Adapting New Data In Intrusion Detection Systems

Aslıhan Akyol *aslihan.ozkaya@gmail.com*
Independent Researcher
Antalya, Turkey

Bekir Karlık *bkarlik@hotmail.com*
McGill University, Neurosurgical Simulation Research
& Training Centre, Montréal, QC, Canada

Barış Koçer *bariskocer@selcuk.edu.tr*
Selcuk University, Department of Computer Engineering,
Konya, Turkey

Abstract

Most of the introduced anomaly intrusion detection system (IDS) methods focus on achieving better detection rates and lower false alarm rates. However, when it comes to real-time applications many additional issues come into the picture. One of them is the training datasets that are continuously becoming outdated. It is vital to use an up-to-date dataset while training the system. But the trained system will become insufficient if network behaviors change. As well known, frequent alteration is in the nature of computer networks. On the other hand it is costly to continually collect and label datasets while frequently training the system from scratch and discarding old knowledge is a waste. To overcome this problem, we propose the use of transfer learning which benefits from the previous gained knowledge. The carried out experiments stated that transfer learning helps to utilize previously obtained knowledge, improves the detection rate and reduces the need to recollect the whole dataset.

Keywords: Intrusion Detection Systems, Transfer Learning, Genetic Transfer Learning, Genetic Algorithms, Artificial Neural Networks.

1. INTRODUCTION

Intrusion Detection Systems (IDSs)monitors the actions taken in a system and decides whether these actions are attacks or legitimate actions [1], [2].Signature based and anomaly based IDSs are the two major types of IDSs [3], [4].The signature based IDSs analyze the characteristics of the actions and compare them with known attack signatures. Signature based IDSs are able to detect intrusions with very low False Alarm Rates (FAR) which means recognizing legitimate actions as intrusions. However signature based IDSs are unable to detect unknown attacks[5]. On the other hand, anomaly based IDSs use wide variety of data mining techniques, statistical modeling and hidden markov models to identify actions that appear to be anomalous with respect to legitimate actions. The advantage of this method is that the system is able to detect unknown attacks[5] but produces high FARs [6].

In anomaly based IDSs, the system is trained using datasets of actions (network packets, log files etc.) where each action is labeled as either legitimate or attack. The most used dataset is the KDD`99[7], [8] which is a version of the DARPA IDEVAL dataset[9], collected in 1998, and used for the Third International Knowledge Discovery and Data Mining Tools Competition. This dataset is mostly used to test and compare different IDS methods. However to have the IDS work in a real environment the IDS should be trained with an up-to-date dataset. But after some time this dataset may become outdated because legitimate network behavior may change or new attack behavior may appear[10], [11], [12], [13]. Therefore the IDS will turn out to be outdated after some time as well[11]. On the other hand the availability of labeled data used to train IDS is usually a major issue [10]. Collecting and labeling new data is a costly process and throwing old data away is a waste[14], [15]. Therefore to train and keep the anomaly IDS updated is difficult. Instead of collecting new data and train/build the IDS from scratch, we suggest to transfer the previous knowledge by using transfer learning so that we could reduce the need and effort to recollect the training data, decrease the time to train the system and obtain higher detection rates.

The rest of this paper is organized as follows: in section 2 research methodology is introduced, the dataset used in the experiments is described in section 3, we stated the experimental results and analysis in section 4 and concluded in section 5.

2. RESEARCH METHODOLOGY

2.1. Transfer Learning

Traditional data mining and machine learning algorithms use labeled or unlabeled data to train the system and then perform predictions on new data that has unknown class labels[14], [16], [17]. On the other hand, in real time applications and classical methods once the data is outdated new data should be re-collected and the system should be retrained from scratch using the newly collected dataset [18]. This process is a costly process and throwing old data is a waste of time. However transfer learning provides the system to benefit from previous knowledge[14], [15], [19]. This helps to make use of previous dataset and the system does not have to be trained all over again from scratch. Moreover the system can be trained only with few up-to-date training data by making use of the previous knowledge[14].

