A Neural Network Architecture Combining Gated Recurrent Unit (GRU) and Support Vector Machine (SVM) for Intrusion Detection in Network Traffic Data

Abien Fred M. Agarap
abienfred.agarap@gmail.com

ABSTRACT

Gated Recurrent Unit (GRU) is a recently-developed variation of the long short-term memory (LSTM) unit, both of which are variants of recurrent neural network (RNN). Through empirical evidence, both models have been proven to be effective in a wide variety of machine learning tasks such as natural language processing[23], speech recognition[4], and text classification[24]. Conventionally, like most neural networks, both of the aforementioned RNN variants employ the Softmax function as its final output layer for its prediction, and the cross-entropy function for computing its loss. In this paper, we present an amendment to this norm by introducing linear support vector machine (SVM) as the replacement for Softmax in the final output layer of a GRU model. Furthermore, the cross-entropy function shall be replaced with a margin-based function. While there have been similar studies[2, 22], this proposal is primarily intended for binary classification on intrusion detection using the 2013 network traffic data from the honeypot systems of Kyoto University. Results show that the GRU-SVM model performs relatively higher than the conventional GRU-Softmax model. The proposed model reached a training accuracy of ≈81.54% and a testing accuracy of ≈84.15%, while the latter was able to reach a training accuracy of ≈63.07% and a testing accuracy of ≈70.75%. In addition, the juxtaposition of these two final output layers indicate that the SVM would outperform Softmax in prediction time - a theoretical implication which was supported by the actual training and testing time in the study.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by classification**; **Support vector machines**; **Neural networks**; • **Security and privacy** → *Intrusion detection systems*;

KEYWORDS

artificial intelligence; artificial neural networks; gated recurrent units; intrusion detection; machine learning; recurrent neural networks; support vector machine

ICMLC 2018, February 26–28, 2018, Macau, China

ACM ISBN 978-1-4503-6353-2/18/02…$15.00
https://doi.org/10.1145/3195106.3195117

ACM Reference Format:
Abien Fred M. Agarap. 2018. A Neural Network Architecture Combining Gated Recurrent Unit (GRU) and Support Vector Machine (SVM) for Intrusion Detection in Network Traffic Data. In *ICMLC 2018: 2018 10th International Conference on Machine Learning and Computing, February 26–28, 2018, Macau, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3195106.3195117

1 INTRODUCTION

By 2019, the cost to the global economy due to cybercrime is projected to reach $2 trillion[10]. Among the contributory felonies to cybercrime is *intrusions*, which is defined as illegal or unauthorized use of a network or a system by attackers[7]. An intrusion detection system (IDS) is used to identify the said malicious activity[7]. The most common method used for uncovering intrusions is the analysis of user activities[7, 13, 17]. However, the aforementioned method is laborious when done manually, since the data of user activities is massive in nature[6, 14]. To simplify the problem, automation through machine learning must be done.

A study by Mukkamala, Janoski, & Sung (2002)[17] shows how *support vector machine* (SVM) and *artificial neural network* (ANN) can be used to accomplish the said task. In machine learning, SVM separates two classes of data points using a hyperplane[5]. On the other hand, an ANN is a computational model that represents the human brain, and shows information is passed from a neuron to another[18].

An approach combining ANN and SVM was proposed by Alalshekmubarak & Smith[2], for time-series classification. Specifically, they combined *echo state network* (ESN, a variant of recurrent neural network or RNN) and SVM. This research presents a modified version of the aforementioned proposal, and use it for intrusion detection. The proposed model will use *recurrent neural network* (RNNs) with *gated recurrent units* (GRUs) in place of ESN. RNNs are used for analyzing and/or predicting sequential data, making it a viable candidate for intrusion detection[18], since network traffic data is sequential in nature.

2 METHODOLOGY

2.1 Machine Intelligence Library

Google TensorFlow[1] was used to implement the neural network models in this study – both the proposed and its comparator.

