6
2D-FFT based Modulation Classification

6.1 INTRODUCTION

In chapters 4 and 5, point density of SDP and constellation have been used as a key feature and two DCNN models are used for classification. Both the classification models provide significantly high classification accuracy with computation complexity. In this chapter, the two-dimensional Fast Fourier Transform (2D-FFT) of constellation structure is used as a classification feature. CDM is formed using the density spread of constellation points and a 2D-FFT matrix is generated through the two-dimensional Fast Fourier Transform of CDM. A light and efficient DCNN model is designed to classify the modulation schemes of different orders of PSK and QAM. The developed method achieves adequate classification performance for considered five modulation schemes in the AWGN channel [Kumar *et al.*, 2020a].

6.2 DCNN-BASED CLASSIFIER

In this section, the developed classification strategy is described. The received complex symbols are used to form a density matrix (DM) and transform into the frequency domain using 2D-FFT, which is used as a feature matrix for classification using DCNN. A light DCNN model with concatenated convolution layers and residual connections is developed for classification.

6.2.1 Signal pre-processing and 2D-FFT matrix generation

Considering N number of constellation points are extracted from the baseband signal waveform. Symbols available in the 4×4 complex plane are normalized to zero mean and unity variance. The selected area of the constellation plane is divided into 100×100 grid sections and the number of symbols in each grid section is calculated to form DM. 2D-FFT is performed to DM generated using 100×100 grid.

For a given $M \times K$ dimensional DM $u(m,k)$, 2D-FFT is given by

$$U(p,q) = \mathscr{F}[u(m,k)] \tag{6.1}$$

where $\mathscr{F}$ denotes Fourier Transform operator.

$$U(p,q) = \sum_{m=0}^{M-1} \sum_{k=0}^{K-1} u(m,k) e^{-j2\pi(\frac{mp}{M})} e^{-j2\pi(\frac{kq}{K})} \tag{6.2}$$

for $m = 0, 1, ..., M-1$ and $k = 0, 1, ..., K-1$.

The transform kernel can be divided into two kernels since one summation depends on m and p while another on k and q.

$$U(p,q) = \sum_{m=0}^{M-1} V(m,q) e^{-j2\pi(\frac{mp}{M})} \tag{6.3}$$

for $p = 0,1,...,M-1$ and $q = 0,1,...,K-1$. Where,

$$V(m,q) = \sum_{k=0}^{K-1} u(m,k) e^{-j2\pi(\frac{kq}{K})} \tag{6.4}$$

for $m = 0,1,...,M-1$ and $q = 0,1,...,K-1$.

The above formulation is a decomposition of 2D-FFT into a row and column-wise 1D-DFT. In this process, the first 1D-DFT is performed along with row and the results are transformed using column-wise 1D-DFT (or vice versa).

6.2.2 DCNN architecture

A DCNN network is developed which extracts important features from the data without manual intervention and provides good efficacy in classification problems. The layered architecture of the proposed DCNN network is shown in Figure 6.1. The basic building blocks of the network are the convolution layer (Conv1D), activation function, pooling layer (AveragePool1D), and dense layer. It has been observed in deep networks, while backpropagation, vanishing gradient problem occurs and the training process does not provide optimum weights. To eliminate this issue, two residual shortcut connections are made across more conjunctive convolution layers.

Consider input to the l^{th} layer to be U^{l-1} and output of that layer is U^l, $U^l \in \mathbb{R}^{M^l \times K^l}$ (here $M^l \times K^l$ represents dimension of l^{th} layer i.e. $M \times K$). For $l = 1$, input is U^0 with the dimension of 100×100. U^0 is scaled to bring in the range $(0,255)$. The input is also visualized as M^l number of vector with the length of K^l. For l^{th} layer processing, with input of vector length K^{l-1} and output of L^l requires $K^{l-1} \times K^l$ number of kernels. Output of the l^{th} convolution layer can be given as

$$U_i^l = h(\sum_{j=1}^{K^{l-1}} U_j^{l-1} * \kappa_{i,j}^l + B_i^l) \tag{6.5}$$

Here, $h(a) = max(0,a)$ is rectified linear unit (ReLU) used for non-linear activation, κ is kernel, B is bias, and $*$ represents convolution operator. Average pooling is used to downsample the input vector with the factor of 2 by taking the average of conjunctive two values.

