

DIRECTIONAL MODULATIONS USING AN ARTIFICIAL NEURAL NETWORK

Rodion Shishkov and Deva K. Borah

Klipsch School of Electrical & Computer Engineering

New Mexico State University

Las Cruces, NM 88003, USA

Email: shishkov@nmsu.edu, dborah@nmsu.edu

ABSTRACT

A directional modulation (DM) system can provide physical layer security by distorting modulations along the eavesdropper directions while maintaining the correct modulation formats at the desired user. One DM approach is to optimize the transmit signals from the transmit antenna array so that a high bit error rate (BER) can be enforced at the eavesdroppers. However, this requires running an optimization algorithm each time the locations of the users change resulting in high numerical computations. To overcome this problem, a multilayer perceptron network from the artificial neural networks (ANN) is used in this paper to obtain the optimized transmit signals. This paper considers only one desired user and one eavesdropper, and the ANN is trained for various orientations of the users. The impact of hyperparameters, e.g., the number of neurons, is studied. The use of shallow and deep networks is investigated. Excellent BER performance is obtained with low numerical computations.

INTRODUCTION

Physical layer security (PLS) can play a significant role in protecting wireless data from an unintended user or eavesdropper (EV). The idea of PLS is based on the channel differences between the desired user (DU) and the EV. One of the PLS techniques uses directional modulation (DM) [1]-[3], where the modulated symbols are distorted in the directions of the EVs. In [4], an orthogonal vector approach is studied using artificial noise (AN). A zero forcing synthesis approach using AN is also given in [5] for multi-beam DM. In [6], an iterative convex optimization of DM with AN (ICODA) algorithm is presented that assigns a transmit symbol vector for each symbol of the DU. The ICODA algorithm designs the optimal transmit symbol vectors so that the data reception at the DU occurs with low bit error rate (BER) while the BER values at the EVs are above a certain value.

Our focus in the paper will be on the ICODA algorithm. The ICODA algorithm designs a set of M transmit symbol vectors for the transmission of an M -ary symbol alphabet. Each symbol vector has N components and is fed to the transmit antenna array consisting of N array elements. The BER penalty at the EV is performed by reducing the inter-symbol distance between specific symbol pairs at the EV. The algorithm consists of two iterative steps. During the first step, the

symbol vectors and the AN average power are obtained by maximizing the received power at the DU while satisfying the total transmit power as well as several other constraints as described in [6]. It is demonstrated that this step becomes a convex optimization procedure. In the second step, symbol pair selection or weighting is optimized to reduce the Euclidean distance between symbol pairs at the EV. The second step is shown to be a linear programming problem. The iterations between these two steps are performed several tens of times to converge to the desired solution.

While the ICODA optimization provides excellent results in terms of BER for both the DU and the EV, the iterative algorithm needs to be run every time the orientations of the DU and the EV change. Therefore, in dynamic situations where the DU/EV change their locations, the algorithm results in high numerical complexity. To overcome this problem, this paper makes the following contributions. First, we present a multilayer perceptron network from the artificial neural networks (ANN) to obtain the optimized transmit signal vectors. We consider only one DU and one EV although the idea can be extended to multiple DU/EV scenarios. Second, the impact of various ANN hyperparameters is studied. The use of shallow and deep networks is investigated. Finally, results are presented to show that the ANN-based approach provides excellent BER performance with low numerical computations. The ANN results also closely match the directly optimized BER performance at random test points.

The notations used in this paper are given as follows. Bold letters denote vectors. The notations $(\cdot)^T$, $(\cdot)^H$, $\|\cdot\|$, $E[\cdot]$, $\text{Re}(\cdot)$, $\text{Im}(\cdot)$ and $\mathbf{0}$ denote transpose, conjugate transpose, Euclidean norm, expectation, real, imaginary, and zero vector respectively. The Q function is defined as $Q(x) = 1/\sqrt{2\pi} \int_x^\infty \exp(-y^2/2)dy$.

