Dynamic RNN-CNN based Malware Classifier for Deep Learning Algorithm

Youngbok Cho
dept. Ioformation Security
Daejeon University
Daejeon, South Korea
lotuscho73@gmail.com

***Abstract*—This study proposes a malware classification model that can handle arbitrary length input data using the Microsoft Malware Classification Challenge dataset. We are based on imaging existing data from malware. The proposed model generates a lot of images when malware data is large, and generates a small image of small data. The generated image is learned as time series data by Dynamic RNN. The output value of the RNN is classified into malware by using only the highest weighted output by applying the Attention technique, and learning the RNN output value by Residual CNN again. Experiments on the proposed model showed a Micro-average F1 score of 92% in the validation data set. Experimental results show that the performance of a model capable of learning and classifying arbitrary length data can be verified without special feature extraction and dimension reduction.**

Keywords—convolution neural network, malware, deep learning, recurrent neural network, kaggle data

I. INTRODUCTION

Malware has also evolved rapidly as the Internet evolved and data transfer and processing accelerated. Conventional malware detection or classification is accomplished through file signatures. But it's a shame that the file signature method isn't fast enough to deal with the spilled variants of malware. Therefore, a lot of researches have recently been conducted on how to detect and classify malware using machine learning / deep learning. In 2015, Microsoft Malware Dataset was unveiled at Kaggle, a competition for machine learning and deep learning data analysis, and a malware classification competition was used [1,2,3]. In addition, about 13,000,000 new malicious codes are created every month, and the number of new malicious codes increases over time. The correct classification of malware families must be preceded and appropriate defenses should be provided in order to provide correct responses to the various types of malicious programs that are poured out every day. To this end, various methods for classifying malware families are being studied. In order to improve the classification performance, it is necessary to understand the characteristics of malware that classify families. It is important to find out the unique characteristics of malicious codes when classifying malicious code families by finding unique characteristics of each malicious code, but it is also important to select the classification algorithm used as classifier correctly.

Recently, the most actively researched field of classification or recognition technique is the research on deep neural network (DNN) called deep neural network made by increasing the number of hidden layers of neural network. In particular, in the field of image and speech recognition, deep neural network-based models record excellent performance, and there are movements to use it in other fields. Malware analysis is one such field.

The top teams in the competition recorded high accuracy by converting malware into fixed images and classifying them into CNN models. However, since malware does not have a fixed size, the process of converting the malware to a fixed image necessarily requires the reduction or expansion of data, which can lead to unintended potential data loss [4, 5]. Depending on how you scale down and scale up, additional time is required, and the task is relatively complex. To overcome these drawbacks, this paper proposes a model that can learn and classify malware data of various sizes without special data reduction.

II. RELATED WORKS

Researches have been conducted to train the machine learning / deep learning model by extracting the calling API, API call sequence and image of the executable from the Windows NT standard operating system [3,7,8]. The model presented in this paper is based on the imaging method and uses ConvLSTM and Attention Mechanism as well as CNN.

A. Malware Imaging

Malware binaries can be visualized as grayscale images in bytes [7,8]. Many malware variants appear to be very similar in layout and texture to images of malware that belong to the same family. This study creates a feature vector by studying the texture that appears when malware is visualized. The model proposed in this paper does not need complicated feature analysis and processing while bringing the advantages of malware imaging of the study.

B. ConvLSTM

Both input and predicted objects are models designed to solve problems in space-time order. The study proposed ConvLSTM in a model that approximates the problem of predicting precipitation. The Convolution LSTM was proposed by extending the Fully Connected LSTM (FC-LSTM) to have a convolution structure in both input-state and state-state transitions. In this paper, we approach the classification problem by imaging the malware. In this paper, we will take advantage of the spatio-temporal processing of images in ConvLSTM as a corresponding aspect.

C. Attention Mechanism

Attention Mechanism is used as a means to overcome the long-term dependency problem of the RNN network.

