SindhUniv. Res. Jour. (Sci. Ser.) Vol. 51 (04) 671-676 (2019)

http://doi.org/10.26692/sujo/2019.12.106

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

Experiments with Neural Networks Training Algorithms for DDoS attacks Detection

F. A. JOKHIO, A. A. JAMALI*,++, M. R. ANJUM**, K. KANWAR*, S. AFRIDI***

Department of Computer Engineering, Quaid-e-Awam University of Engineering, Science and Technology, Nawabshah, Sindh, Pakistan

Received 13th October 2018 and Revised 4th May 2019

Abstract: Attacks on servers to bring their services down is growing rapidly. There are several possible methods through which attackers can attack a server. Distributed denials of service (DDoS) attacks are one of these attacks types in computing environments. In this paper, we discuss the possible ways of DDoS attacks detection using artificial neural network based techniques. Here a feed forward neural network is designed to detect DDoS attacks in a network based environment. Performance of feed forward neural network is analyzed by applying three different training algorithms. These algorithms are (1) Scaled Conjugate Gradient (2) Bayesian Regularization and (3) Gradient Descent. Results show that Bayesian Regularization algorithm is better among these three training algorithms and it has less number of misclassifications.

Keywords: DDoS, Neural Networks**,** Training Algorithms, Training Parameters, Confusion Matrix. Feed Forward Neural Network

1. INTRODUCTION

At present it is possible to provide computing as well as networking services to end user with low cost and less efforts. In a network several computers are connected with each other. A cloud computing environment has also several computing as well as storage resources connected with some sort of network. In such network based environment it is always possible that a malicious user may target a server and bring it down completely or degrade its services. These kinds of attacks on networks and servers are increasing with time. There are several types of attacks, however the most common attacks are Denial of service (DoS) attacks, Distributed denial of service (DDoS) attacks (Mirkovic, 2004), Password attacks, Eavesdropping attack, SQL injection attack etc., (Darwish,2013). These attacks are capable to bring servers down and stop their services (Xiapu, 2005).

DDoS attack is used to make server unavailable to end users by sending several requests from various sources which the server may not handle and become overloaded (Fu, 2012). These types of attacks are not new. However, these attacks are real security challenges and needs prevention in order to provide reliable services to end users. It is very important to learn which request is an attack and which request is coming from legitimate user. The detection of these attacks through machine learning techniques and other genetic based algorithms is possible (Subbulakshmi 2010). There are several types of machine learning algorithms. Neural networks are one of these. There are several types of neural networks available and for each network there are several training algorithms.

The purpose of this paper is to use feed forward neural network and provide it proper training so that it may detect DDoS attacks efficiently. The neural network used will extract useful features and learn whether current request is an attack or it is a normal request. It also classifies different types of attacks based on their characteristics.

2. RELATED WORK

There exists several works on DDoS attacks defense mechanisms with various technologies. These works include both conventional network based and cloud computing based environments with virtual machines and other network and storage resources. Some cloud computing based works on DDoS attacks are described here. Siamak described some challenges in cloud computing and these challenges are due to various attacks including DDoS and other types of attacks (Siamak, 2013). Kilari, *et. al.*, provided an overview of DDoS attacks in a cloud computing environment (Kilari, 2015). Lanjuan, *et. al.*, proposed a defense mechanism to prevent DDoS attacks in a Cloud computing environment (Lanjuan, 2012). Wang also worked on defense mechanism for DDoS attacks in a cloud computing environment (Wang, 2015). Other several researchers have proposed DDoS attacks defense over networks (Peng, 2007). Douligeris have worked on classification of DDoS attacks and networks (Douligeris, 2004). Krishan proposed an entropy based

Corresponding Authors Email: jamali.abdulaleem@quest.edu.pke ngr.muhammadrizwan@gmail.com
*Department of Electronic Engineering, Quaid-e-Awam University of Engineering, Science and Technology, Nawabshah, Sindh, Pakistan.
**Department of Electronic Engineering, Islamia University Bahawalpur, Pakistan.
***Department of Electrical Engineering, University, Sukkur, Pakistan.

