Detection of GPS Spoofing Attacks on Unmanned Aerial Systems

Mohsen Riahi Manesh[1*], Jonathan Kenney[1], Wen Chen Hu[1], Vijaya Kumar Devabhaktuni[2], and Naima Kaabouch[1]
[1]School of Electrical Engineering and Computer Science, University of North Dakota, ND 58202 USA
[2]Department of Electrical and Computer Engineering, University of Purdue Northwest, IN 46323 USA
[*]mohsen.riahimanesh@und.edu

Abstract—**Unmanned Aerial Systems (UAS) have received a huge interest in military and civil applications. Applications of UAS are dependent on successful communications of these systems with different entities in their networks. A UAS network can include the UAS, ground control station, navigation satellite system, and automatic dependent surveillance-broadcast (ADS-B) receiver. Through these entities, a UAS is vulnerable to different cyber-attacks such as GPS spoofing. In this attack, a malicious user transmits fake signals to the GPS receiver on the UAS. The fake signals can mislead not only the aircraft but also air traffic controllers, leading to serious problems. These problems range from aircraft hijacking to collisions and human casualties. This paper proposes a supervised machine learning method based on the artificial neural network to detect GPS spoofing signals. Different features such as pseudo range, Doppler shift, and signal-to-noise ratio (SNR) are used to perform the classification of GPS signals. We examine and compare the efficiency of one- and two-hidden-layer neural networks with various numbers of hidden neurons. The results show that our proposed method provides a high probability of detection and a low probability of false alarm.**

***Index Terms*— UAS; UAV; GPS spoofing; GPS meaconing; cyber-attacks; attack detection, machine learning; neural network**

I. INTRODUCTION

The operation of autonomous systems such as Unmanned Aerial Systems (UAS) is dependent on several sensors including Global Positioning System (GPS). The GPS worldwide coverage has made this system a standard means for navigation and tracking purposes with an accuracy of up to 3 meters [1-3]. In addition, numerous devices use GPS for accurate localization or precise time synchronization. The GPS receiver receives the signals from four or more satellites and calculates its three-dimensional position. However, the public GPS receiver is not secure because of the unencrypted signals of satellites. This makes GPS receivers vulnerable to several types of cyber-attacks including GPS spoofing [4-9]. In this attack, a malicious user transmits fake signals similar to those transmitted by satellites, but at a higher power. The malicious user can also store GPS signals received at a location and later rebroadcast them at another location. This second type of spoofing attack is called GPS meaconing. By modifying the time delays and information in the signals, an attacker can make the receiver calculate an arbitrary position. These attacks can be easily launched using cheap commercial off-the-shelf software and hardware [6, 10]. Therefore, efficient solutions must be found to detect and mitigate these types of attacks.

Several techniques have been proposed for this purpose [11-18]. For instance, the authors of [11] presented an overview of GNSS spoofing attacks, its detection mechanisms, and difficulties associated with these mechanisms. In [12], the authors proposed a method for GPS signal authentication using probabilistic models that combines cryptographic code origin authentication with code timing authentication based on statistical hypothesis tests. Other methods based on symmetric cryptography such as those in [13, 14] were also proposed to deal with GPS spoofing attacks. However, cryptographic approaches cannot provide security against replay attacks. These approaches also require significant modification to the GPS infrastructure. In [15], a spoofing detection method called auxiliary peak tracking in combination with a navigation message inspector was proposed in which the strongest satellite signal as well as other weaker environment signals are tracked, which can also detect takeover attacks. In [16], the authors proposed a detection and mitigation technique based on phase delay and spatial processing to protect GPS receivers from spoofing attacks. First, the attack detection is done via phase delay estimation. Next, the angle of arrival is estimated and the undesired signals are spatially filtered. However, this method requires multiple reception antennas.

In [17], the authors proposed a method based on the analysis of state estimation using Support Vector Machine to detect GPS spoofing attacks on UAS. This method exploits the error distribution between GPS and inertial navigation system of the UAS. The authors found out that in the presence of a GPS spoofing attack, this distribution changes. However, this method suffers from a significant performance degradation in a long duration attack. In [18], a method called Crowd-GPS-Sec is presented which detects and localizes GPS spoofing attacks on aircraft and UAVs. Crowd-GPS-Sec benefits from existing wireless air traffic control systems such as ADS-B devices to detect spoofing attacks from a distant location. This method uses the differences in reported positions by multiple aircraft. However, to achieve a localization accuracy of around 150 meters, this method needs 15 minutes of monitoring time, which is long enough for an attacker to hijack a UAS.