978-1-7281-3673-8/19/$31.00 ©2019 IEEE

Attention Mechanism enables the concentration of specific output values by weighting the output values of sequence inputs [9, 10]. Attention Mechanism has proved effective in solving NLP problems. This study applies Attention Mechanism to output variable dimension data as fixed dimension data through encoder. Malware is classified according to its malicious behavior. By applying the Attention Mechanism to the output sequence of the ConvLSTM model, we assume that the behavior of the malware will occur in some code.

D. Malware coordinate-based imaging

The malware suggests a way to measure the similarity of binaries. It generates a fixed size image corresponding to each binary and classifies malicious codes using the image as feature information [10, 11]. In this study, the binary data of the malware is used as two single-byte coordinates to create a 256 x 256 fixed image. The coordinates are read one by one and the value of corresponding pixel coordinate is increased by one. This study is similar in that it can cope with the model and variable input data presented in this paper. However, the preprocessing method of reading coordinates by moving one byte requires a lot of preprocessing operations. The model proposed in this paper proposes a model that can classify malware with relatively little preprocessing.

III. DYNAMIC RNN-CNN

In this paper, the procedure for experiment is as follows. Read binary dump data of the malware data set and restore it to binary. Then, convert the binary into several 256 x 256 images, and train the model.

A. Data Set

To test the classification performance of the proposed system, we used the Microsoft Malware Classification Challenge (BIG 2015) data set [14] and VXHeaven's 2010 virus collection data set [15]. Kaggle, which shares machine learning and deep learning data and competes for shared data, shared data from the Microsoft Malware Classification for malware classification. In this study, the malware classification deep learning model was trained using this malware data. Malware training data consists of 10,868 malware analysis files. Number of training and validation data for each classes.

TABLE I. NUMBER OF TRAINING AND VALIDATION DATA FOR EACH CLASSES.

Class	Training data	Validation data
Ramnit	1231	308
Lollipop	1982	469
Kelihos_ver3	2354	588
Vundo	380	95
Simda	34	8
Tracur	601	150
Kelihos_ver1	318	80
Obfuscator.ACY	982	246
G	810	203
Total	8692	2174

The data set is divided into two formats per malware. The hexadecimal representation includes a bytes file that records malware binary bytes in a string format, and an asm file that records the meta information extracted from the binary file with the IDA disassembler. For the proposed model, we divided the training data and the verification data in an 8: 2 ratio. The model proposed in this paper is a model that receives the malware binary as an image without using the meta information of the malware. Therefore, data set in ida file format is not used. After reading the asm file and restoring it to the malware binary, we experimented with the recovered binary file. Two outlier data larger than 6MB were excluded from the data set. Table 1 shows the number of data samples for each class of the training and verification data set..

B. Simulation Environment

Programming language environment for experiment was implemented using Python 3.6.2 and Tensorflow-gpu 1.9 version was used to implement Dynamic RNN-CNN model..

TABLE II. COMPUTER HARDWARE SPECIFICATIONS FOR EXPERIMENTS

H/W	Description
CPU	Inter® Core™ i7-8100 CPU @ 3.60GHz (4 CPUs)
GPU	NVIDIA GeForce GTX 1060 6GB
Memory	8192MB

Also, Numpy 1.15.0 and pandas 0.23.4 version were used to process malware data. The specifications of the computer used in the experiment are shown in Table 2.