technique to detect attacks in Internet Service Providers (ISP) domain (Kirshan, 2007). Sachdeva *et. al.*, also worked on defense mechanisms for DDoS attacks in an ISP (Sachdeva 2011). In majority of works, authors use Transmission Control Protocol (TCP) and NS-2 tool. In several other works authors use Software Defined Networks and mininet to provide defense against DDoS attacks in a network and Cloud based environment. In several works KDD CUP 99 data set is used to generate attacks. Here in this paper instead of using NS-2 tool, Matlab is used to create a neural network for DDoS attacks detection. The KDD CUP 99 dataset is used in this paper.

3. FEED FORWARD NEURAL NETWORK DESIGN

It is possible to detect DDoS attacks using Machine Learning based techniques. Such techniques include Support Vector Machines, Decision Trees, Naïve Bayes, and Neural Networks. In this research work, DDoS attacks detection is achieved by using Neural Networks. There exist several neural network architectures such as Multi-Layer Perceptron (MLP), casecade feed forward, fit networks etc. In this research work a Multi-Layer Perceptron neural network or feed forward neural network is used. Each neural network uses some kind of training algorithm to train the network on data. There are several training algorithms such as Levenberg, scale conjugate, gradient decent etc. Here main purpose of this work is to check the performance of these training algorithms for DDoS attacks detection. Here we briefly describe these training algorithms.

Gradient descent back propagation

This is a back propagation algorithm based on convex optimization and is used to train a model. The goal of this function is to select and fine tune those parameters which provide minimum cost function. Parameters are assigned with some initial values and are iteratively changed by using partial derivative with respect to its input until cost function is reduced. The slope of a function is obtained with derivative which specifies how much input needs to be changed to get desired output. If the gradient is higher, the slope will be steeper and model can learn faster. It updates both bias and weight in the direction of the negative gradient. Training parameters for this algorithm include maximum number of epochs which is set to be 1000, initial learning rate is set as 0.01, the number of maximum validation checks is set to be 6, and lowest performance gradient is set to be 1e-5.

Some of these parameters are used to stop training. Here if the maximum number of epochs reaches up to 1000 then training stops. Other parameters used to stop training are maximum validation fails, minimum performance gradient, performance goal and training time. Here in all experiments training time is set to be unlimited. The networks training is possible with gradient descent algorithm until transfer functions, net input and weight have derivatives.

Scaled conjugate gradient back propagation

There are several gradient algorithms and are computationally expensive due to a line search in each iteration. Scaled conjugate gradient algorithm avoids this line search. It uses some conjugate gradient algorithm and model trust region approach.

Training parameters for this algorithm include maximum number of epochs which is set to be 1000, initial learning rate is set as 0.01, the number of maximum validation checks is set to be 6, and lowest performance gradient is set to be 1e-6. Here change in weight is set as 5.0e-5 for second derivative approximation; hessian indefiniteness parameter is set to be 5.0e-7.

Bayesian regularization back propagation

In Bayesian regularization training function updates weight and bias in order to minimize squared error (Foresee 1997). For this algorithm the maximum number of epochs is set to be 1000, marquardt parameter for adjustment is set to be 0.005, for mu the decrease factor is 0.1 and increase factor is 10, maximum value for mu is set to be 1e10, the lowest performance gradient is set to be 1e-7.

