

Journal of Financial and Actuarial Mathematics and Management

A joint edition of the Institute of Information Sciences and Institute of Social Sciences of Perun Academy
Publishing Company: PSP Publishing
ISSN: 2367-5810
Issue 12, Year XII (2024)
Founding Editor and Editor-in-Chief: Pavel Stoynov

Editorial Board: Hiroyuki Ono (Japan), Yasukazu Yoshizawa (Japan), Naoyuki Ishimura (Japan), Boris Melnikov (Russia), Elena Melnikova (Russia), Goran Karanovic (Croatia), James Osondu (Nigeria)

Contents

P. Stoynov and N. Mastorakis. Feed forward network with ST neurons
implementing XOR function……3
P. Stoynov and N. Mastorakis. Feed forward network with ST neurons
for investigation of network traffic……9
P. Stoynov and N. Mastorakis. Feed forward network with ST neurons
for distinguishing normal traffic from icmp-echo attack……20

Feed forward network with ST neurons implementing XOR function

Pavel Stoynov
Technical university of Sofia,
8, Clement Ohridski, Sofia, 1000
Bulgaria
Perun R&D Academy, Sofia, Bulgaria
pavel_todorov_stoynov@yahoo.com
Nikos Mastorakis
English Language Faculty of Engineering
Technical university of Sofia,
8, Clement Ohridski, Sofia, 1000
Bulgaria

Abstract.
In this article we apply ST-based neurons in neural networks and check efficiency of these networks for realizing XOR functions.
Keywords: neural networks, neurons, ST neurons, activation function

Introduction

In this article we apply ST-based neurons in neural networks and check efficiency of these networks for realizing XOR functions. The article is further development of the research in Stoynov (2021, 2022, 2023).

The most natural way to introduce nonlinearity into neural networks is through the nonlinear activation function of the neurons' output.

An overview of the different activation functions used in neurons is for example done by Szandala (2019).

Among the most commonly used are ReLU (Rectified Linear Unit), sigmoid, hyperbolic tangent, Swish Ramachandran et al. (2017, 2017a) and others.

If n_{kt} is the internal output (the output before applying the nonlinearity) of the neuron k at time t, then the ReLU function has the form

$$L(n_{kt}) = \max(0, n_{kt}) = n_{kt}^{+}. \tag{1}$$

The sigmoid function has the form

$$L(n_{kt}) = \frac{1}{1+e^{-n_{kt}}}. \tag{2}$$

The hyperbolic tangent function is defined by the equality

$$T(n_{kt}) = \frac{e^{n_{kt}} - e^{-n_{kt}}}{e^{n_{kt}} + e^{-n_{kt}}}. \tag{3}$$

The Swish function has the form

$$L(n_{kt}) = \frac{n_{kt}}{1+e^{-n_{kt}}}. \tag{4}$$

The cumulative Gaussian function is given by the equality

$$\Phi(n_{kt}) = \int_{-\infty}^{n_{kt}} \frac{1}{\sqrt{2\pi}} e^{-0,5x^2}\, dx. \tag{5}$$

ST neurons

A new proposal to use the so-called ST activation function, which uses the density of the probability distribution of switches (Switch-Time – ST) is presented in Stoynov (2016).

A random variable ξ with a switching distribution ST(n,ß) has a density

$$f_{\xi}(x)=\begin{cases}C(n,\beta)e^{-\beta x}(1+x)^{n},\ x\geq 0\\ 0,\ x<0,\end{cases} \tag{6}$$

where $C(n,\beta)$ are normalization coefficients for which

$$C(n,\beta)=\frac{1}{I(n,\beta)} \tag{7}$$

and

$$I(n,\beta)=\frac{1}{\Gamma(1)}\int_{0}^{\infty}e^{-\beta t}(1+t)^{n}\,dt=\int_{0}^{\infty}e^{-\beta t}(1+t)^{n}\,dt. \tag{8}$$

In private cases when $n=0,\ n=1$ and $n=2$ the equalities $C(0,\beta)=\beta,\ \ C(1,\beta)=\frac{\beta^{2}}{\beta+1}$ and $C(2,\beta)=\frac{\beta^{3}}{\beta^{2}+2\beta+2}$ apply accordingly. The first and second case are exponential and Lindley distribution correspondingly.

A random variable ξ with Lindley distribution has density

$$f_{\xi}(x)=\begin{cases}\frac{\beta^{2}}{\beta+1}e^{-\beta x}(1+x),\ x\geq 0\\ 0,\ x<0.\end{cases} \tag{9}$$

Using the ST distribution thus represented, an ST activation function with a threshold of zero can be introduced where

$$N_{kt}=\begin{cases}\int_{0}^{n_{kt}}C(n,b)e^{-bt}(1+t)^{n}\,dt,\ n_{kt}\geq 0,\\ 0,\ n_{kt}<0.\end{cases} \tag{10}$$

This function depends on two parameters - parameter n and parameter b. Standard ST activation functions can be obtained by certain selection of these two parameters.

The double ST distribution (DST distribution) has the density function

$$s_{\xi}(x)=\begin{cases}\frac{1}{2}C(n,b)e^{-bx}(1+x)^{n},\ x\geq 0,\\ \frac{1}{2}C(n,b)e^{bx}(1-x)^{n},\ x<0.\end{cases} \tag{11}$$

This distribution makes it possible to define non-threshold activation functions. The general appearance of the non-threshold ST activation function is:

$$N_{kt}=\int_{-\infty}^{n_{kt}}s(x)dx=$$
$$=\begin{cases}\frac{1}{2}+\frac{1}{2}\int_{0}^{n_{kt}}C(n,b)e^{-bx}(1+x)^{n}\,dx,\ n_{kt}\geq 0,\\ \frac{1}{2}\int_{-\infty}^{n_{kt}}C(n,b)e^{bx}(1-x)^{n}\,dx,\ n_{kt}<0.\end{cases} \tag{12}$$

The neurons using ST and DST activation function we call ST and DST neurons correspondingly. The neurons using activation functions combining ST and DST with other activation function we call ST-based neurons.

In some of the types of ST neurons, the speed of convergence of the activation function is lower, and therefore a larger number of iterations is needed when training the neural networks.

Therefore, when a higher convergence rate is sought, ST neurons are more useful in combination with neurons with other activation functions, such that ST neurons can be used in the output layers of the network, while the hidden layers can use, for example, a hyperbolic tangent as activation function.

Another possibility is to combine an activation function of type ST with popular types of activation functions such as RELU. Thus, the activation function of a DST01LU neuron is set by the equality

$$N_{kt} = \begin{cases} n_{kt}, n_{kt} \geq 0, \\ \frac{1}{2}\int_{-\infty}^{n_{kt}} e^x dx - \frac{1}{2} = \frac{1}{2}e^{n_{kt}} - \frac{1}{2}, n_{kt} < 0. \end{cases} \quad (13)$$

DST01LU neurons and similar to them are interesting alternatives of the standard conventional neurons used today in neural networks.