2.2. Genetic Transfer Learning

2.2.1. Genetic Algorithms

Genetic algorithm (GA) is a search algorithm that simulates a natural selection to optimize a problem [20], [21]. GA is widely applied on diverse areas such as machine learning, chemistry, economy, algebra, music generation and strategy planning [22], [23]. In GA a population of candidate subsets is evolved to obtain candidate solutions also called as individual. Each individual consists of genes that can be either numerical or binary values [24]. A fitness function (F) is used to measure the suitability of the solutions. The solutions with the best fitness values have higher probability to be selected with the roulette wheel for the next generation [24]. Then crossover will be applied on some individuals of next generation where each individual is selected under a pre-determined probability value (or *crossover rate*) [17], [23], [24]. Also the crossover point is determined randomly for each crossover pair. Then mutation, in which a selected gene is replaced by a random value, is applied on some genes determined by a pre-determined *mutation rate parameter*. As a result a new generation is created. The whole process (creating a new generation) is going to be repeated until a pre-determined iteration or fitness value is met, see figure 1.

2.2.2. Genetic Transfer

In genetic algorithms every time when a new generation is created the old generation is killed. However the killed generations may contain suitable solutions for similar optimization problems [23]. Therefore in genetic transfer learning, before killing each generation some selected solutions are saved into a solution pool. These selected solutions are usually one with the best, one with the median and one with the worst fitness value [17], [23]. The solution pool is later used on a different but similar problem, and this process is named as transferring knowledge in genetic algorithms or genetic transfer learning.

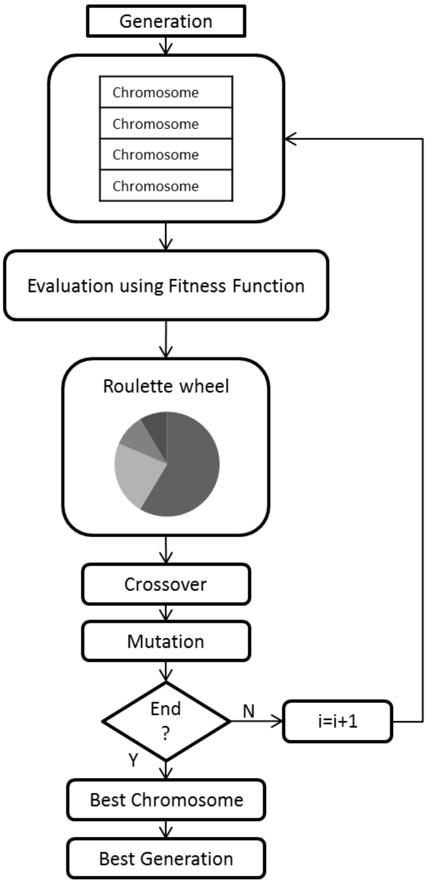


FIGURE 1: Genetic Algorithm Flowchart[24].

2.3. Genetic – ANN Hybrid Transfer Learning

In genetic transfer learning the first generation is usually created randomly. However we used artificial neural network (ANN) with the back-propagation algorithm, therefore the first generation is created from the weights which are obtained during the training phase in ANN. Furthermore, the fitness function is the back-propagation algorithm, in which the best fitness value is zero representing the error rate.

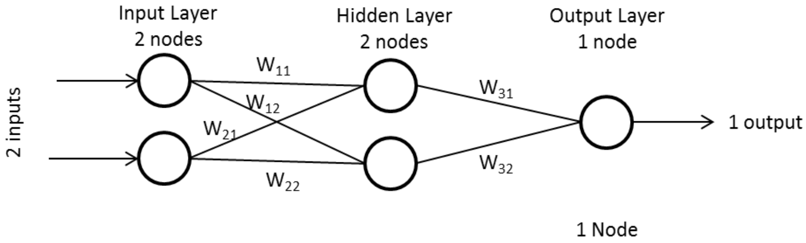
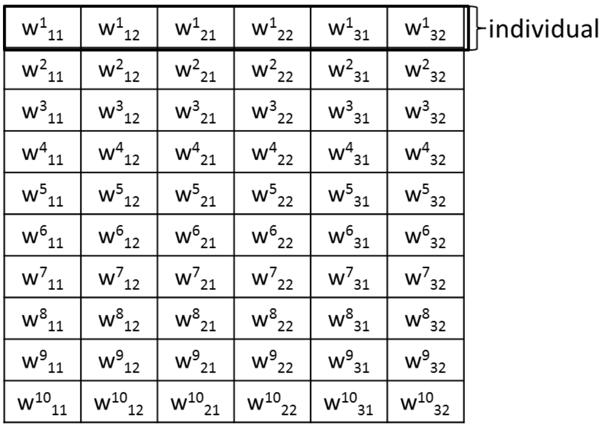


FIGURE 2: ANN Structure.