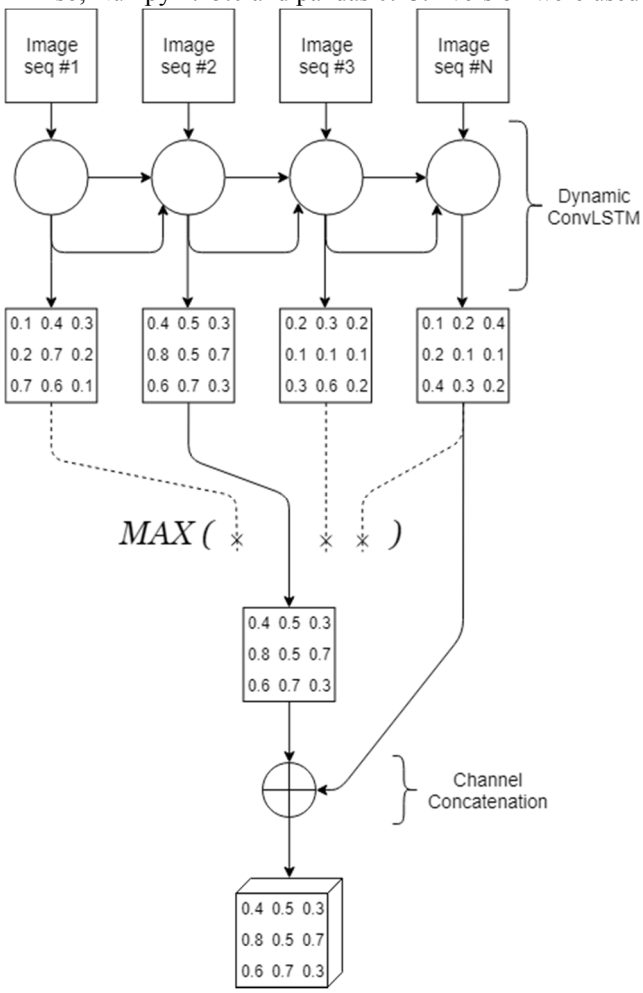

Fig. 1. Variable length encoder

C. *Encoder and Classifier of Proposed Model*

Dynamic RNN-CNN model consists of two parts. It consists of Encoder that outputs variable dimension file and Classifier which classifies malicious code with result of Encoder. Encoder is composed of Dynamic ConvLSTM as shown in Figure 1.

Convert malware binaries into multiple 256 x 256 images. Each image is entered as a time step sequence, and a large file takes a long time step. Because of the dynamic ConvLSTM operation, encoding is possible regardless of the length of the time step. The output will be as long as the input time step of ConvLSTM. The idea of the Attention mechanism is borrowed for outputting fixed dimension data regardless of the time step. The output that has the highest weight among the output results and the channel concatenation of the last output are the final outputs of the encoder.

Classifier is composed as shown in Figure 2. Classifier is composed only of Convolution layer without Fully Connected layer. The Convolution layer adds a Residual Block. The kernel's kernel size was 3 x 3, the number of filters was 16, and the last layer was set to 9.