$$U^{l+1} = downsample(U^l, 2) \tag{6.6}$$

where $downsample(.)$ updates the dimension of output vector to $M^{l+1} = M^l/2$. The first residual short cut connection is introduced after three convolution layers keeping the output dimensions of layer $l+1$ and $l+4$ to be equal for addition ($K^{l+1} = K^{l+4}$). After completion of convolution operation with cascaded convolution layers, activation, and average pooling, output

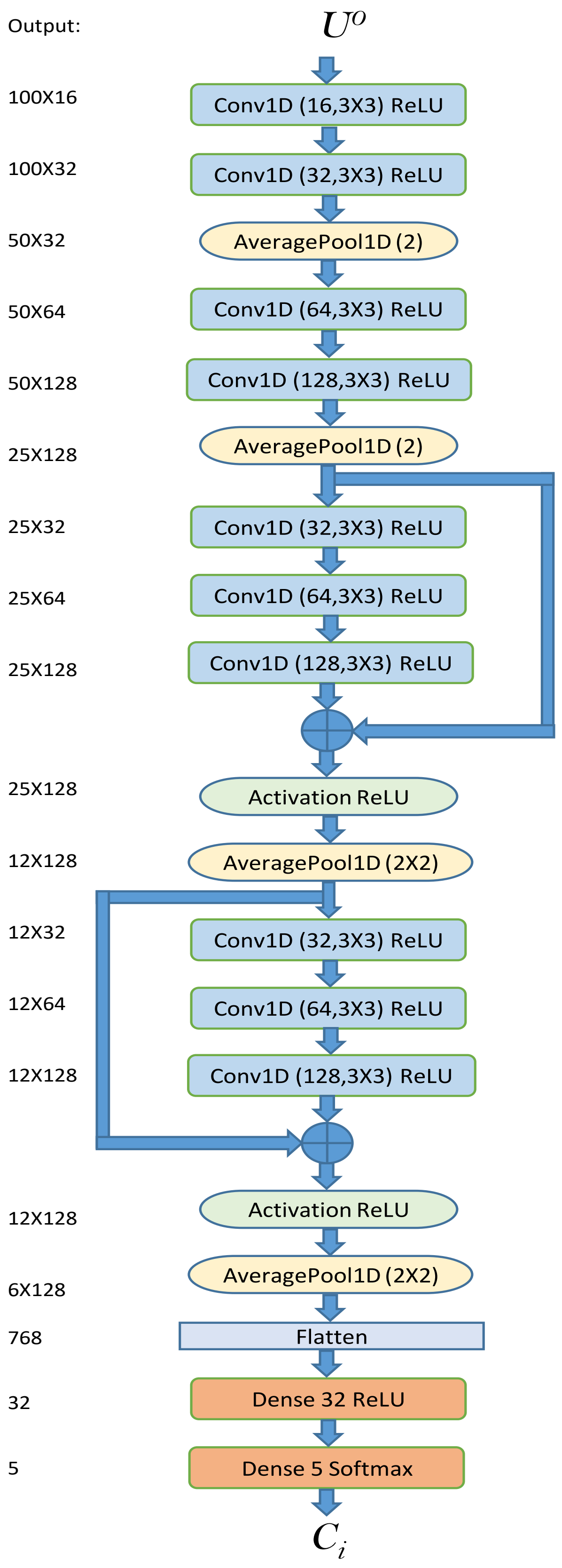


Figure 6.1 : DCNN architecture.