SYSTEM MODEL

Let us consider a wireless communication system consisting of an access point (AP) transmitting data to a DU in the presence of an EV. The AP uses a uniform linear array of N antenna elements with an inter-element spacing of d . Both the DU and the EV use single antenna receivers. Let \mathbf{x} be the transmit vector of length N from the AP. We define $\mathbf{x} = \mathbf{s}_i + \mathbf{n}$, where \mathbf{s}_i is the i -th transmit vector corresponding to an M -ary alphabet so that $\mathbf{s}_i \in \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_M\}$ and \mathbf{n} is the AN vector with $\xi = E[\|\mathbf{n}\|^2]$ as the total average AN power. Let $k = 1$ denote the DU and $k = 2$ denote the EV. Then the channel vector for the k -th user located along the direction θ_k is given by $\mathbf{h}_k = [h_{k,1}, h_{k,2}, \dots, h_{k,N}]^T$, where $h_{k,m} = \exp(-j(2\pi/\lambda)(m - (N + 1)/2)d \cos \theta_k)$, $m = 1, 2, \dots, N$, and λ is the wavelength. The signal received at the k -th user is

$$y_k = \mathbf{h}_k^H \mathbf{x} + z$$

where z is complex additive white Gaussian noise (AWGN) with a variance of σ_n^2 . The ICODA method ensures that the symbol received at the DU is not distorted, i.e., $\mathbf{h}_1^H \mathbf{x} = \nu a_i$, where ν is a constant and a_i is the i -th symbol from the M -ary symbol alphabet. Similarly, to distort the symbols at the EV, a lower bound on the symbol error rate, given by $P_{LB} = (2/M)Q(\sqrt{\Gamma_{min}})$, with $\Gamma_{min} = d_{min}^2/2N_o$, is used, where N_o is the AWGN power spectral density, and d_{min} is the minimum Euclidean distance. The Euclidean distance between the i -th and the j -th symbols at the EV is given by $d_{i,j}^2 = \|\mathbf{h}_2^H (\mathbf{s}_i - \mathbf{s}_j)\|^2$. The lower bound P_{LB} on the symbol error rate is constrained to be higher than a certain threshold.

The first step of the ICODA algorithm maximizes ν over the parameters s_1, s_2, \dots, s_M and the AN parameter ξ so that the DU has maximum received signal power subject to (1) transmit power constraint: $(1/M) \sum_{i=1}^M \|s_i\|^2 + \xi \leq P_{max}$, (2) no distortion of symbols at the DU: $\mathbf{h}_1^H s_i = \nu a_i$, (3) zero-mean symbol vectors: $(1/M) \sum_{i=1}^M s_i = \mathbf{0}$, and (4) desired level of symbol distortion at EV: $\sum_{l=1}^L \omega_l \Gamma_l \leq \delta_{th}$, where $\omega_l, l = 1, 2, \dots, L$, are non-negative weighting coefficients, L is the number of symbol pairs, and δ_{th} is a threshold. After obtaining $s_i, i = 1, 2, \dots, M$ and ξ using Step 1, the second step of linear optimization is performed over $\omega_l, l = 1, 2, \dots, L$, to minimize a variable t subject to $\sum_{l=1}^L \omega_l \Gamma_l \leq t, \sum_{l=1}^L \omega_l = 1$ and $\omega_l \geq 0$ for all l . The purpose of the second step is to take advantage of the symbol pairs that degrade the error performance at the EV and maximize their degrading effect. After obtaining the solution for $\omega_l, l = 1, 2, \dots, L$, Step 1 is performed again and the process is iterated to get the final solutions.

