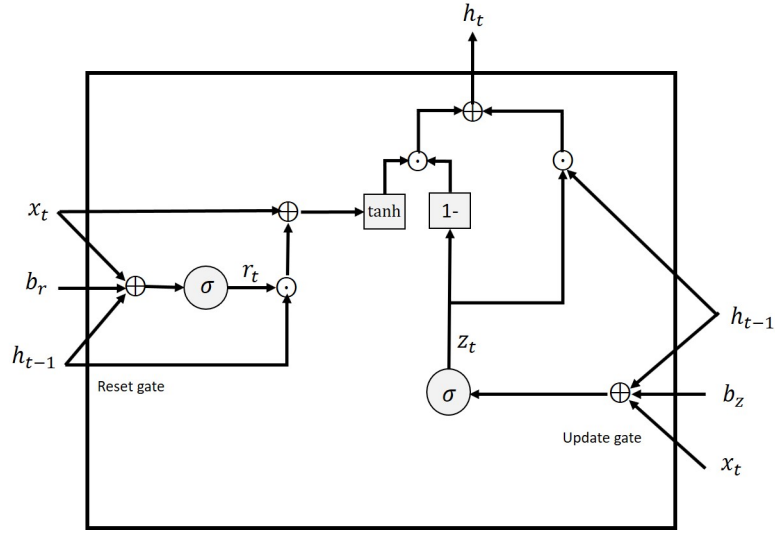

Fig. 4. A GRU block with reset gate, update gate, sigmoid and hyperbolic tangent activation functions.

2.5   Proposed Models

The first proposed model employs two stages, an Autoencoder and LSTM. First, the data set is divided into two sets, positive labeled data (normal) and negative labeled data (abnormal) to train the Autoencoder. The Autoencoder output is then fed into the LSTM network for anomaly detection and classification.

The Autoencoder has two networks (positive and negative) with three hidden layers (two encoder and one decoder layers). For the positive Autoencoder network, the positive labeled data is selected and text preprocessing such as tokenization is applied and the letters changed to lowercase. Next, sentences are truncated or padded to 40 words (100 words for the IMDB data set) and sentences containing less than 5 words are deleted. The word frequency is computed and the data set is shuffled. Then an encoder layer with 400 neurons and L1 regularizer is used, followed by an encoder layer with 200 neurons. Next, a decoder layer with 200 neurons is used and finally an output layer with the same input size is applied. This model is trained with just the positive labeled data without the labels. The categorical cross entropy loss function with ADAM optimizer is used to train this model. To prevent overfitting, dropout with probability 0.8 between each layer is employed and early stopping is used. The batch size is 128 and the maximum number of training epochs is 100.

After training, the positive Autoencoder output (which is the same size as the input) is labeled as positive. The negative Autoencoder network has the same architecture as the positive network, the only difference is that negative labeled data is used. After training, the output is labeled as negative. Now the two outputs are concatenated to obtain a single labeled data set for anomaly detection and binary classification. Duplicates are removed and Gaussian noise with zero mean and variance 0.1 is added to avoid overfitting [28].

The LSTM network is now used for anomaly detection and classification. This network has a single hidden layer. First, the concatenated data set is divided into testing and training sets with 95% for

testing and 5% for training, and these sets are shuffled. The training set is then divided into two sets with 5% for training and 95% for training validation (except for the IMDB and Openstack data sets with 85% for training and 15% for validation). This is input to a Keras[7] embedding layer which converts each element to a vector. A hidden LSTM layer of size 100 is used to classify the data into two labels using softmax activation in the final layer. Categorical cross entropy is used as the loss function and the ADAM optimizer is applied. To prevent overfitting, dropout is used in each LSTM layer with probability 0.8 and early stopping is applied. 10-fold cross validation is used in training with a maximum of 100 epochs and batch size 128.

This model is called Auto-LSTM. Replacing the LSTM network in this model with the BLSTM and GRU networks gives the Auto-BLSTM and Auto-GRU models, respectively. All three models have the same hyperparameters as described above. The Auto-GRU architecture with two Autoencoder networks and a GRU network for anomaly detection and classification is shown in Fig. 5.