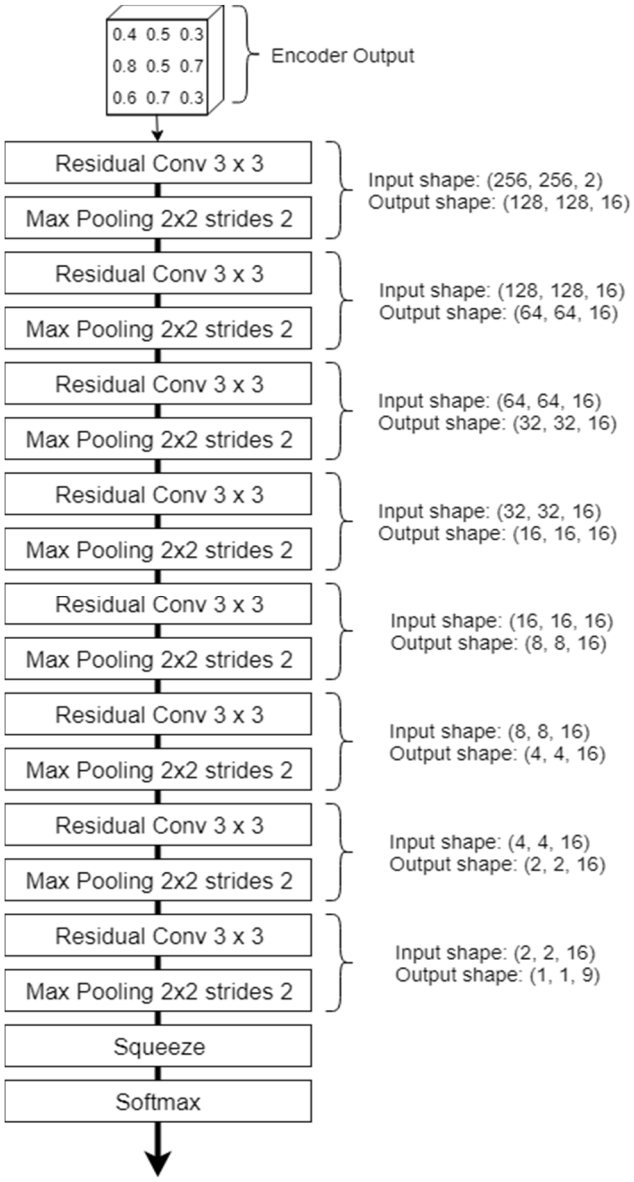

Fig. 2. Classifier with Residual Convolutional Layers

Figure 3 shows the structure of the Residual Conv layer among the classifier layers in Figure 2. Batch normalization was performed in the middle of the convolution layer, and Leaky Relu was used as the activation function.

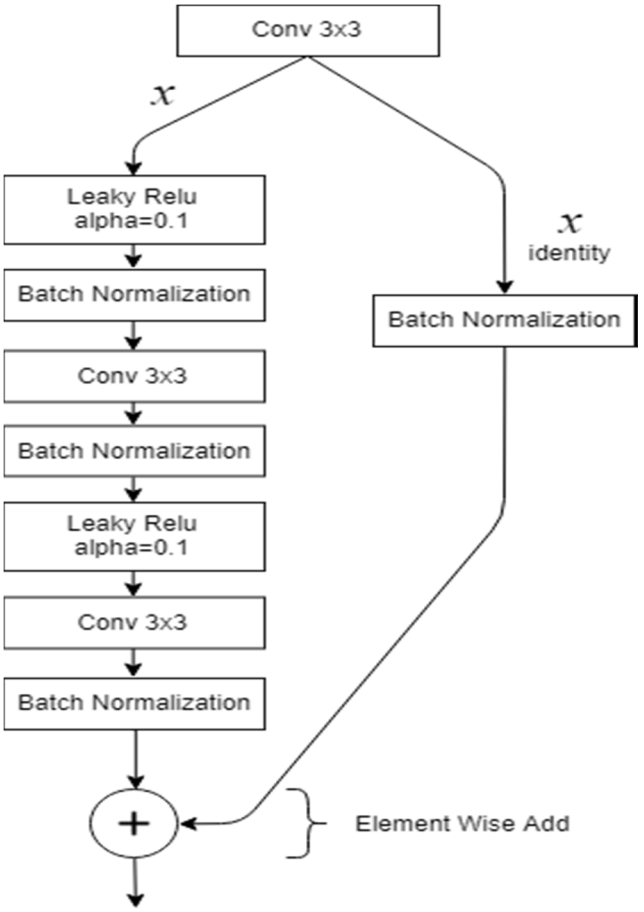

Fig. 3. Residual Conv 3x3 Layer in Fig. 2.

D. Model Training

Mini-Batch size was set to 20 and 40 epochs were learned. The calculation of the validation data set was performed every 100 steps and the accuracy of the validation data set was measured using the confusion matrix. Multiclass cross entropy was used for the loss function metric. The learning optimization algorithm of the proposed model is Adam [11] and the learning rate is 0.0001. Malware binary files were processed in Mini-Batch units due to insufficient main memory.