Therefore, motivated by the limitations of the existing solutions, we propose a supervised machine learning method based on artificial neural networks (NN) in which received GPS signals, legitimate or fake, are directly fed to the algorithm and

978-1-5386-5553-5/19/$31.00 ©2019 IEEE

a decision is made on the presence or absence of the attack. This method does not need any modification of the GPS infrastructure and can be simply implemented using microcontrollers or microcomputers of any GPS-driven autonomous system.

The organization of the paper is as follows. Section II describes structure of the neural networks and the features extracted from GPS signals. Section III describes and discusses the results. Finally, conclusions are drawn in Section IV.

II. Methodology

In this section, we first describe the structure of the proposed neural network and how this algorithm can be trained with known data to make decisions on unknown data. Then, we describe the features that are selected and extracted from the received GPS signal.

A. Neural Networks

In a neural network as shown in Fig. 1, each neuron has an activation and each link between two neurons has a weight. The activation and bias of neuron j in layer l are denoted by $a_j^{(l)}$ and $b_j^{(l)}$, respectively. The first layer in every neural network is the input layer and has the activation $a_j^{(1)} = x_j$ where $x_j$is the input feature vector which is taken from each received GPS signal. The weight from the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer is denoted by $w_{jk}^{(l)}$. Therefore, $a_j^{(l)}$ can be calculated as follows [19]:

$$z_j^{(l)} = \sum_k w_{jk}^{(l)} a_k^{(l-1)} + b_j^{(l)} \quad (1)$$

$$a_j^{(l)} = g\left(z_j^{(l)}\right) \quad (2)$$

where $g(.)$ is called the activation function. The most widely used activation functions are sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU) as follows:

$$g_{sigmoid}(z) = \frac{1}{1+e^{-z}} \quad (3)$$

$$g_{tanh}(z) = \frac{2}{1+e^{-2z}} - 1 \quad (4)$$

$$g_{ReLU}(z) = \max(0, z) \quad (5)$$

We can rewrite the matrix formulation of (1) and (2) as follows:

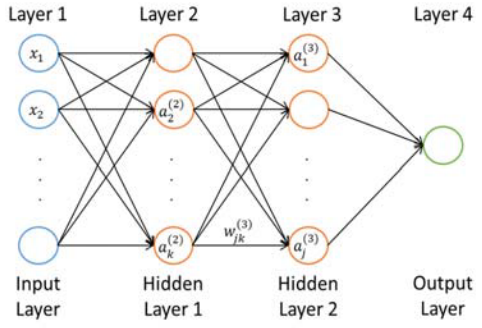

Fig. 1. General flow graph of a neural network with 4 layers.

$$\boldsymbol{z}^{(l)} = \boldsymbol{w}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)} \quad (6)$$

$$\boldsymbol{a}^{(l)} = g(\boldsymbol{z}^{(l)}) \quad (7)$$

Using the input, $\boldsymbol{x}$, initialized weights, and initialized values of bias, the output, $\boldsymbol{a}^{(L)}$, which is a prediction on the presence or absence of the attack, can be computed using (6) and (7). The objective is to minimize a cost function, which is the error between the obtained output and the desired output, $\boldsymbol{y}$. This cost function, C, is defined as:

$$C(w,b) = \frac{1}{2N}\sum_{\boldsymbol{x}}\left\|\boldsymbol{y} - \boldsymbol{a}^{(L)}\right\|^2 \quad (8)$$

where $\boldsymbol{y}$ is the desired output for individual input vector $\boldsymbol{x}$, $\boldsymbol{a}^{(L)}$ is the neural network output for the same given input, N is the total number of training samples, and L is the number of layers in the network.

The backpropagation algorithm is employed to minimize the cost function and find the optimal weight and bias values. This algorithm calculates the cost function partial derivatives $\frac{\partial C}{\partial w}$ and $\frac{\partial C}{\partial b}$ with regards to w and b, respectively [19]. Accordingly, considering an individual training example, we define the error of node j in l^{th} layer as:

$$\delta_j^{(l)} = \frac{\partial C_{\boldsymbol{x}}}{\partial z_j^{(l)}} \quad (9)$$

where $C_{\boldsymbol{x}} = \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{a}^{(L)}\right\|^2 = \frac{1}{2}\sum_{\boldsymbol{x}}(y_j - a_j^{(L)})^2$is the cost for a single training example $\boldsymbol{x}$. We start by calculating the error in the last layer, L, as:

$$\delta_j^{(L)} \overset{\text{def}}{\equiv} \frac{\partial C_{\boldsymbol{x}}}{\partial z_j^{(L)}} = \frac{\partial C_{\boldsymbol{x}}}{\partial a_j^{(L)}}\, g'(z_j^{(l)}) \quad (10)$$

or in the matrix form by:

$$\boldsymbol{\delta}^{(L)} = \nabla_a C_{\boldsymbol{x}} \circ g\left(\boldsymbol{z}^{(L)}\right) = (\boldsymbol{a}^{(L)} - \boldsymbol{y}) \circ g'\left(\boldsymbol{z}^{(L)}\right) \quad (11)$$

where $\nabla_a C_{\boldsymbol{x}} = \boldsymbol{a}^{(L)} - \boldsymbol{y}$ is the vector containing the components of $\frac{\partial C_{\boldsymbol{x}}}{\partial a_j^{(L)}}$ and $\circ$ denotes the Hadamard product. To backpropagate the error in other layers $l = L-1, L-2, \dots, 2$, we can use:

$$\boldsymbol{\delta}^{(l)} = \left(\left(\boldsymbol{w}^{(l+1)}\right)^T \boldsymbol{\delta}^{(l+1)}\right) \circ g'\left(\boldsymbol{z}^{(L)}\right) \quad (12)$$

where T denotes the transpose operation. By using (11) and (12), the error $\boldsymbol{\delta}^{(l)}$ for all the layers in the network can be calculated. Thus, the gradient of the cost function with respect to weight and bias can be calculated as:

$$\frac{\partial C_{\boldsymbol{x}}}{\partial b_j^{(l)}} = \delta_j^{(l)} \quad (13)$$

$$\frac{\partial C_{\boldsymbol{x}}}{\partial w_{jk}^{(l)}} = a_k^{(l-1)}\delta_j^{(l)} \quad (14)$$

Next, we can update the weight and bias values for $l = L-1, L-2, \dots, 2$ using the following equations:

$$\boldsymbol{w}^{(l)} = \boldsymbol{w}^{(l)} - \frac{\lambda}{N}\sum_x \boldsymbol{\delta}^{(l)}(\boldsymbol{a}^{(l-1)})^T \quad (15)$$

$$\boldsymbol{b}^{(l)} = \boldsymbol{b}^{(l)} - \frac{\lambda}{N}\sum_x \boldsymbol{\delta}^{(l)} \quad (16)$$

where λ is the regularization term.

Table 1 summarizes the neural network training process including the feedforward and backpropagation algorithms.

TABLE 1. THE NEURAL NETWORK TRAINING ALGORITHM

1- Start
2- Input a set of training examples
3- For each training example, feed the network with the input features, x, from GPS signal
4- Perform the feedforward operation using
$$\boldsymbol{a}^{(l)} = \boldsymbol{g}(\boldsymbol{z}^{(l)}) \text{ and } \boldsymbol{z}^{(l)} = \boldsymbol{w}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}$$
5- Calculate the error in the last layer, L by
$$\boldsymbol{\delta}^{(L)} = (\boldsymbol{a}^{(L)} - \boldsymbol{y}) \circ \boldsymbol{g}'(\boldsymbol{z}^{(L)})$$
6- Backpropagate the error in other layers by
$$\boldsymbol{\delta}^{(l)} = ((\boldsymbol{w}^{(l+1)})^{T}\boldsymbol{\delta}^{(l+1)}) \circ \boldsymbol{g}'(\boldsymbol{z}^{(L)})$$
7- Compute the gradient of the cost function by
$$\frac{\partial C_x}{\partial b_j^{(l)}} = \boldsymbol{\delta}_j^{(l)} \text{ and } \frac{\partial C_x}{\partial w_{jk}^{(l)}} = \boldsymbol{a}_k^{(l-1)}\boldsymbol{\delta}_j^{(l)}$$
8- Go back to 3 until all training samples are fed to the network.
9- Update the weight and bias values using
$$\boldsymbol{w}^{(l)} = \boldsymbol{w}^{(l)} - \frac{\boldsymbol{\mu}}{\boldsymbol{N}}\sum_x \boldsymbol{\delta}^{(l)}(\boldsymbol{a}^{(l-1)})^{T}$$
$$\boldsymbol{b}^{(l)} = \boldsymbol{b}^{(l)} - \frac{\boldsymbol{\mu}}{\boldsymbol{N}}\sum_x \boldsymbol{\delta}^{(l)}$$

B. Feature Selection

We extract different parameters from the received GPS signals to develop input features for the neural network. These features are:

- Satellite vehicle number (SVN)
- Signal-to-noise ratio (SNR)
- Pseudo range (PR)
- Doppler shift (DO)
- Carrier phase shift (CP)

Satellite vehicle number: A satellite vehicle number is used to identify different satellites orbiting the earth. This number can be simply read from the contents of decoded received GPS signals.