3   Results

In this section, the Auto-LSTM, Auto-BLSTM and Auto-GRU models are evaluated using four data sets, namely BGL, IMDB, Openstack and Thunderbird. The following four criteria are used to evaluate the performance: accuracy, precision, recall and F-measure. Accuracy is the fraction of input data that are correctly predicted and
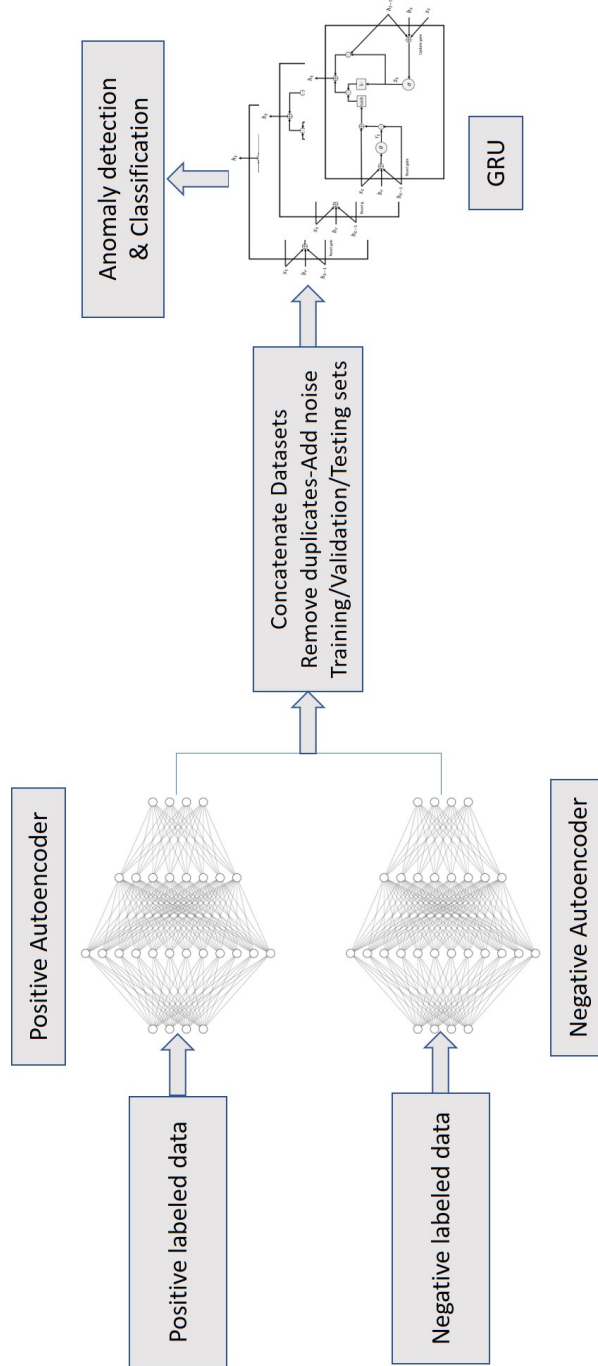
---

[7] https://github.com/keras-team/keras

Fig. 5. The Auto-GRU architecture with two Autoencoder networks and a GRU network for anomaly detection and classification.

is given by

$$A = \frac{T_p + T_n}{T_p + T_n + F_p + F_n},$$

(12)

where T_p is the number of positive instances predicted by the model to be positive, T_n is the number of negative instances predicted to be negative, F_p is number of negative instances predicted to be positive and F_n is number of positive instances predicted to be negative. Precision is given by

$$P = \frac{T_p}{T_p + F_p},$$

(13)

and recall is expressed as

$$R = \frac{T_p}{T_p + F_n}.$$

(14)

The F-measure is the harmonic mean of recall and precision

$$F = \frac{2 \times P \times R}{P + R}.$$

(15)

All experiments were executed on the Compute Canada Graham cluster with 32 CPU cores, 124 GB memory and two P100 GPUs with Python in Keras. The default hyperparameters were used for all data sets so they are not tuned for the proposed models. Tables 1 to 3 give the results for the BGL, IMDB, Openstack and Thunderbird data sets with the Auto-LSTM, Auto-BLSTM and Auto-GRU models, respectively. For each data set, the average training accuracy, average validation accuracy, average training loss, testing accuracy, precision, recall and F-measure are shown.