In this algorithm there is no validation check, stops are set to be zero which means the training will not stop until it finds optimal values for weights and loss function. (Mackay 1992)

4. EXPERIMENTAL SETUP

In this research work KDD-CUP99[1] data set is used. It contains data which includes both normal requests and DDoS attacks. KDDCUP99 is accessible via MIT Lincoln lab which is located in Lexington, Massachusetts. It is a research center of United States department of Defense. The datasets consists of several records, each record has 42 fields. Among these fields first 41 fields indicates the input features such as duration, protocol type, service, flag, src_bytes, dst bytes, etc. The last field indicates DDoS attacks. There are several attacks types such as neptune, smurf, teardrop etc. The KDD data set contains both symbolic and continuous data. To make it Matlab compatible, symbolic data fields are replaced with the continuous data. The symbolic data includes protocol types, service

[1]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

types, flags and attacks type. Protocol types and service types are replaced with values like tcp=1, udp=2, icmp=3, http=4, smtp=5. In the data set there are more than 65 protocol types. There are total ten error flags and are replaced with values like S0=1, SF=2, S1=3, REJ=4, S2=5, RSTO=6, S3=7, RSTR=8, SH=9, and OTH=10. There are more than twenty attacks types and are replaced with values like normal=1, smurf=2, neptune=3, teardrop=5, pod=6, l and=7, etc.

Feed forward Neural Network design

The feed forward neural network uses one hidden layer with 20 neurons. In the hidden layer sigmoid transfer function is used. It also has one output layer with 5 neurons with a linear transfer function as shown in **(Fig. 1)**. Since majority of the requests include either normal or, Neptune and smurf attacks. Here normal indicates that there is no attack. Therefore, for simplicity the attacks are classified into five classes. Therefore the output layers consist of only five neurons.

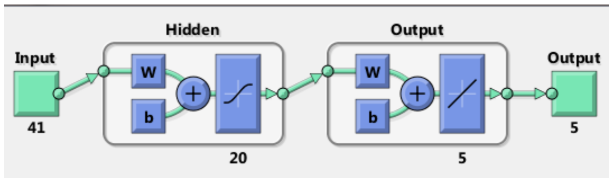


Fig. 1. Feed forward neural network

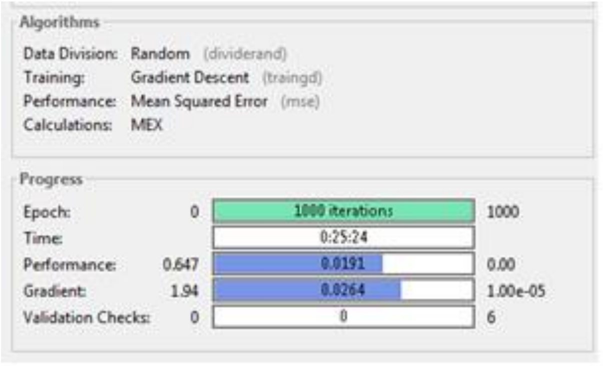
The architecture of feed forward neural network shown in **(Fig. 1)** is used in all experiments. This paper uses three different training algorithms to train the neural network.

5. <u>RESULTS</u>

In this section the DDoS detection and classification results of three different training algorithms are described for the same feed forward neural network. The data set used in these experiments is KDD-CUP99. In KDD-CUP99 there are more than twenty classes of attacks. For simplicity, in this paper these attacks are classified into five categories. The first category shows the normal requests which means that there is no attack. The second category shows smurf attacks, third category shows neptune attacks fourth category shows teardrop and fifth category indicates all other attacks.

Gradient descent back propagation

Neural network's parameters during training are shown in **(Fig. 2)** for gradient descent training algorithm. Here data division is taken as random. Here mean squared error is used to calculate loss or errors. Maximum epochs are 1000. Here iteration is equal to one epoch. Therefore total number of iteration is also 1000. Total time taken to train the neural network is 25 minutes and 24 seconds.



Algorithms

Data Division: Random (dividerand)
Training: Gradient Descent (traingd)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	1000 iterations	1000
Time:		0:25:24	
Performance:	0.647	0.0191	0.00
Gradient:	1.94	0.0264	1.00e-05
Validation Checks:	0	0	6

Fig. 2. Gradient descent training

Training performance of Gradient Decent algorithm is shown in **(Fig. 3).** The best validation performance is 0.018889 at epoch 1000. Since maximum numbers of epoch is set to be 1000. Therefore, training stops once number of epoch reaches to 1000.