For instance let's assume that the structure of the ANN is as shown in figure 2 and it is trained with 10 iterations. 6 weights ($w_{11,}w_{12,}w_{21,}w_{22,}w_{31,}w_{32}$) are obtained during each iteration. If we save the weights of each iteration the dimensions of the generation will be 10 x 6 (see figure 3) where each set of 6 weights is called as individual.


individual

w^{1}_{11}	w^{1}_{12}	w^{1}_{21}	w^{1}_{22}	w^{1}_{31}	w^{1}_{32}
w^{2}_{11}	w^{2}_{12}	w^{2}_{21}	w^{2}_{22}	w^{2}_{31}	w^{2}_{32}
w^{3}_{11}	w^{3}_{12}	w^{3}_{21}	w^{3}_{22}	w^{3}_{31}	w^{3}_{32}
w^{4}_{11}	w^{4}_{12}	w^{4}_{21}	w^{4}_{22}	w^{4}_{31}	w^{4}_{32}
w^{5}_{11}	w^{5}_{12}	w^{5}_{21}	w^{5}_{22}	w^{5}_{31}	w^{5}_{32}
w^{6}_{11}	w^{6}_{12}	w^{6}_{21}	w^{6}_{22}	w^{6}_{31}	w^{6}_{32}
w^{7}_{11}	w^{7}_{12}	w^{7}_{21}	w^{7}_{22}	w^{7}_{31}	w^{7}_{32}
w^{8}_{11}	w^{8}_{12}	w^{8}_{21}	w^{8}_{22}	w^{8}_{31}	w^{8}_{32}
w^{9}_{11}	w^{9}_{12}	w^{9}_{21}	w^{9}_{22}	w^{9}_{31}	w^{9}_{32}
w^{10}_{11}	w^{10}_{12}	w^{10}_{21}	w^{10}_{22}	w^{10}_{31}	w^{10}_{32}

FIGURE 3: One generation obtained from the ANN training with 10 iterations (w^{x}_{y}, x: iteration number, y: index number of weight).

The rest of the process is the same with genetic transfer. The weights are going to be used as the initial generation. This generation will be passed through the genetic algorithm process (figure 1) and a new generation is going to be created. Before killing the old generation individuals with the best, median and worst fitness values will be saved in the solution pool.

3. DATASET AND PREPROCESSING

The 10% KDD`99 dataset has been used in the experiments. This dataset has 41 attributes (see TABLE 1) which are either string or decimal values and has data on 22 different attacks plus normal network packets. In [25], which is under review, a discernibility function based feature selection has been applied to obtain the best feature subsets for each attack category [26]. The feature subset, that gave the highest detection rate for normal packets, consists only of the following 14 attributes: A2, A4, A5, A6, A10, A12, A23, A24, A33, A35, A37, A38, A39, A40 which are shown in bold in TABLE 1.

TABLE 1: Attributes and their index number of each record of the KDD Cup dataset(The bold attributes are used in our experiments.)

Index	Title	Index	Title	Index	Title	Index	Title
A1	Duration	A11	num_failed_logins	A21	is_host_login	A31	srv_diff_host_rate
A2	**protocol_type**	**A12**	**logged_in**	A22	is_guest_login	A32	dst_host_count
A3	Service	A13	num_compromised	**A23**	**count**	**A33**	**dst_host_srv_count**
A4	**Flag**	A14	root_shell	**A24**	**srv_count**	A34	dst_host_same_srv_rate
A5	**src_bytes**	A15	su_attempted	A25	serror_rate	**A35**	**dst_host_diff_srv_rate**
A6	**dst_bytes**	A16	num_root	A26	srv_serror_rate	A36	dst_host_same_src_port_rate
A7	Land	A17	num_file_creations	A27	rerror_rate	**A37**	**dst_host_srv_diff_host_rate**
A8	wrong_fragment	A18	num_shells	A28	srv_rerror_rate	**A38**	**dst_host_serror_rate**
A9	Urgent	A19	num_access_files	A29	same_srv_rate	**A39**	**dst_host_srv_serror_rate**
A10	**Hot**	A20	num_outbound_cmds	A30	diff_srv_rate	**A40**	**dst_host_rerror_rate**
						A41	dst_host_srv_rerror_rate