Signal-to-noise ratio: The SNR is an indicator of how strong the signal that carries the GPS information is after it is mixed with noise and interference. It can be measured and calculated from the received signals using specific algorithms [20].

Pseudo range: Every GPS satellite has a unique Gold code that its autocorrelation function has an equilateral triangle shape, where its peak happens when the correlation is perfect. This characteristic can be used to find the travel time of the signal from the satellite to the receiver by cross-correlating the Gold code with its receiver-generated replica. This calculated time difference, ΔT, along with the speed of light is used to estimate the distance, pseudo range, between the satellite and the receiver.

$$PR = \Delta T.c = (T - T_s).c \tag{17}$$

where T is the reception time at the receiver and T_s is the transmission time at the satellite. This process is shown in Fig. 2.

Doppler shift: At the receiver, the GPS carrier signal, $g(t)$, is multiplied by a reference signal, $f(t)$, as follows:

$$g(t) \times f(t) = A_g \sin\left(2\pi\varphi_g(t)\right) \times A_0 \sin\left(2\pi\varphi_0(t)\right) =$$

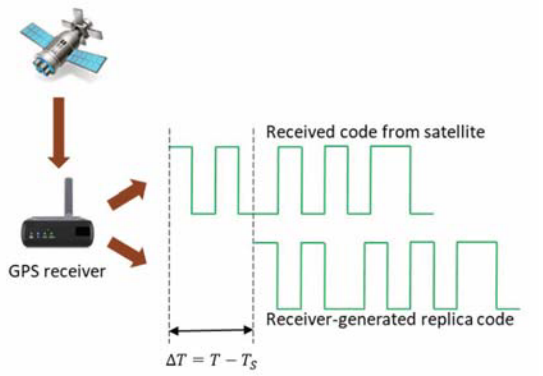

Fig. 2. An example of GPS signal phase difference.

$$\frac{A_g A_0}{2}[cos2\pi\left(\varphi_0(t) - \varphi_g(t)\right) - cos2\pi\left(\varphi_0(t) + \varphi_g(t)\right)] \tag{18}$$

where A_g, $\varphi_g(t)$, A_0, and $\varphi_0(t)$ are the amplitudes and phases of the received GPS signal and reference signal, respectively. The higher frequency component can be filtered out, keeping the lower frequency component, as follows:

$$b(t) = Filter\{g(t) \times f(t)\} = \frac{A_g A_f}{2} cos2\pi\left(\varphi_0(t) - \varphi_g(t)\right) \tag{19}$$

The phase difference of the GPS signal and the reference signal is defined as:

$$\varphi_d(t) = \varphi_0(t) - \varphi_g(t) - P \tag{20}$$

where P is phase ambiguity. By differentiating (20) with respect to time, the frequency difference between the received GPS signal and the reference signal can be calculated. This difference is an indicator of Doppler shift, f_d, given by

$$f_d = \frac{d\varphi_d(t)}{dt} = f_0 - f_g \tag{21}$$

Carrier phase shift: Referring to (20), we can rewrite the carrier phase shift observed at time T as follows:

$$\varphi(T) = \varphi_0(T) - \varphi_g(T) - P \tag{22}$$

We can also write the observation time as a function of phase and frequency as follows:

$$T = \frac{\varphi(T) - \varphi_0}{f_0} \tag{23}$$

Therefore, considering that the incoming signal phase received at time T is identical to the transmitted signal at time T^S, the carrier phase observable from satellite S is:

$$\varphi^S(T) = f_0 T + \varphi_0 - f_0 T^S - \varphi_0^S - P^S \tag{24}$$

Equation (24) can be extended to include multiple receivers and satellites as follows:

$$\varphi_j^i(T_j) = f_0 T_j + \varphi_{0_j} - f_0 T^i - \varphi_0^i - P_j^i \tag{25}$$

where i and j are used to identify an arbitrary satellite and an arbitrary receiver, respectively.

III. Results and Discussions

In this work, we assume that the attacker is able to spoof a GPS receiver via playback or meaconing attack. As mentioned previously, meaconing is receiving GPS signals in a remote location and rebroadcasting them at another location at a slightly higher power. We simulated meaconing signals by collecting GPS signals from a remote location. The data collection was done using real GPS signals received from a software-defined radio. The signals were received using the RTL-SDR V.3 RTL2832U. The antenna used was an active GPS SMA antenna. The software used was open source gnss-sdrgui.exe and open source rtknavi.exe. There are three main steps to receiving data: configure, enable antenna, and enable software. The RTL-SDR was set up with a center frequency of 1575.42 MHz, sampling frequency of 2.048 MHz for the I/Q signals, and looking for all GPS satellites.