E. Algorithm

We used the following algorithm for the malware classifier using C ++.

1 - 1/(N*P)*(sum_{i=1}^{N} Pi), where ,

Pi = the number of passed positive sample at i^{th} negative point, P = total positive number and N = total negative number.

In order to adjust the fixed size for classifier input, the malware image of various sizes finishes the malware image conversion step by enlarging or reducing the malware image. Figure 4 shows an example of converting nine malicious code samples in the Microsoft data set into images.

```
double get_ROC_AUC(std::vector<std::pair<double,bool> >& value) {
    std::sort(value.begin(), value.end(), sort_by_class);
    std::vector<std::pair<double, double> > roc_pair;
    roc_pair.push_back(std::pair<double, double>(0, 0));
    int negative_total = 0;
    int positive_total = 0;
    std::vector<std::pair<double,bool> >::const_iterator pos;
    for (pos = value.begin(); pos != value.end(); pos++){
      if(pos->second == true) positive_total++;
      if(pos->second == false) negative_total++;
    }
    for (pos = value.begin(); pos != value.end(); pos++){
      double classifier = -1;
      if (pos + 1 == value.end()) classifier = (pos->first) + 1;
      else classifier = (pos + 1)->first;
      if (classifier - (pos->first) < 0.00000001) continue;
        double positive_count = 0;
        double negative_count = 0;
        std::vector<std::pair<double, bool> >::const_iterator snsp_pos =
        value.begin()
        for (snsp_pos; snsp_pos <= pos; snsp_pos++){
          if (snsp_pos->second == true) positive_count++;
          else if (snsp_pos->second == false) negative_count++;
        }
        roc_pair.push_back(std::pair<double,
        double>((negative_count/static_cast<double>(negative_total)),
        (positive_count/static_cast<double>(positive_total))));
    }
      double AUC = get_AUC(roc_pair);
      return AUC;
}
```

The classifier used in the malware classification model is CNN. The main operation of classifier is convolution operation. Therefore, in case of CNN that needs to learn a large amount of data, it takes less time to train the model when the convolution operation can be efficiently processed.

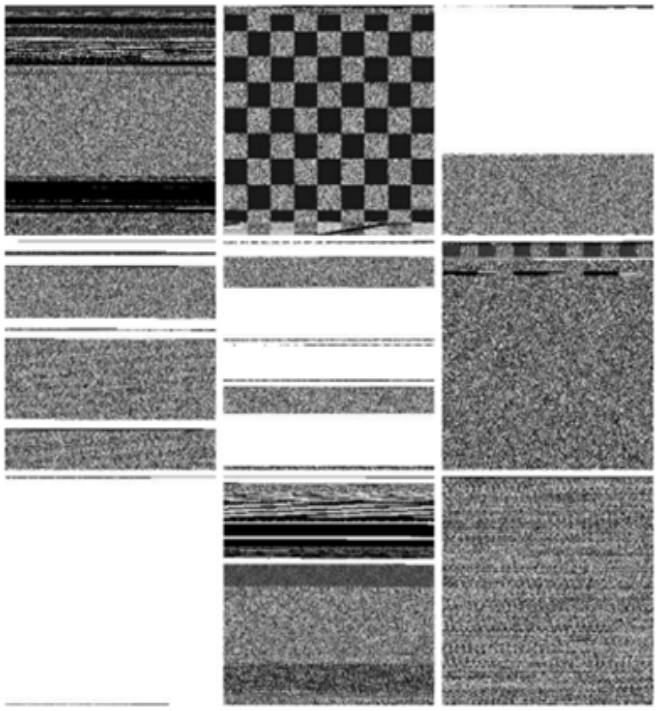

Fig. 4. Figure 1 Example Malware Image of Microsoft Dataset

IV. EXPERIMENT RESULT

It is difficult to evaluate the performance of algorithms proposed in the field of machine learning or deep learning with only classification accuracy. Therefore, in order to measure the performance of the classifier, the performance index is defined by subdividing the relationship between the predicted and actual values of the classifier using a confusion matrix called an error matrix. It can be expressed as an index. In this paper, the learning process and the results can be seen in the graph of Figure 4. In this paper, the string binary dump data of the malware data set is read and recovered to binary, and the result of the classifier that can classify the malware through learning after converting the binary into multiple images of 256 x 256 size, You can see it in the graph in Figure 5.