The 10% KDD`99 dataset has 494.021 records with many duplicates. In our experiments we removed the duplicated data and the number of records has been dropped to 145.585. Then we converted the attributes with text data to numeric values and applied normalization by scaling each attribute between 0 and 1.

4. EXPERIMENTAL RESULTS AND ANALYSIS

The dataset used in our experiments has 14 attributes and one output. The output is a binary value (as either attack or normal). According to the dataset we have 14 nodes in the input layer and one node in the output layer of the ANN, whereas the number of nodes in the hidden layer is 28 (figure 4).

The ANN is a fully connected network, therefore there are 14×28=392 weights between the input and hidden layer and 28×1=28 weights between the hidden and output layer. As a result there are a total of 420 weights. In other words there are 420 genes in each individual.

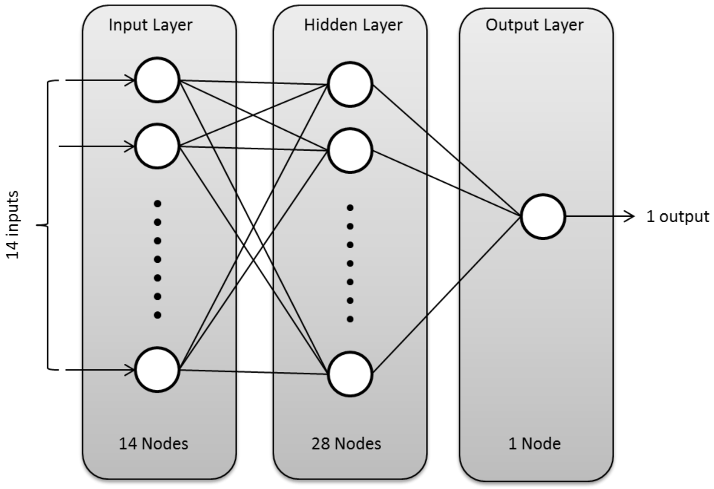


FIGURE 4:The structure of the artificial neural network used in our experiments.

We removed all records that belong to one of the attack types (back, ipsweep, neptune, nmap, pod, portsweep, see TABLE 2) each one at a time from the 10% KDD`99 dataset and used the decreased dataset to create the solution pool. We called the decreased dataset as the outdated dataset. Then we added the removed records back and called the new one as the updated dataset (see table 3). In experiments 7-9 (see TABLE 4), all packets that use icmp protocol were removed from the 10% KDD`99 dataset to create the outdated dataset. Then we added normal packets, attacks and both (attacks&normal) that use icmp protocol respectively to create updated datasets. The main idea of adding the removed data is to simulate a real-time ANN training where a newly collected dataset becomes outdated as soon as a new benign network behavior or a new attack appears in the network. We showed that the training time decreases if we transfer knowledge from the training with the outdated dataset.

TABLE 2: Attack Names and Counts.

Name	Count	Name	Count
back.	968	perl.	3
buffer_overflow.	30	phf.	4
ftp_write.	8	pod.	206
guess_passwd.	53	portsweep.	416
imap.	12	rootkit.	10
ipsweep.	651	satan.	906
land.	19	smurf.	641
loadmodule.	9	spy.	2
multihop.	7	teardrop.	918
neptune.	51820	warezclient.	893
nmap.	158	warezmaster.	20

TABLE 3: The distinction between outdated and updated datasets used in experiments 1-6.

	The attack name that is extracted from the outdated and included into the updated dataset
Experiment 1	back attack
Experiment 2	ipsweep attack
Experiment 3	neptune attack
Experiment 4	nmap attack
Experiment 5	pod attack
Experiment 6	portsweep attack

TABLE 4: The distinction between outdated and updated datasets used in experiments 7-9.