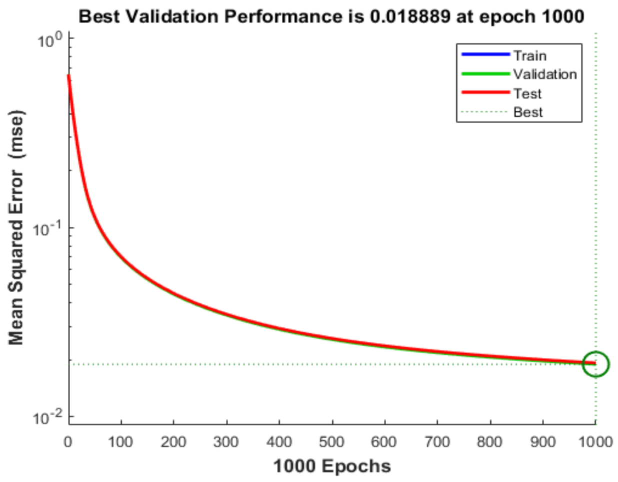
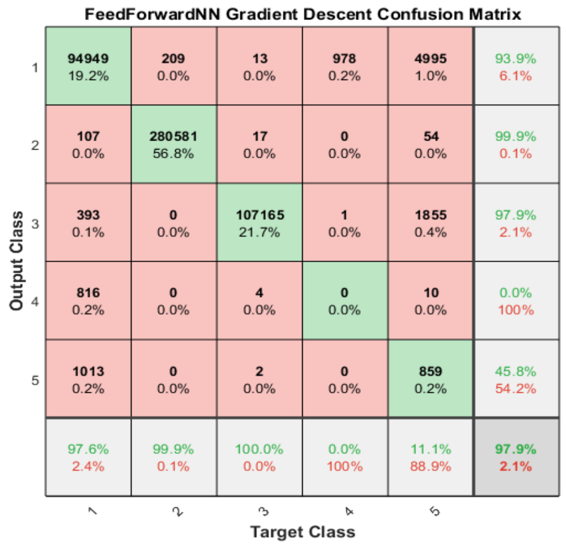


Fig. 3. Performance graph for gradient descent

Confusion Matrix for gradient descent training algorithm is shown in **(Fig. 4)**. Here first class indicates normal requests. For this class there are total 2329 misclassifications. Total 94949 requests are correctly classified; hence accuracy for first class is 97.6%. Second class shows smurf attacks. For this class there are total 209 misclassifications. Total 280581 requests are classified correctly. The accuracy for this class is 99.9%. Third class shows Neptune attacks, for this class there are total 36 misclassifications. Total 107165 requests are correctly classified. The accuracy for this class is approximately 100%. Fourth class shows teardrop attacks. It has 979 misclassifications and not a single request is correctly classified. Hence its accuracy is 0%. Fifth class shows all other types of attacks. For this class there are total 6914 misclassifications. Total 850 requests are correctly classified. Accuracy for this class is only 11.1%. With this algorithm the accuracy for first three classes is very good. However, for other two classes its performance is not satisfactory. This is due to the fact that training data provided in the data set has very less number of teardrop and other types of attacks.