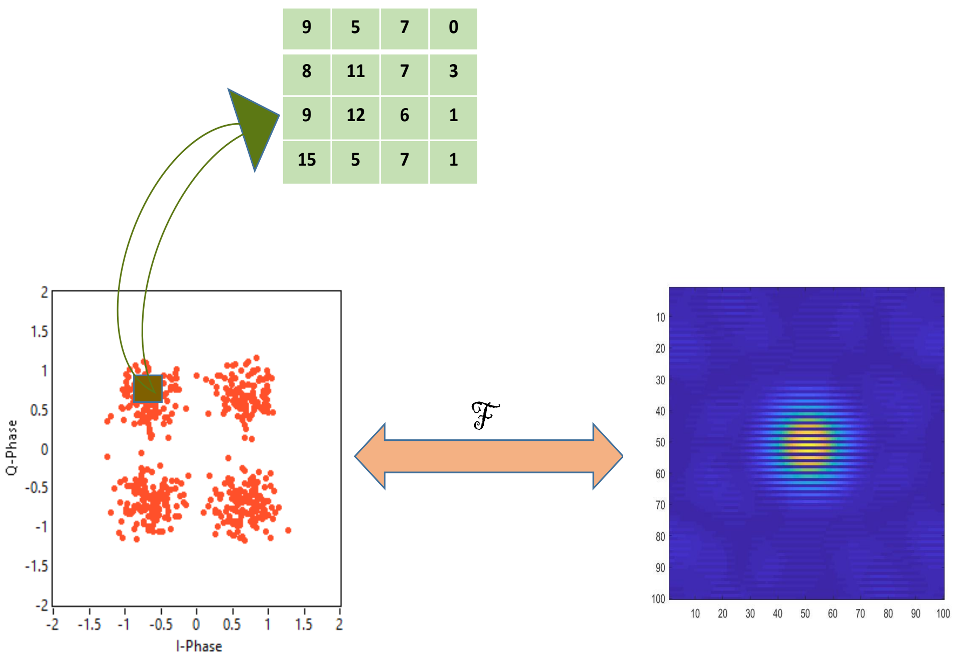


Figure 6.2 : Generation of 2D-FFT from DM of I-Q diagram.

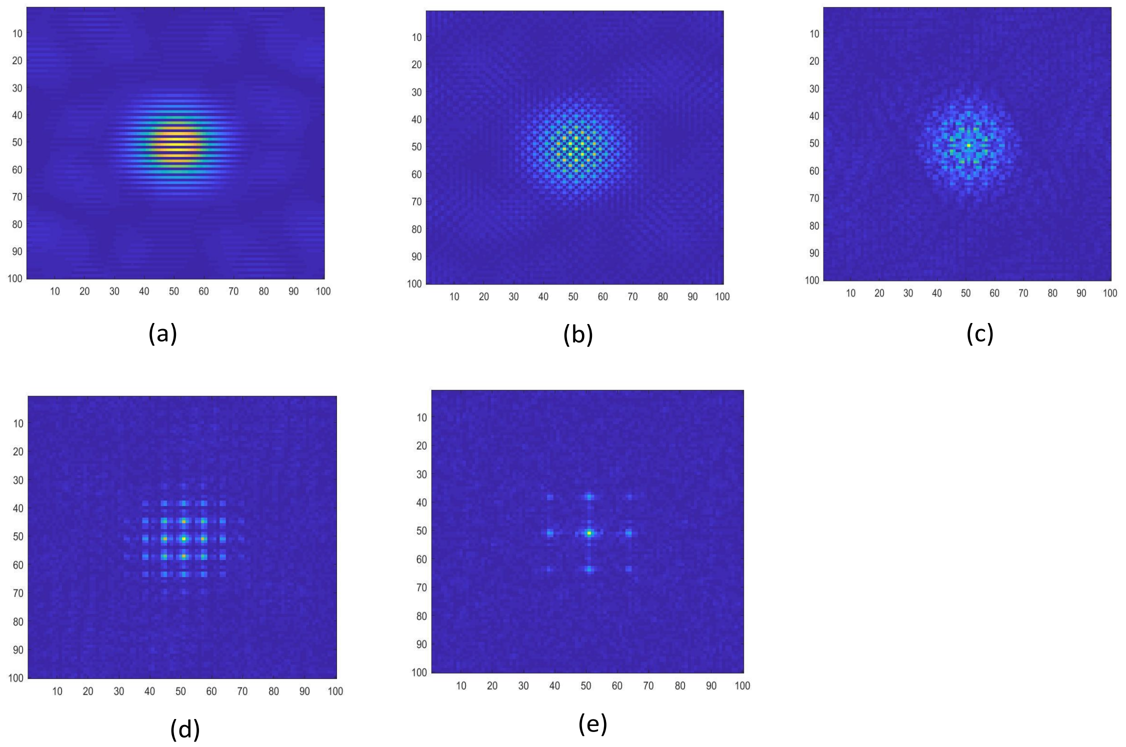


Figure 6.3 : 2D-FFT dataset generated from DM of (a) BPSK, (b) QPSK, (c) 8PSK, (d) 16QAM, and (e) 64QAM.