To train and test the neural network, a large number of received GPS signals was collected at two locations spaced approximately 480 feet apart. Data collected at location 1 was considered to be the legitimate GPS data whereas data collected at location 2 was those that the attackers intend to replay at a later time. The legitimate and spoofing data were labeled by 0's and 1's, respectively.

The two labeled data were combined to get the dataset for testing the neural networks.. We selected features of the received signals that can most effectively distinguish between legitimate GPS signals and GPS spoofing signals. As the spoofing signals were collected at a different location, their characteristics, and thus the extracted features are different from that of the legitimate GPS signals. We tested the model using a K-fold cross validation method. Table 2 illustrates the simulation parameters used in this work. To compare the performance of different methods, we used four metrics namely probability of detection, P_d, probability of false alarm, P_{fa}, probability of miss detection, P_m, and accuracy. These metrics are calculated as follows:

$$P_d = P\left(a^{(L)} = 1 \middle| y = 1\right) \quad (26)$$

$$P_m = P\left(a^{(L)} = 0 \middle| y = 1\right) \quad (27)$$

$$P_{fa} = P\left(a^{(L)} = 1 \middle| y = 0\right) \quad (28)$$

$$\text{Accuracy} = P\left(a^{(L)} = 1 \middle| y = 1\right) + P\left(a^{(L)} = 0 \middle| y = 0\right) \quad (29)$$

where P_d is the probability that GPS spoofing signals are detected correctly as spoofing ones, P_m is the probability that spoofing signals are incorrectly detected as legitimate ones, P_{fa} is the probability that legitimate signals incorrectly detected as spoofing signals, and accuracy is the combined probability that both legitimate and fake signals are detected correctly as legitimate and fake ones, respectively.

Fig. 3 shows the impact of the database size on the detection accuracy of a neural network with 20 hidden neurons when different solver algorithms and activation functions are used. Solvers are used to find the optimal weights for the neural network. These algorithms try to minimize

Table 2. Simulation parameters used in this work

Parameter	Value/Range
Transmission frequency	1575.42 MHz
Type of code	C/A
Total number of training examples, N	2000
Ratio of legitimate to fake samples	50:50
Number of hidden layers	1, 2
Number of neurons in each hidden layer	[1, 25]
Activation function	tanh
Number of folds, K	10
Number of iterations	200

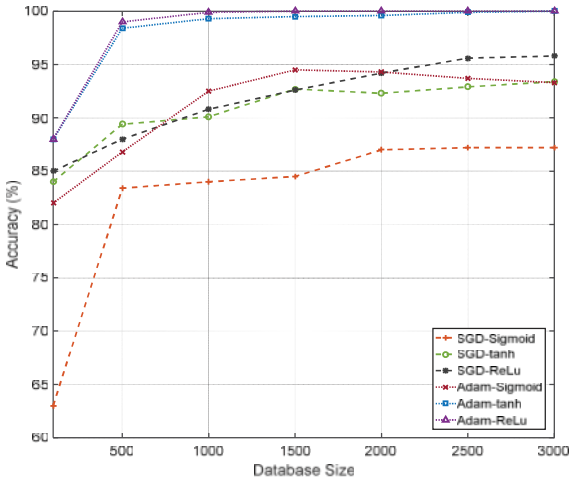

Fig. 3. Accuracy as a function of database size.

the cost function. The stochastic gradient descent (SGD) looks for the minimum using classical Newton-Raphson iterative solving but only operates on a subset of samples. The adaptive moment (Adam) estimation uses a dynamic per-parameter learning rate to find the minimum of the cost function. The first and second moments are used in the estimation of the best learning rate. As it can be seen, after a database size of 2000 samples, the detection accuracy of all the methods slightly changes. Therefore, we select this database size as the default size for the rest of the paper. The figure also shows that the highest and lowest accuracies, regardless of the database size, are obtained by Adam-ReLu and SGD-Sigmoid, respectively.