	Outdated dataset does not have	**Updated dataset has**
Experiment 7	any packet that use icmp protocol	normal (benign) packets that use icmp protocol
Experiment 8		attacks that use icmp protocol
Experiment 9		normal (benign) & attack packets that use icmp protocol

The process of applying transfer learning is done as following: The ANN is trained on the outdated dataset with 100 iterations and the weights obtained from iteration are stored. At the end of this process the first generation is completed. This generation is passed through genetic algorithms to create new generations. Before killing the old generation, two individuals (with the best and median fitness values) from the old generation are saved into the solution pool. Then again the new generation is passed through genetic algorithms. This process is repeated 100 times. As a result the solution pool has 200 individuals each with 420 weights (genes). The crossover rate and the mutation rate parameters are selected as 0.7 and 0.01 respectively.

The solution pool is used when we have anew(updated)dataset which is similar to the outdated dataset. Each individual is applied on the updated dataset with ANN. The individual with the fittest result is used as the initial weights at the ANN training process. We compared the cumulative errors (see Eq. (1)) of the genetic &ANN hybrid transfer learning with the classical ANN. Lower cumulative error is better because lower cumulative error provides higher detection rates. We showed that transfer learning helps the system to benefit from the previously obtained knowledge.

$$\text{Cumulative Error} = \sum_{n=1}^{N}(d - o) \qquad (1)$$

Where *d*, *o* and *N* are the desired output, obtained output and number of inputs respectively. It can be clearly seen that in all experiments (Figure 5-13) the transfer learning method started with better cumulative error values. Additionally, even the beginning error values obtained with transfer learning method in experiments 4-7 (Figure 8-11) were better than or very close to the error values obtained after 100 iterations with the classical ANN. This proves that genetic & ANN hybrid transfer learning decreases the time to train the system and provides better detection rates.

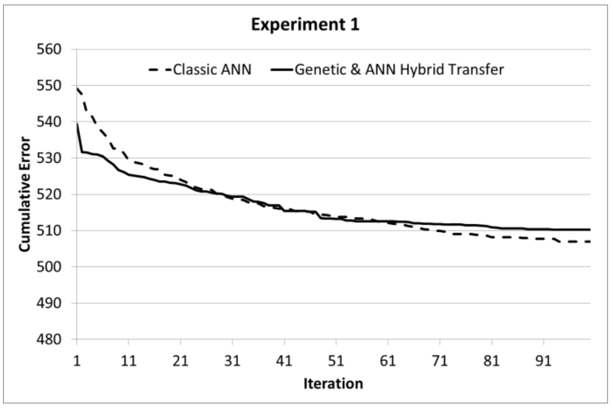


FIGURE 5: Difference between Classical ANN and Genetic & ANN Hybrid Transfer, outdated dataset don't have any back attack while updated dataset has 968 back attacks.

In experiment 1 (Figure 5), the classical ANN showed slightly better result but still very similar with genetic & ANN hybrid method. While in experiment 3 (Figure 7) genetic & ANN transfer learning started with a clear advantage, but after 100 iterations classical ANN showed slightly better result than genetic & ANN Hybrid transfer learning method. In all other experiments (# 2, 4-9) the genetic & ANN hybrid transfer learning method shows clearly better results than the classical ANN. These results makes it obvious that the transfer learning method helps to utilize previously obtained knowledge and improves the detection rate. It can also reduce the need to recollect the whole dataset if we may be able to only collect the data packets of the new network behavior.

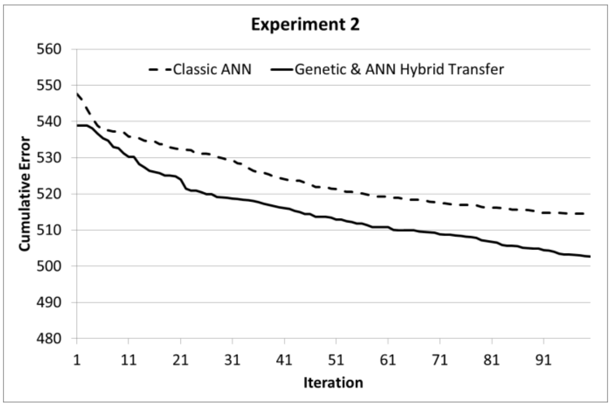


FIGURE 6: Difference between Classical ANN and Genetic & ANN Hybrid Transfer, outdated dataset don't have any ipsweep attack while updated dataset has 651 ipsweep attacks.