Using neural networks with ST neurons and their derivatives for implementing XOR function

As an example of using ST neurons in conjunction with other types of neurons, we consider a neural network including hyperbolic tangent neurons in the inner layer and a DST01 neuron in the output layer to implement the XOR (exclusive or) function.

As an example of neurons that combine ST and other activation functions, we discuss the use of DST01LU neurons to implement the XOR ("exclusive or") function.

As a benchmark, a network with hyperbolic tangent neurons as the activation function is used.

Functional model. We implement "exclusive or" (XOR) function using ST neurons and their derivatives and the effectiveness of such a network to the effectiveness of a forward network with the same structure, but different types of activation functions. The functional model is presented in Figure 1.

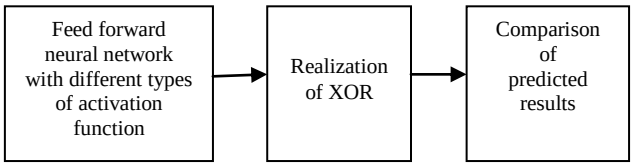


Fig. 1. Comparing the effectiveness of neural networks using ST-based neurons to the effectiveness of standard feed forward neural networks when implementing "exclusive or" (XOR) function. Source: the author.

Experiment scenario and structure of the experimental set-up. The experiment scenario includes the following steps:

1. Simulation of a feed forward neural network with two inputs, three neurons in the hidden layer, and one output neuron. The activation function of the neurons is different considering three cases:

1.1. The activation function of all neurons is a hyperbolic tangent.

1.2. The activation function of the hidden layer neurons is a hyperbolic tangent, and the output neuron is a DST01 neuron.

1.3. All neurons in the network are DST01LU neurons.

2. The neural network is trained for 1000 iterations in each of the three cases.

3. The obtained results are analyzed.

The structure of the experimental setup is presented in Figure 2.

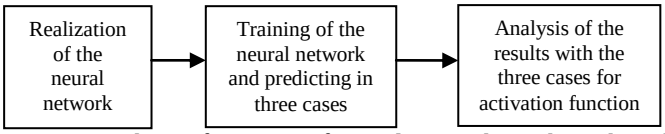


Fig. 2. The experimental setup to test the performance of neural networks with ST-based neurons in realizing the XOR function by comparing it with the performance of conventional feed forward neural network. Source: the author.

Carrying out the experiment.

The network is implemented in the Python language and is then trained and tested for each of the three cases considered.

1. Feed forward neural network with hyperbolic tangent neurons as activation function. Python code used:

```
#xor training data
x_train = np.array([[[0,0]], [[0,1]], [[1,0]], [[1,1]]])
y_train = np.array([[[0]], [[1]], [[1]], [[0]]])
# network
net = Network()
net.add(FCLayer(2, 3))
net.add(ActivationLayer(tanh, tanh_prime))
```

```
net.add(FCLayer(3, 1))
net.add(ActivationLayer(tanh, tanh_prime))
# train
net.use(mse, mse_prime)
net.fit(x_train, y_train, epochs=1000, learning_rate=0.1)
# test
out = net.predict(x_train)
print(out)
```

The results are presented in Table 1.
function and DST01

2. Feed forward neural network with neurons in the hidden layer using hyperbolic tangent activation output neuron.

Table 1. Realization of XOR with feed forward neural network with neurons with hyperbolic tangent as an activation function. Source: the authors.

X1	X2	XOR	Value calculated by the neural network with hyperbolic tangent for activation function
0	0	0	0.00085202
0	1	1	0.97670355
1	0	1	0.97801807
1	1	0	0.00161211

The following code in Python is used:

```
#xor training data
x_train = np.array([[[0,0]], [[0,1]], [[1,0]], [[1,1]]])
y_train = np.array([[[0]], [[1]], [[1]], [[0]]])
# network
net1 = Network()
net1.add(FCLayer(2, 3))
net1.add(ActivationLayer(tanh, tanh_prime))
net1.add(FCLayer(3, 1))
net1.add(ActivationLayer(DST01, DST01_prime))
# train
net1.use(mse, mse_prime)
net1.fit(x_train, y_train,
epochs=1000, learning_rate=0.1)
# test
out1 = net1.predict(x_train)
print(out1)
```

The results are presented in Table 2.

Table 2. A feed forward network for implementation of XOR function with neurons with hyperbolic tangent activation function in the inner layer and DST01 neurons in the output. Source: the authors.

X1	X2	XOR	Value from a neural network with DST01 neuron in the output layer
0	0	0	0.000399618
0	1	1	0.96248401
1	0	1	0.96640301
1	1	0	0.01713484

3. Feed forward neural network with DST01LU neurons. Python code used:

```
#xor training data
x_train = np.array([[[0,0]], [[0,1]], [[1,0]], [[1,1]]])
```

```
y_train = np.array([[[0]], [[1]], [[1]], [[0]]])
# network
net2 = Network()
net2.add(FCLayer(2, 3))
net2.add(ActivationLayer(DST01LU, DST01LU_prime))
net2.add(FCLayer(3, 1))
net2.add(ActivationLayer(DST01LU, DST01LU_prime))
# train
net2.use(mse, mse_prime)
net2.fit(x_train, y_train,
epochs=1000, learning_rate=0.1)
# test
out2 = net2.predict(x_train)
print(out2)
```

The results are presented in Table 3.

Analysis of results and conclusions.

Comparing the results in Table 1, Table 2 and Table 3 shows that DST01 neurons can be used in combination with conventional neurons. Also, DST01LU neurons produce results comparable to those of conventional neurons with hyperbolic tangent as the activation function.

Table 3. Realization of XOR using feed forward network with DST01LU neurons. Source: the author.

X1	X2	XOR	Value from neural network with DST01LU neurons
0	0	0	0.000000055
0	1	1	1.000000100
1	0	1	1.000000390
1	1	0	-0.000000024

The main conclusions of this experiment are:

1. Neural networks with neurons that have a hyperbolic tangent activation function in their hidden layer and DST01 neurons in their outer layer successfully model an XOR function.
2. Neural networks with DST01LU neurons successfully model XOR function

Conclusion

Neural networks with ST-based neurons can be applied successfully for realizing XOR function. Further research could be applied to check efficiency of neural networks with ST-based neurons for solving different applied tasks.

References

Ramachandran, P., B. Zoph, V. Quoc (2017) Searching for activation functions. CoRR, arXiv, vol. abs/1710.05941. https://arxiv.org/pdf/1710.05941.pdf

Ramachandran, P., B. Zoph, V. Quoc (2017) SWISH: a self-gated activation function. CoRR arXiv, https://arxiv.org/pdf/1710.05941v1.pdf?source=post_page

Stoynov, P. (2016) Switch Time Family of distributions and processes and their applications to reflected surplus models. Annual of the Faculty of Economics and Business Administration, Sofia University "St. Kliment Ohridski" - Sofia, 2016, pp. 255-285.