To identify how different features affect the performance of the algorithm, we fed different combinations of features from 1 to 5 to a one-hidden-layer neural network containing 10 hidden neurons with. The results of this analysis are shown in Table 3. As it can be seen, there are a total of 31 combinations of features. With only one feature, the highest accuracy of 73.2% is achieved with pseudo range, PR. As we increase the number of features to two, a significant increase in detection performance is observed. With two features, the best performance with an accuracy of 92.6%, a probability of detection of 85.2%, and a probability of false alarm of 0% is obtained with pseudo range, PR, and Doppler shift, DO as the features. By increasing the number of features to three, an overall improvement in the detection performance can be seen. With three features, the best performance with an accuracy of 98.2%, a probability of detection of 99.4%, and a false alarm probability of 3.1% is obtained with SNR, PR, and DO. With four and five features and taking the above discussion into

TABLE 3. IMPACT OF DIFFERENT FEATURES ON DETECTION PERFORMANCE

No. of Features	Features	Accuracy (%)	P_d (%)	P_{fa} (%)	P_m (%)
1	SVN	67.1	54.4	20.1	45.6
1	PR	73.2	50.0	3.6	50.0
1	CP	61.9	54.2	30.4	45.8
1	DO	62.1	58.1	33.9	41.9
1	SNR	71.8	66.2	22.6	33.8
2	SVN, PR	87.3	78.7	4.2	21.3
2	SVN, CP	80.6	73.7	12.4	26.3
2	SVN, DO	74.1	73.0	24.8	27.0
2	SVN, SNR	77.0	83.1	29.1	16.9
2	PR, CP	88.8	83.1	5.5	16.9
2	PR, DO	92.6	85.2	0.0	14.8
2	PR, SNR	90.8	90.9	9.2	9.1
2	CP, DO	69.9	66.1	26.3	33.9
2	CP, SNR	89.7	91.4	12.0	8.6
2	DO, SNR	88.7	91.7	14.2	8.3
3	SVN, PR, CP	88.4	85.0	8.1	15.0
3	SVN, PR, DO	94.1	89.0	0.8	11.0
3	SVN, PR, SNR	94.8	94.8	5.3	5.2
3	SVN, CP, DO	93.1	92.1	5.8	7.9
3	SVN, CP, SNR	90.5	94.1	13.1	5.9
3	SVN, DO, SNR	91.6	97.4	14.2	2.6
3	PR, CP, DO	92.9	87.1	1.2	12.9
3	PR, CP, SNR	94.6	97.4	8.3	2.6
3	PR, DO, SNR	98.2	99.4	3.1	0.6
3	CP, DO, SNR	93.7	92.4	5.0	7.6
4	SVN, PR, CP, DO	95.3	92.4	1.9	7.6
4	SVN, PR, CP, SNR	95.1	98.5	8.3	1.5
4	SVN, CP, DO, SNR	96.6	98.1	4.9	1.9
4	SVN, PR, DO, SNR	98.2	99.0	2.6	1.0
4	PR, CP, DO, SNR	97.9	98.8	3.0	1.2
5	SVN, PR, CP, DO, SNR	98.3	99.2	2.6	0.8

account, it can be seen that whenever the three features of PR, DO, and SNR are combined with other features, an accuracy of at least 98% is obtained. Therefore, we can reduce the complexity involved in classification by using only PR, DO, and SNR as features for the rest of this paper.

Fig. 4 illustrate the performance of a one-hidden-layer neural network as a function of number of hidden neurons in terms of accuracy, probability of detection, and probability of false alarm. As it can be seen, as the number of hidden neurons increases, the detection performance also increases. For instance, increasing the number of hidden neurons from 2 to 10 improves the accuracy by 9.2%. However, after about 14 hidden neurons, the accuracy does not change significantly because the neural network is saturated and no more relationship of features can be extracted by the algorithm. The same trend can be observed for the probability of detection and probability of false alarm. It must be noted that increasing the number of hidden neurons also increases the complexity of the algorithm.

To understand if adding a hidden layer to the NN improves the detection performance, we equally split the hidden neurons between two layers for a two-hidden-layer NN. Fig. 5 shows the performance of this network as a function of the number of neurons in the hidden layers in terms of accuracy, probability of detection, and probability of false alarm. A pair of (a, b) indicates the number of hidden neurons in the first layer, a, and in the second layer, b. As it can be seen, breaking the hidden later into two hidden layers improves the performance. For instance, considering only the accuracy, the one-hidden-layer network needs 10 neurons for an accuracy of about 98% whereas the two-hidden-layer NN achieves that accuracy with 6 neurons (3 neurons in each hidden layer). This improved performance comes at the cost of the increased complexity of deploying two hidden layers. A comparison of these two neural networks for number of neurons up to 10 is also shown in Fig. 6.

