



Research article

Modulation classification analysis of CNN model for wireless communication systems

Tamizhelakkiya K^{1,3,*}, Sabitha Gauni^{1,2} and Prabhu Chandhar³

¹ Department of Electronics and Communication Engineering, SRM Institute of Science and Technology, SRM Nagar, Kattankulathur, Chennai, Tamil Nadu, India

² Autosys Control Systems India Pvt Ltd, Chennai, Tamil Nadu, India

³ Chandhar Research Labs Pvt Ltd, Chennai, Tamil Nadu, India

* **Correspondence:** Email: tk1045@srmist.edu.in.

Abstract: Modulation classification (MC) is a critical task in wireless communication systems, enabling the identification of the modulation class in the received signals. In this paper, we analyzed a novel multi-layer convolutional neural network (CNN) to extract hierarchical features directly from the raw baseband samples. Moreover, we compared the training and testing accuracy of the CNN model for various decimation rates, input sample size and the number of convolutional layers. The results showed that the three-layer CNN model provided better classification accuracy with less computation cost. Furthermore, we observed that the MC performance of the proposed CNN model was better than the other deep learning (DL) and cumulant-based models.

Keywords: modulation classification (MC); CNN; deep learning (DL); feature map; neural network

1. Introduction

Modulation classification (MC) has been found to be a fundamental task in wireless communication systems that involves identifying the modulation class used to encode information in the received signal. It plays a crucial part in various applications, including signal analysis, spectrum monitoring [1] and cognitive radio (CR) [2]. The primary goal of MC is to blindly recognize the modulation types of the received signal based on characteristics, such as amplitude, phase and frequency variations. This enables the receivers to properly demodulate the signal and extract the transmitted information. To perform MC, several techniques and algorithms have been developed [3]. These methods leverage different features extracted from the received signal and employ various classification algorithms, such as machine learning and statistical analysis. The key steps involved in MC are as follows [4]:

- **Signal Preprocessing:** The received signal is typically processed to remove noise, interference, and other unwanted effects that may affect the classification accuracy. This processing technique involves filtering, signal normalization and sampling rate adjustment.
- **Feature Extraction:** Relevant features are extracted from the preprocessed signal to capture its unique characteristics. Commonly used features include statistical moments, power spectral density, cyclic auto-correlation and higher-order cumulants. These features aim to capture the temporal and spectral properties of the received signal.
- **MC:** Once the feature set is obtained, the model generation process is carried out by either the likelihood-based or the feature-based method. The different machine learning (ML) / deep learning (DL) algorithms have been adopted to perform the classification task. These algorithms are trained using labeled data to learn the patterns and characteristics associated with each modulation class.
- **Performance Evaluation:** The accuracy of the MC system is assessed using performance metrics such as classification accuracy, confusion matrix and detection probability. These metrics provide insights into the system's ability to correctly classify different modulation classes.

In [5], the authors suggested that the MC can be performed by using ML methods such as the random forest (RF), support vector machine (SVM) and neural network (NN). Further, in DL, salient features have been extracted from the received signals [6], such as power spectral density, instantaneous voltage, phase and frequency for MC. The obtained features are suffering from channel impairments and fading environments that may lead to a major challenge for recognizing modulation type under different channel conditions [7].

The improvement in classification accuracy has been achieved at the expense of increased computational complexity by the traditional convolutional neural network (CNN) [8] models such as Alexnet, Inception [9] and GoogleNet [10]. Similarly, the channel state information (CSI) parameters, such as rank indicator (RI), delay spread (DS), and signal to interference noise ratio (SINR), have been classified with higher accuracy than the third generation partnership project (3GPP) recommended value by using a CNN model [11]. The authors in [9] suggest that the combination of the Inception and ResNet network model has a faster convergence rate among all traditional CNN models. Similarly, the hybrid residual and long short term memory (LSTM)-based model has been developed to improve the classification accuracy at low signal to noise ratio (SNR) scenarios [12]. The automatic modulation recognition (AMR)-based NN [13] and CNN [14] have been implemented on field programmable gate arrays (FPGA) for dynamic spectrum access (DSA) and CR systems with a latency of $8\mu\text{s}$. Moreover, the LSTM-based channel estimator has been demonstrated to detect received orthogonal frequency division multiplexing (OFDM) signals [15]. In [16], the authors proposed an adversarial transfer learning architecture (ATLA), which performs the domain-level asymmetric mapping with an accuracy of 17.3% higher than parameter-based transfer methods. Moreover, ATLA is applicable for knowledge transfer between the parameters for the reduced training data and assists the CNN model to improve model accuracy. Although ATLA provides optimal performance, it is also sensitive to parameter variations and becomes impractical for real-time prediction scenarios.

To recognize the radio signals, various datasets are available as listed in Table 1. It has been noticed that the graphics processing unit (GPU) has been used to create the model, which reduces the training overhead more than the central processing unit (CPU). Moreover, Hisarmod2019.1 and RadioML2018.10.a datasets have been adopted with channel impairments such as multipath fading

and frequency offset. The aforementioned simulation-based dataset may not be sufficient for real-world problems. Moreover, improvements in MC models for smaller variations with respect to decimation rate, number of signals and number of layers are still lacking in the literature.

Table 1. Comparison of traditional datasets adopted for MC based on various salient characteristics.

Dataset	Input Samples Format	No.of. Classes	Training Platform	Testing Platform	Channel Impairments	Real time Execution	Real time Prediction
RadioML 2016.10.a [17]	I/Q	10	GPU	GPU	×	×	×
RadioML 2016.10.b [18]	I/Q	10	GPU	GPU	×	×	×
Hisarmod 2019.1 [19]	I/Q	26	GPU	GPU	✓	×	×
RadioML 2018.10.a [20]	Polar	24	GPU	CPU	✓	×	×
RFSC [21]	I/Q	10	CPU	GPU	✓	✓	✓

In resource-constrained applications, the above-mentioned parameters play an important role in decoding physical resource block (PRB), primary synchronization signal (PSS) and secondary synchronization signal (SSS) in the physical channels. In this work, we have utilized radio frequency signal classification (RFSC) dataset [21] to analyze the multi-layer CNN model performance. We leverage the benefits of CNN models to tackle the issues of MC, which severely hinders the practical resource-constrained scenarios. The main contributions of this paper are summarized as follows:

- We propose a multilayer CNN model for MC using the RFSC dataset.
- We investigate the performance analysis of the multilayer CNN model for different decimation rates (D), input signals (K), number of input frames (F) and number of CNN layers (l).
- We evaluate the proposed model performance in terms of confusion matrix, training/testing loss and accuracy.
- We demonstrate the recognition accuracy comparison among different DL models and cumulant-based models for RFSC dataset.
- We also analyze the performance of generated models using RadioML, Hisarmod and RFSC datasets.

The paper is organized as follows: In section two, the proposed CNN architecture with different hyperparameters has been presented in detail. The simulation results are discussed in section three. The conclusions are finally drawn in section four.

2. Proposed model

The proposed CNN architecture incorporates feature extraction capabilities to learn the discriminating representations directly from the raw data. The CNN model receives input in a time-domain representation, which is translated into a 2D matrix containing the amplitude and phase information. The CNN architecture is made up of numerous convolutional layers with increased

complexities that allow for the extraction of low-level and high-level features from the input signal. After each convolutional layer, a pooling layer is applied to perform the down-sampling on the feature map to retain the most important information.

2.1. Convolutional layer

The received signal $r(t)$ is modeled as,

$$r(t) = s(t) * h(t, \tau). \quad (2.1)$$

Here, $s(t)$ denotes the transmitted signal and $h(t, \tau)$ is a channel output described as a complex channel finite impulse response (FIR) filter. The received signal $r(t)$ is down-converted and sampled to derive the baseband digital samples ($r_b[n] = r[nT_s] = r_{b,I}[n] + jr_{b,Q}[n]$). The k -th dataset vector of a $r(t)$ sample is denoted as:

$$\mathbf{r}_k = [r_b[0], r_b[1], \dots, r_b[N-1]]^T. \quad (2.2)$$

The k -th feature vector of the m -th class \mathbf{x}_{mk} that is derived from a dataset of N samples and decimated by a factor D is given by,

$$\mathbf{x}_{mk} = [r'_{b,I}[0], r'_{b,Q}[0], r'_{b,I}[1], r'_{b,Q}[1], \dots, r'_{b,I}[N'-1], r'_{b,Q}[N'-1]], N' = \frac{N}{D}. \quad (2.3)$$

Here, $m = 1, 2, \dots, M$.

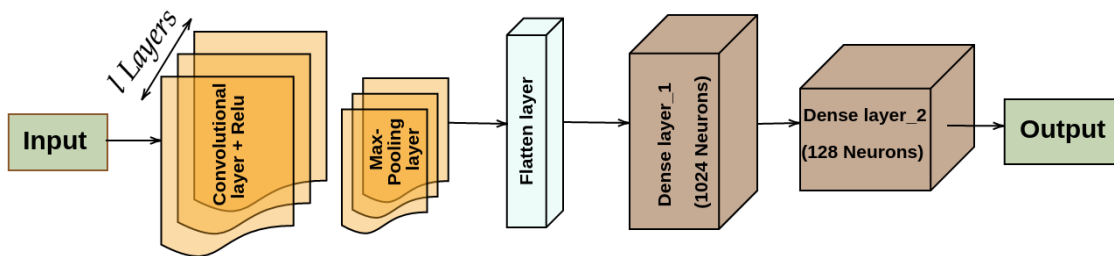


Figure 1. A proposed multilayer CNN model composed of l convolutional layers and two dense layers.

The k -th feature vector is further translated into a matrix (\mathbb{R}) to train the proposed CNN model, $\mathbf{x}_{mk} \in \mathbb{R}^{f_H \times f_W \times C_{in}}$ with size $(f_H \times f_W \times C_{in})$, where f_H is the height, f_W is the width of an image and C_{in} is the number of channels, respectively. The matrix \mathbb{R} is fed as an input to the CNN model to perform feature extraction through the activation function. The features are extracted by the consecutive convolutional layers that contain the multiple kernels to perform the convolution operation. Kernels are acting as feature detectors, generally FIR filters, which perform the convolution operation on the input image samples and produce the transformed version as an output. The proposed CNN architecture with l convolutional layers and two dense layers has been shown in Figure 1.

The mathematical representation of the convolutional layer is as follows: For l -th convolutional layer with n -th filter, the convolution operation using filters has been performed as:

$$\mathbf{z}_{x,y,n}^{[l]} = \left(\sum_{m=0}^{f_H-1} \sum_{n=0}^{f_W-1} \sum_{c=0}^{C_{in}-1} \mathbf{K}_{m,n,c,n}^{[l]} \cdot \mathbf{a}_{x+m,y+n,c}^{[l-1]} + \mathbf{b}_n^{[l]} \right), \quad (2.4)$$

where $\mathbf{z}_{x,y,n}^{[l]} = \text{conv}(\mathbf{a}^{[l-1]}, \mathbf{K}^{(n)})_{x,y,n}$ is the output value at position (x, y) in the n -th feature map (output channel) of the l -th convolutional layer. f_H and f_W are the height and width of the filter \mathbf{K} , respectively. C_{in} is the number of input channels in the previous layer ($\mathbf{a}^{[l-1]}$). Where $\mathbf{a}^{[l-1]} = \mathbf{x}_{mk}, \forall m, k$ is an input to the l -th layer. $\mathbf{K}_{m,n,c,n}^{[l]}$ represents the weight (filter) value at position (m, n) in the c -th input channel and n -th output channel of the l -th convolutional layer. $a_{x+m,y+n,c}^{[l-1]}$ is the input value at position $(x + m, y + n)$ in the c -th feature map of the $(l - 1)$ -th layer. $\mathbf{b}_n^{[l]}$ is the bias term for the n -th output channel.

$$f_H^{[l]} = \begin{cases} \left\lfloor \frac{f_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor, & s > 0; \\ f_H^{[l-1]} + 2p^{[l]} - f^{[l]}, & s = 0. \end{cases} \quad (2.5)$$

$$f_W^{[l]} = \begin{cases} \left\lfloor \frac{f_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor, & s > 0; \\ f_W^{[l-1]} + 2p^{[l]} - f^{[l]}, & s = 0. \end{cases} \quad (2.6)$$

Here, s is a stride parameter used to define the step size of the convolution operation; p represents the amount of zero padding; $\mathbf{K}^{(n)}$ is a kernel represented by $(f^{[l]}, f^{[l]}, C_{in}^{[l-1]})$; $f^{[l]}$ denotes the size of the filter in the l -th layer. The activation function applied at the l -th layer can be defined as:

$$\mathbf{a}_{x,y,n}^{[l]} = \psi^{[l]}(\mathbf{z}_{x,y,n}^{[l]}), \quad (2.7)$$

where $a_{x,y,n}^{[l]}$ is the output value at position (x, y) in the n -th feature map of the l -th activation layer. The activation function $\psi^{[l]}$ is applied element-wise to the output of the convolutional layer and it is defined as:

$$\psi^{[l]}(x) = \max(0, x). \quad (2.8)$$

The output of the l -th convolutional layer can be obtained as

$$\mathbf{a}^{[l]} = \left[\psi^{[l]}(\text{conv}(\mathbf{a}^{[l-1]}, \mathbf{K}^{(1)})), \psi^{[l]}(\text{conv}(\mathbf{a}^{[l-1]}, \mathbf{K}^{(2)})), \dots, \psi^{[l]}(\text{conv}(\mathbf{a}^{[l-1]}, \mathbf{K}^{(C_{in}^{[l]})})) \right], \quad (2.9)$$

with dimension $[\mathbf{a}^{[l]}] = [f_H^{[l]}, f_W^{[l]}, C_{in}^{[l]}]$.

The number of learning parameters at l -th layer is given by

$$\mathbf{f}^{[l]} \times \mathbf{f}^{[l]} \times \mathbf{C}_{in}^{[l-1]} \times \mathbf{C}_{in}^{[l]} + \mathbf{b}_n^{[l]}. \quad (2.10)$$

2.2. Pooling layer

At each convolution layer, CNN uses the pooling layers for feature map dimensionality reduction to avoid overfitting and improves the model accuracy. The down-sampling has been performed using a 2×2 kernel with a stride of two. Here, it uses a max-pooling operation, which takes the maximum value from the set of four values after each stride. The output of the l -th pooling layer is given by,

$$p_{x,y,z}^{[l]} = \max_{i=0}^{p_h-1} \max_{j=0}^{p_w-1} \mathbf{a}_{x \cdot s_x + i, y \cdot s_y + j, z}^{[l-1]}. \quad (2.11)$$

Here, $(p_{x,y,z}^{[l]})$ is the output value at position (x, y) in the z -th feature map of the l -th pooling layer. p_h and p_w are the height and width of the pooling window, respectively. The pooling window is applied with these dimensions to the input feature map. s_x and s_y are the stride values in the horizontal and vertical directions, respectively. The pooling window moves with these strides over the input feature map. $(\mathbf{a}_{x \cdot s_x + i, y \cdot s_y + j, z}^{[l-1]})$ represents the input value at position $(x \cdot s_x + i, y \cdot s_y + j)$ in the z -th feature map of the $(l-1)$ -th layer. The max-pooling operation selects the maximum value within a local region, which is determined by the pooling window size and stride of the previous layer ($a^{[l-1]}$), assigns to the corresponding position in the pooling layer.

2.3. Flatten and dense layers

The extracted features from the convolutional layer have been passed to the flattening layer to create a 1D vector. The flattened layer converts the 2D feature map matrix into 1D vector: $[f_H^{[l-1]} \times f_W^{[l-1]} \times C_{in}^{[l-1]}, 1]$. This 1D vector has been fed into the fully connected layer, which contains the dense layers for the MC technique.

$$\mathbf{o}_n^{[l]} = \sum_{i=0}^{H_{l-1}-1} \sum_{j=0}^{W_{l-1}-1} \sum_{c=0}^{C_{l-1}-1} \mathbf{a}_{i,j,c}^{[l-1]} \cdot \mathbf{W}_{i,j,c,n}^{[l]} + \mathbf{b}_n^{[l]}, \quad (2.12)$$

where $(\mathbf{o}_n^{[l]})$ is the output value of the n -th neuron in the output vector of the fully connected layer (l -th layer). (H_{l-1}) and (W_{l-1}) are the height and width of the feature map in the $(l-1)$ -th layer ($\mathbf{a}^{[l-1]}$), respectively. C_{l-1} is the number of input channels in the $(l-1)$ -th layer. $(\mathbf{W}_{i,j,c,n}^{[l]})$ represents the weight connecting the (i, j, c) position in the $(l-1)$ -th layer to the n -th neuron in the fully connected layer. The input $\mathbf{a}^{[l-1]}$ represents the output from the flattened layer or previous dense layer, and $(\mathbf{b}_n^{[l]})$ represents the bias term. The n -th node of l -th layer of a fully connected network is given by,

$$a_n^{[l]} = \phi^{[l]} \mathbf{o}_n^{[l]}, \quad (2.13)$$

where $\phi^{[l]}$ is an activation function. The last dense layer performs the maximum likelihood probability over M modulation classes, using softmax classifier.

In the output softmax layer, m represents the index of the classes. In a classification problem, the softmax layer computes the probabilities of the input belonging to each class. If there are M classes in the classification task, m ranges from zero to $M-1$, representing the M class labels. The softmax function takes the raw scores $o_n^{[l]}$ from the previous fully connected layer and converts them into a probability distribution over the classes. The probability of the input belonging to class m is denoted by $p_m^{[l]}$. The softmax function is defined as follows:

$$p_m^{[l]} = \frac{e^{o_m^{[l]}}}{\sum_{j=0}^{M-1} e^{o_j^{[l]}}}, \quad (2.14)$$

where $o_m^{[l]}$ is the raw score for class m in the output vector and M is the total number of classes. Each $p_m^{[l]}$ represents the probability of the input being classified into class m and the sum of all $p_m^{[l]}$ values will be equal to one, ensuring that the probabilities form a valid probability distribution. The class with the highest probability ($\max(p_m^{[l]})$) is typically considered as the predicted class (\hat{y}_i) for the applied input during inference.

Table 2. Type, dimension and the learning parameters for each layer of CNN model.

Layer Type	Dimension	Value	Total Parameters
Input Layer	(f_H, f_W, C_{in})	(28, 28, 2)	0
Conv2D_1	$(n_H^{[1]}, n_W^{[1]}, n_C^{[1]})$	(28, 28, 256)	4864
MaxPooling_1	$(n_H^{[1]}, n_W^{[1]}, n_C^{[1]})$	(14, 14, 256)	0
Conv2D_2	$(n_H^{[2]}, n_W^{[2]}, n_C^{[2]})$	(14, 14, 64)	147520
MaxPooling_2	$(n_H^{[2]}, n_W^{[2]}, n_C^{[2]})$	(7, 7, 64)	0
Conv2D_3	$(n_H^{[3]}, n_W^{[3]}, n_C^{[3]})$	(7, 7, 64)	36928
MaxPooling_3	$(n_H^{[3]}, n_W^{[3]}, n_C^{[3]})$	(4, 4, 64)	0
Conv2D_4	$(n_H^{[4]}, n_W^{[4]}, n_C^{[4]})$	(4, 4, 64)	36928
MaxPooling_4	$(n_H^{[4]}, n_W^{[4]}, n_C^{[4]})$	(2, 2, 64)	0
Conv2D_5	$(n_H^{[5]}, n_W^{[5]}, n_C^{[5]})$	(2, 2, 64)	36928
MaxPooling_5	$(n_H^{[5]}, n_W^{[5]}, n_C^{[5]})$	(1, 1, 64)	0
Flatten	$(n_H^{[5]} \times n_W^{[5]} \times n_C^{[5]})$	(64)	0
Dense_1	Number of neurons (n_1)	(128)	8320
Dense_2	Number of neurons (n_2)	(7)	903
Total Trainable parameters			2,72,391

The likelihood probability of a particular modulation type belongs to the input signal x , denoted as $p(y = k|x; \Theta)$, where k is a 1D tensor and $k \in R^M$ is for the different modulation types of the classification task. The CNN parameter (Θ) has been determined to minimize the training loss in the dataset $(x_i, y_i)_{i \in S}$ with training dataset size S ,

$$\min_{\Theta} \sum_{i \in S} l_p(\hat{y}_i, y_i), \quad (2.15)$$

where $l_p(\cdot)$ denotes the categorical cross-entropy. It is defined as a log-likelihood function and represented by

$$l_p(\hat{y}_i, y_i) = \sum_{i=1}^M y_i \log(\hat{y}_i), \quad (2.16)$$

where $i = 1, 2, \dots, M$. The loss function l_p is evaluated at the last dense layer (softmax layer), which evaluates the error between the predicted \hat{y}_i and the actual modulation labels y_i .

The classification testing accuracy (A_{test}) for the given testing dataset can be calculated by

$$A_{test}(\%) = \frac{\sum_{i=1}^M (\hat{y}_i == y_i)}{\sum_{i=1}^M y_i} \times 100\%. \quad (2.17)$$

The layer type, dimension and learning parameters of each layer for a five convolutional layer CNN model have been shown in Table 2. It has been clearly mentioned that the learning parameters have been calculated for each convolution and dense layer. It can be found that the total learning parameters for the five layer model (5-CNN) are 2,72,391, whereas for the three layer (3-CNN) model, the total learnable parameters are 3,21,415.

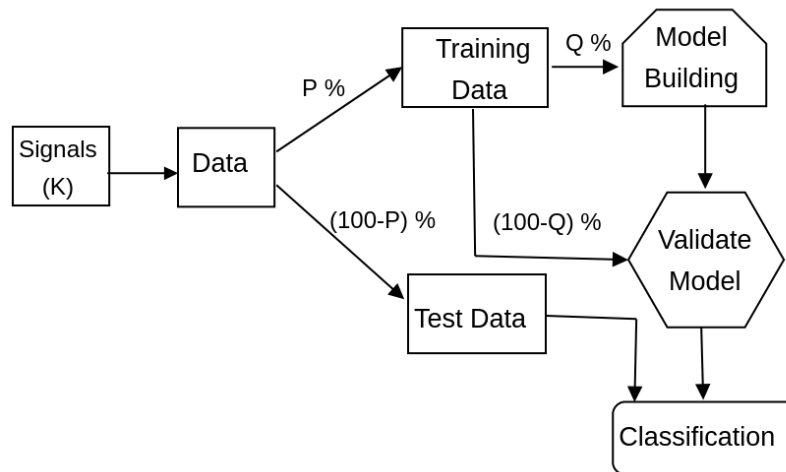
3. Results

The simulations have been conducted with the Intel R Core™ i5 processor. The Ubuntu OS and Python 3.5 have been utilized to carry out the simulation using parameters as listed in Table 3.

Table 3. Simulation parameters.

Parameters	Value
Distance between TX and RX in m	1
SNR in dB	-5 to 10
Sampling Rate (f_s) in MSps	2.4
Center Frequency (f_c) in MHz	810

In this work, we choose samples of different modulation types such as binary phase-shift keying (BPSK), quadrature phase-shift keying (QPSK), continuous-phase frequency-shift keying (CPFSK), 16 quadrature amplitude modulation (16QAM), 64 quadrature amplitude modulation (64QAM), gaussian minimum shift keying (GMSK) and gaussian frequency shift keying (GFSK) at a high SNR position (10 dB) from the RFSC dataset [21] for the generation of the multilayer CNN model. The RFSC dataset created in the laboratory environment includes the exact channel conditions as well as the controlled channel parameters. The verification of the generated CNN models has been performed by using the collected testing dataset from the assumed receiver position.

**Figure 2.** Dataset split-up for CNN model generation, validation and classification.

3.1. Model training performance analysis

The dataset split-up framework for the CNN model generation, validation and classification has been depicted in Figure 2. We can observe that the partition set $\{(P\% : (100 - P)\%) \times \text{input data samples}\}$ represents the number of samples used for training and testing process, and $\{(Q\% : (100 - Q)\%) \times \text{training data samples}\}$ denotes the number of samples used for model generation and validation process, respectively. For the adopted RFSC dataset, we have 1,500 and 500 samples (75:25) for each modulation type for the training and testing process. Further, the dataset for the training process (10,500 samples) has been divided into 7,350 and 3,150 samples (70:30) for model generation and validation, respectively. Note that the models are generated with a training epoch of 70 using the samples collected at locations very close to the transmitter. Here, the training samples are collected from the chosen receiver position with the presence of transmitter at location one (TX1). The proposed CNN model is trained on the RFSC dataset comprised of the labeled signals with known modulation

classes. The back-propagation and stochastic gradient descent algorithms are used to optimize the model parameters and minimize the classification loss.

3.1.1. Training loss and accuracy

Figure 3(a) shows the model training with a minimal training loss. It can be noticed that the CNN model with $l > 2$ provides a similar performance throughout the epoch. Figure 3(b) depicts the training accuracy for all four models. It has been observed that the overshoot gets reduced after 60 epochs, where all the models converge with similar performance.

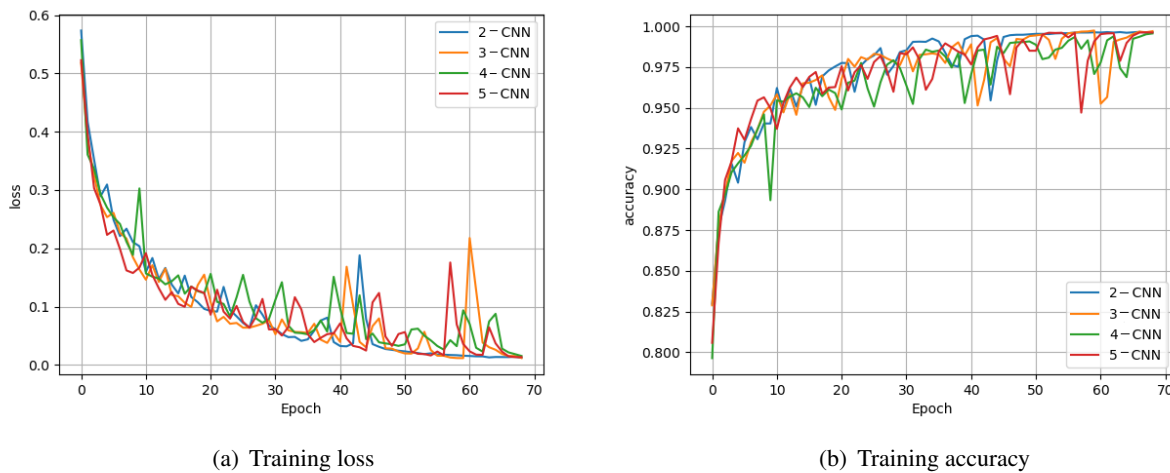


Figure 3. Training performance comparison of various CNN layers for different epoch numbers.

3.1.2. Visualization of feature maps, weights and bias value of the layer

In the training process, the CNN model extracts the features using the filters in each convolutional layer that capture and activate similar patterns. The captured pattern has been visualized as a feature map corresponding to each layer of the CNN model. While passing an input image into a CNN model, each layer generates the feature maps that are moving on to the next successive layer. The initial convolutional layer extracts the horizontal/diagonal edges of an image. The consecutive layers detect the corners of an image. On moving deeper into the network, we can identify even more complex features. The features extracted by the filters used in the third convolutional layer have been depicted in Figure 4.

The mathematical representations of neurons in the convolutional and pooling layers are used to calculate the weighted sum of multiple inputs and outputs. For each neuron, the weights are real values associated with each feature and indicate the significance of that related feature in predicting the modulation type. The weights assigned to each feature play an important role in selecting the output class. The weights close to zero tend to be less important in the classification process than the weights with higher values. The weight value for different index values has been represented in Figure 5(a). The bias value of a neuron is used to push the activation function toward left/right as shown in Figure 5(b).

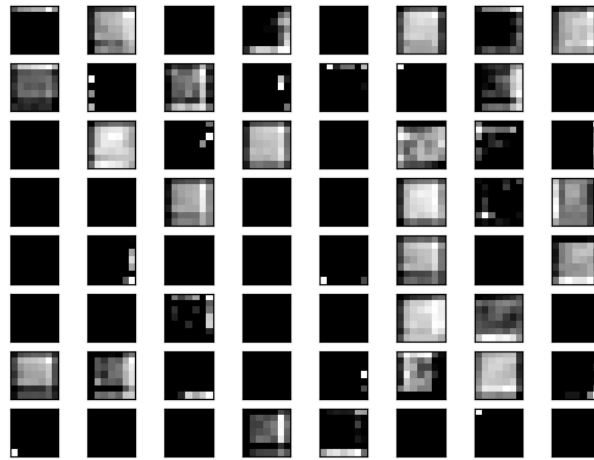


Figure 4. Feature map representation for the fifth layer of 3-CNN model.

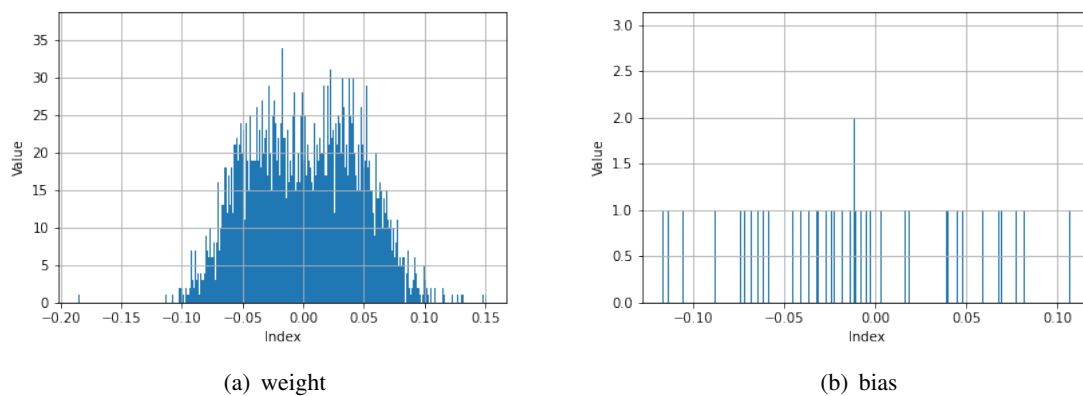


Figure 5. Representation of weights and bias values for a third convolution layer of the 3-CNN model.

3.2. Model testing analysis

In this section, we analyze the multilayer CNN model performance for different input parameters such as decimation factor, image frame size, sample size and the number of convolutional layers.

3.2.1. Different decimation factor

Table 4 shows the testing accuracy of the CNN model for different input frame sizes ($F = 1, 2, 3, 4$) and various decimation rates ($D = 6, 12, 24, 48$). It has been observed that the model generated with four input frames and $D = 12$ yields the highest testing accuracy among other configurations. It has been noticed that accuracy gets reduced with increasing the modulation classes.

Table 4. Testing accuracy of CNN model for a varying number of frames, decimation factor and number of modulation classes.

No. of Frames (F) →	1				2				3				4			
Decimation (D) → No. of classes ↓	6	12	24	48	6	12	24	48	6	12	24	48	6	12	24	48
2 (16QAM, 64QAM)	99.5	99.5	99.2	100	99.5	99.8	99.4	99.8	99.8	99.3	99.0	99.8	99.8	98.9	99.1	99.6
3 (16QAM, 64QAM, BPSK)	98.9	99.1	98.6	97.6	98.1	98.7	98.7	98.5	98.8	99.1	98.4	98.3	98.6	99.1	98.6	98.4
4 (16QAM, 64QAM, BPSK, CPFSK)	84.7	99.0	99.4	97.3	85.0	98.2	98.6	98.1	85.3	99.1	99.0	98.6	85.1	99	98.5	98.3
5 (16QAM, 64QAM, BPSK, CPFSK, GFSK)	86.5	98.2	96.6	96.5	87.5	97.8	97.7	98.0	86.9	97.9	96.9	97.8	86.4	97.8	97.4	97.5
6 (16QAM, 64QAM, BPSK, CPFSK, GFSK, GMSK)	86.0	95.3	94.1	96.1	88.1	97.0	96.7	97.0	87.1	94.7	96.0	97.4	83.4	97.2	96.2	96.8
7 (16QAM, 64QAM, BPSK, CPFSK, GFSK, GMSK, QPSK)	78.6	89.9	95.2	95.5	79.5	91.7	95.6	95.9	78.3	95.9	96.2	96.9	78.3	96.5	96.9	97.6

3.2.2. Different input sample size

In 1D, the actual samples are converted into images with a single frame for a single channel. In 2D, the actual samples are converted into images with four frames. Each frame consists of two channels. For both 1D and 2D cases, the generated CNN model has been tested for $K = 500$ and 1,000 signals. Table 5 compares the model testing accuracy and loss in both cases. It has been observed that the accuracy increases with the respective increment in sample size for both 1D and 2D formats.

Table 5. Comparison of CNN model classification performance for different input sizes, number of samples, image size and number of signals.

Input size	1D						2D					
Actual sample size	1568		6272		25088		1568		6272		25088	
Image size ($f_H \times f_W \times C_{in}$)	$1568 \times 1 \times 1$		$6272 \times 1 \times 1$		$25088 \times 1 \times 1$		$14 \times 14 \times 2$		$28 \times 28 \times 2$		$56 \times 56 \times 2$	
Total number of signals (K)	500	1000	500	1000	500	1000	500	1000	500	1000	500	1000
Testing accuracy $A_{test}(\%)$	86.1	85.7	94.5	94.3	94.8	98.25	88.2	87.8	96.4	96.7	99.0	99.3
Testing loss (l_p)	0.72	0.78	0.27	0.28	0.27	0.06	0.65	0.66	0.22	0.2	0.08	0.04

For $K = 500$ signals, the model classification accuracies (A_{test}) for an image of size ($f_H \times f_W \times C_{in}$) of $14 \times 14 \times 2$, $28 \times 28 \times 2$ and $56 \times 56 \times 2$ are 88.2%, 96.4% and 99%, respectively. Correspondingly, the model testing losses (l_p) are 0.655, 0.22 and 0.084. Similarly, for $K = 1,000$ signals, the model classification accuracies (A_{test}) for $14 \times 14 \times 2$, $28 \times 28 \times 2$ and $56 \times 56 \times 2$ are 87.8%, 96.7% and 99.3%, respectively. Correspondingly, the model testing losses (l_p) are 0.66, 0.2 and 0.0403. It has been noticed that the image size $28 \times 28 \times 2$ gives an optimum classification accuracy with moderate computational complexity compared with the other two image sizes. As K increases from 100 to 500 signals, there is a corresponding increase in the testing accuracy, which is found to be 96.3% as shown

in Figure 6. A gradual increase in the number of signals leads to increased testing accuracy. So, we have chosen K as 500 signals for our work to save computation time.

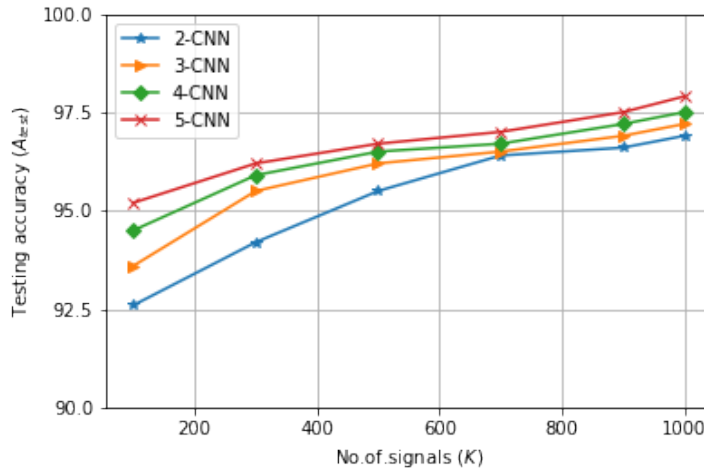


Figure 6. MC accuracy comparison of CNN model for two, three, four and five convolution layers.

3.2.3. Different numbers of CNN layers

Table 6 compares the testing loss and testing accuracy of the CNN model for a varying number of convolution layers (l) in the feature extraction phase. It can be observed that the increasing number of convolution layers in the CNN model yields a reduced testing loss (l_p) and improved accuracy. Moreover, it can be noticed that the testing accuracy remains static after three convolution layers.

Table 6. Testing loss and testing accuracy of CNN model.

No. of Convolutional Layers (l)	Two	Three	Four	Five
Testing loss (l_p)	0.24	0.16	0.15	0.13
Testing accuracy ($A_{test}(\%)$)	95.82	97.0	97.01	97.12

To minimize the computational complexity, we carry out the work with 3-CNN architecture. The classification accuracy increases with increasing convolutional layers as shown in Figure 6. Simultaneously, increasing the number of layers and signals will lead to over-fitting errors. Also, there are chances for noise to be misinterpreted as a signal and vice versa. It is mandatory for the designer to properly tune the convolutional layers based on the number of input signals (K). So, we have chosen a 3-CNN model that provides better testing accuracy and reduced computational time compared with other models.

3.3. Confusion matrix

The confusion matrix for the three-layer CNN at the selected receiver position in the presence of TX1 is shown in Figure 7. The probability values in each box represent the classification accuracy of each modulation type. It has been observed that the model provides an improved performance with

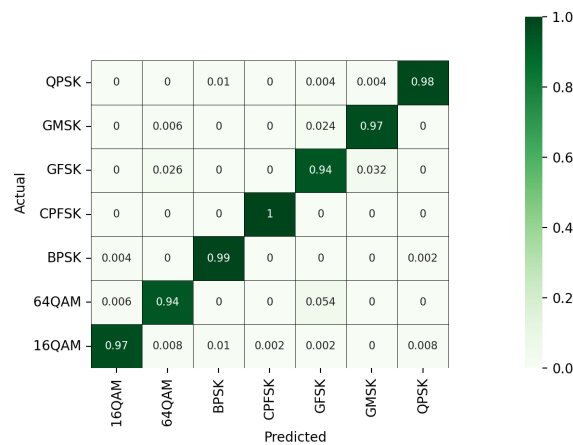


Figure 7. Confusion matrix for three-layer CNN model.

the chosen receiver position. Moreover, the misclassification rate among different modulation classes has been found to be reduced with predicted labels. Due to the nature of MC systems, the multilayer CNN model performs well for the higher SNR values. The proposed model shows better classification performance for the adopted seven modulation classes.

3.4. Analysis of RFSC dataset classification results

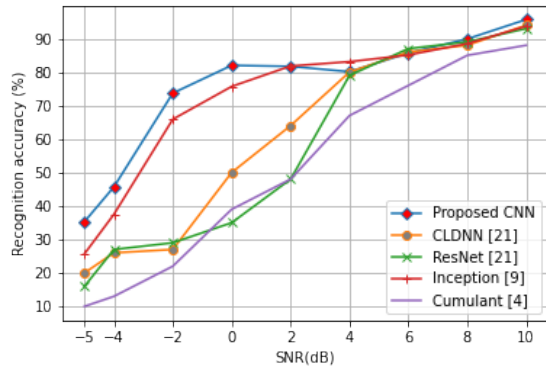
From the RFSC dataset [21] we have selected seven modulation types that are used to test the proposed CNN model for different SNR values. Similarly, the different DL models have been generated and compared against various SNR values as shown in Figure 8(a). It has been observed that the proposed CNN model shows higher performance than the other models at SNR levels lower than 2 dB. The cumulant model has been found to be effective with a maximum accuracy of 88.5%. Moreover, the proposed CNN model performs with a maximum accuracy of 95.7%. Furthermore, the other models provide up to 93% accuracy, along with high computational complexity.

In Table 7, it has been noticed that the proposed CNN model provides the best performance with the least number of convolutional layers and learnable parameters. Furthermore, the proposed 3-CNN model provides a peak training accuracy of 98% for the least time per epoch than the prior models adopted for comparison. Also, the proposed 3-CNN model has been generated with the lowest trainable parameters in a short duration of 15 minutes. The proposed CNN model produces the lowest complexity and CPU inference time at the expense of memory usage allocated for feature extraction.

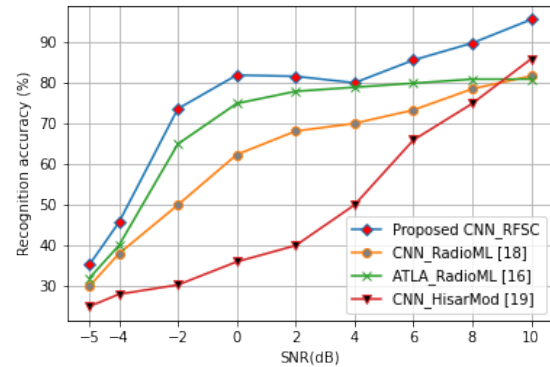
Table 7. Training parameters and computational complexity of different DL models.

Model	Number of Layers	Number of Parameters	Duration for Model Generation (Minutes)	Training Accuracy (%)	Time per Epoch (Seconds)
CNN	6	3,21,415	15	98	8
ResNet	11	6,90,503	30	95	15
CLDNN	9	11,69,607	65	96	25
Inception	15	25,992,327	90	89	10

3.5. Comparison of model performance between RadioML and RFSC dataset



(a) Performance of different DL models



(b) Performance of CNN model among different dataset

Figure 8. Recognition accuracy values against different SNR values.

In this section, we also compare the proposed CNN models generated using RadioML, HisarMod and RFSC datasets. The RadioML and HisarMod consists of different modulation classes that have been created using GNU Radio [22], which include a variety of channel impairments such as fading, frequency offset and sample rate offset [17]. We have analyzed the model performance for the SNR range varying between -5 dB to 10 dB. From Figure 8(b), we can observe that the proposed CNN achieves better recognition accuracy at the low SNR scenario. The RFSC dataset produces the best classification accuracy in the medium SNR scenario. Moreover, the proposed multilayer CNN model has been tested for limited SNR values (-5 dB to 10 dB). Here, the modulation recognition has been restricted to single-carrier modulation types. Moreover, it has been noticed that the CNN model generated using RadioML and the HisarMod dataset provides less than 80% recognition accuracy for SNR values between 4 dB and 10 dB.

In ATLA [16], the authors have frozen the weights of the convolution layers, which reduces the feature extraction capability, whereas in our proposed 3-CNN model, we have not frozen the convolutional layers but adjusted (proper selection/tuning) the weights and bias of the convolutional kernels/filters to achieve the highest classification accuracy. This provides an improvement in the classification accuracy compared with existing DL-based models. Additionally, only two convolutional layers are adopted in ATLA to extract fewer features than the 3-CNN model. Moreover, authors vary the size of the dataset only with sampling frequency. However, in our proposed work, we not only vary the sampling rate but also include the number of signals and decimation rate. This greatly reduces the issues associated with decoding PRB/master information block (MIB), particularly in resource-constrained Internet of Things (IoT) applications.

4. Conclusions and future enhancements

MC has been used extensively in the military, civilian and future-generation wireless systems to detect and classify modulation types, hence preserving the radio spectrum. A detailed mathematical framework for the proposed multilayer CNN model has been presented. We adopted the RFSC

dataset for the generation of the proposed multilayer CNN model. Moreover, improvements in the classification accuracy for better decimation rates, frame size and input sample size for the different convolutional layer model have been discussed. Furthermore, the 3-CNN model achieved nearly 98% classification accuracy. Thus, we conclude that there exists a trade-off while choosing the CNN model between the model parameter size and the computational complexity, which are more viable for resource-limited scenarios. We presented the recognition performance comparison between the DL models and the cumulant-based model. We also showed that the model generated by the RFSC dataset achieves higher accuracy than the models generated by the RadioML and Hisarmod datasets. In the near future, we will investigate MC for various modulation orders by identifying each modulation class. Extensive research for varying the number and size of convolutional filters finds an interesting topic to be investigated in resource-constrained applications.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Challita U, Sandberg D (2021) Deep Reinforcement Learning for Dynamic Spectrum Sharing of LTE and NR. *arXiv e-prints*, *arXiv:2102.11176*.
2. Papageorgiou GK, Voulgaris K, Ntougias K, Ntaikos DK, Butt MM, Galiotto C, et al. (2020) Advanced Dynamic Spectrum 5G Mobile Networks Employing Licensed Shared Access. *IEEE Commun Mag* 58: 21–27. <https://doi.org/10.1109/MCOM.001.1900742>
3. Liao F, Wei S, Zou S (2020) Deep Learning Methods in Communication Systems: A Review. *Journal of Physics: Conference Series* 1617: 012024. <https://doi.org/10.1088/1742-6596/1617/1/012024>
4. Huynh-The T, Pham QV, Nguyen TV, Nguyen TT, Ruby R, Zeng M, et al. (2021) Automatic Modulation Classification: A Deep Architecture Survey. *IEEE Access* 9: 142950–142971. <https://doi.org/10.1109/ACCESS.2021.3120419>
5. Spooner CM, Mody AN, Chuang J, Petersen J (2017) Modulation Recognition Using Second- and Higher-Order Cyclostationarity. *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 1–3. <https://doi.org/10.1109/DySPAN.2017.7920744>
6. Alarabi A, Alkishriwo OAS (2021) Modulation Classification Based on Statistical Features and Artificial Neural Network. *IEEE International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering*, 748–751. <https://doi.org/10.1109/MI-STA52233.2021.9464363>
7. An TT, Lee BM (2023) Robust Automatic Modulation Classification in Low Signal to Noise Ratio. *IEEE Access* 11: 7860–7872. <https://doi.org/10.1109/ACCESS.2023.3238995>

8. PramodKumar A, KiranKumar G (2023) Blind Modulation Classification in Multiple Input and Output-Orthogonal Frequency Division Multiplexing Using Time-Frequency Analysis and Customized Convolutional Neural Network Architecture. *T Emerg Telecommun T* 34: e4720. <https://doi.org/10.1002/ett.4720>
9. Wu P, Sun B, Su S, Wei J, Zhao J, Wen X (2020) Automatic Modulation Classification Based on Deep Learning for Software-Defined Radio. *Math Probl Eng* 2020: 1–13. <https://doi.org/10.1155/2020/2678310>
10. Peng S, Jiang H, Wang H, Alwageed H, Zhou Y, Sebdani M, et al. (2018) Modulation Classification Based on Signal Constellation Diagrams and Deep Learning. *IEEE T Neur Net Lear Syst* 30: 718–727. <https://doi.org/10.1109/TNNLS.2018.2850703>
11. Vora A, Thomas P, Chen R, Kang K (2019) CSI Classification for 5G via Deep Learning. *IEEE Vehicular Technology Conference (VTC2019-fall)*, 1–5. <https://doi.org/10.1109/VTCFall.2019.8891133>
12. Mohamed ME, Safwat MR (2023) A Hybrid Model for Automatic Modulation Classification Based on Residual Neural Networks and Long Short Term Memory. *Alex Eng J* 67: 117–128. <https://doi.org/10.1016/j.aej.2022.08.019>
13. Tang ZL, Li SM, Yu LJ (2018) Implementation of Deep Learning-Based Automatic Modulation Classifier on FPGA SDR platform. *Electronics* 7: 122–138. <http://dx.doi.org/10.3390/electronics7070122>
14. Tridgell S, Boland D, Leong PH, Siddhartha S (2019) Real-Time Automatic Modulation Classification. *International Conference on Field-Programmable Technology (ICFPT)*, 299–302. <https://doi.org/10.1109/ICFPT47387.2019.00052>
15. Sebin JO, Renu J (2023) LSTM Projected Layer Neural Network-Based Signal Estimation and Channel State Estimator for OFDM Wireless Communication Systems. *AIMS Electronics and Electrical Engineering* 7: 187–195. <https://doi.org/10.3934/electreng.2023011>
16. Bu K, He Y, Jing X, Han J (2020) Adversarial Transfer Learning for Deep Learning Based Automatic Modulation Classification. *IEEE Signal Proc Lett* 27: 880–884. <https://doi.org/10.1109/LSP.2020.2991875>
17. Pijackova K, Gotthans T (2021) Radio Modulation Classification Using Deep Learning Architectures. *International Conference Radioelektronika (RADIOELEKTRONIKA)*, 1–5. <https://doi.org/10.1109/RADIOELEKTRONIKA52220.2021.9420195>
18. Liu X, Yang D, Gamal AE (2017) Deep Neural Network Architectures for Modulation Classification. *Asilomar Conference on Signals, Systems, and Computers*, 915–919. <https://doi.org/10.1109/ACSSC.2017.8335483>
19. Tekbıyık K, Ekti AR, Görçin A, Kurt GK, Keçeci C (2020) Robust and Fast Automatic Modulation Classification with CNN Under Multipath Fading Channels. *IEEE Vehicular Technology Conference (VTC)*, 1–6. <https://doi.org/10.1109/VTC2020-Spring48590.2020.9128408>
20. Teng CF, Chou CY, Chen CH, Wu AY (2020) Accumulated Polar Feature-Based Deep Learning for Efficient and Lightweight Automatic Modulation Classification With Channel Compensation Mechanism. *IEEE T Veh Technol* 69: 15472–15485. <https://doi.org/10.1109/TVT.2020.3041843>

21. Tamizhelakkiya, Chandhar P, Gauni S (2021) Comparison of Deep Architectures for Indoor RF Signal Classification. *International Conference on Emerging Techniques in Computational Intelligence (ICETCI)*, 107–112. <https://doi.org/10.1109/ICETCI51973.2021.9574083>
22. GNU Radio. Available from: <https://www.gnuradio.org/>



AIMS Press

© 2023 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0>)