NEURAL NETWORK IMPLEMENTATION

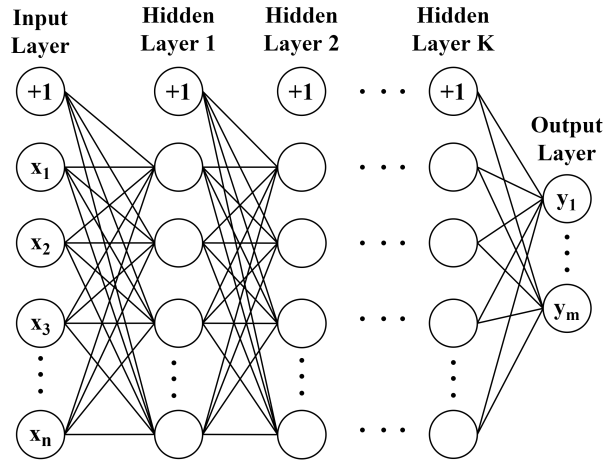


Figure 1: MLP diagram

In this section, we present an overview of the multilayer perceptron (MLP) NNs and discuss their implementation for generating ICODA symbols. A detailed description of MLP NNs is available in [7] and [8].

Consider the block diagram of a general MLP NN given in Fig 1. The MLP consists of an input layer, an output layer and K hidden layers [7]. The input data is introduced to a model at the input layer. The data propagate through hidden layers from left to right until they reach the output layer that produces system outputs. This type of data propagation in MLP is referred to as feedforward NNs. Each hidden layer consists of a bias term illustrated as a circle with a $+1$ as the input and neuron units which are depicted as empty circles on the diagram. All layers of MLP are densely connected which means that each neuron unit of a layer L is connected with all neurons to the following layer $L+1$ as well as with all the neurons in the preceding layer $L-1$. These connections are implemented via weights w_{nj} as shown in Fig 2.

In Fig. 2, we show an artificial neuron (perceptron) model. The inputs to the neuron model can come from the outputs of the neurons from a preceding layer or can come directly from the external

input vector. The scalar output y'_j from the j th neuron is the output of an activation function operating on the weighted linear combination of the input signals so that

$$y'_j = f\left(\sum_{i=1}^{n'} w_{ij}x'_i + w_{0j} \cdot 1\right)$$

where x'_i is an input, and $f(\cdot)$ is the nonlinear activation function such as rectified linear unit (ReLU) that is defined as $y(x) = \max(0, x)$.

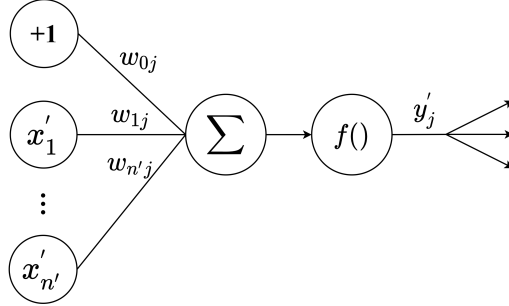


Figure 2: Artificial neuron model where x'_n represents the n -th input applied to a neuron, $+1$ is a bias term with a corresponding weight w_{0j} , w_{ij} is the weight on the i -th input to the neuron, $f(\cdot)$ is the nonlinear activation function, and y'_j is the output of the j -th neuron.

Once an MLP NN structure is chosen, it can be trained to approximate any function with arbitrary accuracy [7]. The back-propagation algorithm is widely used in supervised training of MLP NNs. To train an NN with back-propagation, the input training data is first passed to the network to obtain a corresponding output. Next, the NN response is compared to the expected output to calculate the error. The derivative of the error is calculated with respect to the NN weights so that the weights can be adjusted to minimize the error. One common choice of an error function is the mean squared error (MSE), defined as $MSE = \frac{1}{\zeta} \sum_{i=1}^{\zeta} (y_i - \hat{y}_i)^2$, where y_i is the expected output, \hat{y}_i is NN response and ζ is the number of output samples. The process is repeated for all available training samples usually multiple times. Each such time, where the entire training data is passed to the NN to update the weights, is called a training epoch. Finally, after the NN is trained to a required accuracy it can be used to produce desired outputs.

The training and testing data for the NN are generated using the constrained nonlinear optimization as given in the previous section. The orientation of the DU and EV are determined by angles θ_1 and θ_2 respectively. All orientations within a range $[\theta_{min}, \theta_{max}]$ are allowed with the constraint that $|\theta_1 - \theta_2| \geq 2^\circ$ to avoid placing the DU and EV too close to each other. When the DU and EV are located too close to each other, the algorithm does not work well.