3.1 BGL

From the BlueGene/L (BGL) data set, 4,399,503 positive logs and 348,460 negative logs were obtained from the Autoencoder. Of these,

11,869 log were used for training, 225,529 for validation and the remaining 4,510,565 for testing.

With the Auto-LSTM model, the average training accuracy is 98.2% and validation is 98.4% with a standard deviation of 0.01 in 10-fold cross validation. The average training loss is 0.09 with a standard deviation of 0.01. The testing accuracy is 99.2% with 98.0% precision for the negative logs and 99.3% for the positive logs, and recall of 91.3% and 99.8% for negative and positive logs, respectively. The F-measure is 94.5% and 99.5% for negative and positive logs, respectively.

With the Auto-BLSTM model, the average training accuracy is 98.4% and validation is 98.7% with a standard deviation of 0.01 in 10-fold cross validation. The average training loss is 0.07 with a standard deviation of 0.01. The testing accuracy is 99.3% with precision of 98.9% for negative logs and 99.4% for positive logs, and recall of 92.1% and 99.9% for negative and positive logs, respectively. The F-measure is 95.4% and 99.6% for negative and positive logs, respectively.

With the Auto-GRU model, the average training accuracy is 97.8% and validation is 98.6% with standard deviations of 0.02 and 0.01, respectively, in 10-fold cross validation. The average training loss is 0.07 with a standard deviation of 0.01. The testing accuracy is 99.3% with 98.9% precision for negative logs and 99.3% for positive logs, and recall of 91.6% and 99.9% for negative and positive logs, respectively. The F-measure is 95.1% and 99.6% for negative and positive logs, respectively.

The testing accuracy is 99.2% for Auto-LSTM and 99.3% for both Auto-BLSTM and Auto-GRU which are better than the 98% accuracy with the LogSig algorithm [12]. The average precision, recall and F-measure for Auto-LSTM are 98.7%, 95.6% and 97%, respectively, for AutoLSTM, 99.2%, 96.0% and 97.5%, respectively, for Auto-BLSTM, and 99.1%, 95.8% and 97.4%, respectively, for Auto-GRU. The accuracy is also better than with SVM supervised learning which provided precision, recall and F-measure of 99%, 75% and 85%, respectively [13].

3.2 IMDB

The IMDB data set consists of 50,000 movie review sentences, with equal numbers that are positive and negative. The Autoencoder reduced this to 49,565 sentences with 24,875 positive and 24,690 negative. From these, 2,106 sentences were used for training, 372 for validation and the remaining 47,087 for testing.

With the Auto-LSTM model, the average training accuracy and average validation accuracy are 97.2% and 97.9%, respectively, with standard deviations of 0.01 and 0.02, respectively, in 10-fold cross validation. The average training loss is 0.08 with a standard deviation of 0.03. The testing accuracy is 97.8% with a precision of 97.9% for negative labels and 97.7% for positive labels, and recall of 97.7% and 97.9% for negative and positive labels, respectively. The F-measure is 97.8% for both labels.

With the Auto-BLSTM model, the average training accuracy and average validation accuracy are 98.3% and 99.1%, respectively, with a standard deviation of 0.01 in 10-fold cross validation. The aver-

age training loss is 0.05 with standard deviation 0.01. The testing accuracy is 98.8% with a precision of 99.6% for negative labels and 98.0% for positive labels, and recall of 98.0% and 99.6% for negative and positive labels, respectively. The F-measure is 98.8% and 98.8% for the negative and positive labels, respectively.

With the Auto-GRU model, the average training accuracy and average validation accuracy is 97.5% and 98.4%, respectively, with a standard deviation of 0.02 in 10-fold cross validation in the training. The average training loss is 0.08 with a standard deviation of 0.04. The testing accuracy is 98.8% with a precision of 99.8% for negative labels and 97.9% for positive labels, and recall of 97.8% and 99.8% for negative and positive labels, respectively. The F-measure is 98.8% and 98.8% for the negative and positive labels, respectively. The accuracy is better than the 93.2% with a Convolutional Recurrent Network which is the combination of a Convolutional Neural Network and a Recurrent Neural Network [11]. The accuracy is also better than the 94.5% with a Paragraph Vector and 89.1% for a simple LSTM network [16].

