FOREX Publication
Open Access | Rapid and quality publishing

International Journal of
Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 228-237 | e-ISSN: 2347-470X

Design of a Deep Learning based Intelligent Receiver for a Wireless Communication System

Drakshayini M.N.[1], Manjunath R. Kounte[2] and Chaya Ravindra[3]
[1,2,3]School of Electronics and Communication Engineering, REVA University, Bengaluru, India; [1]mndrakshayini@gmail.com, [2]manjunath.kounte@gmail.com, [3]chaya.ravindra@gmail.com

***Correspondence:** Drakshayini M.N; mndrakshayini@gmail.com

ABSTRACT- In communication systems, deep learning techniques can provide better predictions than model-based methods when the hidden features of the problem are prone to deviating substantially from the formulated assumptions. Severe signal impairments due to multipath fading and higher channel noise levels degrade the performance of conventional receivers. To overcome this, a novel intelligent receiver based on a deep learning network is presented, achieving better performance in terms of reduced bit error rate than a standalone conventional receiver. The experimental result shows that the relative decrement in the symbol error ratio due to the proposed method is about 9% compared to the traditional receiver when the Rician channel fading is relatively high.

Keywords: Confusion matrix, Deep learning, intelligent receiver, Percentage training error, Rician flat channel, Symbol Error Ratio.

ARTICLE INFORMATION

Author(s): Drakshayini M.N., Manjunath R. Kounte and Chaya Ravindra
Received: 30/11/2023; **Accepted**: 14/02/2024; **Published**: 20/03/2024;
e-ISSN: 2347-470X;
Paper Id: IJEER 3011-22;
Citation: 10.37391/IJEER.120132
Webpage-link:
https://ijeer.forexjournal.co.in/archive/volume-12/ijeer-120132.html

Publisher's Note: FOREX Publication stays neutral with regard to Jurisdictional claims in Published maps and institutional affiliations.

1. INTRODUCTION

The basic task of a wireless communication receiver is to faithfully recover the transmitted data despite the distortions due to channel impairments. Conventionally, the channel distortion effects are compensated using channel equalization techniques [1–2] and noise cancellation methods [3–4]. However, the equalization process may be incomplete when the channel state is challenging to estimate due to environmental fluctuations and unseen inter-channel interferences. The noise cancellation could also be imprecise when the actual channel noise differs from the assumed noise model. Then, the traditional model-based methods fail to compensate for the channel distortion fully. In such a scenario, the deep learning-assisted receiver design can provide improved data recovery with reduced latency.

Deep learning (DL) is realized using deep learning networks (DLNs), which are essentially extended versions of artificial neural networks. DLNs implement complex algebraic operations [5] embedded within their intermediate layers to capture and learn the features and patterns of the input data. Inspired by the admirable success of DL techniques in image classification and encouraged by the earlier successful applications of adopting DLNs in digital wireless communication, we have developed a novel intelligent receiver, a combination of the traditional receiver and the proposed DLN. It is basically a symbol-by-symbol detector, unlike a sequence detector. Here, DLN is implemented using a custom Convolutional Neural Network, and the proposed scheme is denoted as CNN-IR (CNN-Assisted IR). The main objective of CNN-IR is to minimize the data recovery error at the receiving end.

2. PREVIOUS WORK

Recently improved techniques of DL with proven designs have paved the way for their applications in modern wireless communication systems like 5G, wireless LANs, software-defined radios (SDRs), OFDM with MIMO, *etc*. Several authors have used machine learning (a superset of DL)-based methods for the channel equalization and the receiver design, where the channel distortion due to multipath fading and the Gaussian noise are compensated.

In [6], Osvaldo Someone has given a substantial introduction to the applications of machine learning to solve problems in wireless communication systems where conventional modeling may not be accurate due to adverse environmental conditions with excessive channel fading and distortions. The author has discussed several scenarios where super skilled learning can be adopted to achieve reliable communication.

In [7], various DL-based techniques available for improved and more reliable communication experiences in IoT and 5G systems have been reviewed. The authors have discussed the application of DL techniques for the design of wireless receivers, transmitters, and channel estimators to achieve their optimal performances. Apart from this, the various challenges to be conquered to adopt DL for wireless communication successfully have been elaborated.
In [8], the authors have reviewed different DL-based schemes available for the physical layer to enhance its performance.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 228-237 | e-ISSN: 2347-470X

Different DL-based designs for signal compression with subsequent detection have been presented, which are useful in the case of redundant data sources. Additionally, a few end-to-end communication system models implemented using DL techniques have been reviewed. Finally, the authors have indicated the possible future research direction in this field.

In [9], the application of DL to five broad topics has been reviewed. These are channel estimation, cognitive radio receivers, communication with edge computing, end-to-end encoder-decoder, and communication using visible light. The prospects and challenges of further research and development in these areas are discussed in detail.

In [10], the author used an AI-aided receiver design that mainly uses DL techniques to realize an intelligent receiver. The receiver is specially designed for linear block-coded (Hamming, Reed Solomon, etc.) transmission. At the receiver, multiple binary classifiers are used for classifying symbols in the range [0 to (M–1)] in the case of M-PSK. Therefore, the DL net gets complex, and the training time will be relatively large. The performance is suboptimal when used without error-correcting channel encoders.

In [11], the end-to-end communication system is treated as an autoencoder, and then its working is optimized to achieve a higher degree of performance jointly by the transmitter as well as the receiver. The authors have introduced the 'Radio Transformer Network' (RTN) to assist the signal processing by the autoencoder to get better overall performance. Additionally, a convolutional neural network has been presented to detect the type of modulation adopted by the transformer by observing the received signal patterns.

In [12], a deep receiver is designed using a one-dimensional convolutional network, DenseNet, that can decode transmitted data streams of different lengths. Multi-bit data classification is achieved using multiple binary classifiers in this scheme. The authors have shown that the deep receiver is capable of discovering the transmitted data in the case of multiple modulation and coding schemes. The use of multiple binary classifiers increases the computational cost of the scheme and may not give the correct result when the modulation type is QAM.

In [13], the DL techniques are used to implement different sub-systems of the receiver for joint optimal performance, and it is shown that these methods provide better information recovery compared to the traditional methods in MIMO communication. The authors have described in detail the use of ResNet and DenseNet to implement an intelligent receiver. Additionally, MobileNetV2 has been adopted with suitable modifications to act as the intelligent receiver.

