Attention-Based LSTM for Insider Threat Detection

Fangfang Yuan[1,2], Yanmin Shang[1], Yanbing Liu[1(✉)], Yanan Cao[1], and Jianlong Tan[1]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{yuanfangfang,shangyanmin,liuyanbing,caoyanan,tanjianlong}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. Insider threat is an important cyber security issue for businesses and organizations. Existing insider threat detection methods can be roughly divided into two categories, statistical features based detection methods and action sequence based detection methods. The first kind of method aggregates all actions that a user has performed over one day and uses these aggregated features to find insider threat. This kind of coarse-grained analytics of user behavior may miss anomalous behavior happening within that day. The second kind of method overcomes the coarser-grained problem and uses fine-grained detection to identify insider threat through user actions. However, the second kind of method considers all user operations to be equally important, without highlighting malicious user actions. To solve this problem, we present an attention-based Long Short-Term Memory (LSTM) model to detect insider threat. In our model, we apply the LSTM to capture the sequential information of user action sequence and employ an attention layer that can learn which user actions contribute more to insider threat detection. Extensive studies are conducted on the public dataset of insider threat. Our results demonstrate that the proposed model outperforms other deep learning models and can successfully identify insider threat.

Keywords: Insider threat detection · Recurrent Neural Network · Anomaly detection · Network security

1 Introduction

Insider threat is one of the most serious challenges in cyber security. Malicious insiders who are trusted by organizations, such as an employee advertently abuse their authorized access to organizational information systems and commit attacks, often causing privacy, credibility and reputation issues [1]. How to detect insider threat early has become a research hotspot in cyber security [2].

However, insider threat detection faces several serious challenges. Firstly, system logs are usually used for insider threat detection. How to identify insider threats from a massive amount of system logs is a crucial issue. Secondly, insider

V. S. Shankar Sriram et al. (Eds.): ATIS 2019, CCIS 1116, pp. 192–201, 2019.
https://doi.org/10.1007/978-981-15-0871-4_15

threat behavior is widely varying, such as a disgruntled employee deleting data from the hard disk or database, using his privileged access to take sensitive data for financial gain, etc. The threat behavior manifests in various forms, thus increasing the difficulty of insider threat detection. Finally, insider's anomalous behavior usually consists of several subtle actions scattered in a lot of users' normal behavior.

In order to detect insider threat, it is necessary to build the profiles representing normal behaviour and recognize abnormal behavior that deviates from the user's normal behavior profiles. Researchers have proposed many approaches to detect and identify insider's anomalous behavior. Tuor et al. [3] use the user's aggregated action features in one day for insider threat detection. However, some anomalous behavior that happened within a day cannot be detected by using this method. For example, an employee logs in his office computer after hours and copys some sensitive data to the removable disk. In order to overcome the coarser-grained problem, Yuan et al. [4] presented the LSTM-CNN framework to detect insider threat. They used the LSTM to capture the temporal features of user behavior from user's action sequences and used the Convolutional Neural Network (CNN) to identify user's abnormal behavior. However, the framework considers all user operations to be equally important, without highlighting malicious user actions.

In this paper, we propose an attention-based LSTM to detect insider threat. Firstly, we apply the LSTM to capture the sequential information of user behavior as far as possible. Secondly, we employ an attention layer that can automatically judge which user actions have more contributions to the classification decision. In summary, the main contributions of the paper are as follows:

(1) We use the LSTM to capture the sequential information of user action sequences.
(2) We apply the attention layer to let the model to pay more or less attention to individual user action when constructing the representation of the user behavior.
(3) We conduct experiments on the CERT insider threat dataset and the results demonstrate that our model outperforms other deep learning models and can successfully identify insider threat.

The rest of this paper is organized as follows. In Sect. 2, we review the related research in the field of insider threat detection. In Sect. 3, we describe our attention-based proposal in detail. We provide the experimental results in Sect. 4. In Sect. 5, We conclude the paper.

