MALWARE CLASSIFICATION WITH LSTM AND GRU LANGUAGE MODELS AND A CHARACTER-LEVEL CNN

Ben Athiwaratkun[*]

Cornell University
Department of Statistical Science
301 Malott Hall
Ithaca, NY 14853

Jack W. Stokes

Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA

ABSTRACT

Malicious software, or malware, continues to be a problem for computer users, corporations, and governments. Previous research [1] has explored training file-based, malware classifiers using a two-stage approach. In the first stage, a malware language model is used to learn the feature representation which is then input to a second stage malware classifier. In Pascanu et al. [1], the language model is either a standard recurrent neural network (RNN) or an echo state network (ESN). In this work, we propose several new malware classification architectures which include a long short-term memory (LSTM) language model and a gated recurrent unit (GRU) language model. We also propose using an attention mechanism similar to [12] from the machine translation literature, in addition to temporal max pooling used in [1], as an alternative way to construct the file representation from neural features. Finally, we propose a new single-stage malware classifier based on a character-level convolutional neural network (CNN). Results show that the LSTM with temporal max pooling and logistic regression offers a 31.3% improvement in the true positive rate compared to the best system in [1] at a false positive rate of 1%.

Index Terms— Malware Classification, Neural Language Model, LSTM, GRU, CNN

1. INTRODUCTION

Microsoft Corporation receives hundreds of thousands of unknown files each day which must be classified as either malicious or benign. These files are submitted by external parties through a sample submission web site [2]. Signatures for malicious files are then either manually or automatically created thereby preventing the files from being installed and executed on a computer running the Microsoft anti-malware engine.

Recent research has explored using deep neural networks (DNNs) for malware classification [3, 4, 5, 6]. DNNs for the static analysis of malware are proposed in [4] and [6]. Similarly, [3] and [5] train DNNs for the output of malware dynamic analysis.

In [1], Pascanu et al. take a different approach to first learn a *language model* for the malware and benign files to construct the feature representation for each file in the training set. The authors propose either using a standard recurrent neural network (RNN) [7] or an echo state network (ESN) [8] as the language model. We note that the structure of an ESN is similar to that of an RNN, but the weights are randomly initialized and are not trained. A logistic regression or multi-layer perceptron (MLP) classifier is then trained based on the feature representations output by the language model. In addition, Pascanu et al. propose using temporal max pooling for both recurrent language models to combine a long sequence of temporal features which helps improve the results.

The best performing architecture in [1] is an ESN with temporal max pooling for generating the feature representation combined with a logistic regression classifier. Unfortunately, the authors found that the RNN failed to learn salient features of the files and has lower performance compared to the untrained ESN.

The main motivation for this work is to find neural architectures that yield improved performance. In particular, we use recurrent neural network architectures that can better capture long-term dependencies than the standard RNN. In this study, we revisit the malware language model architecture to investigate whether these enhanced language models lead to improved results for malware classification. In particular, we experiment with the long short-term memory (LSTM) [10] and the gated recurrent unit (GRU) [11] as language models. We compare all model variants with temporal max pooling and a recently proposed attention mechanism similar to that in [12]. Finally, we also consider whether the recently introduced character-level convolutional neural network (CHAR-CNN) can improve the malware classification performance. In Section 4, we show that the best performing system in this study employs an LSTM for the language model with temporal max pooling and a logistic regression classifier.

This study makes the following contributions. We propose several deep learning architectures for classifying malware including LSTM- and GRU-based language models and a character-level CNN. We show that the features learned from an LSTM language model help improve the performance compared to random-weight architectures, with the LSTM model and temporal max pooling outperforming other competing models.

2. DATA

Before presenting the proposed models in the next section, we first describe the dataset used in our work. A modified version of the production Microsoft anti-malware engine was used to perform dynamic analysis of 75,000 Windows portable executable (PE) format files equally split between malware and benign files. Our dataset is further randomly split into three datasets including a training set with 50,000 files, a validation set with 10,000 files, and a test set with 15,000 files.

[*]The first author worked on this project during an internship at Microsoft Research.

Before a file is allowed to be executed on a Windows computer running the Microsoft anti-malware engine, it is analyzed using lightweight file emulation by the engine. This emulation is one type of dynamic analysis and produces the sequence of system calls executed by the file. This sequence is logged by the modified anti-malware engine for training and testing our classifier.

The system calls logged by the anti-malware engine are high level events. For example, there are multiple user mode, kernel model, and other APIs (e.g. fopen() in C) that can be used to open a file. All of these different API calls are mapped to a single file open event by the anti-malware engine.

There are a total of 114 system calls which are found in our dataset. Each system call event is first converted into a stream of integers ranging from 0 to 113 where each integer is the index of an individual system call event. This sequence of integers is then input into the malware classification system which is described in the next section.

3. MODELS

In this section, we describe the proposed malware classification models. Before doing so, we first review the previously proposed models which we will use for the performance baselines in the following section on *Experimental Results*. We then present the new, two-stage malware classification models which utilize a malware language model to generate the features. Finally, we propose a single-stage, character-level convolutional neural network for malware classification.

3.1. Previously Proposed Baseline Models

Previously proposed malware classification using a language model [1] included an echo state network or a recurrent neural network with temporal max pooling in the first language model stage and logistic regression or a multi-layer perceptron (MLP) in the second classification stage.

ESN and RNN: In its original formulation [8], the ESN has the same recurrent architecture as the RNN. However the recursive matrix and input matrix weights are randomly initialized for an ESN but are learned for the RNN. The authors evaluate ten different models for the feature representations in the first stage. These models include all combinations of an ESN versus RNN for the standard model, a leaky model [13], a bi-directional model [14], a half-frame model [1], and the standard model with temporal max pooling [1]. The classifiers used in [1] include logistic regression trained with the softmax output and an MLP. The standard ESN with max pooling outperforms the RNN with max pooling, and both of these models provide better results than the other eight models. Thus, we only consider the two max pooling architectures in this study.

Temporal Max Pooling: While max pooling was previously used for sequences processed with a convolutional neural network (CNN), Pascanu et al. [1] were the first to propose it for ESNs and RNNs. In temporal max pooling, the maximum hidden unit is chosen across all hidden states when processing the input sequence. Let $\vec{h}^t$ be hidden vectors generated from a recurrent language model for time steps $t = 0, \ldots, T-1$ where each file event sequence is divided into subsequences of length T. Temporal max pooling generates a vector $\vec{h}^{max}$ where $h_i^{max} = \max_{t \in 0, T-1} h_i^t$ for index $i = 0, \ldots, D-1$, and D is the dimension of $\vec{h}^t$. We next describe the newly proposed models for malware classification.

3.2. Language Model-Based Malware Classification

The new malware language model-based classifiers are depicted in Figure 1. In the first stage, a malware language model (LM) utilizing either an LSTM or GRU is initially used to construct the features. In the second stage, these features are classified with either a single-hidden layer MLP or logistic regression with softmax.

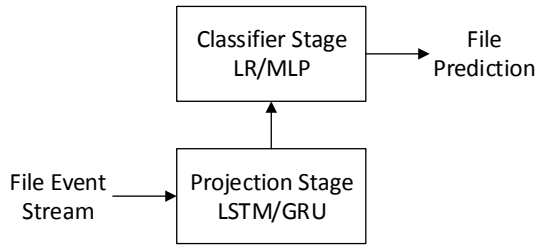

Fig. 1. Two-stage, language model-based malware classification system.

LSTM Malware Language Model: The first malware classification model we propose is to use the long short-term memory model, proposed by Hochreiter and Schmidhuber [10], as the language model. The LSTM replaces the RNN or the ESN in [1]. The LSTM is also a recurrent neural network model which includes state from the input, output and forgetting gates and has been shown to increase the memory capacity compared to the standard RNN. We follow the LSTM implementation described in [15].

GRU Malware Language Model: Similar to the LSTM, Cho et al. [11] proposed the gated recurrent unit for neural machine translation. The second new malware classification model proposed in this paper uses a GRU for the malware language modeling stage. Again, we follow the implementation of the GRU found in [15].

Attention Mechanism: Bahdanau et al. [12] recently proposed a new attention mechanism to align the input and output sentences (i.e. sequences) in the context of neural machine translation. In this work, we adapt their attention mechanism to serve as an alternative to the temporal max pooling proposed in [1]. While temporal max pooling chooses the maximum hidden unit across all of the hidden vectors in the sequence, the attention mechanism instead first learns the attention score for each time step and then computes the temporal average of all hidden vectors. The attention mechanism generates a vector $\vec{h}^{att}$ of length D where

$$\vec{h}^{att} = \sum_{t=0}^{T-1} a^t \vec{h}^t,$$

and the attention scores a^t are calculated from a dense network with $\vec{h}^t$ as an input. The final vector representation to be used for the LM classification is the concatenation of (1) the last hidden vector from the language model $\vec{h}^T$, (2) the feature vector from temporal max pooling $\vec{h}^{max}$ or the attention mechanism $\vec{h}^{att}$, and (3) the bag-of-words representation from the one-hot encoding of the input events.

Second Stage Malware Classifier: We train the second stage of the LM classifier using either logistic regression or a single-hidden layer, multi-layer perceptron (MLP). Logistic regression was the best performing classifier in [1], and the single-hidden layer MLP provided the best classification results in [3]. The MLP uses rectified linear unit (ReLU) activation functions [16].

Layer	Description
1	Conv length = 7, Max Pooling length = 3
2	Conv length = 7, Max Pooling length = 3
3	Conv length = 3
4	Conv length = 3
5	Conv length = 3
6	Conv length = 3, Max Pooling length = 3
7	Batch Normalization, Dropout = 0.5
8	Batch Normalization, Dropout = 0.5
9	Batch Normalization

Table 1. Character-level, CNN layer description.

3.3. Character-Level Convolutional Neural Network

The final model we propose in this paper is the character-level convolutional neural network (CHAR-CNN) [17] which is shown in Figure 2. Unlike the LSTM and GRU language model-based architectures, the CHAR-CNN is a single-stage malware classifier similar to the more traditional malware classifiers [18]. The CHAR-CNN employs nine layers which are summarized in Table 1. It takes a sequence of maximum length 1,014 characters where each character is an event. For sequences fewer than 1,014 characters, we pad the end with eod-of-sequence tokens. The first eight layers use the rectified linear activation function while the sigmoid activation function is employed in the final layer for binary classification. The loss function utilized by this model is the cross entropy loss function.

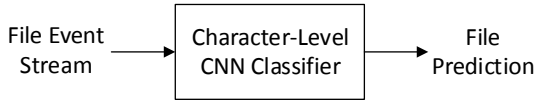

Fig. 2. Single-stage, character-level convolutional neural network malware classification model.

4. EXPERIMENTAL RESULTS

In this section, we evaluate the proposed architectures. All models are implemented using Keras [15] with the Theano [19] backend deep learning engine. We first describe the experimental setup. We then compare the receiver operating characteristic (ROC) curves for the different model variants.

Experimental Setup: To train the language models, we divide all files event sequences into smaller subsequences, with length up to $T = 50$. The maximum number of epochs is set to 15, but we employ early stopping if training additional epochs fails to decrease the cross entropy loss. We train the language models using a mini-batch size of 50. The number of hidden units for the language models (D) and the dimensions of $\vec{h}^{att}$ and $\vec{h}^{max}$ are all equal to 1,500. Since the dimension of the one-hot, event embeddings is 114, the final representation has dimension 3,114.

We next describe the hyperparameters used for the logistic regression and MLP LM classifiers. After training the language models, we obtain features of files by scanning sequences up to length 200 for classification because using a sequence length longer than 200 events degraded the performance. We train the classifiers for a maximum of 20 epochs with early stopping.

The hyperparameters used for the character-level convolutional neural network are given in Table 1. While the language model classifiers use a maximum subsequence length of 200, we can use up to 1,014 file events for training the character-level CNN has since it no temporal memory-loss issue.

Model Performance: We first investigate how the attention mechanism compares to the temporal max pooling for the ESN and RNN language model-based architectures with logistic regression in Figure 3. An echo state, RNN-based network with temporal max pooling (LR-RNN-ESN-MAX) and RNN with temporal max pooling (LR-RNN-MAX) are the two best performing models in [1], and thus are the two primary baseline models. This figure also includes the results for an LR-RNN-based ESN with an attention mechanism (LR-RNN-ESN-ATT) as well as the RNN with attention (LR-RNN-ATT). The figure indicates that for the RNN-based architectures, the LR-RNN-ESN-ATT model clearly outperforms the other three variants including the two baseline models proposed in [1].

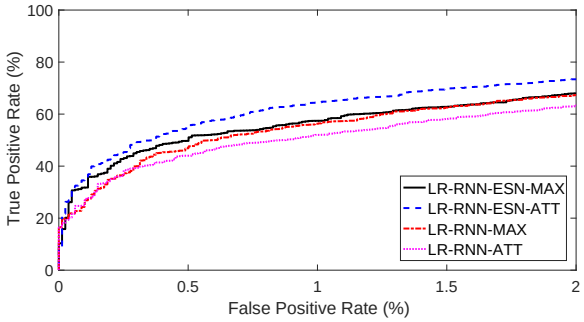

Fig. 3. ROC curves for the RNN-based models with a logistic regression classifier. An untrained RNN-based, echo state network model with an attention mechanism performs best.

After implementing the baseline systems in Figure 3, we next evaluated similar architectures based on the LSTM language model and logistic regression in Figure 4. While the two RNN-based ESNs (LR-RNN-ESN-ATT and LR-RNN-ESN-MAX) are the best two performing models in Figure 3, the LSTM-based ESNs (LR-LSTM-ESN-MAX and LSTM-ESN-ATT) are the two worst performing LSTM models in Figure 4. The LSTM model with temporal max pooling (LR-LSTM-MAX) is the best performing LSTM models overall. LR-LSTM-ATT, the LSTM model with attention, offers similar true positive rates (TPRs) for false positive rates (FPRs) above 1.0% but slightly lower performance for lower FPRs.

The performance of the GRU-based language models with logistic regression classification is depicted in Figure 5. The ROC curves are more tightly clustered compared to the RNN and LSTM-based models. Similar to the RNN models, the echo state network version with attention (LR-GRU-ESN-ATT) is the best performing model in this group. For FPRs greater than 0.80%, the GRU model trained with attention (LR-GRU-ATT) is the second best model.

In general, the ROC results for the logistic regression classifier-based architectures are slightly better than the MLP version with the same language model. For example, Figure 6 depicts the performance for the LSTM language models with an MLP classifier. Similar to Figure 4, the LSTM with max pooling (MLP-LSTM-MAX) outperforms the other LSTM language models with an MLP classifier.

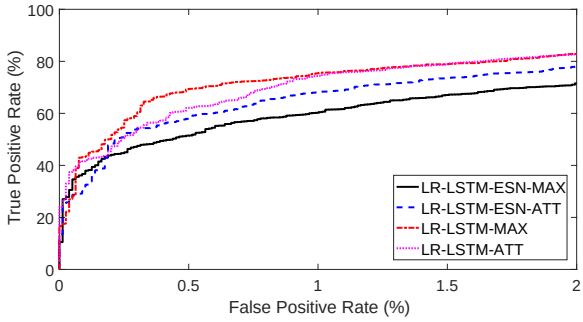

Fig. 4. ROC curves for the LSTM-based, language models with a logistic regression classifier. The standard LSTM with max pooling outperforms the other LSTM architectures on this dataset.

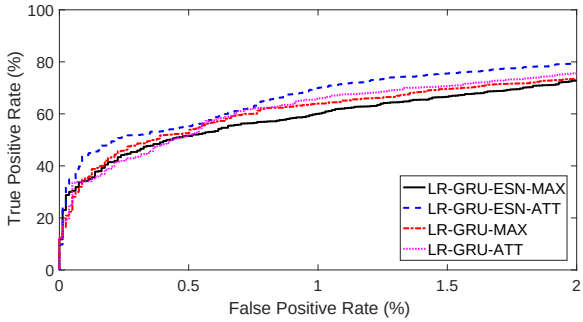

Fig. 5. ROC curves for the GRU-based, language models with a logistic regression classifier. Similar to Figure 3, the untrained GRU-based, echo state network with an attention (GRU-ESN-ATT) is the best performing GRU-based model.

Finally in Figure 7, we compare the ROC curves for the character-level, convolutional neural network (CHAR-CNN) to the best performing model in Figures 3- 6 and the two baselines from [1], LR-RNN-ESN-MAX and LR-RNN-MAX. This figure indicates that the overall best performing model in this study is the LSTM trained with temporal max pooling using the logistic regression classifier (LR-LSTM-MAX). The TPR for LR-LSTM-MAX is 31.3% higher than the TPR for LR-RNN-ESN-MAX at 1.0% FPR. The LSTM with temporal max pooling and an MLP classifier (MLP-LSTM-MAX) offers better performance for low FPRs, but LR-LSTM-MAX has a slightly better TPR at higher FPRs. The performance of the CHAR-CNN classifier is significantly worse than all of the other models at low FPRs but eventually becomes the third best performing model at an FPR of 1.9%.

5. RELATED WORK

Interestingly, the first malware classifier proposed in the literature employed a neural network [20]. Since this initial work, researchers have explored many different machine learning models [18, 21, 22]. Employing system calls as features have previously been used for malware classification [1, 3] and intrusion detection systems [23].

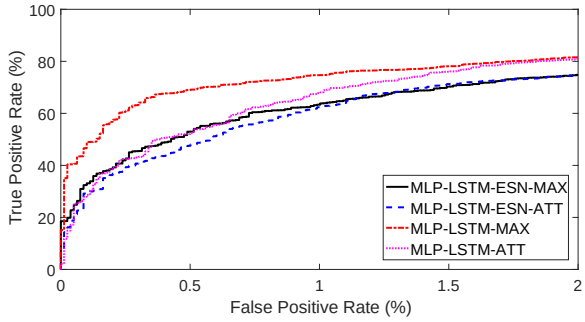

Fig. 6. ROC curves for the LSTM-based, language models with an MLP classifier. The standard LSTM with max pooling outperforms the other LSTM architectures on this dataset.

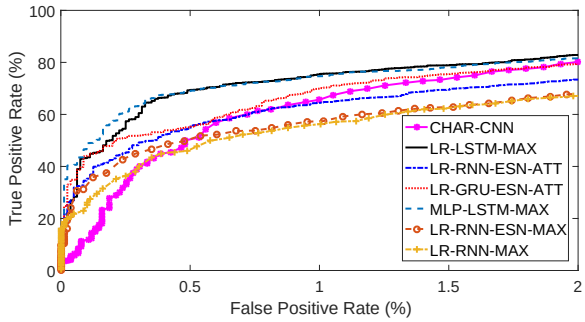

Fig. 7. ROC curves for the character-level CNN and the best performing models in Figures 3- 6. The LSTM with max pooling and a logistic regression classifier is the best performing model.

Our work is most closely related to that of Pascanu et al. [1]. This study extends their system by introducing new malware language models including the LSTM and GRU as well as adding the attention mechanism. We also propose the character-level CNN [12] as the malware classifier. Deep neural networks have been proposed for malware classification in [3, 4, 5, 6] as noted in the introduction.

The attention mechanism used in our work has yielded excellent results for machine translation [12].

6. CONCLUSIONS

Most importantly, our work shows the benefits of semi-supervised learning for malware classification where we leverage the knowledge from unsupervised learning on a large amount of unlabelled sequences captured in the neural language model of these file sequences. We show that with proper architectures that can handle long-term dependencies, the representation from the neural language models can help increase the performance on downstream tasks such as malware classification. This work is also based on an assumption that a language model for file stream events exists. The fact that our learned model performs better than the random weight counterpart confirms this hypothesis.

7. REFERENCES

[1] Razvan Pascanu, Jack W. Stokes, Hermineh Sanossian, Mady Marinescu, and Anil Thomas, "Malware classification with recurrent networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 1916–1920.

[2] Microsoft Malware Protection Center, "Submit a sample," `https://www.microsoft.com/en-us/security/portal/submission/submit.aspx/`, 2016.

[3] George E. Dahl, Jack W. Stokes, Li Deng, and Dong Yu, "Large-scale malware classification using random projections and neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[4] Joshua Saxe and Konstantin Berlin, "Deep neural network based malware detection using two dimensional binary program features," *arXiv preprint arXiv:1508.03096v2*, 2015.

[5] Wenyi Huang and Jack W. Stokes, "Mtnet: A multi-task neural network for dynamic malware classfication," in *Proceedings of Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2016, pp. 399–418.

[6] Seonhee Seok and Howon Kim, "Visualized malware classification based-on convolutional neural network," in *Proceedings of Korea Institutes of Information Security and Cryptology*, 2016, pp. 197–208.

[7] T. Mikolov, M Karafiat, L. Burget, J. Cernocky, and S Khundanpur, "Recurrent neural network based language model," in *Proceedings of Interspeech*, 2010.

[8] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," in *Science*, 2004.

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014.

[10] Sepp Hochreiter and Jurgen Schmidhuber, "Long short-term memory," in *Proceedings of Neural Computation*, 1997, pp. 1735–1780.

[11] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Proceedings of the Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST)*, 2014.

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[13] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[14] Mike Schuster and Kuldip K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, pp. 2673–2681, November 1997.

[15] Keras Development Team, "Keras: Deep learning library for theano and tensorflow," `https://keras.io/`, 2016.

[16] Vinod Nair and Geoffrey E Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[17] Xiang Zhang, Junbo Zhao, and Yann LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems (NIPS)*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., pp. 649–657. Curran Associates, Inc., 2015.

[18] N. Idika and A.P. Mathur, "A survey of malware detection techniques," Tech. Rep., Purdue Univ., February 2007.

[19] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[20] Jeffrey O. Kephart, "A biologically inspired immune system for computers," in *In Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*. 1994, pp. 130–139, MIT Press.

[21] M.G. Schultz, Eleazar Eskin, E. Zadok, and S. Stolfo, "Data mining methods of detection of new malicious executables," in *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, 2001, pp. 38–49.

[22] J.Z. Kolter and M.A. Maloof, "Learning to detect and classify malicious executables in the wild," in *Journal of Machine Learning Research*, 2006, pp. 2721–2744.

[23] Wenke Lee, Saivatore J. Stolfo, and Kui W. Mok, "A Data Mining Framework for Building Intrusion Detection Models," *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, pp. 120 – 132, 1999.