In [14], the authors have described the application of DL for decoding the polar-encoded data packets. Even though this method works well in the presence of AWGN in the channel, it cannot handle distortion due to the fading channels.

In [15], DL is used to solve two associated problems in digital communication systems: algorithmic approximation and signal detection. In algorithmic approximation, a known iterative algorithm that takes a long time (hence may not be suitable for online high-speed communication) is replaced by a DL network to get quick results. For example, power allocation in MIMO, iterative estimation of channel coefficients, etc. In signal detection, DL is used as an intelligent receiver that can replace a complex hardware unit.

In [16], the author has used DL techniques to detect the transmitted symbols. A CNN carries out symbol-by-symbol detection. Additionally, sequence-by-sequence detection has been implemented using a Long Short-Term Memory (LSTM) network. Here, molecular communication is used to test the algorithms.

In [17], the LSTM deep learning (DL) scheme is implemented to detect the signal in a multiple-access multi-carrier modulation scenario with generalized Gaussian noise and fading. Signal detection in the uplink and downlink modes is realized without the use of the Successive Interference Cancellation Unit.

In [18], the authors have presented a DL-based detector for a single-carrier non-orthogonal multi-access communication system with index modulation. The detector eliminates the successive interference cancellation unit and is found to perform better in the presence of severe interchannel interference.

2.1 Main contributions

Several DL-based techniques have been published for the detection of the transmitted signal without direct knowledge of the present channel state or the noise levels. However, their performances are application-specific and not fully error-free. Therefore, in DLA-IR, our contribution is to provide error-free symbol detection in the case of the M-PSK single carrier modulation system.

3. PRELIMINARIES

This paper considers an M-PSK transmitter-receiver system with M = 4. However, the principle can be easily extended to other values of M, namely 2, 8, 16, etc., and also to the various QAM configurations. Here, the wireless communication system is simulated using the MATLAB communication toolbox, and the various signals involved therein are obtained by running the simulation to cover the desired range of values.

3.1 Baseband Communication Model

The baseband communication model under consideration, with a Recian Flat Fading Channel, is shown in *figure 1*.

The different symbols used here are represented as follows;
Symbol $\boldsymbol{x}$ is the input data stream of length N expressed as

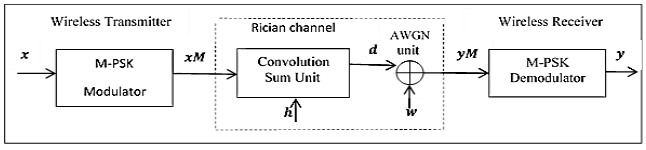

Figure 1: Base band block diagram of a basic Wireless Communication System

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 228-237 | e-ISSN: 2347-470X

$$\boldsymbol{x} = \left[x(t_0),\ x(t_0+\tau), \ldots, x(t_0+n*\tau), \ldots, x\big(t_0+(N-1)\big)*\tau\right]$$

Without loss of generality, for simplicity in explaining, t_0 is taken as 0, and the symbol interval τ s set to 1. We also use array indexing, starting with 1 instead of zero. Then, in its simplified form, the column vector is written as,

$$\boldsymbol{x} = [x(1), x(2), \ldots, x(n), \ldots, x(N)]^{\mathrm{T}} \quad (1)$$

Here, T stands for the transpose of the array. In the M-PSK modulation scheme, element $x(n) \in [0 \text{ to } (M-1)]$, for n = 1 to N. The size of x is N×1. As a baseband unit, the wireless transmitter essentially operates as an M-PSK modulator. The output of the modulator is given by $\boldsymbol{xM}$ as,

$$\boldsymbol{xM} = [xM(1), xM(2), \ldots, xM(n), \ldots, xM(N)]^{\mathrm{T}} \quad (2)$$

In (2), the M-PSK modulated Output $xM(n)$ corresponds to the data input $x(n)$ for n = 1 to N and $xM(n)$ is a complex number related to $x(n)$ as [19],

$$xM(n) = \exp\big(i*\theta(n,M)\big) \text{ where } \theta(n,M) = \pi*\left(\tfrac{2*x(n)+1}{M}\right) \quad (3)$$

For two consecutive x(n)'s, the corresponding phase vectors differ by $(2*\pi)/M$ radians. In *figure 1*, in terms of the baseband, the vector (stream) xM is the output of the modulator, which is the input to the Rician Flat channel.

3.1.1 Rician Flat Channel

In the proposed scheme (CNN-IR), the Rician Flat Channel (RFC) model is adopted, assuming the existence of a dominant line of sight (LoS) path from the transmitter to the receiver. The RFC is assumed to be a non-frequency selective channel with stationary fading characteristics. The RFC is represented by the Tapped Delay Line (TDL) model, as shown in *figure 2*.

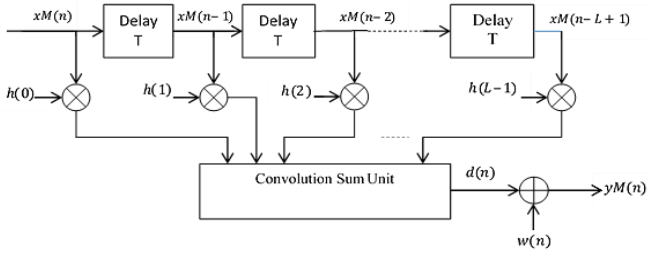

Figure 2: L-tap Rician Flat Channel Model

The channel gain coefficients are taken as follows:

$$h = [h(1), h(2), \ldots, h(L)]^{\mathrm{T}} \quad (4)$$

The coefficients are generated based on the Rician distribution [20]. In *eq. (4)* L represents the number of wireless paths from the transmitter to the receiver. Here, $h(1)$ is the gain of the LoS path. The normalized path gain of successive paths are taken as [pg(1), pg(2),…,pg(L)] and pg(1) is set to 1. The output of the TDL unit is represented by $\boldsymbol{d}$, and it is obtained by the convolution of xM and h as $\boldsymbol{d} = \boldsymbol{xM} \otimes \boldsymbol{h}$.