A single training sample consists of an input vector and a corresponding output vector. We obtain the DU and EV channels, \mathbf{h}_1 and \mathbf{h}_2 respectively, for all possible values of the orientation angles, θ_1 and θ_2 . These channel values are used as the input data. The optimized ICODA symbol vectors \mathbf{s}_i for the corresponding channel responses are used as the desired output vectors from the NNs.

Input and output vectors are complex-valued. However, the MLP NNs take one-dimensional vectors as input and output. Therefore, the training data are reshaped to one-dimensional vectors by

stacking the real part of a vector on top of the imaginary part. Thus, the l -th input and output vectors \mathbf{X}_l and \mathbf{Y}_l are formed in the following way: $\mathbf{X}_l = [\text{Re}(\mathbf{h}_1^T), \text{Im}(\mathbf{h}_1^T), \text{Re}(\mathbf{h}_2^T), \text{Im}(\mathbf{h}_2^T)]^T$ $\mathbf{Y}_l = [\text{Re}(\mathbf{s}^T), \text{Im}(\mathbf{s}^T)]^T$, where $\mathbf{s} = [\mathbf{s}_1^T, \mathbf{s}_2^T, \dots, \mathbf{s}_M^T]^T$.

The training data for the ANN are generated by running the nonlinear ICODA algorithm for various orientations of the DU and the EV. Unfortunately, the solutions obtained in this way are found to be ineffective in training the network. The reason is that the solution obtained for a given DU/EV orientation may considerably differ from the solution obtained for another orientation of the users due to the nature of the nonlinear optimization algorithm and the possibility of local convergence. The ANN, therefore, finds it hard to see a pattern in the data. To address this issue, we propose a novel approach where the solution obtained for a given DU/EV orientation is used to initialize the algorithm for the next closest DU/EV orientation. Toward this end, we choose the DU orientation in steps and for each DU orientation step we consider all possible EV orientations in steps of $\delta\theta$. Therefore, the ICODA transmit vector solution obtained for the DU/EV orientation pair of (θ_1, θ_2) is used to initialize the algorithm for the second orientation pair of $(\theta_1, \theta_2 + \delta\theta)$. The solution obtained for the second orientation pair is then used to initialize the algorithm for the third orientation pair of $(\theta_1, \theta_2 + 2\delta\theta)$ and the process continues. The solution patterns now are captured well by the ANN as will be demonstrated in the numerical results section.

To establish model accuracy and optimal selection of training epochs, test data are used for cross-validation. The test data are generated using random orientation angles of the DU and the EV within the range $[\theta_{min}, \theta_{max}]$, and they are different from the training orientation data. To compare the performance of the ANN based solution, we also directly run the ICODA algorithm for all test orientations. The algorithm for each test point orientation is initialized with the optimized solution obtained from the closest training orientation data. The number of test points is 10% of the overall number of data points.

On implementation of the ANN, we use a high-level application programming interface of TensorFlow called Keras in Python programming language. We define the MLP as densely connected layers and assign activation functions to each layer. The *ReLU* activation function is used in the hidden layers. The output layers should have a linear activation function so it can return output values directly without changing weighted sum of signals from the last hidden layer. We use the MSE loss function during the training that is optimized with ADAM optimizer for updating the NN weights. Training of NNs should be performed in mini-batches for better NN stability. Batches of 32 samples are used. Next, we assign neurons to each hidden layer by first choosing the overall number of neurons for an NN which is divided evenly among all hidden layers.

To characterize the numerical complexity, consider an MLP NN with a single-hidden layer that takes n input samples, and has p_1 and p_2 neurons in hidden and output layers respectively. There is also a bias term of +1. The number of multiplications and additions to process a single input sample x_n by a fully connected NN with a single-hidden layer is given by $(n + 1)p_1 + (p_1 + 1)p_2$ and $np_1 + p_1p_2$ respectively. In addition, a fully connected NN also requires $p_1 + p_2$ applications of activation functions.