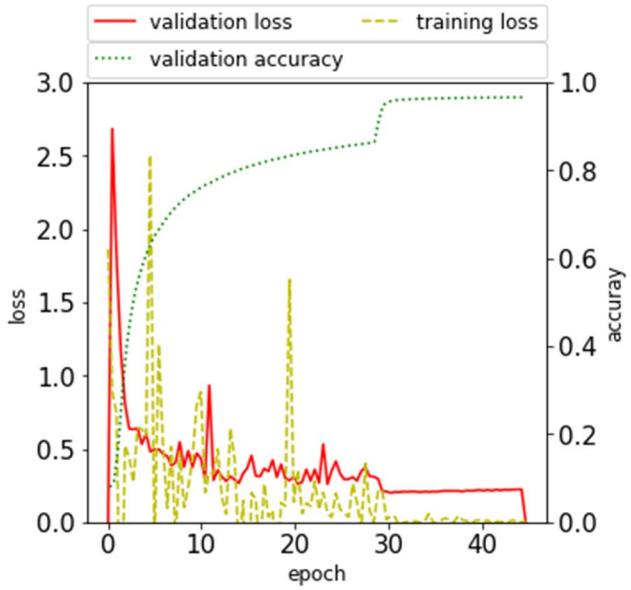

Fig. 5. Loss and accuracy graph

In addition, we can compare the model classification and label data of each class for accurate malware classification and plot it with the Confusion Matrix.

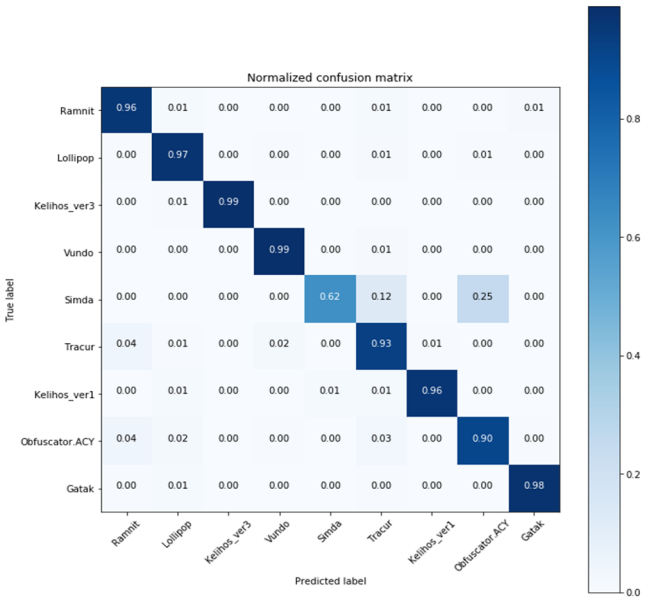

Fig. 6. Normalized confusion matrix.

As can be seen in the error matrix of Figure 6, the Simda class classification shows relatively low accuracy. This seems to be because the data of Simda class has less number than the data of other class. When evaluating the model with the Micro-average F1 score, considering the size of the unbalanced data set, 92% of results were obtained. In order to give a more accurate reliability of learning, we have plotted the ROC curves of Micro-average and Macro-average, respectively, in Figures 7 and 8. The ROC curve is used to evaluate the performance of the model and can be explained by reflecting the sensitivity and specificity of the proposed model.