3.1.2 AWGN Noise

The AWGN noise of the channel is added after the tapped delay line. The AWGN noise vector $\boldsymbol{w}$ of length N is represented by,

$$\boldsymbol{w} = [w(1), w(2), \ldots w(n), \ldots, w(N)]^{\mathrm{T}} \quad (5)$$

The value of w is chosen to provide the specified *snr*. The resulting signal stream after the noise addition is denoted by the vector yM (see *figure 1*). $\boldsymbol{yM}$ is given by,

$$\boldsymbol{yM} = \boldsymbol{d} + \boldsymbol{w} = \boldsymbol{xM} \otimes \boldsymbol{h} + \boldsymbol{w} \quad (6)$$

The elements of $\boldsymbol{yM}$, which are complex numbers, are represented as,

$$yM = [yM(1), yM(1), \ldots yM(n), \ldots, yM(N)]^{\mathrm{T}} \quad (7)$$

Signal stream $\boldsymbol{yM}$ is the input to the M-PSK demodulator, as shown in *figure 1*. The size of the output of the demodulator is $\boldsymbol{y}$ which is a vector (discrete signal stream) of length **N** as,

$$\boldsymbol{y} = [y(1), y(2), \ldots y(n), \ldots, y(N)]^{\mathrm{T}} \quad (8)$$

3.2 Simulation of the communication system

The baseband communication system, shown in *figure 1*, is realized using the built-in functions available from the MATLAB communication toolbox as shown in *table 1*.

Table 1: Realization of the variables and parameters in *figure 1*

Variables and parameters	Brief description
M = 4;	%QPSK
x = randi([0 M-1],*N*,1);	%Random Input vector of size *N*×1
xM = pskmod(*x*,M);	%Modulated samples(complex) %of size *N*×1
L = 1; *h* = andn(*L*,1)+1i*randn(*L*,1);	% Length specification %normal random value of length 1
h = [1, 0.2**h*/norm(*h*)];	%Normalized Rician channel % coefficients of length 2
d = conv(*xM*,*h*);	%convolution
d = *d*(1: end-1);	%adjust the length(d)= N while %neglecing the last term
yM = awgn(*d*,10,'measured');	%snr =10 dB Noise is added
y = pskdemod(*yM*,M);	%Demodulated output of %conventional Receiver for ground %truth verification

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 228-237 | e-ISSN: 2347-470X

3.3 Convolution Receiver and the Intelligent Receiver

The Conventional Receiver (CR) and the intelligent receiver of CNN-IR are shown in *figures 3(a) and 3(b)*. The CR has the channel estimator and the equalizer units to compensate for the channel distortion. On the other hand, the Intelligent Receiver (IR), shown in *figure 3(b)*, is essentially a well-trained Convolutional Neural Network for the Intelligent Receiver (CNN-IR) that compensates for the Channel Estimator (CE) and the Equalizer.

The CNN-IR accepts the channel distorted and noisy input yM and generates the output y. The CNN-IR is trained such that its output y is equal to the original transmitted data x, when the training is perfect. Therefore, under ideal conditions, the output y of the IR is equal to the transmitted input x itself. *Thus, the IR* eliminates the need for the *CE and the Equalizer*. In this way, the errors due to the inaccurate design/realization of the CE and the Equalizer are eliminated by using the IR.

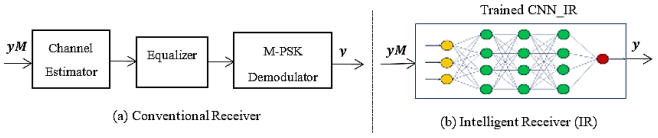

Figure 3: Conventional Receiver and Intelligent Receiver

3.4 Learning Process in CNN

CNN-IR operates on the principle of supervised deep learning. To appreciate it better, let us consider the functional relation between the input data vector x at the transmission side and the corresponding receiver input vector yM of *figure 3(b)*.

3.5 Functional dependency between x and yM

From *figure 3(b)*, it can be seen that the receiver output $\boldsymbol{y}$ depends on its input yM. Let this complex dependency be represented by a function $G(\ldots)$ as,

$$\boldsymbol{y} = G(\mathrm{yM}) \tag{9}$$

However, for an ideal receiver, the expected (target) output $\boldsymbol{y}$ should be equal to x. That is,

$$\boldsymbol{y} = \boldsymbol{x} \tag{10}$$

From *equation (9) and (10),*

$$\boldsymbol{x} = G(\boldsymbol{yM}) \tag{11}$$

Functional relation *equation (11)* is nonlinear [20], and cannot be accurately expressed in closed form. However, based on the principle of machine learning, a CNN can learn the complex relation G (…), between x and yM, when it is trained using a large set of known samples of x and yM.

Once the CNN learns the functional relation *eq. (11),* it can predict the value of x given yM. In M-PSK system, the elements of vector x are integers in the finite range [0 to (M–1)], and there are N elements in each x. Therefore, the recovery of the discrete integer vector x of size N×1 is achieved using the classification mode where the prior knowledge that the variable to be predicted is a discrete integer in a finite range improves the classification accuracy.

4. BASIC WORKING OF THE CNN-IR

4.1 Block diagram of the CNN-IR

The block diagram of the CNN used in CNN-IR is shown in *figure 4*. The CNN operates in three stages, namely, the training stage, the test stage, and the deployment stage.

4.2 Generation of Training Data

For the CNN-IR the basic input is yM, and the target output is $\boldsymbol{x}$**.** However, a relatively large dataset is required for efficient learning during the training stage. Therefore, four arrays, namely ***xTrain, xValid***, and ***yTrain, yValid*** are used in training the CNN-IR. The detailed objectives of these four arrays are given in [21-22].

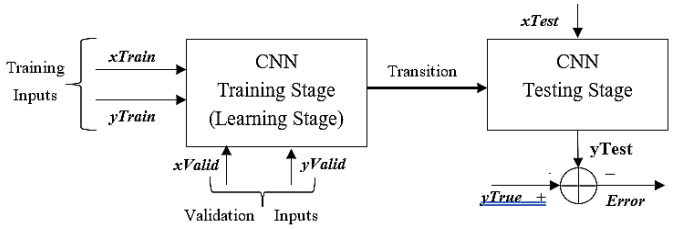

Figure 4: Basic Block Diagram of the CNN

4.2.1 Generation of *yTrain, yValid and yTrue*

Here, ***yTrain*** and ***yValid*** are obtained by splitting the vector $\boldsymbol{x}$ (as given in *table 1*) of length N into three parts of length R, V, and S as,

$$\boldsymbol{yTrain} = \boldsymbol{x}(1{:}\,R) \tag{12}$$
$$\boldsymbol{yValid} = \boldsymbol{x}(1 + R{:}\,R + V) \tag{13}$$

Here, the first R elements of $\boldsymbol{x}$ form ***yTrain,*** and the next V elements of $\boldsymbol{x}$ form

yValid. The remaining N–(R+V) elements form the ***yTrue*** array (see *figure 4*) as,

$$\boldsymbol{yTrue} = \boldsymbol{x}(1 + \ R + V{:}\,R + V + S) = \boldsymbol{x}(1 + \ R + V{:}\,N) \tag{14}$$

Where,

$$R{+}V{+}S = N \tag{15}$$

In CNN -IR training, the ratio (R:V:S) is taken at (80:10:10) as

$$(R{:}V{:}S) = (80{:}10{:}10) \tag{16}$$

Here, it is assumed that N is an integer multiple of 100. The (R:V:S) ratio could be slightly different as (70:10:20) or (60:10:30) or near these values.