2D data is flattened to a vector that contains significant feature values. This array vector of the dimension of 768×1 is connected with a dense layer of 32 neurons. Each neuron of the previous layer is sharing information with each neuron of the dense layer, which makes the parameter dimension to be (768×32). The output of l^{th} dense layer can be described by

$$U^l = h(W^l U^{l-1} + b^l) \tag{6.7}$$

Where W is weight matrix of size of (768×32), and b is bias vector of length 32. Similarly, another dense layer of 5 neurons is connected to produce 5 output values corresponding to the 5 classes. The activation function in this layer is softmax, which provides the probability of each class to be the classified output. For the input to softmax activation $d_i, i \in (1,5)$, output probability can be defined as

$$p_i = \frac{e^{d_i}}{\sum\limits_{j=1}^{5} e^{d_j}} \tag{6.8}$$

Random 25% dropout is introduced into the 32 neurons dense layer to provide regularization and avoid the network to overfit with the training data.

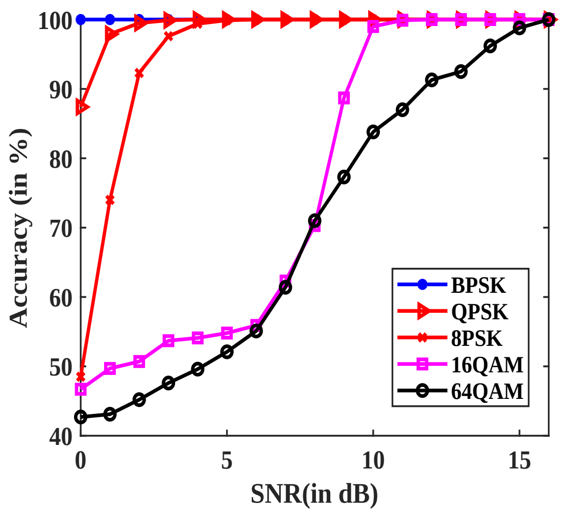
6.2.3 Implementation and training

The classification process includes five modulation schemes BPSK, QPSK, 8PSK, 16QAM, and 64QAM. All the complex base-band signals are generated with RadioML software. Received complex symbols are mapped to the complex plane of dimension 4×4 as shown in Figure 6.2. This dimension has been chosen to include most of the symbol points for the considered SNR range. Mapped constellation points are used to generate DM of dimension 100×100. DM is transformed into the frequency domain using 2D-FFT to extract the classification features which is shown in Figure 6.2. The generated 2D-FFT matrices for all five considered modulation schemes are shown in Figure 6.3, which are directly given to the developed DCNN for training. The network has trained for 50 epochs with the data shuffled by random state-2. The best-trained model weights are saved during the training procedure. The complete dataset is divided into small batches of size N_b (considered batch size is 64) and the model is trained batch-wise. Considered loss function for training procedure is categorical cross-entropy given as

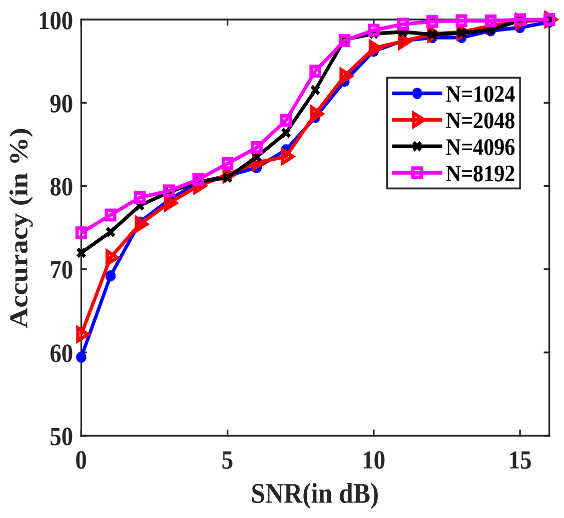
$$\mathscr{L} = -\frac{1}{N_b} \sum_{i=1}^{N_b} y_i \log p_i \tag{6.9}$$