2 Related Work

The topic of insider threat has recently received increasing attention both in academic and industry fields. There has been many studies on insider threat detection.

Insider threat detection based on machine learning is the main direction of current studies. Schonlau et al. [5] built the Schonlau dataset (SEA dataset) based on UNIX user truncated commands and compared six different insider threat detection methods. Maxion et al. [6,7] used the same dataset and showed better performance by using the Naive Bayes classifier to detect insider threat. Oka et al. [8] employed the Eigen Co-occurrence Matrix (ECM) approach for insider threat detection on the SEA dataset. Szymanski and Zhang [9] used One-Class Support Vector Machine (OC-SVM) to detect insider threat. However, their results are not good enough. More recently, Rashid et al. [10] used the Hidden Markov model (HMM) to build each user's normal behavior profile and identify the deviations that may potentially indicate insider threats. The advantage of their model is learning from the sequential data. However, the increasing number of states leads to the increasing computational cost of the HMM model, while the number of the states would highly impact the effectiveness of this method.

Recently, the rapid development of deep neural networks has brought new inspiration to insider threat detection. Veeramachaneni et al. [11] used time-aggregated statistics as features and applied an Autoencoder neural network to insider threat detection. However, they did not explicitly model individual user behavior over time. Lu and Wong [12] used the LSTM to build user's behavior patterns and find anomalous events. However, the LSTM is a biased model, where later user actions are more dominant than earlier user actions. Hence, using the recurrent model to directly classify user's behavior is not efficient. Tuor et al. [3] proposed a deep learning based insider threat detection system. They trained the LSTM models to recognize each user's characteristic and classified user behavior as anomalous or normal. While they aggregated features by one day for each user, this has the potential to miss anomalous behavior happening within a single day. To solve the coarser-grained problem, Yuan et al. [4] presented the LSTM-CNN framework to detect insider threat. However, they failed to highlight the user actions that contribute more to detect insider threat. Instead, our model combines the LSTM and the attention layer. Therefore, our model can identity anomalous behavior happening within a single day and provides insight into which user actions contribute more to insider threat detection.

3 Attention-Based LSTM Model

The aim of our work is to find the user's anomalous behavior which is an indicator of insider threat. The individual operation of a user represents a user action; a user action sequence that the user performs in one day represents user behavior. We firstly feed a user action sequence to the LSTM layer and obtain the abstract feature vectors. Secondly, the abstract feature vectors are fed into the attention layer. Finally, we obtain the representation of user behavior and feed it to the output layer to classify the user behavior as anomalous or normal.

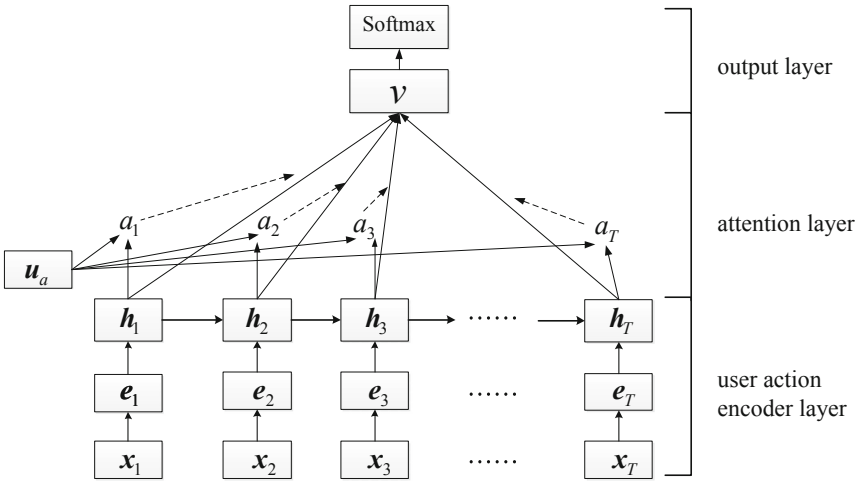


Fig. 1. Structure of attention-based LSTM

Figure 1 shows the structure of the attention-based LSTM model. It contains three layers: a user action encoder layer, an attention layer and an output layer. The different components of the structure are described in detail as follows.