Stoynov, P. (2021) Applications of ST Distributions to Neural Networks and Regression Models. Proceedings of Eighth International Conference on New Trends in the Applications of Differential Equations in Science (NTADES'21), 6-9 September 2021, St. Constantine and Helena, Bulgaria.

Stoynov, P. (2022) Switch Time Activation Function and Stopit Regression – Some Examples. Proceedings of XXXI International Scientific Conference Electronics-ET2022, 13-15 September 2022, Sozopol, Bulgaria.

Stoynov, P. (2023) Switch Time Neurons. Definition and Types. Proceedings of XXXII International Scientific Conference Electronics-ET2023, 13-15 September 2023, Sozopol, Bulgaria. Accepted for printing.

Szandala, T. (2019) Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. arXiv. Cornell University, 2019 https://arxiv.org/ftp/arxiv/papers/2010/2010.09458.pdf

Feed forward network with ST neurons for investigation of network traffic

Pavel Stoynov
Technical university of Sofia,
8, Clement Ohridski, Sofia, 1000
Bulgaria,
Perun R&D Academy, Sofia, Bulgaria
pavel_todorov_stoynov@yahoo.com
Nikos Mastorakis
English Language Faculty of Engineering
Technical university of Sofia,
8, Clement Ohridski, Sofia, 1000
Bulgaria

Abstract.
Network security includes analysis of network traffic as a key process for intrusion detection. This analysis can be performed using neural etworks. In this article, we apply ST-based neurons in neural networks and check efficiency of these networks for investigation of network traffic based on indicator variables.
Keywords: network security, neural networks, ST activation function, ST neurons

Introduction

In this article, we apply ST-based neurons in neural networks and check efficiency of these networks for investigation of network traffic based on indicator variables. The article is further development of the research in Stoynov (2016, 2021, 2022) and Mastorakis and Stoynov (2024).

The most natural way to introduce nonlinearity into neural networks is through the nonlinear activation function of the neurons' output.

A random variable ξ with a switch-time distribution ST(n,ß) has a density

$$f_{\xi}(x)=\begin{cases}C(n,\beta)e^{-\beta x}(1+x)^{n},\ x\geq 0\\ 0,x<0,\end{cases} \tag{1}$$

where $C(n,\beta)$ are normalization coefficients for which

$$C(n,\beta)=\frac{1}{I(n,\beta)} \tag{2}$$

and

$$I(n,\beta)=\frac{1}{\Gamma(1)}\int_{0}^{\infty}e^{-\beta t}(1+t)^{n}\,dt= \\ =\int_{0}^{\infty}e^{-\beta t}(1+t)^{n}\,dt. \tag{3}$$

Using the ST distribution thus represented, an ST activation function with a threshold of zero can be introduced:

$$N_{kt}=\begin{cases}\int_{0}^{n_{kt}}C(n,b)e^{-bt}(1+t)^{n}\,dt,\ n_{kt}\geq 0,\\ 0,\ n_{kt}<0.\end{cases} \tag{4}$$

This function depends on two parameters - parameter n and parameter b. Standard ST activation functions can be obtained by certain selection of these two parameters.

The double ST distribution (DST distribution) has the density function

$$s_{\xi}(x)=\begin{cases}\frac{1}{2}C(n,b)e^{-bx}(1+x)^{n},\ x\geq 0,\\ \frac{1}{2}C(n,b)e^{bx}(1-x)^{n},x<0.\end{cases} \tag{5}$$

This distribution permits to define non-threshold ST activation functions. The general appearance of the non-threshold ST activation function is:

$$N_{kt}=\begin{cases}\frac{1}{2}+\frac{1}{2}\int_{0}^{n_{kt}}C(n,b)e^{-bx}(1+x)^{n}dx,n_{kt}\geq 0,\\ \frac{1}{2}\int_{-\infty}^{n_{kt}}C(n,b)e^{bx}(1-x)^{n}dx,n_{kt}<0.\end{cases}. \tag{6}$$

The neurons using ST and DST activation function we call ST and DST neurons correspondingly. The neurons using activation functions combining ST and DST with other activation function we call ST-based neurons.

In some of the types of ST neurons, the speed of convergence of the activation function is lower than in the commonly used neurons, and therefore a larger number of iterations is needed when training the neural networks. So, when a higher convergence rate is sought, ST neurons are more useful in combination with neurons with other activation functions, such that ST neurons can be used in the output layers of the network, while the hidden layers can use, for example, a hyperbolic tangent as activation function.

Another possibility is to combine an activation function of type ST with popular types of activation functions such as RELU. Thus, the activation function of a DST01LU neuron is set by the equality

$$N_{kt}=\begin{cases}n_{kt},\ n_{kt}\geq 0,\\ \frac{1}{2}\int_{-\infty}^{n_{kt}}e^{x}dx-\frac{1}{2}=\frac{1}{2}e^{n_{kt}}-\frac{1}{2},n_{kt}<0.\end{cases} \tag{7}$$

DST01LU neurons and neurons similar to them are interesting alternatives of the standard conventional neurons used today in neural networks.

The data set

In this article we investigate the network traffic using data including different characteristics of the traffic. Some of these characteristics may indicate if the traffic is normal or due to network intrusion. So, the type of traffic influences these characteristics

The data used are provided by Al-Kasassbeh, Al-Naymat, Al-Hawari (2018). The dataset contains 4998 records for anomaly detection. The data is based on SNMP variables collected from network devices in a real network under test. The variables are grouped into five categories: Interface, IP, ICMP, TCP, UDP.

The Internet standard Simple Network Management Protocol (SNMP, 2023) is a protocol for collecting and organizing information about controlled devices on IP computer networks and for modifying that information to change the behaviour of the devices. Network devices that support SNMP include: cable modems, routers, switches, servers, workstations, printers, and more.