To compare the computational differences between the ANN based approach and the ICODA based optimization algorithm, we also estimate the elapsed execution times on 100 random orientations of DU and EV. The results show that the optimization based ICODA requires 0.89 sec to process one orientation of the DU/EV, whereas the ANN based method with two hidden layers and $p = 108$ neurons requires 4.14×10^{-4} sec on average. As we can see, ANN based approach provides

significant computational advantage over direct ICODA optimization.

NUMERICAL RESULTS

Our numerical results use $N = 5$, Quadrature Phase Shift Keying (QPSK) modulation ($M = 4$), $d = \lambda/2$ and a carrier frequency of 2.35×10^9 Hz. Unless otherwise stated, we use a two-layer NN, the AWGN variance is 0.05 and the number of neurons is $p = 108$. We also set AN power $\xi = 0$ for simplicity. The training data size used for orientation angle step size $\delta\theta = 1$ degree is about 16,000.

Figure 3 compares the performance of both shallow and deep NNs for varying number of hidden layer neurons p . The single-layer MLP curve represents the performance of shallow NN whereas two and three-layer MLP curves are deep NNs with 2 and 3 hidden layers. The neurons are distributed evenly among all hidden layers for each model. Thus, if a shallow NN has p neurons in its hidden layer, then two and three-layer MLP models have $p/2$ and $p/3$ neurons in each layer respectively. The MSE is obtained over the test data points used in the cross-validation during the training. The figure also demonstrates that the training data obtained using random initialization of the ICODA algorithm for various DU/EV orientations are not useful. Our proposed initialization, where the previous solution from the nearest orientation angles is used to initialize the ICODA algorithm, allows successful training of the NNs. The figure shows that the single-layer MLP curve has the highest averaged MSE. Two and three-layer MLP models have similar MSE performance starting from $p = 72$. The results show that the performance of a two or three-layer MLP improves significantly by a factor of about 10 when p increases from 40 to 100. The improvement beyond $p = 100$ is slow and is by about a factor of two when p is raised to 180.

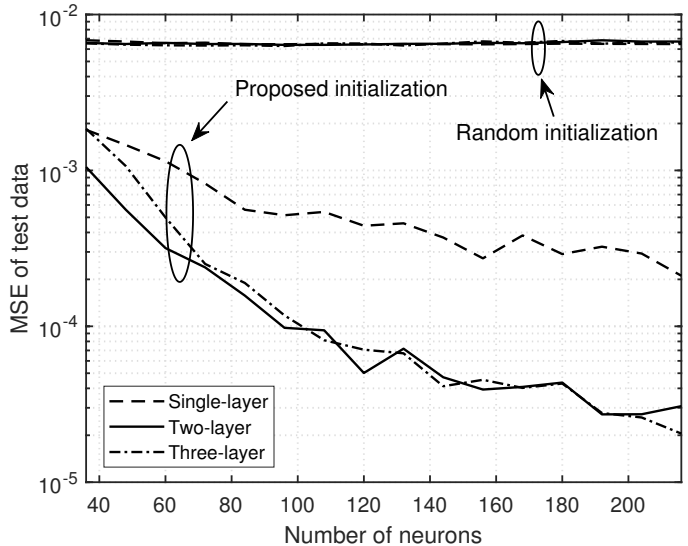


Figure 3: Impact of the number of ANN neurons

Figure 4 demonstrates how training data resolution affects MSE of test data for one, two and three-layer MLP NNs. The figure shows that the use of the training data with a higher step size, $\delta\theta$,

increases the MSE on the test data. A step size of 0.5 or 1 degree gives nearly similar results. Each point in this figure corresponds to an ensemble averaged MSE over 5 NNs. These NNs have the same architecture but have different initialization parameters. The performance of the shallow NNs degrades quickly with higher $\delta\theta$, whereas the performance of two-layer NN is quite similar to three-layer NNs. For $\delta\theta = 1^\circ$, the averaged MSE of a two-layer NNs is approximately 0.8×10^{-4} while it is 1.0×10^{-4} for a three-layer NN. For the same value of $\delta\theta$, the MSE for a shallow NN is about 4.8×10^{-4} .