3.1 LSTM-Based Sequence Encoder

Recurrent Neural Network (RNN) extends the conventional feed-forward neural network. Unfortunately, Bengio et al. [13] found that training the RNN to capture long-term dependencies is a difficult task because the vanishing gradient problem or the exploding gradient problem may occur during the training process. In order to address this problem, Hochreiter and Schmidhuber [14] developed the Long Short-Term Memory (LSTM) nerual network. Unlike to the RNN, the LSTM maintains a memory cell that is capable of storing information. The LSTM has three gates which are used to modulate the flow of information inside the unit. The input gate decides how much of the new information should be added to the memory cell. The forget gate modulates how much of the previous cell state should be forgotten. The output gate decides which part of the memory should be seen.

The updation of the LSTM is implemented as follows:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \boldsymbol{e}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{1}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \boldsymbol{e}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \tag{2}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \boldsymbol{e}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \tag{3}$$

$$\boldsymbol{g}_t = \tanh(\boldsymbol{W}_g \boldsymbol{e}_t + \boldsymbol{U}_g \boldsymbol{h}_{t-1} + \boldsymbol{b}_g) \tag{4}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t \tag{5}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \tag{6}$$

where $\boldsymbol{W}_i$, $\boldsymbol{W}_f$, $\boldsymbol{W}_o$ are the weighted matrices and $\boldsymbol{b}_i$, $\boldsymbol{b}_f$, $\boldsymbol{b}_o$ are biases of the LSTM. These parameters are learned during training. σ denotes sigmoid function. $\odot$ is an element-wise multiplication. $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, $\boldsymbol{o}_t$ represent the input gate, the forget gate and the output gate respectively. $\boldsymbol{e}_t$ is the sequence vector at time t, representing the user action embedding vector $\boldsymbol{x}_t$ in Fig. 1. The hidden state is $\boldsymbol{h}_t$.

3.2 Attention Layer

The standard LSTM cannot pick out which is the important part for insider threat detection. To solve the problem, we design that the LSTM is followed by an attention layer that can capture the important user actions.

As Fig. 1 shows, given the user $u_k (k \in [0, K])$, his action sequence on the j-th day can be represented as $S_{u_k} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T]$, where $\boldsymbol{x}_t (1 \leq t \leq T)$ denotes the user action at time instance t. Firstly, the user action is embedded into a vector representation $\boldsymbol{e}_t = \boldsymbol{W}_e \boldsymbol{x}_t$, where $\boldsymbol{W}_e$ is the embedding matrix. Next, a single layer LSTM takes the embedded vector $\boldsymbol{e}_t$ as input, and outputs the hidden status $\boldsymbol{h}_t$.

$$\boldsymbol{e}_t = \boldsymbol{W}_e \boldsymbol{x}_t, t \in [1, T] \tag{7}$$

$$\boldsymbol{h}_t = LSTM(\boldsymbol{e}_t), t \in [1, T] \tag{8}$$

Not all user actions contribute equally to detect insider threat. Hence, we employ attention mechanism to self-adaptively pick out important user actions that play key roles in insider threat detection. Specifically,

$$\boldsymbol{u}_t = \tanh(\boldsymbol{W}_a \boldsymbol{h}_t + \boldsymbol{b}_a) \tag{9}$$

$$\alpha_t = \frac{\exp(\boldsymbol{u}_t^T \boldsymbol{u}_a)}{\sum_t \exp(\boldsymbol{u}_t^T \boldsymbol{u}_a)} \tag{10}$$

$$\boldsymbol{v} = \sum_t \alpha_t \boldsymbol{h}_t \tag{11}$$

where $\boldsymbol{W}_a$ is the weighted matrix and $\boldsymbol{b}_a$ is the bias. $\boldsymbol{u}_a$ is a context vector, which is randomly initialized and jointly learned during training.