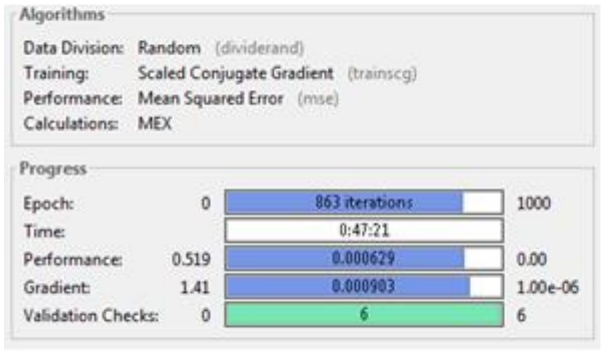

FeedForwardNN Gradient Descent Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	
1	94949 19.2%	209 0.0%	13 0.0%	978 0.2%	4995 1.0%	93.9% 6.1%
2	107 0.0%	280581 56.8%	17 0.0%	0 0.0%	54 0.0%	99.9% 0.1%
3	393 0.1%	0 0.0%	107165 21.7%	1 0.0%	1855 0.4%	97.9% 2.1%
4	816 0.2%	0 0.0%	4 0.0%	0 0.0%	10 0.0%	0.0% 100%
5	1013 0.2%	0 0.0%	2 0.0%	0 0.0%	859 0.2%	45.8% 54.2%
	97.6% 2.4%	99.9% 0.1%	100.0% 0.0%	0.0% 100%	11.1% 88.9%	97.9% 2.1%

Fig. 4. Confusion matrix for gradient descent

Scaled conjugate gradient back propagation

Neural Network's parameters during training for scaled conjugate gradient algorithm are shown in **(Fig. 5).** Here data division is taken as random. Here mean squared error is used to calculate loss or errors. Maximum epochs are set to be 1000. However training stopped at 863 iterations. Here iteration is equal to one epoch. This was due to the fact that maximum validation checks were set to be 6 and at 863 iterations these 6 validation checks were achieved. Total time taken to train the neural network is 47 minutes and 21 seconds.


Algorithms

Data Division: Random (dividerand)
Training: Scaled Conjugate Gradient (trainscg)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	863 iterations	1000
Time:		0:47:21	
Performance:	0.519	0.000629	0.00
Gradient:	1.41	0.000903	1.00e-06
Validation Checks:	0	6	6

Fig. 5. Scaled conjugate gradient training

Training Performance for scaled conjugate gradient algorithm is shown in **(Fig. 6).** Here the mean squared error is used to calculate error. The best validation performance is 0.00062119 at epoch 857. After this for six consecutive iterations mean square error remained the same and training was stopped at 863 iterations.

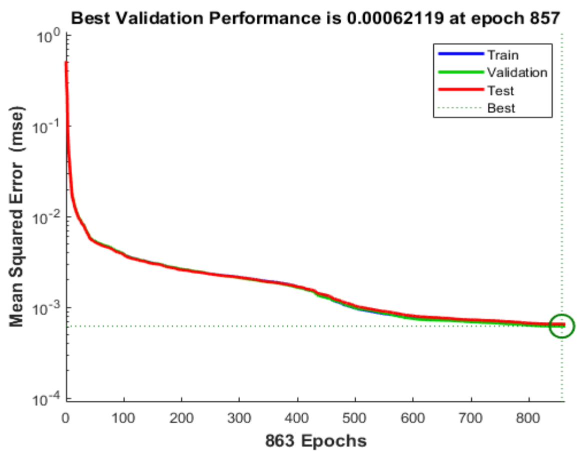
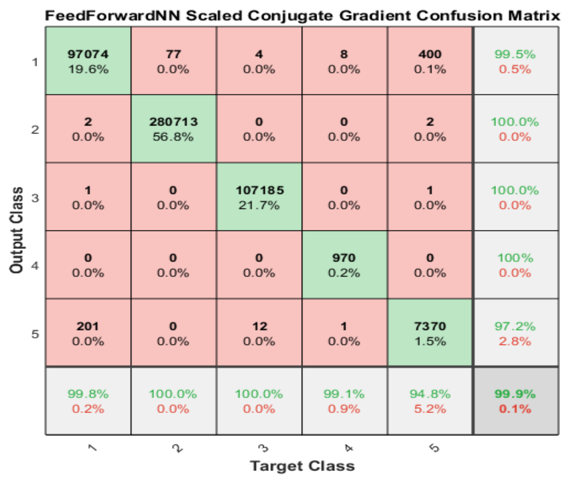