To be compliant with the SNMP standard, an IP-capable device must support a Management Information Base (MIB), i.e. it must have an interface table, which is one of the most useful objects in network monitoring. The table contains information about the system network interfaces. This information includes the interface characteristics:

- ifDescr – description of the interface;
- ifType – interface type;
- ifOpenStatus – indicates whether the interface is on, off, or in some intermediate test state;
- ifMtu – the size of the largest packet that can be sent through the interface;
- ifSpeed – the maximum frequency (bandwidth) of the interface;
- ifInOctets – the number of bytes (octets) received through the interface, including the framing symbols;
- ifOutOctets - the number of bytes sent through the interface, including the framing characters;
- ifPhysAddress – the hardware address of the interface;
- ifInErrors – the number of input packets that contain errors and cannot be delivered to a higher-level protocol;

- ifInUcast – the number of unicast packets delivered to a higher level protocol;
- ifOutUcast – the number of packets that higher-level protocols require to be transmitted to a unicast network address, including packets that are rejected or not sent;
- ifInUnknownProtos – the number of packets received through the interface that were rejected due to unknown or unsupported protocols;
- ifInMulticastPkts – number of delivered packets from the current to a higher layer or sublayer addressed to a multicast address, and for a MAC layer protocol this means group or functional addresses (the 64-bit analogue of this indicator is HCINMCAST);
- ifOutMultiCastPkts – the total number of packets received by higher-level protocols for transmission over the network to multicast addresses of the current layer (the 64-bit analogue of this indicator is HCOUTMCAST);
- ifInBroadcastPkts – the number of packets delivered by the current layer, which were addressed to broadcast addresses of this layer (the 64-bit analogue of this indicator is HCINBCAST);
- ifOutBroadcastPkts – the total number of packets that higher-level protocols requested to be transmitted from the current level to the network, and which were addressed to broadcast addresses of the current level (the 64-bit analogue of this indicator is HCOUTBCAST);
- deltaTime – time interval between two consecutive readings in milliseconds.

Various network traffic metrics are calculated based on these interface characteristics.

Incoming traffic is calculated as the difference of ifInOctets values within a given time interval:

$$\begin{aligned} &InTraffic = \\ &= \frac{ifInOctets(t_2) - ifInOctets(t_1)}{deltaTime} * \\ &*8*1000, Mbps \\ &deltaTime = t_2 - t_1. \end{aligned} \tag{8}$$

Accordingly, for outgoing traffic we have

$$\begin{aligned} &outTraffic = \\ &= \frac{ifOutOctets(t_2) - ifOutOctets(t_1)}{deltaTime} * \\ &*8*1000, Mbps \\ &deltaTime = t_2 - t_1. \end{aligned} \tag{9}$$

The rate of interface utilization (Interface Utilization) is the ratio of network traffic to the speed of the interface at a given moment in percentage:

The input utilization rate is

$$InUtilization = \frac{InTraffic}{InSpeed} * 100, \%. \tag{10}$$

The output utilization rate is

$$OutUtilization = \frac{OutTraffic}{OutSpeed} * 100, \%. \tag{11}$$

The total number of received packets per second is set with the formula

$$\begin{aligned} &RPackets = \\ &= \frac{1}{deltaTime * \frac{1}{1000}} * \\ &*(InUcast + InMulticastPkts + \\ &+ InBroadcastPkts + InErrors + \\ &+ InUnknown) \end{aligned} \tag{12}$$

The total number of packets sent per second is set with the formula

$$TPackets = \frac{1}{deltaTime * \frac{1}{1000}} * \\ * (OutUcast + OutMultiCast + \\ + OutBroadcast). \quad (13)$$

In the context of Ethernet, the concept of a packet (frame) is considered. To form the packet, the source and destination addresses are added to the data to form a datagram, and then the source and destination MAC addresses are added to the datagram to produce the Ethernet frame. The maximum size of a packet (frame) in Ethernet is 1513 bytes. Any larger packet should be fragmented. Thus packets on the network are either IP/UDP datagram envelopes or fragments of datagram envelopes.

In the typical version of SNMP, one or more administrative computers, called managers, are tasked with monitoring and managing a group of hosts or devices on a computer network. Each managed system has a running software component called an agent that sends information via SNMP to managers. A managed system is any network node that implements an SNMP interface. Network Management Station (Network Management Station) is a station running applications that monitor and control network devices (SNMP, 2023).

The data is grouped into eight traffic categories: bruteForce, httpFlood, icmp-echo, normal, slowloris, slowpost, tcp-syn, udp-flood.

A bruteForce attack is an attempt to guess a username and password by guessing a combination of multiple predefined values (dictionary). Most often, automated software or scripts are used for this purpose, which try to enter the combinations without human intervention.

The httpFlood attack is a type of Distributed Denial of Service (DDoS). It represents mass sending of http requests to a server. It takes place at level 7 of the OSI model, i.e. in the application layer. There are two types of http attacks – http POST and http GET. They are often carried out using botnets.

The icmp-echo attack is related to the Internet Control Message Protocol – ICMP (2023). This protocol is described in RFC 792 as an Internet standard. Defines different types of packets used to log problems in the Internet Protocol – (IP, 2023).

Among the messages within ICMP are ICMP Redirect, ICMP Destination Unreachable, ICMP Time Exceeded, ICMP Echo Request, ICMP Echo Reply. The last two messages are a stimulus-response pair that is actually a ping. The ping mechanism involves two steps:

1. The client server prepares and sends an ICMP Echo Request to the accessed computer.

2. The accessed computer, upon receiving the ICMP Echo Request message, prepares and sends an ICMP Echo Reply packet to the client computer.

ICMP packets are typically 56 bits long and are supplemented with ICMP and IP packets up to 64 bits long. Accessed computers and routers can impose a limit on incoming ICMP Echo Request packets, which they accept or drop, respectively, as a precaution. Like TCP and UDP, the ICMP protocol is encapsulated in IP. Unlike them, however, it is a network layer protocol, not a transport layer protocol.

The slowloris attack is Denial of Service (DoS). With it, one machine can crash a web server with minimal effects on other services and ports. With slowloris, an attempt is made to maintain many connections to the attacked server for as long as possible. This is done by opening connections and sending a partial request. The attacker periodically sends additions to this partial request through subsequent http headers, but always such that they do not fully complete the request. Affected servers will keep connections open, thus filling their quota for the maximum number of open connections and denying requests to other clients.

The slowpost attack is a type of distributed denial of service (DDoS). The attacker sends correct http POST headers to the web server. In these headers, the sizes of the message bodies that will follow are correctly specified. But the message bodies are sent at a very slow rate. The rate can be, for example, one byte every two minutes. As a result, the functioning of the web server is disrupted.

The tcp-syn flood attack is a type of Distributed Denial of Service (DDoS). It uses part of the normal TCP three-way handshake to consume resources on the attacked server and prevent it from responding to other clients. The attacker sends TCP connection requests faster than the attacked machine can process them, causing the network to overflow.

The udp-flood attack is a type of Distributed Denial of Service (DDoS). In it, the attacker blocks random ports on the attacked server with IP packets that contain UDP datagrams. The receiving server checks for applications associated with these charts and, finding no such programs, sends back a Destination Unreachable packet. As more and more UDP packets are received and answered, the system becomes overloaded and cannot respond to requests from other clients.