4.2.2 Generation of *xTrain, xValid* and *xTest*

From *figure 3(b)*, the CNN input is yM, a complex vector. But *xTrain* and *xValid* have to be arrays of real numbers. Hence, the contribution of yM towards the formation of *xTrain, xValid, and xTest* is obtained innovatively using the complex type property

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 228-237 | e-ISSN: 2347-470X

of *yM*. To achieve this, let a multi-dimensional array *XT* be formed as,

$$\boldsymbol{XT} = [\boldsymbol{xt}[1],\ \boldsymbol{xt}[2], \dots, \boldsymbol{xt}[n], \dots, \boldsymbol{xt}[N]] \quad (17)$$

The n^{th} term $xt[n]$ is formed based on the components of the complex number *yM*(*n*), as shown in *table 2*.

Table 2: Formation of the Primary Segment of *xt* [n]

Part	Description	Built-in function used for parts 1 to 4
Part 1	Real part of *yM*(*n*)	real(*yM*(*n*))
Part 2	Imaginary part of *yM*(*n*)	imag(*yM*(*n*))
Part 3	Absolute value of *yM*(*n*)	abs(*yM*(*n*))
Part 4	Phase angel of *yM*(*n*)	atan2(imag(*yM*(*n*)), real(*yM*(*n*)))

Here, the real part, the imaginary part, the absolute value (magnitude), and the argument (phase angle) of *yM* are used in the formation of the Primary Feature Vector (*PFV*). The *PFV* is formed by stacking the four parts vertically, one below the other as,

$$PFV[n] = \begin{bmatrix} \text{real}(yM(n)) \\ \text{imag}(yM(n)) \\ \text{abs}(yM(n)) \\ \text{atan2}\left(\text{imag}(yM(n)), \text{real}(yM(n))\right) \end{bmatrix} \quad (18)$$

In *eq. (18)*, the size of the column vector $PFV[n]$ is (4 ×1). The CNN-IR requires a large-sized *xt* [n] for better performance. Hence, the $PFV[n]$ is extended by vertically cascading the reversed $PFV[n]$ with the original $PFV[n]$ as,

$$EPFV[n][1] = \begin{bmatrix} PFV[n] \\ flipud(PFV[n]) \end{bmatrix} \quad (19)$$

In [19], $EPFV[n][1]$ represents the level 1 extended $PFV[n]$. The flip up-down function $flipud(PFV[n])$ gives the the vertically reversed $PFV(n)$. Now, the length of *EPFV*[*n*][1] is twice that of $PFV[n]$. Thus, the size of $EPFV[n][1]$ is ((4×2) ×1). To increase the size further, $EPFV[n][1]$ is extended to level 2 as,

$$EPFV[n][2] = \begin{bmatrix} EPFV[n][1] \\ flipud(EPFV[n][1]) \end{bmatrix}$$

Now, the length is doubled. The recursive extension is carried out further to get the u^{th} level extension as,

$$EPFV[n][\text{u}] = \begin{bmatrix} EPFV[n][u-1] \\ flipud(EPFV[n][u-1]) \end{bmatrix} \quad (20)$$

The size of $EPFV[n][\text{u}]$ will be $(4 \times 2^u) \times 1$. The u^{th} level extended $EPFV[n][\text{u}]$ represents the hidden feature vector of the received data *yM*(*n*) corresponding to the transmitted data *x*(*n*). Now, $EPFV[n][\text{u}]$ forms the training vector *xtrain*(*n*) as

$$xt[n] = EPFV[n][u] \quad (21)$$

Here, u is the extension parameter corresponding to the n^{th} symbol transmission. (Experimentally, it is found that u = 8 gives good training performance). For good diversity, the elements of the column vector *xt* [*n*] are thoroughly shuffled. For successful training as well as validation, *xt* [*n*] is generated for *n* = 1 to *N* to get the multi-dimensional array *XT* as,

$$XT = [xt[1],\ xt[2], \dots, xt[n], \dots, xt[N]] \quad (22)$$

The size of *XT* is $(4 \times 2^u) \times N)$. The matrix *XT* is split into three parts *XTR*, *XTV*, *and XTS*, where *XTR* is used for training, *XTV* for validation, and XTS for testing. The first *R* columns of *XT* form *XTR* as,

$$XTR = [xt[1],\ xt[2], \dots, xt[\text{R}]] \quad (23)$$

The size of *XTR* is $(4 \times 2^u) \times R)$.

The next *V* columns form *XTV* as,

$$XTV = [xt[\text{R}+1],\ xt[\text{R}+2], \dots, xt[R+V]] \quad (24)$$

The size of *XTV* is $(4 \times 2^u) \times V)$. The remaining columns of *XT* form the test set *XTS* as,

$$XTS = [xt[1+\text{R}+\text{V}], xt[\text{R}+\text{V}+2], \dots, xt[R+V+S]] \quad (25)$$

The size of *XTS* is $(4 \times 2^u) \times S)$.