Fig. 6. Performance with scaled conjugate gradient

Confusion Matrix for gradient descent training algorithm is shown in **(Fig. 7)**. It can be seen that for the normal requests class there are 204 misclassifications. Total 97074 requests are correctly classified; hence accuracy for first class is 99.8%. For class 2 which shows smurf attacks, there are total 77 misclassifications. Total 280713 requests are classified correctly. The accuracy for this class is approximately 100%. For third class which shows Neptune attacks, there are total 16 misclassifications. Total 107185 requests are correctly classified. The accuracy for this class is approximately 100%. For fourth class which shows teardrop attacks, there are total 9 misclassifications and 970 requests are correctly classified. Hence its accuracy is 99.1%. For fifth class which shows all other types of attacks, there are total 403 misclassifications. Total 7370 requests are correctly classified. Accuracy for this class is only 94.8%. With this algorithm the overall average accuracy for all classes 99.9% which is very good. This algorithm is able to learn even is less data samples are provided for a class.

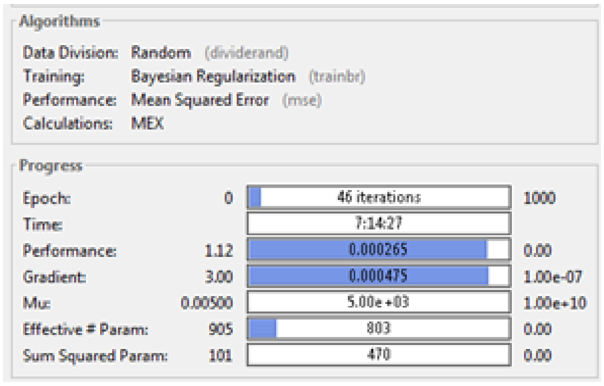

FeedForwardNN Scaled Conjugate Gradient Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	
1	97074 19.6%	77 0.0%	4 0.0%	8 0.0%	400 0.1%	99.5% 0.5%
2	2 0.0%	280713 56.8%	0 0.0%	0 0.0%	2 0.0%	100.0% 0.0%
3	1 0.0%	0 0.0%	107185 21.7%	0 0.0%	1 0.0%	100.0% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	970 0.2%	0 0.0%	100% 0.0%
5	201 0.0%	0 0.0%	12 0.0%	1 0.0%	7370 1.5%	97.2% 2.8%
	99.8% 0.2%	100.0% 0.0%	100.0% 0.0%	99.1% 0.9%	94.8% 5.2%	99.9% 0.1%

Fig. 7. Confusion matrix scaled conjugate gradient

Bayesian regularization back propagation

Training for Bayesian regularization training algorithm is shown in **(Fig. 8).** Here data division is taken as random. Here mean squared error is used to calculate loss or errors. Maximum epochs are set to be 1000. However training stopped at 46 iterations. Here iteration is equal to one epoch. Total time taken to train the neural network is 7 hours 14 minutes and 27 seconds. This algorithm takes the highest training time among three algorithms discussed in this paper.



Algorithms

Data Division:	Random	(dividerand)
Training:	Bayesian Regularization	(trainbr)
Performance:	Mean Squared Error	(mse)
Calculations:	MEX	

Progress

Epoch:	0	46 iterations	1000
Time:		7:14:27	
Performance:	1.12	0.000265	0.00
Gradient:	3.00	0.000475	1.00e-07
Mu:	0.00500	5.00e+03	1.00e+10
Effective # Param:	905	803	0.00
Sum Squared Param:	101	470	0.00

Fig. 8. Bayesian regularization training

Training performance for Bayesian regularization training algorithm is shown in **(Fig. 9)**. Here the mean squared error is used to calculate error. The best validation performance is 0.00026451 at epoch 46.