All input variables are: ifInOctets11, ifOutOctets11, ifOutDiscards11, ifInUcastPkts11, ifInNUcastPkts11, ifInDiscards11, ifOutUcastPkts11, ifOutNUcastPkts11, tcpOutRsts, tcpInSegs, tcpOutSegs, tcpPassiveOpens, tcpRetransSegs, tcpCurrEstab, tcpEstabResets ActiveOpens, udpInDatagrams,

udpOutDatagrams, udpInErrors, udpNoPorts, ipInReceives, ipInDelivers, ipOutRequests , ipOutDiscards, ipInDiscards, ipForwDatagrams, ipOutNoRoutes, ipInAddrErrors, icmpInMsgs, icmpInDestUnreachs, icmpOutMsgs, icmpOutDestUnreachs, icmpInEchos, icmpOutEchoReps, class.

Since the quantities ifInOctets11 and ifOutOctets11 are counted cumulatively, the quantities InOctets and OutOctets are formed as the differences of two consecutive values for ifInOctets11 and ifOutOctets11 respectively.

Analogously, the quantities tcpInSegs, tcpOutSegs, udpInDatagrams, udpOutDatagrams, ipInReceives, ipInDelivers, ipOutRequests, ipOutDiscards, ipInDiscards, icmpInMsgs, icmpOutMsgs are registered with accumulation and they are replaced by tcpInSegsDif, tcpOutSegsDif, udpInDatagramsDif, udpOutDatagramsDif, ipInReceivesDif, ipInDeliversDif, ipOutRequestsDif, ipOutDiscardsDif, ipInDiscardsDif , icmpInMsgsDif, icmpOutMsgsDif respectively, by subtracting the values for adjacent time points.

The model

In this article, we investigate the network traffic using data including different characteristics of the traffic. Some of these characteristics may indicate if the traffic is normal or due to network intrusion. So, the type of traffic influences these characteristics

To compare the performance of the proposed ST neurons to that of conventional neurons, it is considered how the input variables udpInDatagramsDif and ipInReceivesDif classify the traffic using a neural network presented at Figure 1.

Table 1 shows the transformation of the input variable udpInDatagramsDif into ranks.

Table 2 shows the ranks from Table 1, distributed by traffic category. All categories except slowloris have negative deviations. The category slowpost has code 1. The categories udp-flood, bruteForce and icmp-echo have code 2. The traffic categories httpFlood and tcp-syn have code 3.

Table 1. Ranks of the input UDP diagrams (udpInDatagramsDif). Source: author's data analysis of Al-Kasassbeh, Al-Naymat, Al-Hawari (2018).

Average number of incoming UDP datagrams (udpInDatagramsDif)	Code
<290	1
290-300	2
300-310	3
>310	4

The traffic categories normal and slowloris have code 4.

Table 2. Average number of incoming UDP datagrams and ranks by traffic category. Source: author's data analysis of Al-Kasassbeh, Al-Naymat, Al-Hawari (2018).

Traffic category	Average number of incoming UDP datagrams (udpInDatagramsDif)	Code
bruteForce	298,1313131	2
httpFlood	309,2578947	3
icmp-echo	297,6279809	2
normal	312,1588629	4
slowloris	314,0592021	4
slowpost	288,3912134	1
tcp-syn	302,7018828	3
udp-flood	297,9844156	2
All categories	303,4348875	3

Table 3 shows the transformation of the input variable ipInReceivesDif in ranks.

Table 3. Ranks defined for the received input IP packets (variable ipInReceivesDif). Source: author's data analysis of Al-Kasassbeh, Al-Naymat, Al-Hawari (2018).

Average number of incoming IP packets (ipInReceivesDif)	Code
0-100000	1
100 000 -200 000	2
200 000-300 000	3
>300 000	4

Table 4 shows the ranks from Table 3, distributed by traffic category. Normal rank is 1. The categories icmp-echo, tcp-syn and udp-flood have moderate positive deviations (code 2). The categories httpFlood (rank 3) and bruteForce (rank 4) have large positive deviations.

Table 4. Average number of received input IP packets and ranks by categories traffic. Source: author's data analysis of Al-Kasassbeh, Al-Naymat, Al-Hawari (2018).

Traffic categories	Average number of received input IP packets (ipInReceivesDif)	Code
bruteForce	341844,4798	4
httpFlood	247230,9649	3
icmp-echo	141183,8792	2
normal	73596,98829	1
slowloris	81517,34106	1
slowpost	93165,15063	1
tcp-syn	152949,5251	2
udp-flood	113951,7221	2
All categories	137310,4912	2

The traffic category code is defined by eight binary variables. The coding of the traffic category is presented in Table 5. The corresponding variable takes a value of one if the observed traffic category is its own, and a value of zero if a category other than its own is observed.

The model used is a feed-forward artificial neural network with 8 binary output variables presented in Table 5 and eight binary input variables presented in Table 6.

Every input variable accepts values 0 or 1. The input variable takes value 1, when the corresponding observed variable takes value in the corresponding range.

Table 5. Output variables of the neural network – code of the traffic class (class_code). Source: author's data analysis of Al-Kasassbeh, Al-Naymat, Al-Hawari (2018).

Variable	Traffic category
y_1	normal
y_2	bruteForce
y_3	httpFlood
y_4	icmp-echo
y_5	slowloris
y_6	slowpost
y_7	tcp-syn
y_8	udp-flood

The neural network has one hidden layer with 4 hidden neurons.

Table 6. Input variables of the neural network. Source: author's data analysis of Al-Kasassbeh, Al-Naymat, Al-Hawari (2018).

Variable	Observed variable	Rank
x_1	input UDP datagrams	1
x_2	input UDP datagrams	2
x_3	input UDP datagrams	3
x_4	input UDP datagrams	4
x_5	input received IP packets	1
x_6	input received IP packets	2
x_7	input received IP packets	3
x_8	input received IP packets	4

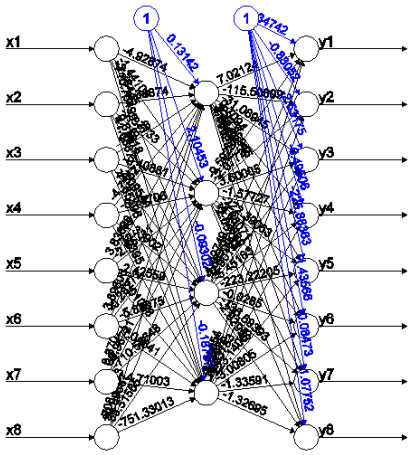


Fig 1. Scheme of the neural network with eight inputs, four hidden neurons and eight outputs.

Traffic type classification based on SNMP variables, which are significantly influenced by traffic type and can be used as indicators of different traffic types.

The experiments and the software

The input variables udpInDatagramsDif and ipInReceivesDif are considered, as examples of one variable that does not differentiate traffic categories well (udpInDatagramsDif) and one variable that differentiates some traffic categories well enough ipInReceivesDif).