Here also, (*R*+*V*+*S*) = *N* as in (14). For correct training of CNN-IR, the split ratio (*R*:*V*:*S*) should be the same as used in (15), which is used for *yTrain*, *yValid*, *and yTest*

4.3 Reshaping of XTR, XTV and XTS

The actual training input ***xTrain, xValid,*** and ***xTest*** are obtained by reshaping ***XTR, XTV,*** and ***XTS*** to get the corresponding 4D matrices (as required by the input layer of the CNN) as,

$$xTrain = \text{reshape}(XTR, N_1,\ N_2,\ N_3,\ R) \quad (26)$$
$$xValid = \text{reshape}(XTV,\ N_1,\ N_2,\ N_3,\ V) \quad (27)$$
$$xTest = \text{reshape}(XTS,\ N1,\ N2,\ N3,\ S) \quad (28)$$

The dimensional parameters N_1, N_2 and N_3 are chosen for minimum training error, subjected to the constraint,

$$N_1 * N_2 * N_3 = 4 * 2^u \quad (29)$$

4.4 Test stage of CNN

The testing scheme is shown in *figure 4*. The purpose of testing a trained CNN_IR is to verify whether it is performing its task correctly or not by comparing its output with the expected (target) values. A known sequence of test input, denoted by ***xTest***, as given by (28), is applied to the input of the CNN, and the resulting predicted output, ***ytest,*** is compared with the known ground truth values represented by ***yTrue*** as given by *eq. (14)*.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 228-237 | e-ISSN: 2347-470X

4.4.1 Test Error and the Percentage Test Error

The test error is represented by *Error* vector, and it is given by,

$$Error = yTrue - yTest \quad (30)$$

The size of *YTrue* is $S \times 1$. In *eq. (30)*, *YTest* is the output of the CNN classifier. The sizes of *YTest* and *Error* vectors are also $S \times 1$. Ideally, for zero error, all the elements of *Error* have to be zeros. Otherwise, the degree of error is given by the number of non-zero elements of *Error*. Hence, the Test Error, *TE* is given by,

$$TE = \text{Number of non-zero elements of } Error \quad (31)$$

Consequently, the Percentage Test Error (*PTE*) is given by,

$$PTE = \frac{TE*100}{Total\ number\ of\ elements\ in\ Error} = \frac{TE*100}{S} \quad (32)$$

4.5 Deployment of CNN-IR

After satisfactory training and testing, the CNN-IR, in the classification mode, is ready for the in-situ deployment at the receiver side, where it accepts the incoming data stream and classifies it into its correct class. Thus, the trained CNN-IR detects the present transmitted data stream by accepting the received distorted/noisy signal. Thus, the CNN-IR that gets trained offline for a reasonably long time provides a fast online response.

4.6 CNN-IR Network

The custom CNN_IR network is formed by the interconnection of neurons arranged in multiple cascaded layers, as shown in *figure 3(b)*. The CNN-IR that is specifically constructed for the Intelligent receiver is inspired by the CNN models designed for image classification [22]. The first layer, imageInputLayer([N_1, N_2, N_3]), accepts the training and test inputs and forwards them to the succeeding layer in the proper format. The last layer, classification Layer generates the classified output.

```
layers = [imageInputLayer([N1, N2, N3])   %Input is generated in the form of a 3D matrix

convolution2dLayer(3,8,'Padding','same') %Kernal size = 3×3. Eight such units

  batchNormalizationLayer %Refer [25]
  reluLayer               %Rectifier linear unit
  maxPooling2dLayer(2,'Stride',2) %Refer [26]

  convolution2dLayer(3,16,'Padding','same')
  batchNormalizationLayer
  reluLayer
  maxPooling2dLayer(2,'Stride',2)

  convolution2dLayer(3,32,'Padding','same')
  batchNormalizationLayer
  reluLayer
  maxPooling2dLayer(2,'Stride',2)

  convolution2dLayer(3,64,'Padding','same')
  batchNormalizationLayer
  reluLayer

    convolution2dLayer(12,16,'Padding','same')
  batchNormalizationLayer
  reluLayer

  dropoutLayer(0.2)   % 20% deleted
  fullyConnectedLayer(fcL) %Dense layer
  softmaxLayer          % Gets the highest probability score
  classificationLayer];     %which is the classified output
```

The architectural arrangement of different layers of CNN-IR is given in *figure 5*.

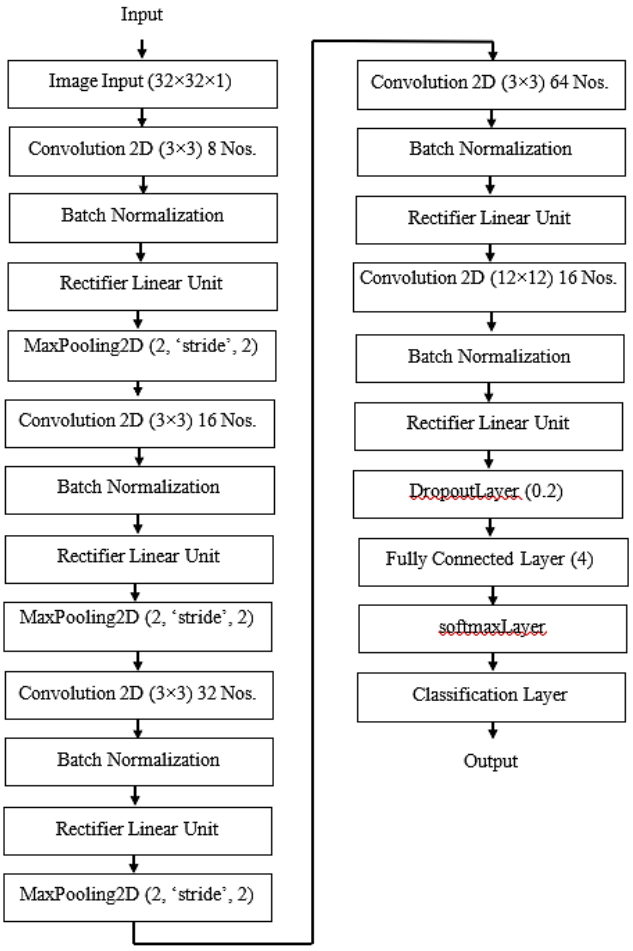

Figure 5: Architecural arrangement of different layers of CNN-IR

4.7 CNN-IR Training

The training is carried out based on the specified *training options* where the number of epochs, the rate of learning, and other hyperparameters are listed. In CNN-IR, the initial transmits, receive, and other signals required for training, testing, and deployment stages are generated using the MATLAB Communication toolbox and the Deep Learning toolbox. The training options used are as follows.