That is, a one-layer neural network takes the user action hidden status $\boldsymbol{h}_t$ as input and outputs $\boldsymbol{u}_t$ which is the hidden representation of $\boldsymbol{h}_t$. Next, the importance of each user action is measured by computing the similarity of $\boldsymbol{u}_t$ with the context vector $\boldsymbol{u}_a$. Then, we use the softmax function to obtain the normalized importance weights. At last, we compute the user behavior vector $\boldsymbol{v}$ as a weighted sum of the user action hidden status $\boldsymbol{h}_t$. Hence, the user behavior vector $\boldsymbol{v}$ summarizes all the information of the user action sequence.

3.3 Output Layer

The last part of our model is an output layer. For insider threat detection, the user behavior vector $\boldsymbol{v}$ is the whole representation of the user action sequence.

Table 1. Selected user action set

Activities	Actions	The number of actions
Logon activity	Logon	4
	Logoff	4
File activity	Copy exe file	4
	Copy doc file	4
	Copy pdf file	4
	Copy txt file	4
	Copy jpg file	4
	Copy zip file	4
HTTP activity	Visit neutral website	4
	Visit hacktivist website	4
	Visit cloudStorage website	4
	Visit jobHunting website	4
Email activity	Internal email	4
	External email	4
Device activity	Connect	4
	Disconnect	4

The softmax classifier takes the $\boldsymbol{v}$ vector as input,

$$p(\hat{y} = k|\boldsymbol{v}) = \frac{\exp(\boldsymbol{W}_k^T \boldsymbol{v} + \boldsymbol{b}_k)}{\sum_{k'=1}^{K} \exp(\boldsymbol{W}_{k'}^T \boldsymbol{v} + \boldsymbol{b}_{k'})} \tag{12}$$

where $\hat{y}$ is the predicted label of the user action sequence, the number of classes is K, $\boldsymbol{W}_k$ and $\boldsymbol{b}_k$ are the parameters of the softmax function for the k-th class. In order to train our model, we use the standard cross-entropy as training loss,

$$L = -\frac{1}{M} \sum_{i=1}^{M} y_i * log(p(\hat{y}_i)) \tag{13}$$

where y_i is the true label of the i-th user action sequence, M is the number of training user action sequences.

4 Experiments

We evaluate the proposed model on the publicly available CMU-CERT Insider Threat [15]. The model is implemented using Keras [16] with Theano [17] backend. Firstly, we introduce the dataset and comparison of methods. Next, we describe the experiment setup. Finally, we compare the performances of different models.

4.1 Dataset

Since the number of insider threat instances in CERT Insider Threat Dataset version r4.2 is larger than other versions of datasets, we conduct experiments on the version r4.2. The dataset consists of five different types of system logs. We can parse the system logs and obtain detailed user activity information. Furthermore, we find that the user behavior of normal users is different from the user behavior of abnormal users during after hours. Compared with normal users, some abnormal users who did not previously work after hours begin logging on their office computers after hours and copying sensitive data to the removable disk. Therefore, we divide a single day into 2 time segments: working hours (8am–5pm) and after hours (5pm–8am). In addition, we regard user's action performed on an assigned PC and user's action performed on an unassigned PC as two different user actions. Finally, we obtain a total of 64 user actions over five categories. Take an user action for example, a user sends an external email working hours on an unassigned computer. Table 1 shows the full set of user actions.

In our experiments, we use the activity record data of 100 users and build 100 users' specific profiles. After data preprocessing, we obtain a total of 25,274 action sequences among which only 954 action sequences represent the anomalous activities. The entire dataset is splitted into two subsets: 80% of the dataset for training and 20% of the dataset for testing.

4.2 Baselines

We compare our method with several deep learning models. Specifically, we test the RNN, LSTM and GRU model to find out which performs better when constructing the feature vectors of user behavior for each user. In addition, We perform experiments to assess the effectiveness of the attention mechanism. Therefore, we design several deep learning models as baseline methods.