The input and output are split into a training set and a test set. The neural network is trained using the training set of inputs and tested for its ability to predict 100 outputs from the test set.

Three experiments are conducted with different types of neural networks:

1. Neural network with hyperbolic tangent neurons for activation functions.
2. Neural network with neurons with activation function of DST01LU neuron.
3. Hyperbolic tangent activation function in the hidden layer of the neural network and DST01LU neurons in the output layer of the neural network.

The neural network for classification of the network traffic using ST-based neurons is simulated with **Python code** presented below:

```
    import pandas as pd
df= pd.read_csv("C:\\input_file.txt",
header=1,sep="\t",names=["tcpInSegs",
"tcpInSegsDif","r1","r2","r3","r4","udpInDatagrams","udpInDatagramsDif","x1","x2","x3","x4","ipInReceives",
"ipInReceivesDif","x5","x6","x7","x8","icmpInMsgs","icmpInMsgsDif","z5","z6","z7","z8","class","class_code
","y1","y2","y3","y4","y5","y6","y7","y8"])
```

```
#DATA NORMALIZATION
udpDatagramsDifNormalized=(df.loc[:,"udpInDatagramsDif"]-
min(df.loc[:,"udpInDatagramsDif"]))/(max(df.loc[:,"udpInDatagramsDif"])-
min(df.loc[:,"udpInDatagramsDif"]))
df['udpDatagramsDifNormalized'] = udpDatagramsDifNormalized
ipInReceivesDifNormalized=(df.loc[:,"ipInReceivesDif"]-
min(df.loc[:,"ipInReceivesDif"]))/(max(df.loc[:,"ipInReceivesDif"])-min(df.loc[:,"ipInReceivesDif"]))
df['ipInReceivesDifNormalized'] =ipInReceivesDifNormalized

#HISTOGRAM BY TRAFFIC
class_traffic=df.loc[:,"class"]
import matplotlib.pyplot as plt
plt.hist(class_traffic)

#TRAIN AND TEST DATA
traffic_all_data_train=df.sample(frac=0.7, replace=False)
traffic_all_data_test=df.drop(traffic_all_data_train.index)

#Input and output as data frames
x_train_df=traffic_all_data_train[["udpDatagramsDifNormalized","ipInReceivesDifNormalized"]]
y_train_df=traffic_all_data_train[["y1","y2","y3","y4","y5","y6","y7","y8"]]
x_train_nr=x_train_df.to_numpy()
y_train_nr=y_train_df.to_numpy()
x_train=x_train_nr.reshape(3482,1,2)
y_train=y_train_nr.reshape(3482,1,8)
x_test_df=traffic_all_data_test[["udpDatagramsDifNormalized","ipInReceivesDifNormalized"]]
y_test_df=traffic_all_data_test[["y1","y2","y3","y4","y5","y6","y7","y8"]]
x_test_nr=x_test_df.to_numpy()
y_test_nr=y_test_df.to_numpy()
x_test=x_test_nr.reshape(1493,1,2)
y_test=y_test_nr.reshape(1493,1,8)

# Network
net = Network()
net.add(FCLayer(2, 4))
net.add(ActivationLayer(tanh, tanh_prime))
net.add(FCLayer(4, 8))
net.add(ActivationLayer(tanh, tanh_prime))

# train on 3482 samples
# as we didn't implemented mini-batch GD, training will be pretty slow if we update at each iteration on 60000
samples...
net.use(mse, mse_prime)
net.fit(x_train, y_train, epochs=35, learning_rate=0.1)

# test on 100 samples
out = net.predict(x_test[0:100])
max=np.zeros(100)
max_y=np.zeros(100)
correct=np.zeros(100)
for i in range(100):
        max[i]=np.argmax(out[i])
        max_y[i]=np.argmax(y_test[i])
        if max[i]==max_y[i]:
                correct[i]=1
>>> correct
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
    0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
```

```
      1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
      1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1.,
      1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1.])

#correct 51
print("\n")
print("predicted values : ")
print(out, end="\n")
print("true values : ")
print(y_test_resh[0:100])

#DST01LU network
import math
net1 = Network()
net1.add(FCLayer(2, 4))
net1.add(ActivationLayer(DST01LU, DST01LU_prime))
net1.add(FCLayer(4, 8))
net1.add(ActivationLayer(DST01LU,DST01LU_prime))


# train on 3482 samples
# as we didn't implemented mini-batch GD, training will be pretty slow if we update at each iteration on 60000 samples...
net1.use(mse, mse_prime)
net1.fit(x_train, y_train, epochs=35, learning_rate=0.1)

out1 = net1.predict(x_test[0:100])
max=np.zeros(100)
max_y=np.zeros(100)
correct=np.zeros(100)
for i in range(100):
        max[i]=np.argmax(out1[i])
        max_y[i]=np.argmax(y_test[i])
        if max[i]==max_y[i]:
                correct[i]=1

correct
>>> correct
array([0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
      0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0., 0.,
      0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
      1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
      1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0.,
      0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])

#correct 49

#combined tanh DST01LU network
net2 = Network()
net2.add(FCLayer(2, 4))
net2.add(ActivationLayer(tanh, tanh_prime))
net2.add(FCLayer(4, 8))
net2.add(ActivationLayer(DST01LU,DST01LU_prime))

# train on 3482 samples
# as we didn't implemented mini-batch GD, training will be pretty slow if we update at each iteration on 60000 samples...
net2.use(mse, mse_prime)
net2.fit(x_train, y_train, epochs=35, learning_rate=0.1)
```

```
out2 = net2.predict(x_test[0:100])
max=np.zeros(100)
max_y=np.zeros(100)
correct=np.zeros(100)
for i in range(100):
        max[i]=np.argmax(out2[i])
        max_y[i]=np.argmax(y_test[i])
        if max[i]==max_y[i]:
                correct[i]=1

correct
>>> correct
array([0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0.,
    0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
    1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
    1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0.,
    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.])
#correct 49
```

Analysis of results and conclusions. The values that are output are between zero and one. Traffic is assigned to that type whose output has the maximum value.

Since the input variables are purposefully chosen so that one of them is not a good indicator of the type of traffic, a high rate of correctly classified test data is not expected, but rather a commensurability of the rates of correctly classified data is sought in each of the three cases.

Table 7 presents the percentages of correctly classified data in each of the three cases.

The data in Table 7 show that in all three cases the network is equally effective at predicting test data.

Table 7. Percentage of correctly classified traffic in the three cases of the type of neurons used.

Network type	Percentage of test records correctly classified
Neural network with neurons with hyperbolic tangent as activation function	51%
Network with DST01LU neurons	49%
Neural network with hyperbolic tangent as activation function in the hidden layer and DST01LU neurons in the output layer	49%

The main conclusions of these experiments are:

1. Neural networks with hyperbolic tangent activation function neurons in their hidden layer and DST01 neurons in their output layer show similar traffic classification accuracy to that of conventional neural networks with hyperbolic tangent activation function neurons.