3.3   Openstack

For the Openstack data set, 137,074 positive log messages and 18,434 negative log messages were obtained from the Autoencoder. From these, 6,608 messages were used for training, 1,167 for validation and the remaining 147,733 logs for testing.

With the Auto-LSTM model, the average training accuracy is 98.5% and average validation accuracy is 98.6% with a standard deviation of 0.01 in 10-fold cross validation. The average training

loss is 0.05 with a standard deviation 0.01. The testing accuracy is 99.1% with precision of 99.4% for negative logs and 99.0% for positive logs, and recall of 92.8% and 99.9% for negative and positive logs, respectively. The F-measure is 96.0% and 99.5% for negative and positive logs, respectively.

With the Auto-BLSTM model, the average training accuracy is 98.5% and the average validation accuracy is 99.4% with a standard deviation of 0.01 in 10-fold cross validation. The average training loss is 0.05 with a standard deviation of 0.02. The testing accuracy is 99.4% with precision of 99.6% for negative logs and 99.3% for positive logs, and recall of 95.2% and 99.9% for negative and positive logs, respectively. The F-measure is 97.3% and 99.6% for negative and positive logs, respectively.

With the Auto-GRU model, the average training accuracy is 98.4% and the average validation accuracy is 97.2% with a standard deviation of 0.01 in 10-fold cross validation. The average training loss is 0.05 with a standard deviation of 0.01. The testing accuracy is 98.3% with precision of 97.9% for negative logs and 98.3% for positive logs, and recall of 87.1% and 99.8% for the negative and positive logs, respectively. The F-measure is 92.2% and 99.0% for negative and positive logs, respectively.

The accuracy is better than the 87.1% with the IPLoM algorithm [47]. The average precision and F-measure of 99.2% and 97.8% for Auto-LSTM, 99.5% and 98.5% for Auto-BLSTM and, 98.1% and 95.6% for Auto-GRU, respectively, are similar to the 94%, 99% and 97% with the Deeplog network [5]. However, the average precision

and F-measure accuracy with Auto-LSTM and Auto-BLSTM are better.

3.4   Thunderbird

For the Thunderbird data set, 3,000,000 positive log messages and 3,248,239 negative log messages were obtained from the Autoencoder output. From these, 15,620 messages were used for training, 296,791 for validation and the remaining 5,935,828 for testing.

With the Auto-LSTM model, the average training accuracy is 97.3% and the average validation accuracy is 98.9% with a standard deviation of 0.01 in 10-fold cross validation. The average training loss is 0.09 with a standard deviation of 0.02. The testing accuracy is 99.0% with precision of 98.4% and 99.8% and recall of 99.8% and 98.2% for negative and positive logs, respectively. The F-measure is 99.1% and 99.0% for negative and positive logs, respectively.

With the Auto-BLSTM model, the average training accuracy is 98.0% and the average validation accuracy is 99.6% with a standard deviation of 0.01 in 10-fold cross validation. The average training loss is 0.07 with a standard deviation of 0.04. The testing accuracy is 99.4% with precision of 99.0% and 99.8% and recall of 99.9% and 98.9% for negative and positive logs, respectively. The F-measure is 99.4% and 99.3% for negative and positive logs, respectively.

With the Auto-GRU model, the average training accuracy is 97.5% and the average validation accuracy is 99.3% with a standard deviation of 0.01 in 10-fold cross validation. The average training loss is 0.08 with a standard deviation of 0.02. The testing accuracy is 99.2% with precision of 98.7% and 99.9% and recall of 99.9% and

98.5% for negative and positive logs, respectively. The F-measure is 99.3% and 99.2% for negative and positive logs, respectively. The accuracy is better than the 95.5% with the Drain algorithm [47].