RNN. This model consists of a single layer RNN network and a softmax layer. The RNN layer takes a user action sequence as input and feeds the last hidden state to the softmax layer for insider threat detection.

RNN with attention (RNN-Att). This model combines the basic RNN network with an attention mechanism and is used to compared with the RNN. We use this model to asess the effectiveness of the attention mechanism in the RNN.

GRU. This model consists of a single layer GRU network and a softmax layer. The GRU layer takes a user action sequence as input and feeds the last hidden state to the softmax layer for insider threat detection.

GRU with attention (GRU-Att). This model combines the basic GRU network with an attention mechanism and is used to compared with the GRU. We use this model to asess the effectiveness of the attention mechanism in the GRU.

LSTM. This model consists of a single layer LSTM network and a softmax layer. The LSTM layer takes a user action sequence as input and feeds the last hidden state to the softmax layer for insider threat detection.

Table 2. Parameters of the RNN, LSTM, GRU and Softmax

Models	Input	Hidden layer
RNN	Embedding dimension: 128	Units size: 64(128, 256) Dropout: 0.5 Activate function: tanh
LSTM	Embedding dimension: 128	Units size: 64(128, 256) Dropout: 0.5 Activate function: tanh The offset of forget gate: True
GRU	Embedding dimension: 128	Units size: 64(128, 256) Dropout: 0.5 Activate function: tanh
Softmax	Input dimension: 128	No

4.3 Experiment Setup

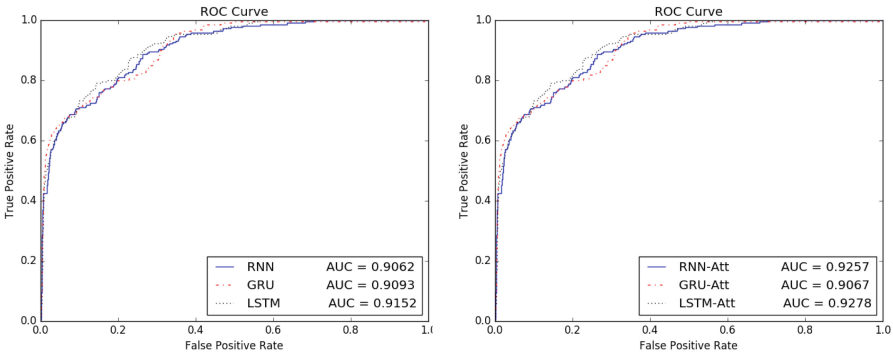
The proposed method is an end-to-end architecture. We hand-tuned the hyper-parameters of the RNN, LSTM and GRU by sweeping over a range of possible values. We tune the number of the batch size (between 5 and 30) and the epoch (between 10 and 30). The parameters of the RNN, LSTM, GRU and Softmax are shown in Table 2. The optimizer is the RMSprop optimizer and the loss function is the cross entropy loss. We set the learning rate to be 0.001.

4.4 Results

Figure 2 shows the experimental results on the CERT insider threat dataset version r4.2. As the dataset is imbalanced, we use the AUC-ROC (Area Under Curve - Receiver Operating Characteristics) curve as the evaluation metric. We analyze these results in detail in the following.

We first evaluate the ROC curves of different models under the same parameter settings. We fix the batch size to 30 samples, the epoch number to 20 and the units size to 128. Figure 2(a) shows the ROC curves when the RNN model, the LSTM model and the GRU model, respectively, are used for insider threat detection. Figure 2(b) shows the ROC curves when these models with attention mechanism, respectively, are used for insider threat detection. We can see that the performances of these models differ slightly when using the same parameter settings. The attention-based LSTM (LSTM-Att) is the best performing model and achieves an area under the ROC curve 0.9278.

We compare RNN with RNN-Att, GRU with GRU-Att, LSTM with LSTM-Att, finding that the addition of attention mechanism improves the performance for both the RNN and the LSTM. The attention mechanism improves the performance of the RNN more significantly because it can highlight the user actions



(a) The models without attention (b) The models with attention

Fig. 2. ROC curves for different models

which are more likely to be malicious user actions. Note that the AUC of GRU-Att is close to that of GRU, we suspect that the GRU-Att will yield superior performance when applied to large-scale dataset with more complicated temporal patterns.