2. Neural networks with DST01LU neurons show similar traffic classification accuracy to conventional neural networks with hyperbolic tangent activation function neurons.

Conclusion

The general conclusion that can be drawn from the experiments conducted in this article is that neural networks which use ST-based neurons can be applied with similar efficiency to classify network traffic as the neural networks using common types of neurons.

Further research can be provided to distinguish specific types of intrusion against normal network traffic using neural networks with ST-based neurons.

References

Al-Kasassbeh, Al-Naymat, Al-Hawari (2018) Using machine learning methods for detecting network anomalies within SNMP-MIB, International Journal of Wireless and Mobile Computing, Vol.15, No.1, 2018.

ICMP (2023) Internet Control Message Protocol –ICMP. Protocol Specification. RFC 792. https://www.rfc-editor.org/rfc/rfc792. Last seen: 17.03.2023.

IP (2023) Internet Protocol. RFC 791. https://www.rfc-editor.org/rfc/rfc791. Last seen: 17.03.2023.

Stoynov, P. and N. Mastorakis (2024) Feed forward network with ST neurons implementing XOR function. Proceedings of Eleventh International Conference "IT in Education, Art, Research and Business" – ITERB-2024. Financial and Actuarial Mathematics – FAM-2024, ISSN1314-460X , p. 3-10.

Stoynov, P. (2016) Switch Time Family of distributions and processes and their applications to reflected surplus models, Annual of the Faculty of Economics and Business Administration, Sofia University "St. Kliment Ohridski", 2016, 255-285.

Stoynov, P. (2021) Applications of ST Distributions to Neural Networks and Regression Models, Proceedings of Eighth International Conference on New Trends in the Applications of Differential Equations in Science (NTADES'21), 6-9 September, St. Constantine and Helena, Bulgaria.2021.

Stoynov, P. (2022) Switch Time Activation Function and Stopit Regression – Some Examples, Proceedings of XXXI International Scientific Conference Electronics-ET2022, 13-15 September 2022, Sozopol, Bulgaria. 2022.

SNMP (2023) Simple Network Management Protocol https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol, Last seen: 17.03.2023.

Feed forward network with ST neurons for distinguishing normal traffic from icmp-echo attack

Pavel Stoynov
Technical university of Sofia,
8, Clement Ohridski, Sofia, 1000
Bulgaria,
Perun R&D Academy, Sofia, Bulgaria
pavel_todorov_stoynov@yahoo.com
Nikos Mastorakis
English Language Faculty of Engineering
Technical university of Sofia,
8, Clement Ohridski, Sofia, 1000
Bulgaria

Abstract.
In this article we apply ST-based (Switch-Time-based) neurons in neural networks and check efficiency of these networks for distinguishing normal traffic from icmp-echo attack in computer networks**.**
Keywords: ST activation function, DST01 neurons, DST01LU neurons, normal traffic, icmp-echo attack

Introduction

In this article we apply ST-based neurons in neural networks and check efficiency of these networks for investigation of network traffic based on indicator variables. The article is further development of the research in Stoynov (2016, 2021, 2022) and Stoynov and Mastorakis (2024, 2024a, 2024b).

A random variable ξ with a switch-time distribution ST(n,ß) has a density

$$f_{\xi}(x)=\begin{cases}C(n,\beta)e^{-\beta x}(1+x)^n, x\geq 0\\ 0, x<0,\end{cases}. \quad (1)$$

where $C(n,\beta)$ are normalization coefficients for which

$$C(n,\beta)=\frac{1}{I(n,\beta)} \quad (2)$$

and

$$I(n,\beta)=\frac{1}{\Gamma(1)}\int_0^{\infty}e^{-\beta t}(1+t)^n dt=\int_0^{\infty}e^{-\beta t}(1+t)^n dt \quad (3)$$

Using the ST distribution thus represented, an ST activation function with a threshold of zero can be introduced where

$$N_{kt}=\begin{cases}\int_0^{n_{kt}}C(n,b)e^{-bt}(1+t)^n dt, n_{kt}\geq 0,\\ 0, n_{kt}<0.\end{cases} \quad (4)$$

This function depends on two parameters - parameter n and parameter b. Standard ST activation functions can be obtained by certain selection of these two parameters.

The double ST distribution (DST distribution) has the density function

$$s_{\xi}(x)=\begin{cases}\frac{1}{2}C(n,b)e^{-bx}(1+x)^n, x\geq 0,\\ \frac{1}{2}C(n,b)e^{bx}(1-x)^n, x<0.\end{cases} \quad)$$

This distribution permits to define non-threshold ST activation functions. The general appearance of the non-threshold ST activation function is:

$$N_{kt} = \int_{-\infty}^{n_{kt}} s(x)dx = \\ = \begin{cases} \frac{1}{2} + \frac{1}{2}\int_{0}^{n_{kt}} C(n,b)e^{-bx}(1+x)^n dx, \, n_{kt} \geq 0, \\ \frac{1}{2}\int_{-\infty}^{n_{kt}} C(n,b)e^{bx}(1-x)^n dx, \, n_{kt} < 0. \end{cases} \tag{6}$$

The neurons using ST and DST activation function we call ST and DST neurons correspondingly. The neurons using activation functions combining ST and DST with other activation function we call ST-based neurons.

In some of the types of ST neurons, the speed of convergence of the activation function is lower, and therefore a larger number of iterations is needed when training the neural networks.

Therefore, when a higher convergence rate is sought, ST neurons are more useful in combination with neurons with other activation functions, such that ST neurons can be used in the output layers of the network, while the hidden layers can use, for example, a hyperbolic tangent as activation function.

Another possibility is to combine an activation function of type ST with popular types of activation functions such as RELU. Thus, the activation function of a DST01LU neuron is set by the equality

$$N_{kt} = \begin{cases} n_{kt}, \, n_{kt} \geq 0, \\ \frac{1}{2}\int_{-\infty}^{n_{kt}} e^x dx - \frac{1}{2} = \frac{1}{2}e^{n_{kt}} - \frac{1}{2}, \, n_{kt} < 0. \end{cases} \tag{7}$$

DST01LU neurons and similar to them are interesting alternatives of the standard conventional neurons used today in neural networks.

The data set

The data set used in this article is from Al-Kasassbeh, Al-Naymat, Al-Hawari (2018). The authors provide a realistic data with 4998 records for anomaly detection. The data is based on SNMP variables collected from network devices in a real network under test. The variables are grouped into five categories: Interface, IP, ICMP, TCP, UDP.

This data set is described in details in Stoynov and Mastorakis (2024a).

Here, we use the input SNMP (2023) variable ipInReceivesDif to distinguish normal traffic from an icmp-echo attack using a neural network with different types of activation functions.