3.5   Discussion

The proposed models provided good results for all four data sets. The results for the Auto-BLSTM model were slightly better than for the Auto-LSTM and Auto-GRU models except for the IMDB data set which was the same as with Auto-GRU. Further, Auto-GRU was slightly better than Auto-LSTM except for the Openstack data set. In the proposed models, very little training data (less than 1 %) was used whereas deep learning algorithms typically need significant data for learning. This was because the input data was trained and features extracted so the LSTM (BLSTM and GRU) network needed to learn less data during training. Although the focus here was on log anomaly detection and classification, the proposed models were evaluated using the IMDB data set to show that they can also provide good results on other text classification tasks such as sentiment classification. Hyperparameter tuning such as the learning rate, network structure and number of hidden layers should improve the results obtained.

Training the Autoencoder network is the most important stage. With text data, words can be converted to digits using methods such as word frequency, Bag-of-Words or TF-IDF. However, the relationship between the data with these methods is not the most suitable for the network. In recent years, embedding methods such as Word2vec [26] and GloVe [31] have been proposed to deal with

Table 1. Auto-LSTM results for the BGL, IMDB, Openstack and Thunderbird data sets (the numbers in parenthesis are standard deviation). Positive labels are denoted by 1 and negative labels by 0.

Data set	Average Training Accuracy	Average Validation Accuracy	Average Training Loss	Testing Accuracy	Label	Precision	Recall	F-measure
BGL	98.2%	98.4%	0.09		0	98.0%	91.3%	94.5%
	(0.01)	(0.01)	(0.01)	99.2%	1	99.3%	99.8%	99.5%
IMDB	97.2%	97.9%	0.08		0	97.9%	97.7%	97.8%
	(0.01)	(0.02)	(0.03)	97.8%	1	97.7%	97.9%	97.8%
Openstack	98.5%	98.6%	0.05		0	99.4%	92.8%	96.0%
	(0.01)	(0.01)	(0.01)	99.1%	1	99.0%	99.9%	99.5%
Thunderbird	97.3%	98.9%	0.09		0	98.4%	99.8%	99.1%
	(0.01)	(0.01)	(0.02)	99.0%	1	99.8%	98.2%	99.0%

Table 2. Auto-BLSTM results for the BGL, IMDB, Openstack and Thunderbird data sets (the numbers in parenthesis are standard deviation). Positive labels are denoted by 1 and negative labels by 0.

Data set	Average Training Accuracy	Average Validation Accuracy	Average Training Loss	Testing Accuracy	Label	Precision	Recall	F-measure
BGL	98.4%	98.7%	0.07	99.3%	0	98.9%	92.1%	95.4%
	(0.01)	(0.01)	(0.01)		1	99.4%	99.9%	99.6%
IMDB	98.3%	99.1%	0.05	98.8%	0	99.6%	98.0%	98.8%
	(0.01)	(0.01)	(0.01)		1	98.0%	99.6%	98.8%
Openstack	98.5%	99.4%	0.05%	99.4%	0	99.6%	95.2%	97.3%
	(0.01)	(0.01)	(0.02)		1	99.3%	99.9%	99.6%
Thunderbird	98.0%	99.6%	0.07	99.4%	0	99.0%	99.9%	99.4%
	(0.01)	(0.01)	(0.04)		1	99.8%	98.9%	99.3%

Table 3. Auto-GRU results for the BGL, IMDB, Openstack and Thunderbird data sets (the numbers in parenthesis are standard deviation). Positive labels are denoted by 1 and negative labels by 0.

Data set	Average Training Accuracy	Average Validation Accuracy	Average Training Loss	Testing Accuracy	Label	Precision	Recall	F-measure
BGL	97.8%	98.6%	0.07	99.3%	0	98.9%	91.6%	95.1%
	(0.02)	(0.01)	(0.01)		1	99.3%	99.9%	99.6%
IMDB	97.5%	98.4%	0.08	98.8%	0	99.8%	97.8%	98.8%
	(0.02)	(0.02)	(0.04)		1	97.9%	99.8%	98.8%
Openstack	98.4%	97.2%	0.05	98.3%	0	97.9%	87.1%	92.2%
	(0.01)	(0.01)	(0.01)		1	98.3%	99.8%	99.0%
Thunderbird	97.5%	99.3%	0.08	99.2%	0	98.7%	99.9%	99.3%
	(0.01)	(0.01)	(0.02)		1	99.9%	98.5%	99.2%

the relationship between words. In the proposed models, the original data is converted from text to digits using the word frequency. Then the Autoencoder is trained and information extracted using this converted data as input which provides a better relationship with the original data for machine learning algorithms. This trained data is then used as input to the embedding method for algorithms such as LSTM, BLSTM and GRU networks. These proposed models are particularly suited to log messages because these messages contain unstructured data which can be challenging for anomaly detection and classification algorithms.