Training Options

```
options = trainingOptions('sgdm', ...
  'MaxEpochs', 200, ...
  'InitialLearnRate', 2.0*1e-3, ...
  'LearnRateSchedule','piecewise', ...
```

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 228-237 | e-ISSN: 2347-470X

```
'LearnRateDropFactor', 0.075, ...
'LearnRateDropPeriod', 300, ...
'Shuffle','every-epoch', ...
'Plots','training-progress', ...
'ValidationData',{xValid,yValid}', ...
'ValidationFrequency', 40, ...
'MiniBatchSize', 30,...
'Verbose', false);
```

Here, 'sgdm' is the abbreviation of 'stochastic gradient descent with momentum.' For details, refer [21-22].

4.7.1 CNN-IR Training Algorithm

The CNN-IR Training Algorithm denoted by 'Train-CNN-IR' is given below. It is assumed that the training/test data set is readily available for implementing the Algorithm.

#Algorithm Train-CNN-IR

Training/test inputs: $xTrain,\ yTrain,\ xValid, yValid,\ XTest,$ and $YTrue$
Output: Satisfactorily Trained CNN-IR.

1. Construct the CNN-IR.
2. Choose the initial hyperparameters (say as in *table 3*).
3. Train CNN-IR using the *trainNetwork*(…)function as,
   Net = *trainNetwork*(*xTrain*,*yTrain*,layers,options);
4. Get the output of the trained network as,
   yTest = classify(net,*XTest*);
5. Evaluate the, *TE* and *PTE* using *eq. (31)* and *(32)*
   If $TE == 0$ // indicates successful training
   Goto *step 6*
   else
   Adjust the layers parameters and the training parameters like [N_1, N_2, N_3],
   number of epochs, training rate etc., (fine tuning), and then
   goto *step 3* (retraining)
   end
6. Over // Now the trained CNN_IR is ready for online deployment

5. PERFORMANCE AND EXPERIMENTAL RESULTS

5.1 Initial parameters Chosen for the preliminary Experiment

The parameters chosen to demonstrate the basic working of the CNN-IR experiment are listed in *table 3*.

Table 3: Values of the Parameters and Variables

Parameters and variables	Symbol	Values
Order of M-PSK	*M*	4
Length of vectors **x**, **w**, **y**	*N*	10000
Signal to Noise Ratio	SNR	10 dB
Number of taps for ***h***	*L*	2
Path gain for h coefficients (normalized)	[pg(1), pg(2)]	[1, 0.2]
Length of training dataset	*R*	8000
Length of validation datasets	*V*	1000
Length of test dataset	*S*	1000
Training datasets (Percentage)	100**R/N*	80%
Validation datasets (Percentage)	100**V/N*	10%
Testing datasets (Percentage)	100**S/N*	10%
Train/Test extension parameter	*u*	8
Data dimensions for Input layer	N_1, N_2, N_3	32, 32, 1
No. of epochs	*NE*	200
Optimizer	-----	sgdm

Other unassigned values that are not covered in *table 3* are selected according to [23].

Training Progress

Here, the CNN-IR is constructed as given in *section 4.6*. The training and test data sets are generated by simulation, as specified in *tables 2 and 3*. The CNN-IR is trained according to the corresponding parameters from *table 3*. The Rician *h* vector is obtained using the Gaussian distribution with normalized path gain = [1, 0.2]. The training is carried out by the function net = *trainNetwork*(…). During training, the training accuracy and the training loss [24] are progressively plotted and as shown in *figure 6*. Here, the *PTE* is found to be **zero**.

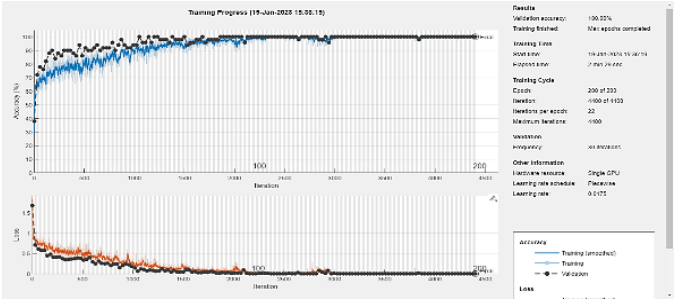

Figure 6: Training progress plots having 200 epochs

5.2 Performance of CNN-IR

The performance of the CNN-IR is mainly measured by the Percentage Test Error (*PTE*) incurred during testing. The effect of variations of different parameters on *PTE* is experimentally demonstrated in this section.

PTE vs the channel noise

In this experiment, *PTE* vs the channel noise, expressed in *snr*, is explored. Here the *snr* is decremented from 12 dB to 5 dB, and for each *snr*, the number of epochs (NE) is varied from 100 to 200 in steps of 25. All other parameters are the same as in Experiment 1. In a given trial, the resulting variations in *PTE* are shown in *table 4*.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 228-237 | e-ISSN: 2347-470X

Table 4: *PTE* versus *snr* for different NE's

snr ↓	Number of Epochs (*NE*)				
	100	125	150	175	200
12 dB	0.10	0.10	0.00	0.00	0.00
11 dB	0.20	0.15	0.10	0.00	0.00
10 dB	0.30	0.20	0.15	0.10	0.00
9 dB	1.20	1.00	0.85	0.65	0.50
8 dB	2.50	2.00	1.50	1.20	1.00
7 dB	4.40	3.90	3.40	2.50	2.20
6 dB	7.90	7.00	6.60	5.40	4.30
5 dB	12.60	11.00	10.20	9.40	7.30

From *table 4*, it can be seen that the *PTE* increases monotonically as the *snr* decreases while the reduction in *PTE* gets saturated for higher values of *NE*.

From *table 4*, *PTE* versus *snr* is plotted for *NE* = 100, 150, and 200, as shown in *figure 7*.

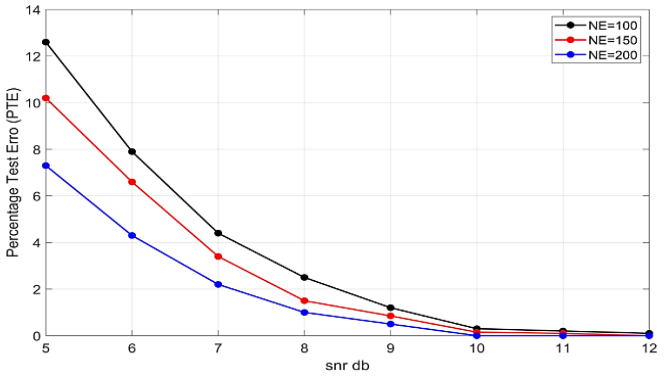

Figure 7: *PTE* versus Number of Epochs (NE)

5.3 Confusion Matrix

The Confusion Matrix (CM) identifies the true and false classification counts over the full set of classes. The following Experiment demonstrates the classification details of the 4-class (M = 4) problem solved by CNN-IR.