2.2 The Dataset

The 2013 Kyoto University honeypot systems' network traffic data[20] was used in this study. It has 24 statistical features[20]; (1) 14 features from the KDD Cup 1999 dataset[21], and (2) 10 additional features, which according to Song, Takakura, & Okabe (2006)[20],

might be pivotal in a more effective investigation on intrusion detection. Only 22 dataset features were used in the study.

2.3 Data Preprocessing

For the experiment, only 25% of the whole 16.2 GB network traffic dataset was used, i.e. ≈4.1 GB of data (from January 1, 2013 to June 1, 2013). Before using the dataset for the experiment, it was normalized first – standardization (for continuous data, see Eq. 1) and indexing (for categorical data), then it was binned (discretized).

$$z = \frac{X - \mu}{\sigma} \tag{1}$$

where X is the feature value to be standardized, μ is the mean value of the given feature, and σ is its standard deviation. But for efficiency, the `StandardScaler().fit_transform()` function of Scikit-learn[19] was used for the data standardization in this study.

For indexing, the categories were mapped to $[0, n-1]$ using the `LabelEncoder().fit_transform()` function of Scikit-learn[19].

After dataset normalization, the continuous features were binned (decile binning, a discretization/quantization technique). This was done by getting the 10^{th}, 20^{th}, ..., 90^{th}, and 100^{th} quantile of the features, and their indices served as their bin number. This process was done using the `qcut()` function of `pandas`[16]. Binning reduces the required computational cost, and improves the classification performance on the dataset[15]. Lastly, the features were one-hot encoded, making it ready for use by the models.

2.4 The GRU-SVM Neural Network Architecture

Similar to the work of Alalshekmubarak & Smith (2013)[2] and Tang (2013)[22], the present paper proposes to use SVM as the classifier in a neural network architecture. Specifically, a Gated Recurrent Unit (GRU) RNN (see Figure 1).

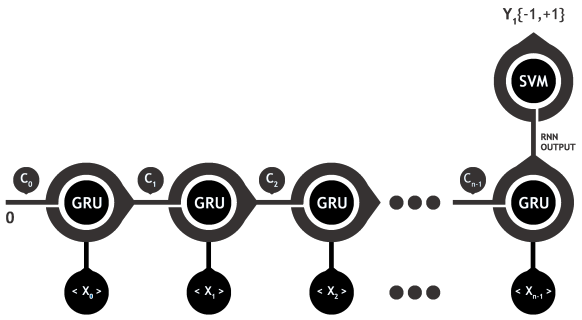

Figure 1: The proposed GRU-SVM architecture model, with $n-1$ GRU unit inputs, and SVM as its classifier.

For this study, there were 21 features used as the model input. Then, the parameters are learned through the gating mechanism of GRU[3] (Equations (2) to (5)).

$$z = \sigma(\mathbf{W}_z \cdot [h_{t-1}, x_t]) \tag{2}$$

$$r = \sigma(\mathbf{W}_r \cdot [h_{t-1}, x_t]) \tag{3}$$

$$\tilde{h}_t = tanh(\mathbf{W} \cdot [r_t * h_{t-1}, x_t]) \tag{4}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{5}$$

But with the introduction of SVM as its final layer, the parameters are also learned by optimizing the objective function of SVM (see Eq. 6). Then, instead of measuring the network loss using cross-entropy function, the GRU-SVM model will use the loss function of SVM (Eq. 6).

$$min \frac{1}{2}\|\mathbf{w}\|_1^2 + C\sum_{i=1}^{n} max(0, 1 - y_i'(\mathbf{w}^T\mathbf{x}_i + b_i)) \tag{6}$$

Eq. 6 is known as the unconstrained optimization problem of L1-SVM. However, it is not differentiable. On the contrary, its variation, known as the L2-SVM is differentiable and is more stable[22] than the L1-SVM:

$$min \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n} max(0, 1 - y_i'(\mathbf{w}^T\mathbf{x}_i + b_i))^2 \tag{7}$$

The L2-SVM was used for the proposed GRU-SVM architecture. As for the prediction, the decision function $f(x) = sign(\mathbf{w}\mathbf{x} + b)$ produces a score vector for each classes. So, to get the predicted class label y of a data x, the *argmax* function is used:

$$predicted\_class = argmax(sign(\mathbf{w}\mathbf{x} + b))$$

The *argmax* function will return the index of the highest score across the vector of the predicted classes.