4   Conclusions

Anomaly detection and classification of log messages is a very important task in Machine Learning. In this paper, the Auto-LSTM, Auto-BLSTM and Auto-GRU models were proposed for this purpose. The first stage of these models employs an Autoencoder network to extract information and features from the input data and the second stage is anomaly detection and classification with an LSTM, BLSTM or GRU network. The proposed models were tested on four different data sets. Three of these data sets are log messages for anomaly detection and classification while the fourth is a sentiment movie review classification data set. The models were shown to provide good results for these data sets with only a very small portion used for training. This indicates that these models can be used for tasks other than log message anomaly detection and classification such as text classification.

Future research can consider the effect of hyperparameter tuning on the proposed models. Further, algorithms such as a Convolutional Neural Network (CNN) can be used for log message anomaly detection and classification.

References

1. Chavan, V., Malage, A., Mehrotra, K., Gupta, M.K.: Printed Text Recognition using BLSTM and MDLSTM for Indian Languages. In: International Conference on Image Information Processing. pp. 1–6 (2017). https://doi.org/10.1109/ICIIP.2017.8313738
2. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: Conference on Empirical Methods in Natural Language Processing. pp. 1724–1734 (10 2014). https://doi.org/10.3115/v1/D14-1179
3. Ding, R., Zhou, H., Lou, J.G., Zhang, H., Lin, Q., Fu, Q., Zhang, D., Xie, T.: Log2: A Cost-Aware Logging Mechanism for Performance Diagnosis. In: USENIX Annual Technical Conference. pp. 139–150 (2015), https://www.usenix.org/node/190487
4. Donahue, J., Hendricks, L.A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., Darrell, T.: Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(4), 677–691 (4 2017). https://doi.org/10.1109/TPAMI.2016.2599174
5. Du, M., Li, F., Zheng, G., Srikumar, V.: DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In: ACM Conference on Computer and Communications Security. pp. 1285–1298 (2017). https://doi.org/10.1145/3133956.3134015
6. Fu, Q., Lou, J.G., Wang, Y., Li, J.: Execution Anomaly Detection in Distributed Systems Through Unstructured Log Analysis. In: IEEE International Conference on Data Mining. pp. 149–158 (2009). https://doi.org/10.1109/ICDM.2009.60
7. Fu, X., Ren, R., Zhan, J., Zhou, W., Jia, Z., Lu, G.: LogMaster: Mining Event Correlations in Logs of Large-Scale Cluster Systems. In: IEEE Symposium on Reliable Distributed Systems. pp. 71–80 (10 2012). https://doi.org/10.1109/SRDS.2012.40

8. Graves, A.: Supervised Sequence Labelling with Recurrent Neural Networks. Studies in Computational Intelligence, Springer (2012). https://doi.org/10.1007/978-3-642-24797-2, https://www.springer.com/gp/book/9783642247965

9. Graves, A., Fernández, S., Schmidhuber, J.: Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) Artificial Neural Networks: Formal Models and Their Applications, Lecture Notes in Computer Science. vol. 3697, pp. 799–804 (2005)

10. Graves, A., Schmidhuber, J.: Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. Neural Networks **18**(5), 602–610 (2005). https://doi.org/10.1016/j.neunet.2005.06.042

11. Hassan, A., Mahmood, A.: Convolutional Recurrent Deep Learning Model for Sentence Classification. IEEE Access **6**, 13949–13957 (2018). https://doi.org/10.1109/ACCESS.2018.2814818

12. He, P., Zhu, J., He, S., Li, J., Lyu, M.R.: An Evaluation Study on Log Parsing and Its Use in Log Mining. In: IEEE/IFIP International Conference on Dependable Systems and Networks. pp. 654–661 (6 2016). https://doi.org/10.1109/DSN.2016.66