The icmp-echo attack is related to the Internet Control Message Protocol – ICMP (1981). This protocol is described in RFC 792 as an Internet standard. Defines different types of packets used to log problems in the Internet Protocol - IP (1981).

Among the messages within ICMP are ICMP Redirect, ICMP Destination Unreachable, ICMP Time Exceeded, ICMP Echo Request, ICMP Echo Reply. The last two messages are a stimulus-response pair that is actually a ping. The ping mechanism involves two steps:

1. The client server prepares and sends an ICMP Echo Request to the accessed computer.

2. The accessed computer, upon receiving the ICMP Echo Request message, prepares and sends an ICMP Echo Reply packet to the client computer.

ICMP packets are typically 56 bits long and are supplemented with ICMP and IP packets up to 64 bits long. Accessed computers and routers can impose a limit on incoming ICMP Echo Request packets, which they accept or drop, respectively, as a precaution. Like TCP and UDP, the ICMP protocol is encapsulated in IP. Unlike them, however, it is a network layer protocol, not a transport layer protocol.

The model

In order to compare the performance of the proposed ST neurons compared to that of conventional neurons, it is considered how the input variable ipInReceives helps to distinguish normal traffic from an icmp-echo attack using a neural network with different types of activation functions.

Since the ipInReceives is registered cumulatively, the quantity ipInReceivesDif are formed as the differences of two consecutive values for ifOutOctets.

Differentiating normal traffic from an icmp-echo attack is based on the functional model presented in Fig. 1.

Experiment scenario includes the following steps:

1. Define SNMP variable which will be used as an input – total number of output octets (ipInReceives).



Fig.1. Classification of the type of traffic with using of selected SNMP variable and AI.

2. Define traffic categories which will be distinguished – normal and Brute Force attack.
3. Selecting the type of the neural network and simulations with it.
4. Analysis of results.

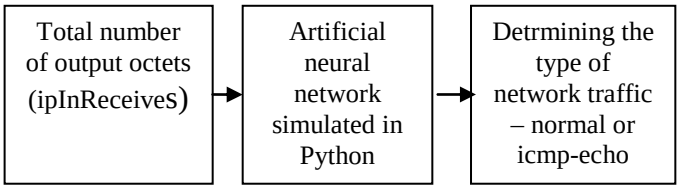


Fig. 2. Scheme of experimental set-up for distinguishing normal traffic from icmp-echo attack attack using SNMP variable ipInReceives and artificial neural network of type Feed Forward. Source: the authors.

The experimental set-up is presented on Fig. 2. The neural network is simulated with the Python language.

The experiment

The values of the ipInReceivesDif variable are used as input to the neural network. The output represents the type of traffic, with normal traffic coded as 0 and icmp-echo attack coded as 1. The input variable is pre-normalized.

The input and output are split into a training set and a test set. The neural network is trained using the training set of inputs and tested for its ability to predict 100 outputs from the test set.

Three experiments are conducted with different types of neural networks:

1. Neural network with hyperbolic tangent neurons for activation functions.
2. Neural network with DST01LU neurons.
3. Neural network with neurons with hyperbolic tangent activation function in the hidden layer and with DST01LU neurons in the output layer.

The neural network is simulated with Python code. Training is performed with 1000 iterations.

The values that are output are between zero and one. They are modified such that those less than or equal to 0.5 are treated as zeros and those greater than 0.5 as ones.

Table 1 presents the percentages of correctly classified data in each of the three cases.

The data in Table 1 show that networks with a hyperbolic tangent activation function in the hidden layer and with DS01LU neurons at the output show similar performance in predicting the test data to those with conventional hyperbolic tangent neurons as the activation function.

Table 1. Percentage of correctly classified traffic
in the three cases of the type of neurons used to distinguish normal schedule from ICMP-echo attack. Source: The authors.

Network type	Percentage of test records correctly classified
Neural network with neurons with hyperbolic tangent as activation function	96%
Network with DST01LU	49%

neurons	
Neural network with hyperbolic tangent as activation function in the hidden layer and DST01LU neurons in the output layer	91%

The main conclusion that can be drawn from this experiment is that a network with hyperbolic tangent activation function neurons in the hidden layer and DST01LU neurons in the outer layer correctly classifies the traffic type 91% of the time, which is comparable to the correct rate of a neural network where all neurons are hyperbolic activation function tangent.

Conclusion

The general conclusion that can be drawn from the experiments conducted in this article is that neural networks which use ST-based neurons can be applied with similar efficiency to distinguish normal network from icmp-echo attack.

Further research can be provided to distinguish other specific types of intrusion against normal network traffic using neural networks with ST-based neurons.

References

Al-Kasassbeh, Al-Naymat, Al-Hawari (2018) Using machine learning methods for detecting network anomalies within SNMP-MIB, in International Journal of Wireless and Mobile Computing, vol. 15, No. 1, 2018.

ICMP (1981) Internet Control Message Protocol –ICMP. Protocol Specification. RFC 792. https://www.rfc-editor.org/rfc/rfc792. Last seen: 17.03.2023.

IP (1981) Internet Protocol. RFC 791. https://www.rfc-editor.org/rfc/rfc791. last seen: 17.03.2023.

Stoynov, P. and N. Mastorakis (2024) Feed forward network with ST neurons implementing XOR function. Proceedings of Eleventh International Conference "IT in Education, Art, Research and Business" – ITERB-2024. Financial and Actuarial Mathematics – FAM-2024, ISSN1314-460X , p. 3-10.

Stoynov, P. and N. Mastorakis (2024a) Feed forward network with ST neurons for investigation of network traffic. Proceedings of Eleventh International Conference "IT in Education, Art, Research and Business" – ITERB-2024. Financial and Actuarial Mathematics – FAM-2024, ISSN1314-460X , p. 11-24.

Stoynov, P. and N. Mastorakis (2024b) Feed forward network with ST neurons for distinguishing normal traffic from Brute Force attack. WSEAS Transactions on Computers. 2024. Accepted for printing.

SNMP (2023) Simple Network Management Protocol https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol last seen: 17.03.2023.

Stoynov, P. (2016) Switch Time Family of distributions and processes and their applications to reflected surplus models. Annual of the Faculty of Economics and Business Administration, Sofia University "St. Kliment Ohridski" - Sofia, 2016, pp. 255-285.

Stoynov, P. (2021) Applications of ST Distributions to Neural Networks and Regression Models. Proceedings of Eighth International Conference on New Trends in the Applications of Differential Equations in Science (NTADES'21), 6-9 September, St. Constantine and Helena, Bulgaria. 2021.

Stoynov, P. (2022) Switch Time Activation Function and Stopit Regression – Some Examples. Proceedings of XXXI International Scientific Conference Electronics-ET2022, 13-15 September 2022, Sozopol, Bulgaria. 2022.