The proposed GRU-SVM model may be summarized as follows:

(1) Input the dataset features $\{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbb{R}^m\}$ to the GRU model.
(2) Initialize the learning parameters `weights` and `biases` with arbitrary values (they will be adjusted through training).
(3) The cell states of GRU are computed based on the input features $\mathbf{x}_i$, and its learning parameters values.
(4) At the last time step, the prediction of the model is computed using the decision function of SVM: $f(x) = sign(\mathbf{w}\mathbf{x} + b)$.
(5) The loss of the neural network is computed using Eq. 7.
(6) An optimization algorithm is used for loss minimization (for this study, the Adam[12] optimizer was used). Optimization adjusts the `weights` and `biases` based on the computed loss.
(7) This process is repeated until the neural network reaches the desired accuracy or the highest accuracy possible. Afterwards, the trained model can be used for binary classification on a given data.

The program implementation of the proposed GRU-SVM model is available at https://github.com/AFAgarap/gru-svm.

Table 1: Hyper-parameters used in both neural networks.

Hyper-parameters	GRU-SVM	GRU-Softmax
Batch Size	256	256
Cell Size	256	256
Dropout Rate	0.85	0.8
Epochs	5	5
Learning Rate	1e-5	1e-6
SVM C	0.5	N/A

2.5 Data Analysis

The effectiveness of the proposed GRU-SVM model was measured through the two phases of the experiment: (1) training phase, and (2) test phase. Along with the proposed model, the conventional GRU-Softmax was also trained and tested on the same dataset.

The first phase of the experiment utilized 80% of total data points (≈3.2 GB, or 14, 856, 316 lines of network traffic log) from the 25% of the dataset. After normalization and binning, it was revealed through a high-level inspection that a duplication occurred. Using the `DataFrame.drop_duplicates()` of pandas[16], the 14, 856, 316-line data dropped down to 1, 898, 322 lines (≈40MB).

The second phase of the experiment was the evaluation of the two trained models using 20% of total data points from the 25% of the dataset. The testing dataset also experienced a drastic shrinkage in size – from 3, 714, 078 lines to 420, 759 lines (≈9 MB).

The parameters for the experiments are the following: (1) Accuracy, (2) Epochs, (3) Loss, (4) Run time, (5) Number of data points, (6) Number of false positives, (7) Number of false negatives. These parameters are based on the ones considered by Mukkamala, Janoski, & Sung (2002)[17] in their study where they compared SVM and a feed-forward neural network for intrusion detection. Lastly, the statistical measures for binary classification were measured (true positive rate, true negative rate, false positive rate, and false negative rate).

3 RESULTS

All experiments in this study were conducted on a laptop computer with Intel Core(TM) i5-6300HQ CPU @ 2.30GHz x 4, 16GB of DDR3 RAM, and NVIDIA GeForce GTX 960M 4GB DDR5 GPU. The hyperparameters used for both models were assigned by hand, and not through hyper-parameter optimization/tuning (see Table 1).

Both models were trained on 1,898,240 lines of network traffic data for 5 epochs. Afterwards, the trained models were tested to classify 420,608 lines of network traffic data for 5 epochs. Only the specified number of lines of network traffic data were used for the experiments as those are the values that are divisble by the batch size of 256. The class distribution of both training and testing dataset is specified in Table 2.

The experiment results are summarized in Table 3. Although the loss for both models were recorded, it will not be a topic of further discussion as they are not comparable since they are in different scales. Meanwhile, Tables 4 & 5 show the statistical measures for binary classification by the models during training and testing.

Figure 2 shows that for 5 epochs on the 1,898,240-line network traffic data (a total exposure of 9,491,200 to the training dataset), the GRU-SVM model was able to finish its training in 16 minutes and 43 seconds. On the other hand, the GRU-Softmax model finished its training in 17 minutes and 11 seconds.