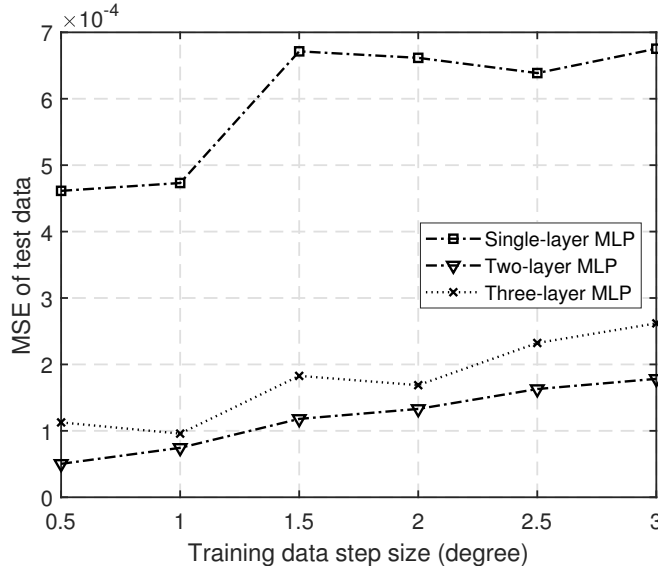


Figure 4: Effect of the resolution of the training data.

In Fig. 5, we display the BER performance against the inverse of the AWGN variance ($1/\sigma_n^2$) at the DU. The ANN based solution is obtained from the trained MLP model. Therefore, there is no need to run the ICODA algorithm for the ANN based curves. The optimized curves are obtained by running the ICODA algorithm for each data point. In the figure, Case A refers to low separation of the DU and the EV with $\theta_1 = 78.32^\circ$ and $\theta_2 = 80.55^\circ$. Case B refers to medium separation between the DU and the EV with $\theta_1 = 95.12^\circ$ and $\theta_2 = 102.13^\circ$. Finally, Case C uses a high separation between the DU and the EV with $\theta_1 = 122.42^\circ$ and $\theta_2 = 156.52^\circ$. The EV noise variance is assumed constant so that a BER of 0.1 is ensured at the EV. The figure shows that the ANN based performance is very close to the optimized performance. Also, Case C shows the best BER performance due to the high separation between the DU and the EV, while Case A's performance is relatively poor as the separation between the DU and the EV is only about 2 degree. Finally, a more detailed BER performance study for both the DU and the EV is described in Table 1. For each DU orientation, two EV orientations on each side of the DU are considered. The EVs are oriented close to the DU with a separation of ± 2 and ± 4 degrees respectively. The results show that the performance of the ANN based method is similar to the optimized results for both DU and the EV. In most of the cases where the ANN BER at the EV is found to be slightly falling below the requirement, there is improvement in the BER performance at the DU.

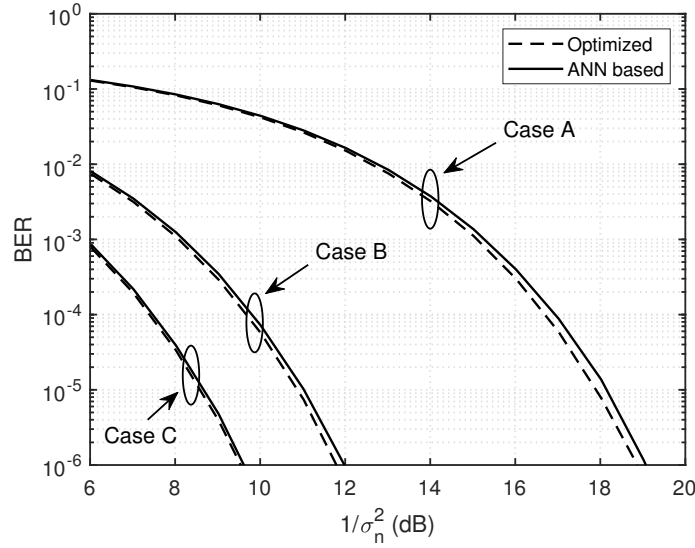


Figure 5: BER Performance of the DU for various EV orientations. Case A, B and C refer to low, medium and high separation of the DU and EV orientations respectively.