13. He, S., Zhu, J., He, P., Lyu, M.R.: Experience Report: System Log Analysis for Anomaly Detection. In: IEEE International Symposium on Software Reliability Engineering. pp. 207–218 (10 2016). https://doi.org/10.1109/ISSRE.2016.21

14. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In: International Conference on Learning Representations (2017)

15. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Comput. **9**(8), 1735–1780 (11 1997). https://doi.org/10.1162/neco.1997.9.8.1735

16. Hong, J., Fang, M.Y.S.: Analysis with Deeply Learned Distributed Representations of Variable Length Texts. Technical report, Stanford University (2015)

17. Irie, K., Tüske, Z., Alkhouli, T., Schlüter, R., Ney, H.: LSTM, GRU, Highway and a Bit of Attention: An Empirical Overview for Language Modeling in Speech Recognition. In: Interspeech. pp. 3519–3523 (2016). https://doi.org/10.21437/Interspeech.2016-491

18. Kc, K., Gu, X.: ELT: Efficient Log-based Troubleshooting System for Cloud Computing Infrastructures. In: IEEE International Symposium on Reliable Distributed Systems. pp. 11–20 (10 2011). https://doi.org/10.1109/SRDS.2011.11

19. Kingma, D.P., Welling, M.: Auto-Encoding Variational Bayes (12 2013), http://arxiv.org/abs/1312.6114

20. LeCun, Y., Bengio, Y., Hinton, G.: Deep Learning. Nature **521**(7553), 436–444 (5 2015). https://doi.org/10.1038/nature14539

21. Lin, Q., Zhang, H., Lou, J., Zhang, Y., Chen, X.: Log Clustering Based Problem Identification for Online Service Systems. In: IEEE/ACM International Conference on Software Engineering. pp. 102–111 (5 2016)

22. Luong, M.T., Sutskever, I., Le, Q., Vinyals, O., Zaremba, W.: Addressing the Rare Word Problem in Neural Machine Translation. In: Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing. vol. 1, pp. 11–19 (7 2015). https://doi.org/10.3115/v1/P15-1002

23. Marchi, E., Ferroni, G., Eyben, F., Gabrielli, L., Squartini, S., Schuller, B.: Multi-Resolution Linear Prediction Based Features for Audio Onset Detection with Bidirectional LSTM Neural Networks. In: IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 2164–2168 (5 2014). https://doi.org/10.1109/ICASSP.2014.6853982

24. Mescheder, L., Nowozin, S., Geiger, A.: Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks (2017), http://arxiv.org/abs/1701.04722

25. Mi, H., Wang, H., Zhou, Y., Lyu, M.R., Cai, H.: Toward Fine-Grained, Unsupervised, Scalable Performance Diagnosis for Production Cloud Computing Systems. IEEE Transactions on Parallel and Distributed Systems **24**(6), 1245–1255 (6 2013). https://doi.org/10.1109/TPDS.2013.21

26. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space (1 2013), http://arxiv.org/abs/1301.3781

27. Nagaraj, K., Killian, C., Neville, J.: Structured Comparative Analysis of Systems Logs to Diagnose Performance Problems. In: USENIX Conference on Networked Systems Design and Implementation. pp. 26–26 (2012), http://dl.acm.org/citation.cfm?id=2228298.2228334

28. Noh, H., You, T., Mun, J., Han, B.: Regularizing Deep Neural Networks by Noise: Its Interpretation and Optimization. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30, pp. 5109–5118 (2017)

29. Otte, S., Krechel, D., Liwicki, M., Dengel, A.: Local Feature Based Online Mode Detection with Recurrent Neural Networks. In: International Con-

ference on Frontiers in Handwriting Recognition. pp. 533–537 (9 2012). https://doi.org/10.1109/ICFHR.2012.229

30. Pecchia, A., Cotroneo, D., Kalbarczyk, Z., Iyer, R.K.: Improving Log-based Field Failure Data Analysis of Multi-Node Computing Systems. In: IEEE/IFIP International Conference on Dependable Systems Networks. pp. 97–108 (6 2011). https://doi.org/10.1109/DSN.2011.5958210