Figure 3 shows that for 5 epochs on the 420,608-line network traffic data (a total test prediction of 2,103,040), the GRU-SVM model was able to finish its testing in 1 minute and 22 seconds. On the other hand, the GRU-Softmax model finished its testing in 1 minute and 40 seconds.

Table 2: Class distribution of training and testing dataset.

Class	Training data	Testing data
Normal	794,512	157,914
Intrusion detected	1,103,728	262,694

Table 3: Summary of experiment results on both GRU-SVM and GRU-Softmax models.

Parameter	GRU-SVM	GRU-Softmax
No. of data points – Training	1,898,240	1,898,240
No. of data points – Testing	420,608	420,608
Epochs	5	5
Accuracy – Training	≈81.54%	≈63.07%
Accuracy – Testing	≈84.15%	≈70.75%
Loss – Training	≈131.21	≈0.62142
Loss – Testing	≈129.62	≈0.62518
Run time – Training	≈16.72mins	≈17.18mins
Run time – Testing	≈1.37mins	≈1.67mins
No. of false positives – Training	889,327	3,017,548
No. of false positives – Testing	192,635	32,255
No. of false negatives – Training	862,419	487,175
No. of false negatives – Testing	140,535	582,105

Table 4: Statistical measures on binary classification: Training performance of the GRU-SVM and GRU-Softmax models.

Parameter	GRU-SVM	GRU-Softmax
True positive rate	≈84.3726%	≈91.1721%
True negative rate	≈77.6132%	≈24.0402%
False positive rate	≈22.3867%	≈75.9597%
False negative rate	≈15.6273%	≈8.82781%

Table 5: Statistical measures on binary classification: Testing performance of the GRU-SVM and GRU-Softmax models.

Parameter	GRU-SVM	GRU-Softmax
True positive rate	≈89.3005%	≈55.6819%
True negative rate	≈75.6025%	≈95.9149%
False positive rate	≈10.6995%	≈4.08513%
False negative rate	≈24.3975%	≈44.3181%

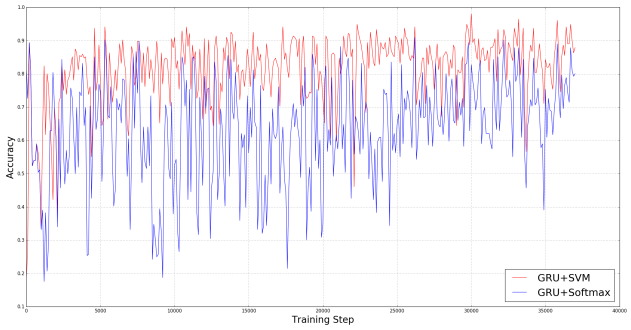

Figure 2: Training accuracy of the proposed GRU-SVM model, and the conventional GRU-Softmax model.

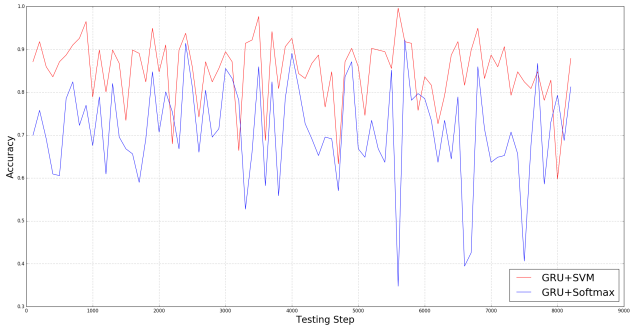

Figure 3: Testing accuracy of the proposed GRU-SVM model, and the conventional GRU-Softmax model.

4 DISCUSSION

The empirical evidence presented in this paper suggests that SVM outperforms Softmax function in terms of prediction accuracy, when used as the final output layer in a neural network. This finding corroborates the claims by Alalshekmubarak & Smith (2013)[2] and Tang (2013)[22], and supports the claim that SVM is a more practical approach than Softmax for binary classification. Not only did the GRU-SVM model outperform the GRU-Softmax in terms of prediction accuracy, but it also outperformed the conventional model in terms of training time and testing time. Thus, supporting the theoretical implication as per the respective algorithm complexities of each classifier.