IV. CONCLUSIONS AND FUTURE WORK

Spoofing is one of the most important threats on GPS receivers. In this paper, we proposed a neural network based method to detect GPS spoofing messages that are previously recorded and retransmitted back to a GPS receiver. We selected five features: satellite number, carrier phase, pseudo range, Doppler shift, and SNR. These features were extensively analyzed. Those that maximize the accuracy and probability of detection and minimize the probability of false alarm were chosen. The efficiencies of different neural networks with different number of hidden neurons were analyzed. To evaluate these efficiencies, four metrics were used, namely probability of detection, probability of miss detection, probability of false alarm, and accuracy. Future work includes using other improved machine learning techniques. One possible avenue of the research is to include online learning in the neural network. Using online learning, the neural network is trained as new data comes in. Another possible direction is the use of unsupervised machine learning algorithms, which are able to classify unlabeled data. Unsupervised machine learning can be a suitable candidate to cluster unknown received messages before any further processing.

V. ACKNOWLEDGMENT

This research was supported by the North Dakota Established Program to Stimulate Competitive Research (ND EPSCoR) through National Science Foundation Grant No. OIA-1355466.

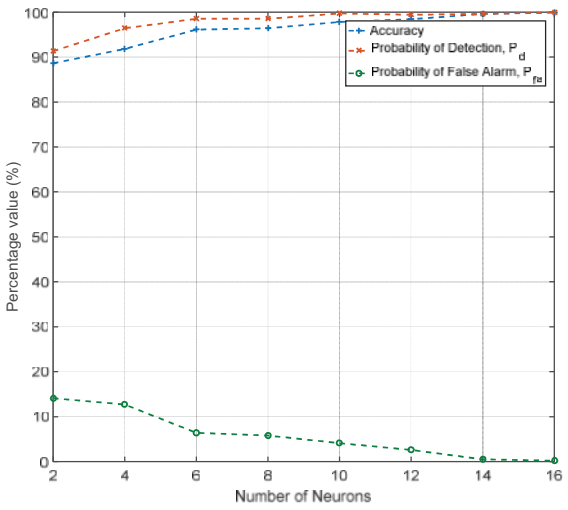

Fig. 4. Accuracy, probabilities of detection, and false alarm as functions of number of neurons in a one-hidden-layer NN.

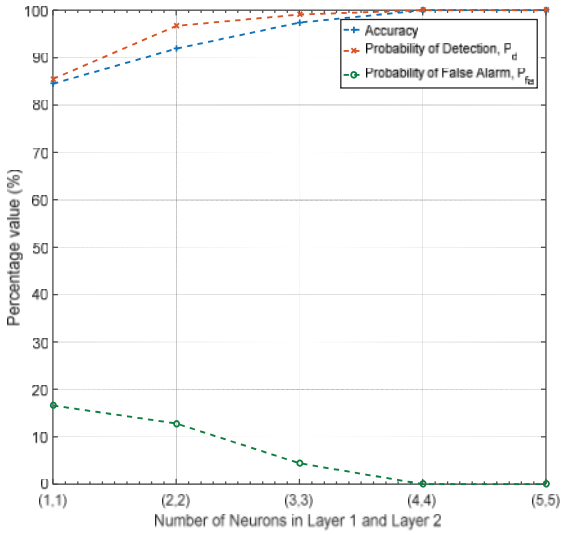

Fig. 5. Accuracy, probabilities of detection, and false alarm as functions of number of neurons in a two-hidden-layer NN.

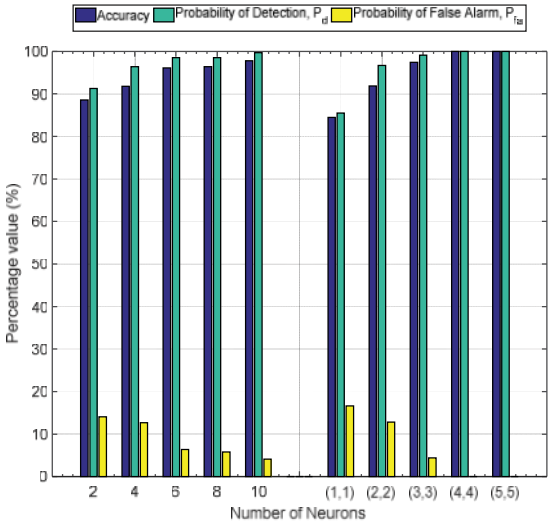

Fig. 6. Comparison of the detection performance of one- and two-hidden-layer neural networks.

REFERENCES

[1] Global Positioning System Standard Positioning Service Performance Standard, 4th ed., U.S. Department of Defense, Sep. 2008.