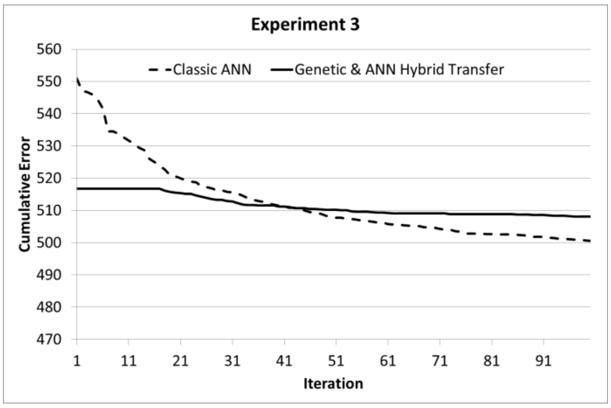


FIGURE 7: Difference between Classical ANN and Genetic & ANN Hybrid Transfer, outdated dataset don't have any neptune attack while updated dataset has 51820 neptune attacks.

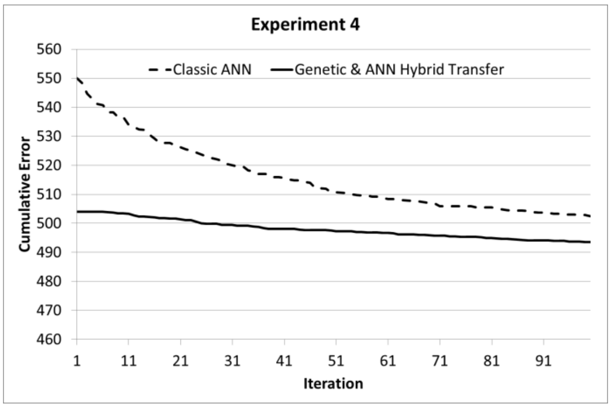


FIGURE 8: Difference between Classical ANN and Genetic & ANN Hybrid Transfer, outdated dataset don't have any nmap attack while updated dataset has 158 nmap attacks.

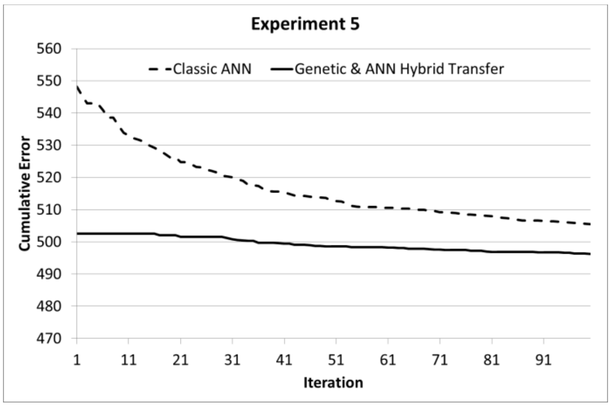


FIGURE 9: Difference between Classical ANN and Genetic & ANN Hybrid Transfer, outdated dataset don't have any pod attack while updated dataset has 206 pod attacks.

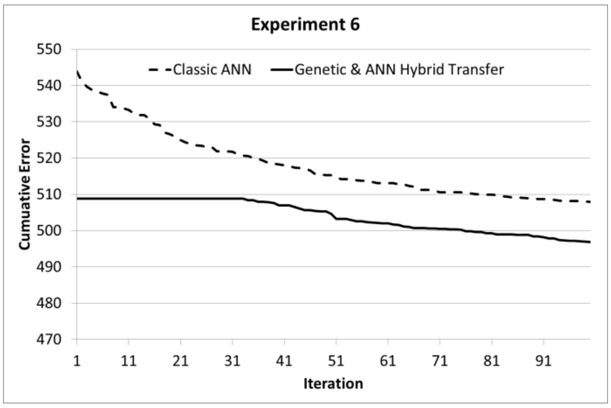


FIGURE 10: Difference between Classical ANN and Genetic & ANN Hybrid Transfer, outdated dataset don't have any portsweep attack while updated dataset has 416 portsweep attacks.