Here, the classification is carried out separately, with the length of S set at 40 for easy verification. In this case, SNR is set to 5 db, and the value of NE at 150. Other parameters are the same as in *table 3*. Training is carried out, and the classification result is shown in the CM of *figure 8*. Here, the classes [1, 2, 3, 4] correspond to the data symbols [0, 1, 2, 3] respectively. In *figure 8*, the off-diagonal elements give the number of errors (total 10) and the diagonal elements give the correct number (total 30) of classifications of each class. The filled element CM(I, J) gives the number of members of true class I predicted as class J. Thus, in *figure 8*, CM(4, 3) = 3 means that 3 members of class 4 are wrongly predicted as the members of class 3. On the other hand, CM (3, 3) = 6 means, 6 members of class 3 are correctly classified into class 3 itself.

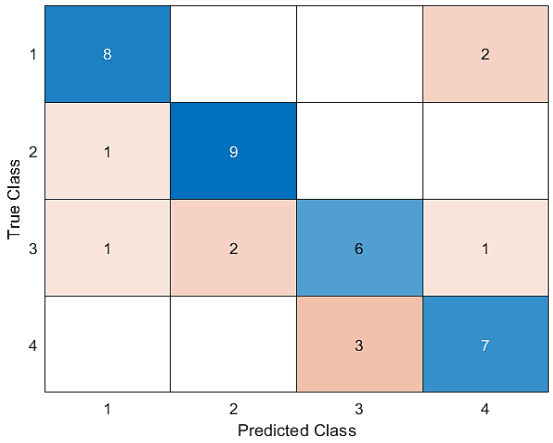

Figure 8: Confusion matrix with S = 40

5.4 *PTE* versus the path gain

Here, the ***snr*** in dB is varied from 10 to 12, and for each ***snr***, the normalized path gain pg(2) of the Rician coefficient h(2) is varied from 0.2 to 0.8 in steps of 0.1. Other parameters are the same as in *table 2*. The resulting variations in *PTE* is shown in *table 5*.

Table 5. *PTE* versus the path gain *pg*(2)

Snr ↓ pg(2) →	**0.10**	**0.15**	**0.20**	**0.25**	**0.30**	**0.35**	**0.40**	**0.45**
CNN-IR, ***snr*** = 10 dB	0.00	0.00	0.00	0.24	1.32	2.88	4.32	6.84
CNN-IR, ***snr*** = 11 dB	0.00	0.00	0.00	0.20	1.10	2.40	3.60	5.70
CNN-IR, *snr* = 12 dB	0.00	0.00	0.00	0.16	0.88	1.92	2.88	4.56
Traditional Receiver *snr* = 10 dB	0.00	0.00	0.00	0.25	1.45	3.26	5.22	8.95

The corresponding plots are shown in *figure 9*.

From *figure 9*, it can be seen that PTE increases nonlinearly with the Rician path gain coefficient *pg*(2), as it contributes a product term as indicated in *figure* 2. On the other hand, *PTE* decreases as the channel *snr* increases, which implies a decrease in the noise level compared to the signal strength.

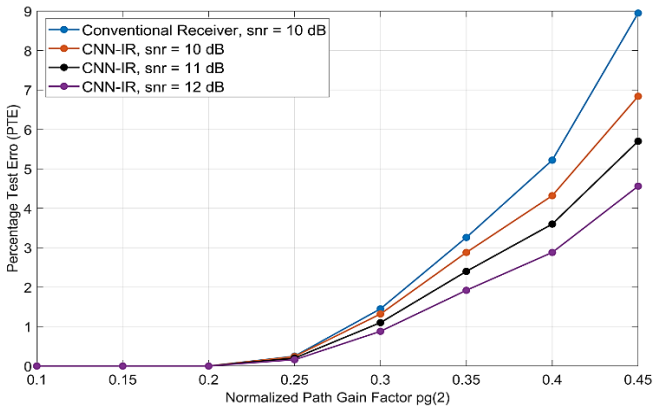

Figure 9: *PTE* versus Normalized Path Gain Factor pg(2)

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 228-237 | e-ISSN: 2347-470X

5.4.1 Comparison of CNN-IR with the Conventional Receiver

The *PTE* values obtained using a Conventional Receiver while the ***snr*** is kept at 10 dB is also included in *table 5*. Here, at *pg*(2) = 0.3, *PTE* by the Conventional Receiver is 1.45 while that due to CNN-IR is 1.32. Therefore, the percentage reduction in the error is:

$$\text{percentage reduction in the error } = \frac{(1.45-1.32)*100}{1.45} = 8.97\% \approx 9\% \quad (33)$$

5.5 Performance with Respect to the Degree of Modulation M

In this case, the degree of modulation (M) varies as M = 2, 4, and 8. The plots of *PTE* vs *snr* are plotted for each M for CNN-IR and the corresponding Conventional receiver. The result is displayed in *figure 10*, which shows that the *PTE* decreases as the channel *snr* increases. A higher *snr* means the signal strength is higher compared to the noise level. On the contrary, as M (degree of modulation) increases *PTE* increases. In this case, for a given M, the receiver has to choose 1 out of M symbols at a time. Hence, the probability of a correct decision depends on (1/M) {inverse relation}. Thus, as M increases, the likelihood of a correct decision decreases.

Here, the size of the test set is selected as S = 10,000 to get a good resolution for PTE. From the plots of *figure 10*, it can be observed that the PTE increases with M while it decreases with an increase in *snr* values.