DU direction (degree)	EV direction (degree)	BER at DU		BER at EV	
		Optimized	ANN based	Optimized	ANN based
30	26	1.31×10^{-3}	1.60×10^{-3}	0.11	0.12
	28	4.32×10^{-2}	2.70×10^{-2}	0.13	0.11
	32	3.57×10^{-2}	1.50×10^{-2}	0.12	0.08
	34	4.26×10^{-4}	2.40×10^{-4}	0.11	0.08
60	56	5.70×10^{-7}	1.44×10^{-6}	0.11	0.12
	58	2.69×10^{-3}	3.15×10^{-3}	0.11	0.11
	62	2.31×10^{-3}	2.25×10^{-3}	0.11	0.12
	64	2.75×10^{-7}	2.50×10^{-7}	0.11	0.11
90	86	1.50×10^{-8}	1.00×10^{-7}	0.11	0.12
	88	7.52×10^{-4}	8.47×10^{-4}	0.11	0.11
	92	7.45×10^{-4}	3.63×10^{-4}	0.11	0.10
	94	1.50×10^{-8}	4.50×10^{-8}	0.11	0.11
120	116	2.85×10^{-7}	3.35×10^{-7}	0.11	0.10
	118	2.32×10^{-3}	1.03×10^{-3}	0.11	0.07
	122	2.69×10^{-3}	1.51×10^{-3}	0.11	0.09
	124	5.50×10^{-7}	5.90×10^{-7}	0.11	0.08

Table 1: BER performance comparison of the ANN based solution with the optimized solution.

CONCLUSIONS

This paper studies the ANN implementation of a directional modulation technique that can provide physical layer security to a desired user in the presence of an eavesdropper. Conventionally, this method has been used in the literature to obtain the optimal transmit symbol vectors using an iterative algorithm. This approach, however, requires running the optimization algorithm each time the orientations of the users change resulting in high numerical computations. To overcome this problem, the proposed method trains an ANN using various orientations of the desired user and the eavesdropper. The trained ANN can then be used to provide the optimal transmit symbols for any possible orientations of the users. Our results show that an ANN with two layers and more than 100 neurons can provide high performance. The BER results obtained for various test data points using the ANN are very close to the results obtained by directly running the optimization algorithm at each data point.

References

- [1] M. P. Daly and J. T. Bernhard, "Directional modulation technique for phased arrays," *IEEE Trans. Antennas Propag.*, vol. 57, pp. 2633–2640, Sept 2009.
- [2] Y. Ding and V. F. Fusco, "MIMO-inspired synthesis of directional modulation systems," *IEEE Antennas and Wireless Propagation Letters*, vol. 15, pp. 580–584, 2016.
- [3] R. M. Christopher and D. K. Borah, "Physical layer security for weak user in MISO NOMA using directional modulation (NOMAD)," *IEEE Communications Letters*, vol. 24, no. 5, pp. 956–960, 2020.
- [4] Y. Ding and V. Fusco, "Orthogonal vector approach for synthesis of multi-beam directional modulation transmitters," *IEEE Antennas and Wireless Propagation Letters*, vol. 14, pp. 1330–1333, 2015.
- [5] T. Xie, J. Zhu, and Y. Li, "Artificial-noise-aided zero-forcing synthesis approach for secure multi-beam directional modulation," *IEEE Communications Letters*, vol. 22, pp. 276–279, Feb 2018.
- [6] R. M. Christopher and D. K. Borah, "Iterative convex optimization of multi-beam directional modulation with artificial noise," *IEEE Communications Letters*, vol. 22, no. 8, pp. 1712–1715, 2018.
- [7] R. Reed and R. MarksII, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. A Bradford Book, MIT Press, 1999.
- [8] C. Bishop, P. Bishop, G. Hinton, and O. U. Press, *Neural Networks for Pattern Recognition*. Advanced Texts in Econometrics, Clarendon Press, 1995.