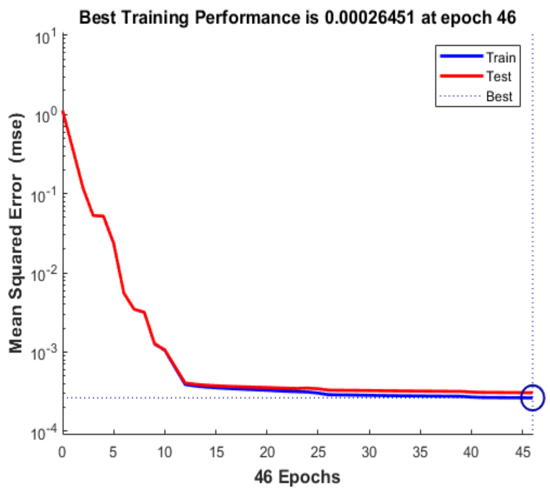
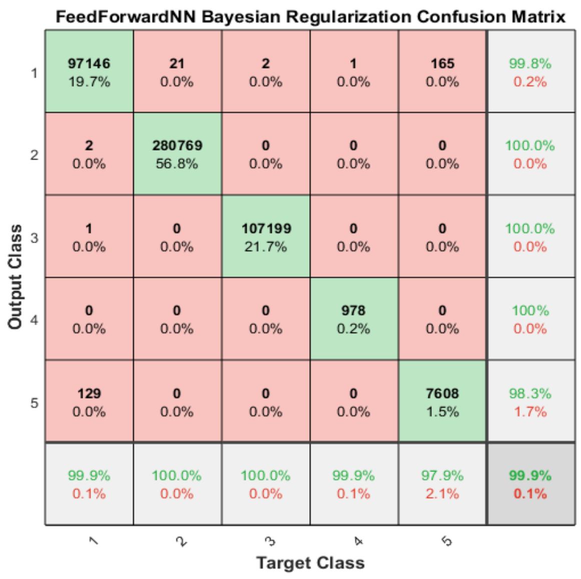


Fig. 9. Training performance with Bayesian Regularization

Confusion Matrix for bayesian regularization training algorithm is shown in **(Fig. 10)**. It can be seen that for normal requests class there are total 132 misclassifications. Total 97146 requests are correctly classified; hence accuracy for first class is 99.9%. For class 2 which shows smurf attacks, there are total 21 misclassifications. Total 280769 requests are correctly classified.



FeedForwardNN Bayesian Regularization Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	
1	97146 19.7%	21 0.0%	2 0.0%	1 0.0%	165 0.0%	99.8% 0.2%
2	2 0.0%	280769 56.8%	0 0.0%	0 0.0%	0 0.0%	100.0% 0.0%
3	1 0.0%	0 0.0%	107199 21.7%	0 0.0%	0 0.0%	100.0% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	978 0.2%	0 0.0%	100% 0.0%
5	129 0.0%	0 0.0%	0 0.0%	0 0.0%	7608 1.5%	98.3% 1.7%
	99.9% 0.1%	100.0% 0.0%	100.0% 0.0%	99.9% 0.1%	97.9% 2.1%	99.9% 0.1%

Fig. 10. Confusion matrix with Bayesian regularization

The accuracy for this class is approximately 100%. For third class which is Neptune attacks, only two requests are misclassified. Total 107199 requests are correctly classified. The accuracy for this class is approximately 100%. For fourth class which shows teardrop attacks, only one request is misclassified and 978 requests are correctly classified. Hence its accuracy is 99.9%. For fifth class which shows all other attacks, there are total 165 misclassifications. Total 7608 requests are correctly classified. Accuracy for this class is 97.9%. With this algorithm the accuracy for all classes is very good. Overall average accuracy obtained with this training algorithm is 99.9%.