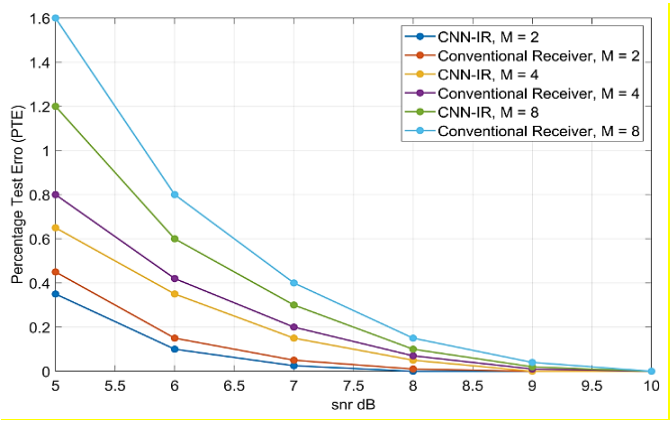

Figure 10: *PTE* versus snr for M= 2, 4, and 8

6. CONCLUSION

A novel intelligent receiver, assisted by a deep learning network, is presented that predicts the transmitted discrete data with almost zero error. The intelligent receiver, on average, reduces the symbol error rate by 20 to 30 percent compared to the stand-alone conventional receiver. The deep learning network is trained offline and then deployed online to assist the conventional receiver. The proposed scheme is well suited for systems using error-correcting block channel codes like Read Solomon, BCH, and so on. The accuracy of detection can be improved further using transfer or reinforced learning.

REFERENCES

[1] Oyerinde, Olutayo & Mneney, S.H. (2012). Review of Channel Estimation for Wireless Communication Systems. IETE Technical Review. 29. 2012, pp. 282-298. doi: 10.4103/0256-4602.101308.

[2] Ahmed, Sheeraz & Khan, Dr.Yousaf & Wahab, Asad. (2019). A Review on Training and Blind Equalization Algorithms for Wireless Communications. Wireless Personal Communications (online), 108, 2019, pp 1-25. doi:10.1007/s11277-019-06495-8.

[3] Deepanjali Jain, Dr. Poonam Beniwal, 2022, Review Paper on Noise Cancellation using Adaptive Filters, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 11, Issue 01, 2022, pp. 241-244

[4] Ladvánszky, J. (2020) Noise Reduction for Digital Communications—The Masterpiece, a Modified Costas Loop. Circuits and Systems, 11, 2021, pp. 57-64. doi: 10.4236/cs.2020.116006.

[5] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," J Big Data 8, 53 2021. pp. 1-74. https://doi.org/10.1186/s40537-021-00444-8

[6] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," IEEE Trans. Cogn. Commun. Netw., vol. 4, no. 4, pp. 648–664, Dec. 2018.

[7] L. Dai, R. Jiao, F. Adachi, H. V. Poor and L. Hanzo, "Deep Learning for Wireless Communications: An Emerging Interdisciplinary Paradigm," in IEEE Wireless Communications, vol. 27, no. 4, pp. 133-139, August 2020, doi: 10.1109/MWC.001.1900491.

[8] Z. Qin, H. Ye, G. Y. Li, and B. F. Juang, "Deep learning in physical layer communications," IEEE Wireless Commun., vol. 26, no. 2, pp. 93–99, Apr. 2019.

[9] Jiao, J.; Sun, X.; Fang, L. An overview of wireless communication technology using deep learning. China Commun. 2021, 18, pp. 1–36.

[10] Xu, Wenjie, "Artificial Intelligence Aided Receiver Design for Wireless Communication Systems" (2021). Theses, Dissertations and Capstones. 1376.

[11] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," in IEEE Transactions on Cognitive Communications and Networking, vol. 3, no. 4, pp. 563-575, Dec. 2017, doi: 10.1109/TCCN.2017.2758370.

[12] S. Zheng, S. Chen and X. Yang, "DeepReceiver: A Deep Learning-Based Intelligent Receiver for Wireless Communications in the Physical Layer," in IEEE Transactions on Cognitive Communications and Networking, vol. 7, no. 1, pp. 5-20, March 2021, doi: 10.1109/TCCN.2020.3018736.

[13] B. Wang, K. Xu, S. Zheng, H. Zhou and Y. Liu, "A Deep Learning-Based Intelligent Receiver for Improving the Reliability of the MIMO Wireless Communication System," in IEEE Transactions on Reliability, vol. 71, no. 2, pp. 1104-1115, June 2022, doi: 10.1109/TR.2022.3148114.

[14] A. Irawan, G. Witjaksono, and W. K. Wibowo, "Deep learning for polar codes over flat fading channels," in Proc. Int. Conf. Artif. Intell. Inf. Commun., Okinawa, Japan, 2019, pp. 488–491.

[15] E. Bjornson and P. Giselsson, "Two Applications of Deep Learning in the Physical Layer of Communication Systems [Lecture Notes]," in IEEE Signal Processing Magazine, vol. 37, no. 5, pp. 134-140, Sept. 2020, doi: 10.1109/MSP.2020.2996545.

[16] Nariman Farsad, "Detection Algorithms for Communication Systems Using Deep Learning," arXiv: 1705.08044v2 [cs.LG]. 2017, pp. 1-10.

[17] A. K. Kowshik, A. H. Raghavendra, S. Gurugopinath and S. Muhaidat, "Deep Learning-Based Signal Detection for Rate-Splitting Multiple Access Under Generalized Gaussian Noise," in IEEE Open Journal of Vehicular Technology, doi: 10.1109/OJVT.2023.3238034.

[18] T. Gian, V. -D. Ngo, T. -H. Nguyen, T. T. Nguyen and T. van Luong, "Deep Neural Network-Based Detector for Single-Carrier Index Modulation NOMA," 2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Chiang Mai, Thailand, 2022, pp. 1805-1809, doi: 10.23919/APSIPAASC55919.2022.9980150.

International Journal of
Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 228-237 | e-ISSN: 2347-470X

[19] https://in.mathworks.com/help/comm/ref/mpskmodulatorbaseband.html

[20] Yong Soo Cho, Jaekwon Kim, Won Young Yang, Chung G. Kang (2010), MIMO-OFDM Wireless Communications with MATLAB, John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop, # 02-01, Singapore 129809.

[21] https://machinelearningmastery.com/how-machine-learning-algorithms-work/

[22] "Create Simple Deep Learning Network for Classification," https://in.mathworks. com/help /deep-learning/ug/ preprocess-images-for- deep-learning .html

[23] "Set Up Parameters and Train Convolutional Neural Network," Matlab. https://in.mathworks.com/help/deeplearning/ug/setting-up-parameters-and-training-of-a-convnet.html.

[24] "Train neural network for deep learning" https:// in.mathworks. com/ help/deeplearning/ref/trainnetwork.html

[25] https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/, by Jason Brownlee (accessed on 31-july-2023).

[26] https://in.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.maxpooling2dlayer.html