The reported training accuracy of ≈81.54% and testing accuracy of ≈84.15% posits that the GRU-SVM model has a relatively stronger predictive performance than the GRU-Softmax model (with training accuracy of ≈63.07% and testing accuracy of ≈70.75%). Hence, we propose a theory to explain the relatively lower performance of Softmax compared to SVM in this particular scenario. First, SVM was designed primarily for binary classification[5], while Softmax is best-fit for multinomial classification[11]. Building on the premise, SVM does not care about the individual scores of the classes it predicts, it only requires its margins to be satisfied[11]. On the contrary, the Softmax function will always find a way to improve its predicted probability distribution by ensuring that the correct class has the higher/highest probability, and the incorrect classes

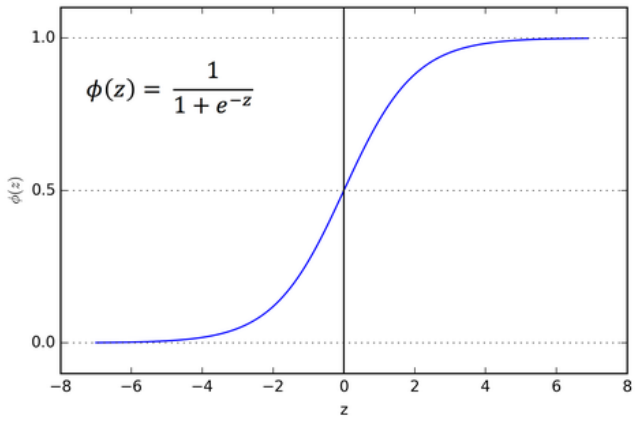

Figure 4: Image from [9]. Graph of a sigmoid σ function.

have the lower probability. This behavior of the Softmax function is exemplary, but excessive for a problem like binary classification. Given that the sigmoid σ function is a special case of Softmax (see Eq. 8-9), we can refer to its graph as to how it classifies a network output.

$$\sigma(y) = \frac{1}{1+e^{-y}} = \frac{1}{1+\frac{1}{e^{y}}} = \frac{1}{\frac{e^{y}+1}{e^{y}}} = \frac{e^{y}}{1+e^{y}} = \frac{e^{y}}{e^{0}+e^{y}} \tag{8}$$

$$softmax(y) = \frac{e^{y_i}}{\sum_{i=0}^{n=1} e^{y_i}} = \frac{e^{y_i}}{e^{y_0}+e^{y_1}} \tag{9}$$

It can be inferred from the graph of sigmoid σ function (see Figure 4) that y values tend to respond less to changes in x. In other words, the gradients would be small, which gives rise to the "vanishing gradients" problem. Indeed, one of the problems being solved by LSTM, and consequently, by its variants such as GRU[3, 8]. This behavior defeats the purpose of GRU and LSTM solving the problems of a traditional RNN. We posit that this is the cause of misclassifications by the GRU-Softmax model.

The said erroneous manner of the GRU-Softmax model reflects as a favor for the GRU-SVM model. But the comparison of the exhibited predictive accuracies of both models is not the only reason for the practicality in choosing SVM over Softmax in this case. The amount of training time and testing time were also considered. As their computational complexities suggest, SVM has the upper hand over Softmax. This is because the algorithm complexity of the predictor function in SVM is only $O(1)$. On the other hand, the predictor function of Softmax has an algorithm complexity of $O(n)$. As results have shown, the GRU-SVM model also outperformed the GRU-Softmax model in both training time and testing time. Thus, it corroborates the respective algorithm complexities of the classifiers.

5 CONCLUSION AND RECOMMENDATION

We proposed an amendment to the architecture of GRU RNN by using SVM as its final output layer in a binary/non-probabilistic classification task. This amendment was seen as viable for the fast prediction time of SVM compared to Softmax. To test the model,

we conducted an experiment comparing it with the established GRU-Softmax model. Consequently, the empirical data attests to the effectiveness of the proposed GRU-SVM model over its comparator in terms of predictive accuracy, and training and testing time.