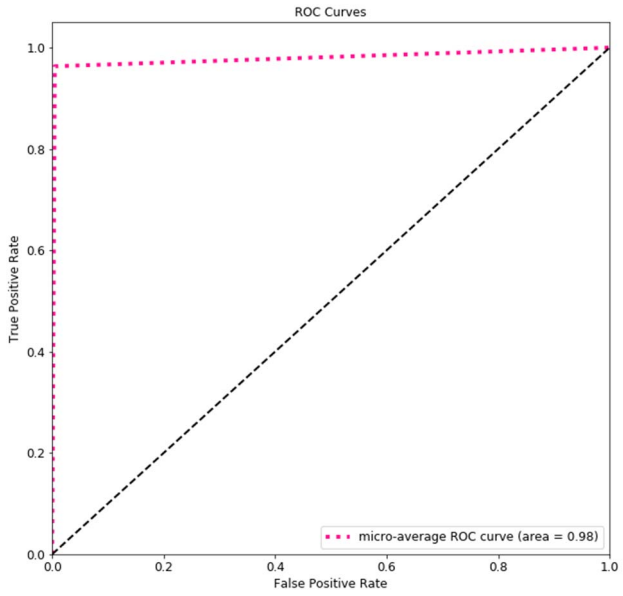

Fig. 7. Micro-average ROC curve

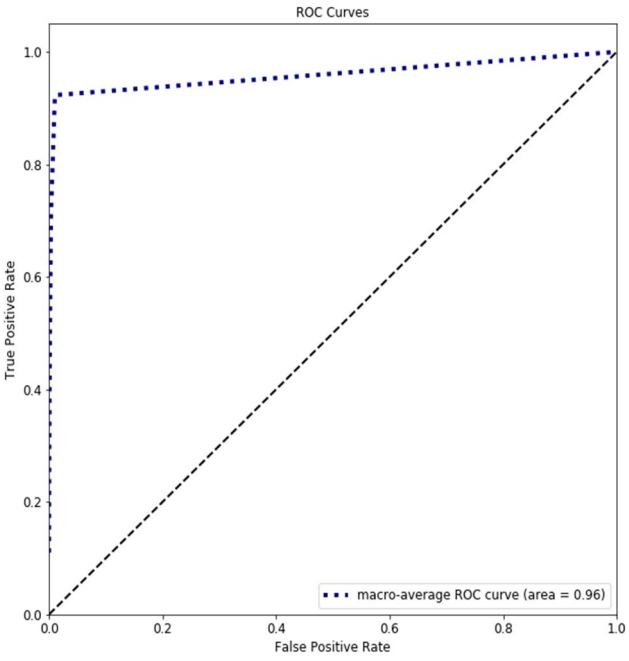

Fig. 8. Macro-average ROC curve

As shown in the results of Fig. 7 and Fig. 8, the proposed model can confirm the accuracy of the precision and recall in the malware classifier. In addition, the proposed model evaluation was demonstrated through the following program.

V. CONCLUSION

In this study, we proposed a deep learning model that can effectively classify malware files with various sizes using dynamic RNN and CNN without complex feature engineering and preprocessing that require many operations. In the existing research, there was a technique to classify families by converting malicious codes into images, but in this study, there was a disadvantage that the malware images could not be directly learned by the classifier and additional process of extracting feature points was needed. As the number of code samples increased, the time required for family classification was longer. We did not use the metadata for the malware data, but preprocessed the binary into multiple images of 256 x 256 size. Experimental results recorded 96% accuracy of the validation data set. Due to data set imbalance, the classification accuracy was observed to be relatively poor in certain classes. When evaluated by micro-average, 92% of results were obtained. Performance degradation due to data set imbalance is expected to be overcome when the number of data sets increases. Also, if you change the model with Bidirectional RNN and Inception structure, you can expect better performance. The paper proposed in this study can cope with variable input data, and it does not need complicated preprocessing for input data, but adds ConvLSTM Encoder and Residual Block, etc. Need research Using the proposed model, the learning process and results can be seen in the graph in Figure 4. The model proposed in this paper shows a high performance with 96% accuracy of verification data set.