[2] M. Riahi Manesh, M. Mullins, K. Foerster, and N. Kaabouch, "A Preliminary Effort Toward Investigating the Impacts of ADS-B Message Injection Attack," IEEE Aerospace Conference, 2018.

[3] M. Riahi Manesh and N. Kaabouch. "Analysis of Vulnerabilities, Attacks, Countermeasures and Overall Risk of the Automatic Dependent Surveillance-Broadcast (ADS-B) System," International Journal of Critical Infrastructure Protection, 2017.

[4] Volpe, J. A., "Vulnerability Assessment of the Transportation Infrastructure Relying on the Global Position System," U.S. Department of Transportation, 2001.

[5] T. E. Humphreys, B. M. Ledvina, M. L. Psiaki, B. W. O'Hanlon, and P. M. Kintner Jr., "Assessing the Spoofing Threat: Development of a Portable GPS Civilian Spoofer," in International Technical Meeting of the Satellite Division of the Institute of Navigation, ser. ION GNSS, pp. 2314–2325, 2008.

[6] M. L. Psiaki and T. E. Humphreys, "Attackers Can Spoof Navigation Signals Without Our Knowledge. Here's How to Fight Back GPS Lies," IEEE Spectrum, vol. 53, no. 8, pp. 26–53, 2016.

[7] T. E. Humphreys, "Statement on the Vulnerability of Civil Unmanned Aerial Vehicles and Other Systems to Civil GPS Spoofing," The University of Texas at Austin, Tech. Rep., 2012.

[8] D.-Y. Yu, A. Ranganathan, T. Locher, S. Capkun, and D. Basin, "Short Paper: Detection of GPS Spoofing Attacks in Power Grids," ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp. 99–104, 2014.

[9] N. O. Tippenhauer, C. Popper, K. B. Rasmussen, and S. Capkun, "On the Requirements for Successful GPS Spoofing Attacks," ACM Conference on Computer and Communications Security, pp. 75–86, 2011.

[10] OSQZSS. (2017) Software-Defined GPS Signal Simulator. [Online]. Available: https://github.com/osqzss/gps-sdr-sim

[11] M. L. Psiaki and T. E. Humphreys, "GNSS Spoofing and Detection," Proceedings of the IEEE, vol. 104, no. 6, pp. 1258–1270, 2016.

[12] K. Wesson, M. Rothlisberger, and T. E. Humphreys, "Practical Cryptographic Civil GPS Signal Authentication," International Technical Meeting of The Satellite Division of the Institute of Navigation, pp. 3335–3345, 2012.

[13] B. W. O'Hanlon, M. L. Psiaki, J. A. Bhatti, D. P. Shepard, and T. E. Humphreys, "Real-Time GPS Spoofing Detection via Correlation of Encrypted Signals," Navigation, vol. 60, no. 4, pp. 267–278, 2013.

[14] A. J. Kerns, K. D. Wesson, and T. E. Humphreys, "A Blueprint for Civil GPS Navigation Message Authentication," IEEE/ION PLANS Meeting, Monterey, pp. 262–269, 2014.

[15] A. Ranganathan, H. Olafsdottir, and S. Capkun, "SPREE: A Spoofing Resistant GPS Receiver," ACM Conference on Mobile Computing and Networking, pp. 348–360, 2016.

[16] J. Magiera and R. Katulski "Detection and Mitigation of GPS Spoofing Based on Antenna Array Processing", Journal of Applied Research and Technology, Vol 13, pp. 45-47, Feb 2015.

[17] Panice, G., Salvatore Luongo, Gabriella Gigante, Domenico Pascarella, Carlo Di Benedetto, Angela Vozella, and A. Pescapè, "A SVM-Based Detection Approach for GPS Spoofing Attacks To UAV," IEEE International Conference on Automation and Computing (ICAC), pp. 1-11, 2017.

[18] Jansen, Kai, Matthias Schäfer, Daniel Moser, Vincent Lenders, Christina Pöpper, and Jens Schmitt, "Crowd-GPS-Sec: Leveraging Crowdsourcing to Detect and Localize GPS Spoofing Attacks," IEEE Symposium on Security and Privacy, pp. 1-14, 2018.

[19] M. A. Nielsen, "Neural Networks and Deep Learning," Determination Press, vol. 25, 2015.

[20] M. Riahi Manesh, A. Quadri, N. Kaabouch. "An Optimized SNR Estimation Technique Using Particle Swarm Optimization Algorithm," IEEE Computing and Communication Workshop and Conference, pp. 1–7, 2017.