Further work must be done to validate the effectiveness of the proposed GRU-SVM model in other binary classification tasks. Extended study on the proposed model for a faster multinomial classification would prove to be prolific as well. Lastly, the theory presented to explain the relatively low performance of the Softmax function as a binary classifier might be a pre-cursor to further studies.

6 ACKNOWLEDGMENT

An appreciation to the open source community (Cross Validated, GitHub, Stack Overflow) for the virtually infinite source of information and knowledge; to the Kyoto University for their intrusion detection dataset from their honeypot system.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

[2] A. Alalshekmubarak and L.S. Smith. 2013. A Novel Approach Combining Recurrent Neural Network and Support Vector Machines for Time Series Classification. In *Innovations in Information Technology (IIT), 2013 9th International Conference on*. IEEE, 42–47.

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[4] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*. 577–585.

[5] C. Cortes and V. Vapnik. 1995. Support-vector Networks. *Machine Learning* 20.3 (1995), 273–297. https://doi.org/10.1007/BF00994018

[6] Jeremy Frank. 1994. Artificial intelligence and intrusion detection: Current and future directions. In *Proceedings of the 17th national computer security conference*, Vol. 10. Baltimore, USA, 1–12.

[7] Anup K Ghosh, Aaron Schwartzbard, and Michael Schatz. 1999. Learning Program Behavior Profiles for Intrusion Detection.. In *Workshop on Intrusion Detection and Network Monitoring*, Vol. 51462. 1–13.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[9] Yohan Grember (https://stackoverflow.com/users/7672928/yohan grember). [n. d.]. Binary classification with Softmax. Stack Overflow. ([n. d.]). arXiv:https://stackoverflow.com/questions/45793856 https://stackoverflow.com/questions/45793856 URL:https://stackoverflow.com/questions/45793856 (version: 2017-08-21).

[10] Juniper. May 12, 2015. Cybercrime will cost Businesses over $2 Trillion by 2019. https://www.juniperresearch.com/press/press-releases/cybercrime-cost-businesses-over-2trillion. (May 12, 2015). Accessed: May 6, 2017.

[11] Anrej Karpathy. [n. d.]. CS231n Convolutional Neural Networks for Visual Recognition. http://cs231n.github.io/. ([n. d.]).

[12] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Sandeep Kumar and Eugene H Spafford. 1994. An application of pattern matching in intrusion detection. (1994).

[14] MIT Lincoln Laboratory. 1999. 1999 DARPA Intrusion Detection Evaluation Data Set. https://www.ll.mit.edu/ideval/data/1999data.html. (1999).

[15] Jonathan L Lustgarten, Vanathi Gopalakrishnan, Himanshu Grover, and Shyam Visweswaran. 2008. Improving classification performance with discretization on biomedical datasets. In *AMIA annual symposium proceedings*, Vol. 2008. American Medical Informatics Association, 445.

[16] Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman (Eds.). 51 – 56.

[17] Srinivas Mukkamala, Guadalupe Janoski, and Andrew Sung. 2002. Intrusion detection: support vector machines and neural networks. In *proceedings of the IEEE International Joint Conference on Neural Networks (ANNIE), St. Louis, MO*. 1702–1707.

[18] M. Negnevitsky. 2011. *Artificial Intelligence: A Guide to Intelligent Systems* (3rd ed.). Pearson Education Ltd., Essex, England.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[20] Jungsuk Song, Hiroki Takakura, and Yasuo Okabe. 2006. Description of kyoto university benchmark data. *Available at link: http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v5.pdf.[Accessed on 15 March 2016]* (2006).

[21] J Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip K Chan. 2000. Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection. *Results from the JAM Project by Salvatore* (2000).

[22] Yichuan Tang. 2013. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239* (2013).

[23] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745* (2015).

[24] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical Attention Networks for Document Classification.. In *HLT-NAACL*. 1480–1489.