ACKNOWLEDGMENT

The This paper was supported by the Daejeon University LINC + Project.

REFERENCES

[1] S. J. Park, S.M. Choi, H.J.Lee and J.B. Kim, “Spatial analysis using R based Deep Learning”, Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology, vol. 6, no. 4, pp. 1-8, April.2016.

[2] J.M Kim and J.H Lee, “Text Document Classification Based on Recurrent Neural Network Using Word2vec”, Journal of korean Institute of Intelligent System, vol27, no.6, pp 560-565, Jun.2017.

[3] P. Baudis, S. Stanko and J.Sedivy, “Joint Learning of Sentence Embeddings for Relevance and Entailment”, in The Workshop on Representation Learning for NLP, Berlin, Germany, pp. 18-26, 2016.

[4] J.Y. Kim and E. H. Park, "e-Learning Course Reviews Analysis based on Big Data Analytics," Journal of the Korea Institute of Information and Communication Engineering, Vol. 21, No. 2, pp. 423~428, Feb. 2017.

[5] J.M. Kim and J.H. Lee, "Text Document Classification Based on Recurrent Neural Network Using Word2vec," Journal of Korean Institute of Intelligent Systems, Vol. 27, No. 6, pp. 560~565, Dec. 2017.

[6] J. Mueller and T. Aditya "Siamese Recurrent Architectures for Learning Sentence Similarity." in Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Arizona, pp.2786~2792, 2016.

[7] Y. Kim, Y. Jernite, S. David and M. R. Alexander, “Character-Aware Neural Language Models.”, CoRR, abs/1508.06615, 2015.

[8] Naver ai hackerton 2018 Team sadang solution [Internet]. Available:https://github.com/moonbings/naver-ai-hackathon-2018.

[9] R. Dey and F. M. Salem. “Gate-variants of gated recurrent unit (GRU) neural networks.”, CoRR, abs/1701.05923, 2017.

[10] Y. B. Cho and S. H. Woo,” Automated ROI Detection in Left Hand X-ray Images using CNN and RNN”, International Journal of Grid and

[11] Distributed Computing Vol. 11, No. 7 (2018), pp.81-92

[12] wiki fast .ai Logloss [Internet]. Available: http://http://wiki.fast.ai/index.php/Log_Loss

[13] Diederik P. Kingma, Jimmy Ba, “Adam: A Method for Stochastic Optimization” in 3rd International Conference for Learning Representations, San Diego, 2015.

[14] G. Y Lim and Y. B. Cho, “The Sentence Similarity Measure Using Deep-Learning and Char2Vec. ” Journal of the Korea Institute of Information and Communication Engineering, Vol. 22, No. 10: 1300~1306, Oct. 2018

[15] C. N. Andres, S. M. Kim,” Nonlinear Compensation Using Artificial Neural Network in Radio-over-Fiber System”, Journal of information and communication convergence engineering Vol. 16, No. 1, pp. 1-5, March 2018,

[16] Y. B. Cho and S. H. Woo, “An Extraction Algorithm for Deep Characterization Using DeepCNN in Diagnostic Ultrasound Images”, 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems and 19th International Symposium on Advanced Intelligent Systems, pp. 514-517

[17] J. Kotchakorn, A. I. Ngamnij and A. I. Somjit, “Ontology Mapping and Rule-Based Inference for Learning Resource Integration”, Journal of information and communication convergence engineering, vol. 14, no. 2, (2016), pp. 97-105.

[18] Y. B. Cho, S. H Woo and S. H Lee, “Genetic lesion matching algorithm using medical image”, Journal of the Korea Institute of Information and Communication Engineering, vol. 21, no.5.pp. 960- 966.2017.