6. <u>CONCLUSION</u>

This paper describes a mechanism to detect DDoS attacks using feed forward neural network. It provides comparative analysis of three different training algorithms for the same neural network. These training algorithms are (1) gradient descent back propagation, (2) scaled conjugate gradient back propagation, (3) bayesian regularization back propagation. All three algorithms are useful. Bayesian regularization has very low rate of misclassification and is the best among all algorithms. However training time for this algorithm required is much higher as compared with other algorithms. Accuracy of gradient descent training algorithm is also very good. Its training time is also low as compared with bayesian regularization. The performance of gradient descent training algorithm for fourth and fifth class is very power and overall performance is also low as compared with other two algorithms. Hence it is not an efficient one. It is suggested that any one algorithm from second and third algorithms can be used to detect DDoS attacks. Total misclassifications for gradient descent back propagation algorithm are 1046, while total misclassifications for scaled conjugate gradient back propagation are 709 and total misclassifications for Bayesian regularization back propagation are 327.

REFERENCES:

Akamai (2016) (State of the Internet), https://www.akamai.com/us/en/multimedia/Documents /state-of-the-internet/akamai-q2-2016-state-of-the-internet-security-report.pdf (Accessed on 4.10.2016).

Douligeris, C. and A. Mitrokotsa, (2004) DDoS attacks and defense mechanisms: classification and state-of-the-art, Computer Networks, vol. 44, No. 5, 643-666.

Darwish, M., A. Ouda, and L. F. Capretz, (2013), June). Cloud-based DDoS attacks and defenses. In Information Society (i-Society), International Conference on 67-71. IEEE.

Fu, D. and F. Shi, (2012) "Buffer Overflow Exploit and Defensive Techniques," Fourth International Conference on Multimedia Information Networking and Security, 87-90.

Foresee, F. D, and M. T. Hagan. (1997) "Gauss-Newton approximation to Bayesian learning." Proceedings of the International Joint Conference on Neural Networks.

Kilari, N., and R. Sridaran, (2015). An Overview of DDoS Attacks in Cloud Environment. International Journal of Advanced Networking & Applications.

Krishan K., R.C. Joshi, and K. Singh. (2007) "A Distributed Approach using Entropy to Detect DDoS Attacks in ISP Domain", 2007 International Conference on Signal Processing, Communications and Networking.

Lanjuan Y., T. Zhang. J. Song, J. Wang and P. Chen, (2012) “Defence of DDoS attack for cloud computing”, In Computer Science and Automation Engineering, IEEE International Conference, vol. 2, 626-629.

Mirkovic J. and P. Reiher, (2004) A taxonomy of DDoS attack and DDoS Defense Mechanisms, ACM-SIGCOMM Computer Communications Review, vol. 34, no. 2, 39-53.

MacKay, J. and C. David (1992) "Bayesian interpolation." *Neural computation.* vol. 4, 3, 415–447.

Peng, T., C. Leckie, and K. Ramamohanarao, (2007) Survey of network-based defense Mechanisms Countering the DoS and DDoS problems, ACM Comput. Survey. 39, 1 Pp.

Siamak A., P. Wieder, R. Yahyapour. (2013). Cloud computing networking: challenges and opportunities for innovations. IEEE Communications Magazine, 51(7), 54-62.

Sachdeva, M., G. Singh, and K. Kumar, (2011) “Deployment of Distributed Defense against DDoS Attacks in ISP Domain,” International Journal of Computer Applications, vol. 15, no. 2, 25–31, published by Foundation of Computer Science.

Subbulakshmi. T. (2010) "Feature Selection and Classification of Intrusions Using Genetic Algorithm and Neural Networks", Communications in Computer and Information Science.

Wang, B., Y. Zheng, W. Lou, and Y. T. Hou, (2015). DDoS attack protection in the era of cloud computing software-defined networking. Computer Networks, 81, 308-319.

XiapuLuo, R.K.C. and E.W.W. Chan. (2005) "Performance Analysis of TCP/AQM Under Denial-of-Service Attacks", 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems.