ATTENTION IN RECURRENT NEURAL NETWORKS FOR RANSOMWARE DETECTION

Rakshit Agrawal[⋆⋆] *Jack W. Stokes*[†] *Karthik Selvaraj, Mady Marinescu*[±]

[⋆] University of California, Santa Cruz, Santa Cruz, CA 95064 USA
[†] Microsoft Research, One Microsoft Way, Redmond, WA 98052 USA
[±] Microsoft Corp., One Microsoft Way, Redmond, WA 98052 USA

ABSTRACT

Ransomware, as a specialized form of malicious software, has recently emerged as a major threat in computer security. With an ability to lock out user access to their content, recent ransomware attacks have caused severe impact at an individual and organizational level. While research in malware detection can be adapted directly for ransomware, specific structural properties of ransomware can further improve the quality of detection. In this paper, we adapt the deep learning methods used in malware detection for detecting ransomware from emulation sequences. We present specialized recurrent neural networks for capturing local event patterns in ransomware sequences using the concept of attention mechanisms. We demonstrate the performance of enhanced LSTM models on a sequence dataset derived by the emulation of ransomware executables targeting the Windows environment.

Index Terms— Ransomware Detection, Attention Mechanisms, Deep Learning, Long Short-Term Memory, LSTM

1. INTRODUCTION

Ransomware, with recent massive scale attacks, has demonstrated the adverse effects of malicious software and has exposed a severe cybersecurity threat. A widespread ransomware attack has the potential to impact many worldwide organizations within a short period of time. Ransomware detection systems incorporate both expert-based as well as machine learning-based methods to increase detection rates. Machine learning methods for ransomware detection can be inspired by the wide body of research in malware detection.

A major family of malware detection methods emerges from the use of emulation sequences derived using Portable Executable (PE) files. These files are processed in a secure environment, and their actions are captured as a sequence of events with each event corresponding to a specific system call. Deep learning methods for sequence learning, such as [1, 2, 3, 4], have demonstrated strong results in malware detection. Since the ransomware executables can be emulated using an antimalware engine, similar methods can be adapted for ransomware detection.

The high detection accuracy of sequence learning models is mostly powered by the ability of recurrent neural networks, such as an LSTM (Long Short-Term Memory) [5, 6], to bind inter-relatedness between events occurring in a certain sequential order. While such order can be captured similarly in ransomware sequences, we believe that these sequences exhibit certain additional properties. As presented in the paper, an analysis of ransomware executables reveals the presence of a large number of short repeating event sequences within the longer sequence. An ability to capture this local repeating behavior along with the general sequence learning can therefore improve the performance of ransomware detection systems.

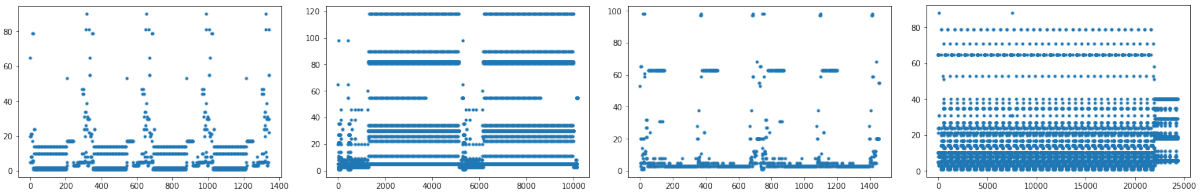
In the recent years, sequence learning methods have incorporated the use of attention mechanisms [7, 8] to strengthen learning by focusing at specific regions within the sequence during overall learning. In case of ransomware sequences, we believe that attention mechanisms can help capture the short locally repeating patterns.

In this paper, we present an enhanced neural cell to incorporate attention in learning from ransomware sequences, known as ARI (Attended Recent Inputs). The ARI cell, while processing the input sequence, also learns from a recent history in the form of a subsequence. It learns attention weights corresponding to each recent input and uses their corresponding significance when processing the input.

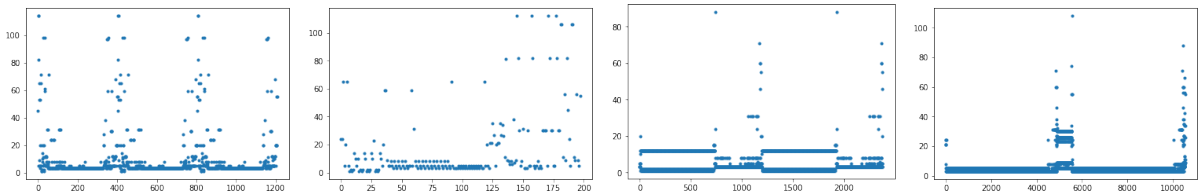
We present an implementation of the ARI cell with LSTM networks, called ARI-LSTM. We enhance the LSTM cell by incorporating ARI mechanism within the cell, and use the resulting neural network for sequence learning with ransomware. Through evaluation on a ransomware dataset for the Windows operating system environment, we show that ARI-LSTM improves the performance of an LSTM in detecting ransomware from emulation sequences.

The paper first explains the significance of repeated local patterns in ransomware sequences and relates the use of attention mechanisms for such tasks. We then describe the ARI cell in detail with its LSTM adaptation. This is followed by the system description for using ARI-LSTM in ransomware detection. We then present results on a large dataset in Windows environment, concluding with a discussion of the described approach.

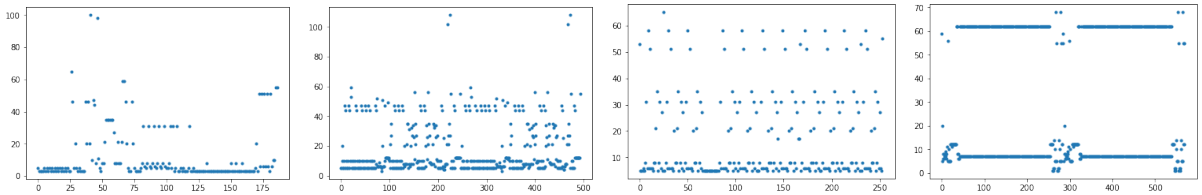
[⋆] The first author performed the work while at Microsoft Research.



(a) Ransomware PE Sequences



(b) General Malware PE Sequences



(c) Benign PE Sequences

Fig. 1: Visual representation of commonly observed ransomware, general malware and benign executable files. Each value on the y-axis corresponds to a Windows API call. The x-axis represents each timestep in the sequential run of the file.

2. BACKGROUND AND MOTIVATION

The adverse impact caused by ransomware on computing systems poses a major threat to everyday users and society in general. With continuous growth in ransomware with newer malicious families emerging every month, the need for strong defensive methods increases every day. While expert-based systems are developed over time, this rate of growth in ransomware creates a need for self evolving methods of defense that can learn from available data and improve over time. Deep learning methods, in particular, can provide this ability to improve learning with the increasing availability of data.

Learning methods inspired by malware detection, such as [1, 2, 4], act as a base design for using deep learning in ransomware detection. However, such methods do not learn specific properties of ransomware, which may not be observed in general malware. We performed an analysis of the execution behavior of ransomware Portable Executable files and compared them with regular malware, as well as with benign executable files. Figure 1 illustrates the behavior of executables in the three categories. Each plot represents the execution of a commonly observed file, where the y-axis corresponds to enumerated event IDs, and the x-axis represents the time steps. Each file category refers to the same list of event APIs. Therefore, a single plot displays the pattern of different API calls observed during the file execution.

As can be seen in the figure 1, ransomware executables display a significantly high repetition of small local patterns. While repeating patterns are often observed in both malware and benign files as well, the counts of repeating events is exceedingly high in the case of ransomware. Intuitively, this can be assumed as expected ransomware behavior, since such software often repeatedly apply an encryption operation on the files in the system. However, in order to efficiently use this behavior in ransomware detection, we must use methods that can utilize repeating behaviors while still maintaining a learning of the outer sequence of events.

Attention Mechanisms [7, 8] provide a family of deep learning tools where significance of specific data within a large structure can be directly related to its use within the remaining problem. They have demonstrated superior performance in machine translation [7, 8, 9], speech [10], language [11, 12], and image captioning [13] tasks. Attention is also used in more complex neural systems such as the Memory networks [14], Neural Turing Machine [15] and the Differential Neural Computer [16]. Improved architectures of attention [17, 9, 18, 19, 20] have also been developed allowing finer use of the information focused within specific regions in the input. Inspired by these models, we believe that the objective of utilizing smaller repeating patterns while processing a longer sequence can be addressed by attention mechanisms.

3. METHODS

In the previous section, we discussed the motivation behind using attention mechanisms while processing executable sequences for ransomware detection. In this section, we describe a neural component, called the Attended Recent Inputs (ARI) cell. An ARI cell, while processing a sequence, can simultaneously provide additional input information by learning attention weights for upto L recent inputs.

For a given primary input $\mathbf{x}_t \in \mathbb{R}^n$ at timestep t, where n is the input dimension to the ARI, and a set $\mathbb{S}_t = \{\mathbf{x}_{t-L}, \mathbf{x}_{t-(L-1)} \cdots, \mathbf{x}_{t-1}\}$ of nearby inputs needs to be attended, we first represent the set $\mathbb{S}_t$ as a matrix $R_t \in \mathbb{R}^{n \times L}$ where each row is a recent input vector at timestep t. The computation process for ARI using $\mathbf{R}_t$ is then defined as:

$$\begin{aligned} \mathbf{R}_t &= \textsc{Matrix}(\mathbb{S}_t) \\ \mathbf{M}_t &= \textsc{Dense}(\mathbf{W}_d * \mathbf{R}_t) \\ \alpha_t &= \text{softmax}(\omega^T \mathbf{M}_t) \\ \mathbf{r}_t &= \mathbf{R}_t \alpha_t^T \end{aligned} \tag{1}$$

where L is the number of recent inputs in $\mathbf{R}_t$ and $\mathbf{M}_t \in \mathbb{R}^{n \times L}$ is the attended vector over recent inputs using an attention learning function $fn = \textsc{Dense}$ neural network layer. $\alpha_t \in \mathbb{R}^{1 \times L}$ is the computed soft weight distribution across $\mathbf{R}_t$. $\mathbf{r}_t \in \mathbb{R}^n$ is the derived vector for input x_t providing a combined measured attention of recent inputs to be used along with x_t. $\mathbf{W}_d$ is the weight matrix for the dense layer, and ω^T is the transposed weight vector ω used for aligning the attended vector. The ARI cell is also illustrated in Figure 2, where fn refers to the DENSE learning operation used by the ARI cell.

The ARI cell, therefore, performs attention at the input of a recurrent neural network (RNN). In order to use such cells in sequence learning, we need to adapt them with an RNN architecture. For instance, the ARI cell can be used as Simple Recurrent Neural Network (SimpleRNN) and Long Short-Term Memory variants.

SimpleRNN: At each timestep t, a Simple Recurrent Neural Network uses the activation $\mathbf{h}_{t-1}$ from the previous timestep when processing input $\mathbf{x}_t$ in order to influence the activation $\mathbf{h}_t$. With σ denoting a non-linearity, we can express a Simple RNN as:

$$\mathbf{h}_t = \sigma(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t) \tag{2}$$

where $\mathbf{W}_h \in \mathbb{R}^{k \times k}$ and $\mathbf{W}_x \in \mathbb{R}^{k \times n}$ are trainable projection matrices for the hidden input $\mathbf{h}_{t-1}$ and input $\mathbf{x}_t$, respectively. k is the output/hidden dimension of the RNN cell, and n is the input dimension. Using the additional input $\mathbf{r}_t$ from ARI at each timestep t, we derive the equations for ARI-RNN as:

$$\mathbf{h}_t = \sigma(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{W}_r \mathbf{r}_t) \tag{3}$$

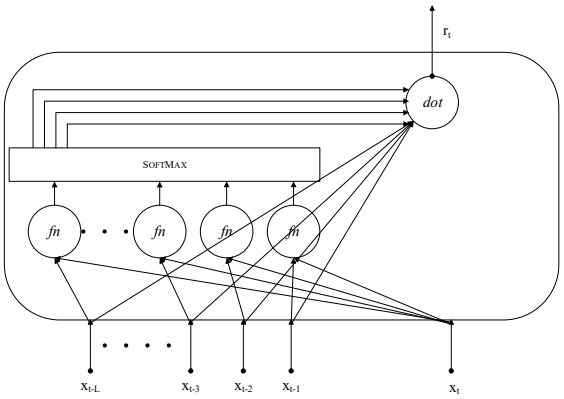


Fig. 2: An illustration of the ARI cell which operates on recurrent inputs and generates a learned vector $\mathbf{r}_t$ using the attention mechanism fn.

where $\mathbf{W}_r \in \mathbb{R}^{k \times n}$ is a trainable projection matrix for the ARI input in the cell.

Long Short-Term Memory: An LSTM [5] is a memory-based gated cell for RNNs most commonly used with long sequences. LSTMs use three kinds of gates (input, output and forget), along with an explicit cell memory. For input $\mathbf{x}_t$ at timestep t, LSTMs using ARI can be adapted into ARI-LSTM as:

$$\begin{aligned} \mathbf{i}_t &= \sigma(\mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{ri}\mathbf{r}_t) \\ \mathbf{f}_t &= \sigma(\mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{rf}\mathbf{r}_t) \\ \mathbf{o}_t &= \sigma(\mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ro}\mathbf{r}_t) \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{rc}\mathbf{r}_t) \\ \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \end{aligned} \tag{4}$$

where σ is the logistic sigmoid function, $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t, \mathbf{c}_t$ are input gate, forget gate, output gate and cell activation, respectively. $\mathbf{W}_h$ are the recurrent weight matrices for each gate, and $\mathbf{W}_x$ are the input weight matrices. $\mathbf{W}_r$ in this equation refers to the recent weight matrices, which are the weight matrices associated with information on recent inputs for each element in the sequence.

While we use the basic definition of attention mechanisms in ARI, more complex cells can also be generated to perform larger attentions similar to [21, 22]. Our objective in defining ARI is the ability to integrate recent input attention within the larger cell operating directly on a sequence. By providing an implicit input attention, we help utilize the attention weights at each timestep in measuring both the hidden activation from the cell, as well as the cell memory. For problems sensitive to relations with recent inputs, we believe this use of attention

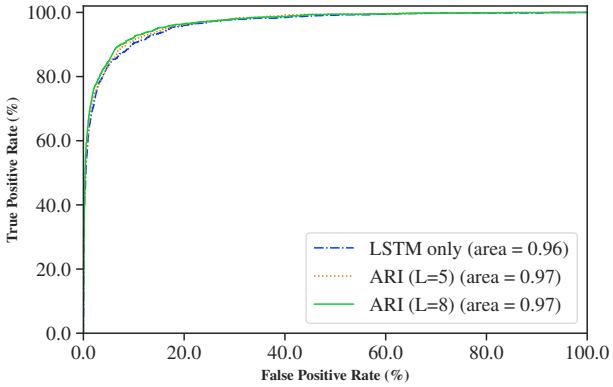


Fig. 3: ROC curves for the LSTM and ARI-LSTM cells in the LaMP model for ransomware detection.

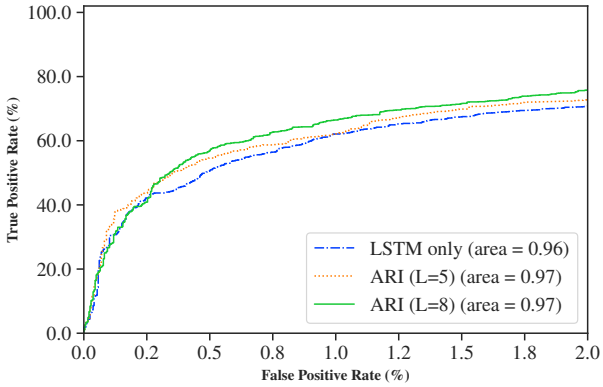


Fig. 4: ROC curves for the LSTM and ARI-LSTM cells in the LaMP model for ransomware detection, zoomed into a maximum FPR = 2%.

within the cell provides a direct influence factor that cannot easily be captured by the LSTM.

4. EVALUATION

Inspired by the ransomware analysis in section 2, the design of the Attended Recent Inputs (ARI) cell was described in the previous section. In this section, we present results from their evaluation on ransomware detection.

4.1. Experiment Setup

We based our learning setup for ransomware detection on the LSTM and Max Pooling (LaMP) model for malware detection presented in [4]. While the LaMP model uses the LSTM as the recurrent neural network, we evaluate the ransomware dataset with both an LSTM and an ARI-LSTM . For our task, the learning objective of LaMP is to perform binary classification on input sequences where a label of 1 corresponds to an inference as 'ransomware' and a label of 0 means 'benign'. We used a dataset of unique file sequences consisting of ransomware and benign executables for the Windows operating system captured from client computers. We trained the model on 12,500 sequences, with a 50% distribution over the labels. We used the Keras [23] deep learning framework, with Tensorflow [24] backend for the training and inference stages of our experiments. All the models were trained using back-propagation with the Adam optimizer [25].

4.2. Results

We evaluate the performance of each model on a larger testing dataset with 26,300 samples. In order to evaluate performance of each model, we compare the receiver operating characteristic (ROC) curves. As shown in the Figure 3, ARI-LSTM performs consistently better than the standard LSTM for the objective of ransomware detection. We further observe the performance at a much finer scale with False Positive Rate (FPR) set at 2% in Figure 4. This focused observation further shows better performance by ARI-LSTM even at very small values of FPR. We also compare the overall accuracy of each model under this setting, with results presented in table 1. Across both the metrics, ARI-LSTM shows significantly better performance than LSTM, proving the efficiency of learning local patterns through attention mechanisms.

Table 1: Accuracy comparison for ransomware detection

Model	Accuracy
LSTM	0.87
ARI-LSTM (L=5)	**0.93**
ARI-LSTM (L=8)	**0.91**

5. CONCLUSION

This paper serves an important problem in cybersecurity for ransomware detection. We perform a detailed analysis of ransomware executables in order to identify structural properties that can be exploited by machine learning systems. We identify an existence of small repeating patterns within long sequences of ransomware potentially corresponding to repeated encryption operations. We present a novel recurrent neural network component for exploiting the repeating patterns by incorporating attention mechanisms on the inputs of a sequence learning module. We present an LSTM variant of our cell called ARI-LSTM . With empirical results on a ransomware dataset, we show that ARI-LSTM performs significantly better than an LSTM for the task of ransomware detection. With the ARI cell, we present an approach for incorporating attention at the inputs of a sequence, which can be used by problems sensitive to relations within recent inputs.

6. REFERENCES

[1] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 1916–1920.

[2] B. Athiwaratkun and J. W. Stokes, "Malware classification with lstm and gru language models and a character-level cnn," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 2482–2486.

[3] R. Agrawal, J. W. Stokes, M. Marinescu, and K. Selvaraj, "Neural sequential malware detection with parameters," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 2656–2660.

[4] R. Agrawal, J. W. Stokes, M. Marinescu, and K. Selvaraj, "Robust neural malware detection models for emulation sequence learning," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, Oct 2018, pp. 1–8.

[5] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[6] Felix A. Gers, Jürgen Schmidhuber, and Fred A. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, vol. 12, pp. 2451–2471, 2000.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2015.

[8] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning, "Effective approaches to attention-based neural machine translation," *CoRR*, vol. abs/1508.04025, 2015.

[9] Michal Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel, "Frustratingly short attention spans in neural language modeling," *CoRR*, vol. abs/1702.04521, 2017.

[10] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015.

[11] Alexander M. Rush, Sumit Chopra, and Jason Weston, "A neural attention model for abstractive sentence summarization," in *EMNLP*, 2015.

[12] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst, "Text understanding with the attention sum reader network," *CoRR*, vol. abs/1603.01547, 2016.

[13] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015.

[14] Jason Weston, Sumit Chopra, and Antoine Bordes, "Memory networks," *CoRR*, vol. abs/1410.3916, 2014.

[15] Alex Graves, Greg Wayne, and Ivo Danihelka, "Neural turing machines," *CoRR*, vol. abs/1410.5401, 2014.

[16] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, et al., "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, pp. 471–476, 2016.

[17] Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston, "Key-value memory networks for directly reading documents," in *EMNLP*, 2016.

[18] Scott E. Reed and Nando de Freitas, "Neural programmer-interpreters," *CoRR*, vol. abs/1511.06279, 2016.

[19] Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu, "Using fast weights to attend to the recent past," in *NIPS*, 2016.

[20] Çaglar Gülçehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio, "Dynamic neural turing machine with continuous and discrete addressing schemes," *Neural Computation*, vol. 30, no. 4, 2018.

[21] Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu, "Using fast weights to attend to the recent past," 2016.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, L ukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., pp. 5998–6008. Curran Associates, Inc., 2017.

[23] François Chollet et al., "Keras," `https://keras.io`, 2015.

[24] Martín Abadi, Ashish Agarwal, Paul Barham, et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.

[25] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.