5 Conclusion

In order to achieve fine-grained analysis of user behavior and improve the detection rate of insider threat, we propose an attention-based LSTM for insider threat detection. Since the threat behavior manifests in different forms, we cannot explicitly define the anomalous behavior pattern of insiders. Instead, we build the user's normal behavior profiles and take the user's anomalous behavior as an indicator of insider threat. Our model captures sequential information of user action sequences with the LSTM and constructs the representation of user behavior using the attention mechanism. We evaluate our method on the public CMU-CERT Insider Threat dataset Version r4.2. The experimental results demonstrate that our model outperforms other baseline methods and can successfully identify insider threat.

Acknowledgement. This work was partly supported by the National Key Research and Development Program (Grant No. 2017YFC0820700), Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No.XDC02030000, the National Natural Science Foundation of China under grant No. 61602466.

References

1. Costa, D.L., Albrethsen, M.J. Collins, M.L: Insider threat indicator ontology. Technical report, Carnegie-Mellon University, Pittsburgh, PA, United States (2016)
2. Azaria, A., Richardson, A., Kraus, S., Subrahmanian, V.S.: Behavioral analysis of insider threat: a survey and bootstrapped prediction in imbalanced data. IEEE Trans. Comput. Soc. Syst. **1**(2), 135–155 (2014)
3. Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N., Robinson, S.: Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In: Workshops at the Thirty-First AAAI Conference on Artificial Intelligence (2017)
4. Yuan, F., Cao, Y., Shang, Y., Liu, Y., Tan, J., Fang, B.: Insider threat detection with deep neural network. In: Shi, Y., et al. (eds.) ICCS 2018. LNCS, vol. 10860, pp. 43–54. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93698-7_4
5. Schonlau, M., DuMouchel, W., Ju, W.-H., Karr, A.F., Theusan, M., Vardi, Y., et al.: Computer intrusion: detecting masquerades. Stat. Sci. **16**(1), 58–74 (2001)
6. Maxion, R.A., Townsend, T.N.: Masquerade detection using truncated command lines. In: Proceedings International Conference on Dependable Systems and Networks, pp. 219–228. IEEE (2002)
7. Maxion, R.A.: Masquerade detection using enriched command lines. In: Proceedings of 2003 International Conference on Dependable Systems and Networks, pp. 5–14. IEEE (2003)
8. Oka, M., Oyama, Y., Kato, K.: Eigen co-occurrence matrix method for masquerade detection. Publications of the Japan Society for Software Science and Technology (2004)
9. Szymanski, B.K., Zhang, Y.: Recursive data mining for masquerade detection and author identification. In: 2004 Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, pp. 424–431. IEEE (2004)
10. Rashid, T., Agrafiotis, I., Nurse, J.R.C.: A new take on detecting insider threats: exploring the use of hidden markov models. In: Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats, pp. 47–56. ACM (2016)
11. Veeramachaneni, K., Arnaldo, I., Korrapati, V., Bassias, C., Li, K.: AI^2: training a big data machine to defend. In: 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), pp. 49–54. IEEE (2016)
12. Lu, J., Wong, R.K.: Insider threat detection with long short-term memory. In: Proceedings of the Australasian Computer Science Week Multiconference, p. 1. ACM (2019)
13. Bengio, Y., Simard, P., Frasconi, P., et al.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
15. Glasser, J., Lindauer, B.: Bridging the gap: a pragmatic approach to generating insider threat data. In: 2013 IEEE Security and Privacy Workshops, pp. 98–104. IEEE (2013)
16. Al-Rfou, R., et al.: Theano: a python framework for fast computation of mathematical expressions. arXiv preprint arXiv:1605.02688 (2016)
17. Chollet, F., et al.: Keras: The python deep learning library. Astrophysics Source Code Library (2018)