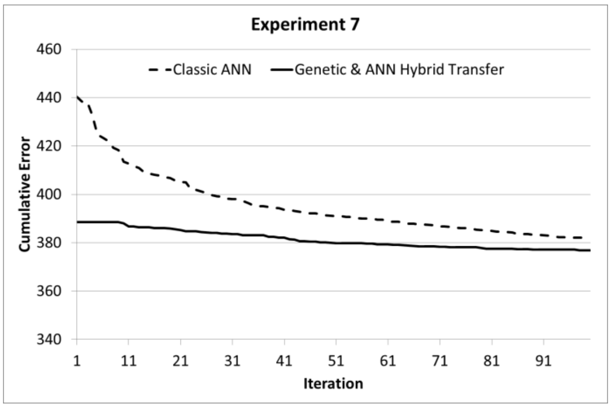


FIGURE 11: Difference between Classical ANN and Genetic & ANN Hybrid Transfer, outdated dataset don't have any packets with icmp protocol while updated dataset has normal packets with icmp protocol.

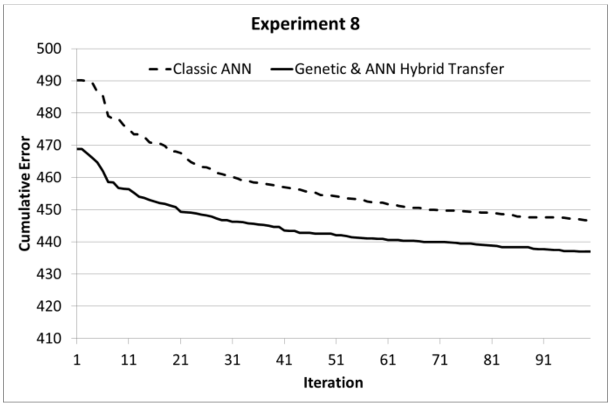


FIGURE 12: Difference between Classical ANN and Genetic & ANN Hybrid Transfer, outdated dataset don't have any packets with icmp protocol while updated dataset has attack packets with icmp protocol.

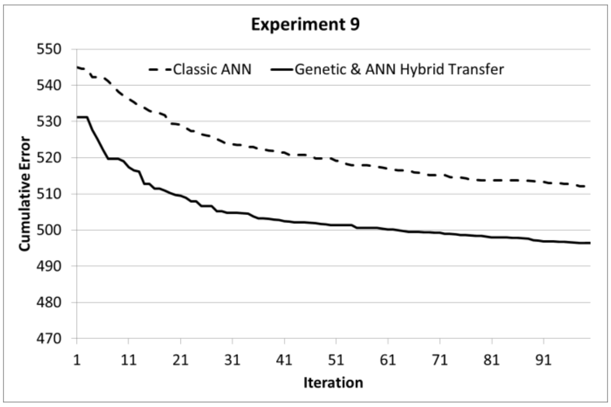


FIGURE 13: Difference between Classical ANN and Genetic & ANN Hybrid Transfer, outdated dataset don't have any packets with icmp protocol while updated dataset has normal & attack packets with icmp protocol.

5. CONCLUSION

In this study we analyzed the effects of transferring knowledge in anomaly based intrusion detection systems (IDS). In our previous study, we have proposed a novel multilevel hybrid classifier that uses different feature sets on each classifier. It has provided better performance than well-known individual classifiers and other proposed hybrid classifiers by using KDD'99 Cup and ISCX datasets[25]. In this study, we discussed that collecting dataset is a costly process and throwing outdated data is a waste. However using transfer learning provides to convert the knowledge obtained from the old dataset into an advantage. We showed that if we transfer previous knowledge, the new training gives lower (better) cumulative errors in sooner iterations. In other words the training process becomes faster if we train the system until a pre-determined error value instead of a pre-determined iteration. In our future work we are going to study transfer learning on the KDD test set by making use of the knowledge obtained from the training set.

6. REFERENCES

[1] H. Debar, M. Dacier, and A. Wespi, “Towards a taxonomy of intrusion-detection systems,” *Comput. Networks*, vol. 31, no. 8, pp. 805–822, Apr. 1999.