31. Pennington, J., Socher, R., Manning, C.D.: Glove: Global Vectors for Word Representation. In: Conference on Empirical Methods in Natural Language Processing. pp. 1532–1543 (10 2014). https://doi.org/10.3115/v1/D14-1162

32. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In: International Conference on Machine Learning. vol. 32, pp. II–1278–II–1286 (2014), http://dl.acm.org/citation.cfm?id=3044805.3045035

33. Rifai, S., Vincent, P., Muller, X., Glorot, X., Bengio, Y.: Contractive Autoencoders: Explicit Invariance During Feature Extraction. In: International Conference on Machine Learning. pp. 833–840 (2011), http://dl.acm.org/citation.cfm?id=3104482.3104587

34. Roberts, A., Engel, J., Raffel, C., Hawthorne, C., Eck, D.: A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music (2018), http://arxiv.org/abs/1803.05428

35. Roy, S., König, A.C., Dvorkin, I., Kumar, M.: PerfAugur: Robust Diagnostics for Performance Anomalies in Cloud Services. In: IEEE International Conference on Data Engineering. pp. 1167–1178 (4 2015). https://doi.org/10.1109/ICDE.2015.7113365

36. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1. pp. 318–362. MIT Press (1986), http://dl.acm.org/citation.cfm?id=104279.104293

37. Sak, H., Senior, A., Beaufays, F.: Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition (2014), http://arxiv.org/abs/1402.1128

38. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and Composing Robust Features with Denoising Autoencoders. In: International Conference on Machine Learning. pp. 1096–1103 (2008). https://doi.org/10.1145/1390156.1390294

39. Wöllmer, M., Metallinou, A., Eyben, F., Schuller, B., Narayanan, S.: Context-Sensitive Multimodal Emotion Recognition from Speech and Facial Expression Using Bidirectional LSTM Modeling. In: Interspeech. pp. 2362–2365 (2010)

40. Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.I.: Detecting Large-scale System Problems by Mining Console Logs. In: ACM Symposium on Operating Systems Principles. pp. 117–132 (2009). https://doi.org/10.1145/1629575.1629587

41. Yen, T.F., Oprea, A., Onarlioglu, K., Leetham, T., Robertson, W., Juels, A., Kirda, E.: Beehive: Large-scale Log Analysis for Detecting Suspicious Activity in Enterprise Networks. In: Annual Computer Security Applications Conference. pp. 199–208 (2013). https://doi.org/10.1145/2523649.2523670

42. Yu, X., Joshi, P., Xu, J., Jin, G., Zhang, H., Jiang, G.: CloudSeer: Workflow Monitoring of Cloud Infrastructures via Interleaved Logs. In: ACM International Conference on Architectural Support for Programming Languages and Operating Systems. vol. 44, pp. 489–502 (2016). https://doi.org/10.1145/2872362.2872407

43. Yuan, D., Mai, H., Xiong, W., Tan, L., Zhou, Y., Pasupathy, S.: SherLog: Error Diagnosis by Connecting Clues from Run-time Logs. In: Architectural Support for Programming Languages and Operating Systems. pp. 143–154 (2010). https://doi.org/10.1145/1736020.1736038

44. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent Neural Network Regularization (9 2014), http://arxiv.org/abs/1409.2329

45. Zhang, K., Xu, J., Min, M.R., Jiang, G., Pelechrinis, K., Zhang, H.: Automated IT System Failure Prediction: A Deep Learning Approach. In: IEEE International Conference on Big Data. pp. 1291–1300 (12 2016). https://doi.org/10.1109/BigData.2016.7840733

46. Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., Xu, B.: Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling. In: International Conference on Computational Linguistics. pp. 3485–3495 (2016)

47. Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., Lyu, M.R.: Tools and Benchmarks for Automated Log Parsing. In: International Conference on Software Engineering: Software Engineering in Practice. pp. 121–130 (2019). https://doi.org/10.1109/ICSE-SEIP.2019.00021

48. Zou, D.Q., Qin, H., Jin, H.: UiLog: Improving Log-Based Fault Diagnosis by Log Analysis. Journal of Computer Science and Technology **31**(5), 1038–1052 (09 2016). https://doi.org/10.1007/s11390-016-1678-7