Where y_i is the ground truth of the output class, p_i is the output of the softmax activation function, and $\mathscr{L}$ is the cumulative loss value for all batches. Model weights are updated using stochastic gradient descent (SGD) algorithm with an adaptive learning rate to minimize the loss value. Adam optimizer is used to adapt the learning rate for the fast learning process.

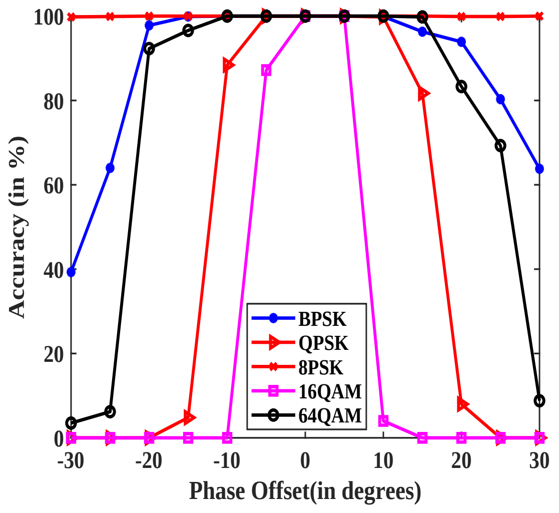
For the training process 250 signals are generated at each SNR. Between 0 dB and 20 dB SNR, step size of 1 dB, and between 25 dB and 50 dB SNR step size of 5 dB is considered. The size for the training dataset is 33750, out of which 20% (i.e. 6750) signals are used for validation.



(a)



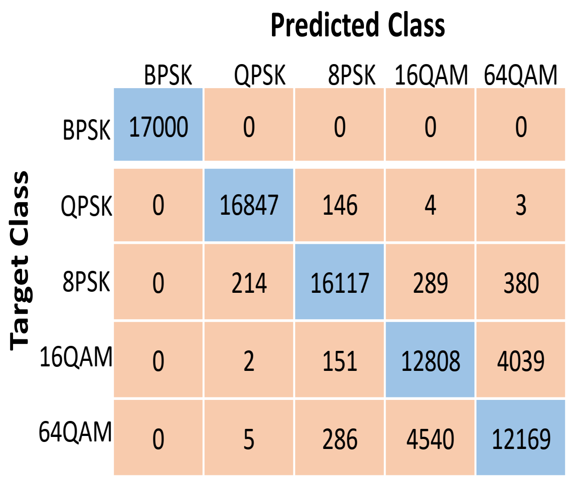
(b)



(c)

Figure 6.4 : (a) Accuracy of the developed method with SNR for all five modulation schemes. (b) Performance comparison of the model with signal length. (c) Classification performance of the model with phase offset. BPSK, QPSK, and 8PSK modulation schemes are tested at 5 dB SNR and 16QAM, and 64QAM are tested at 20 dB SNR.

DCNN model is implemented in python with Keras, and NVIDIA DGX-2 GPU has been used to train the model. Parameters of the solver configuration have been adjusted for better classification and a faster training process.



Target Class \ Predicted Class	BPSK	QPSK	8PSK	16QAM	64QAM
BPSK	17000	0	0	0	0
QPSK	0	16847	146	4	3
8PSK	0	214	16117	289	380
16QAM	0	2	151	12808	4039
64QAM	0	5	286	4540	12169

Figure 6.5 : Confusion matrix of proposed DCNN model for all five classes. Each class is tested with 1000 signal realization for a range of SNR (0, 16) dB.

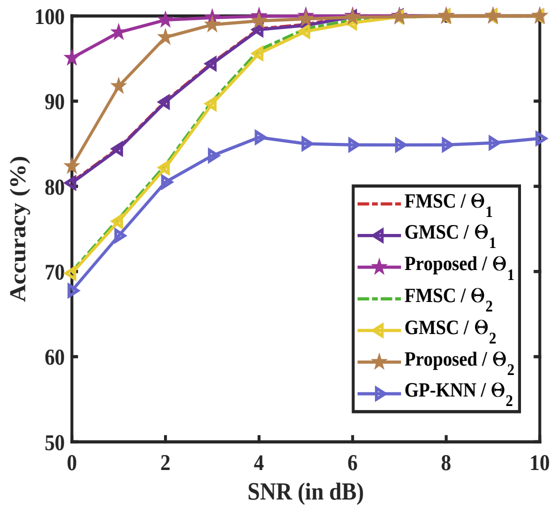


Figure 6.6 : Comparison of proposed method with naive-based and SVM classifiers.

6.3 SIMULATION RESULTS

In this section performance of the DCNN model for the classification of five modulation schemes in the AWGN channel is shown. Classification accuracy for each modulation scheme with SNR is shown in Figure 6.4(a). Accuracy results are obtained for 1000 signal realization, with each signal containing 2048 symbols. Figure 6.4(b) shows the average classification accuracy of the proposed model with signal length. As shown in Figure 6.4(a), the classification accuracy of the model for BPSK, QPSK, and 8PSK are adequate in comparison to 16QAM and 64QAM. Three modulation schemes viz. BPSK, QPSK, and 8PSK are classified with 100% accuracy at 5 dB SNR while for 16QAM and 64QAM, 11 dB and 16 dB SNR are required for reliable results. The reason for the higher SNR requirement for 16QAM, and 64QAM is the confusion between each other shown in a confusion matrix given in Figure 6.5.

The performance of the DCNN model has been tested with the phase offset for practical channel conditions. In Figure 6.4(c), the average classification accuracy with varying phase offset is shown for all five modulation schemes. Results of classification accuracy with phase offset for BPSK, QPSK, 8PSK are given at 5dB SNR while for 16QAM, and 64QAM, 20dB SNR is considered.

6.4 COMPARISON WITH EXISTING WORK

In this section performance comparison of the developed method has been done with some of the existing work in literature. In [Wang *et al.*, 2020a], authors have developed a modulation classification method for varying SNR values by training the model with the signals from a range of SNR values. For a set of modulation schemes $\Theta_1 = \{BPSK, QPSK, 8PSK\}$, classification accuracy comparison of their method generalized modulation signal classification (GMSC) and proposed method is shown in Figure 6.6. Comparison with another method, fixed modulation signal classification (FMSC) is given in Figure 6.6. The performance of the proposed method is better than both the above-mentioned works. We have also compared the developed method with these two methods for another set of modulation schemes $\Theta_2 = \{BPSK, QPSK, 8PSK, 16QAM\}$ and shown the better classification results.

In [Aslam *et al.*, 2012], the authors have used genetic programming (GP) for feature optimization, and classification has been done using k-nearest neighbor (KNN). The considered modulation set taken is Θ_2. For the same set of modulation schemes, we have compared the classification results in the AWGN environment. Figure 6.6 shows the better performance of the proposed method over the GP-KNN method for all values of the SNR range. In [Wong *et al.*, 2008], authors have considered higher-order statistical features and classification has been done using naive-based, SVN, and maximum likelihood (ML) modulation classifier (MC). For same set of modulation schemes $\Theta_3 = \{BPSK, QPSK, 16QAM, 64QAM\}$ and SNR range (10, 20) dB, performance comparison of the proposed method with naive-based, SVM, and ML MC is done in TABLE 6.1.

Table 6.1 : Comparison between different classifiers given in [Wong *et al.*, 2008] and proposed work for Θ_3 modulation set

SNR (dB)	Naive	SVM	ML MC	Proposed
10	97.67	97.59	75	98.3
12	98.69	98.55	75	99.05
14	99.43	99.04	98.53	99.4
16	99.42	99.26	100	100
18	99.4	99.36	100	100
20	99.66	99.4	100	100

...