[2] N. Weng, L. Vespa, and B. Soewito, “Deep packet pre-filtering and finite state encoding for adaptive intrusion detection system,” *Comput. Networks*, vol. 55, no. 8, pp. 1648–1661, Jun. 2011.

[3] S. Axelsson, “Intrusion detection systems: A survey and taxonomy,” Göteborg, Sweden, 2000.

[4] S. Lee, D. Kim, and J. Park, “A survey and taxonomy of lightweight intrusion detection systems,” *J. Internet Serv. Inf. Secur.*, vol. 2, no. 1/2, pp. 119–13, 2012.

[5] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, “A survey of intrusion detection techniques in Cloud,” *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 42–57, Jan. 2013.

[6] E. Lundin and E. Jonsson, “Anomaly-based intrusion detection: privacy concerns and other problems,” *Comput. Networks*, vol. 34, no. 4, pp. 623–640, Oct. 2000.

[7] “KDD Cup 1999 Data,” *The UCI KDD Archive*, 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. [Accessed: 05-Jul-2013].

[8] A. Özkaya and B. Karlık, “Protocol type based intrusion detection using RBF neural network,” *Int. J. Artif. Intell. Expert Syst.*, vol. 3, no. 4, pp. 90–99, 2012.

[9] MIT, “MIT Lincoln Laboratory: Communications & Information Technology.” [Online]. Available: http://www.ll.mit.edu/mission/communications/ist/index.html. [Accessed: 21-Jun-2014].

[10] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly Detection : A Survey,” *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–72, 2009.

[11] H. Bensefia and N. Ghoualmi, “A new approach for adaptive intrusion detection,” *2011 Seventh Int. Conf. Comput. Intell. Secur.*, pp. 983–987, Dec. 2011.

[12] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhuhast, “IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection” December 11, 2017, vol. 6, pp. 1792–1806, 2018.

[13] A Pektaş, and T. Acarman, “A deep learning method to detect network intrusion through flow-based features” *International Journal of Network Management*, special issue paper, pp. 1–19, 2018.

[14] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[15] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, “Boosting for transfer learning,” in *Proceedings of the 24th international conference on Machine learning - ICML ’07*, 2007, pp. 193–200.

[16] E. Baralis, S. Chiusano, and P. Garza, “A lazy approach to associative classification,” *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 2, pp. 156–171, Feb. 2008.

[17] B. Koçer and A. Arslan, “Genetic transfer learning,” *Expert Syst. Appl.*, vol. 37, no. 10, pp. 6997–7002, Oct. 2010.

[18] S. Gou, Y. Wang, L. Jiao, J. Feng, and Y. Yao, “Distributed transfer network learning based intrusion detection,” in *2009 IEEE International Symposium on Parallel and Distributed Processing with Applications*, 2009, pp. 511–515.

[19] A. J. Storkey, “When training and test sets are different : Characterising learning transfer,” *Dataset shift Mach. Learn.*, pp. 3–28, 2013.

[20] U. Maulik and S. Bandyopadhyay, “Genetic algorithm-based clustering technique,” *Pattern Recognit.*, vol. 33, no. 9, pp. 1455–1465, Sep. 2000.

[21] H.-T. Lin, Y.-Y. Lin, and J.-W. Chiang, “Genetic-based real-time fast-flux service networks detection,” *Comput. Networks*, vol. 57, no. 2, pp. 501–513, Feb. 2013.

[22] M. Srinivas and L. M. Patnaik, “Genetic algorithms: A survey,” *Computer (Long. Beach. Calif).*, vol. 27, no. 6, pp. 17–26, Jun. 1994.

[23] B. Koçer, “Transfer öğrenmede yeni yaklaşımlar,” PhD thesis (in Turkish), Selcuk University, 2012.

[24] D. Hermawanto, “Genetic algorithm for solving simple mathematical equality problem,” *arXiv Prepr. arXiv1308.4675*, 2013.

[25] A. Akyol, M. Hacıbeyoğlu, and B. Karlık, “Design of multilevel hybrid classifier with variant feature sets for intrusion detection system” *IEICE Transactions on Information and Systems,* vol. 99, no.7, pp.1810-1821, 2016.

[26] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, “Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation,” in *DARPA Information Survivability Conference and Exposition, 2000. DISCEX’00. Proceedings*, vol. 2, pp. 12–26.