Neural Network based Modulation & Channel Coding Identification for SATCOM Systems

2nd Milestone's Performance Evaluation

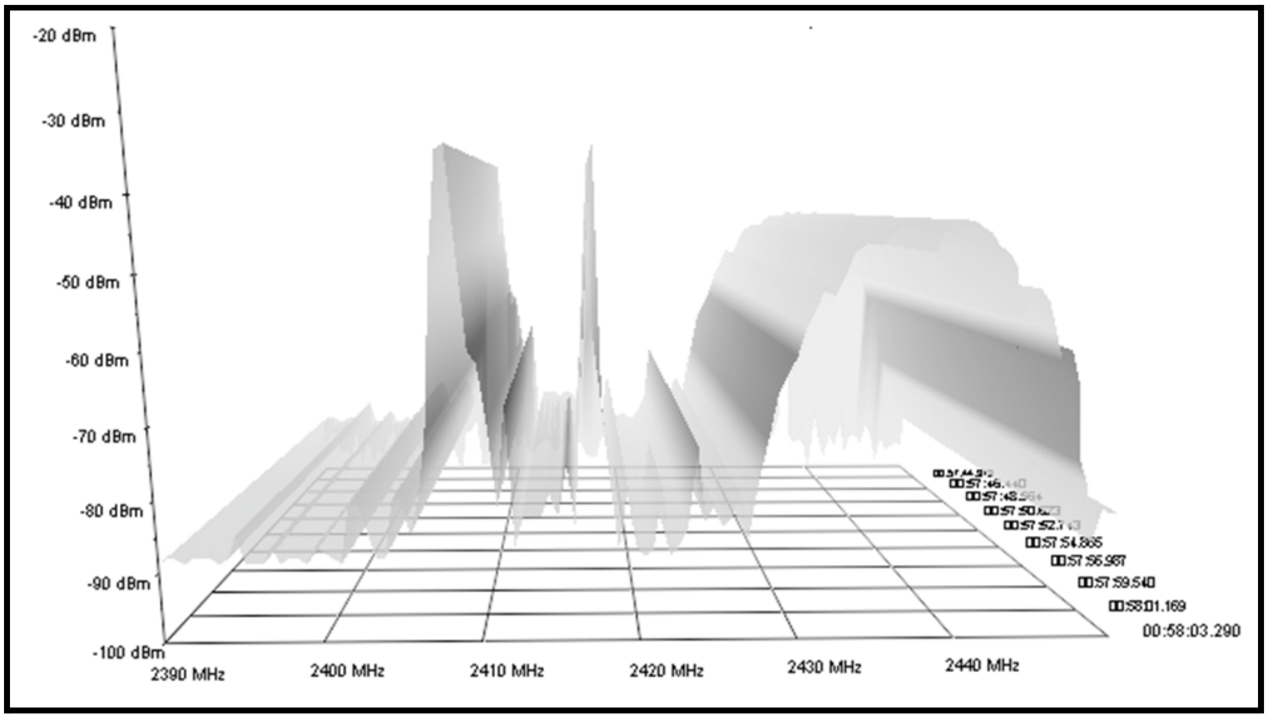
Principal Investigator:
Syed Ali Hassan, Ph.D.
Electrical Engineering

Department of Electrical Engineering
School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST)
Islamabad, Pakistan

2022

Proposed Neural Network for MODCOD Project Report

Neural Network based Modulation & Channel Coding Identification for SATCOM Systems



Prepared by:

Research Assistant: **Saad Iqbal, BE**
Electrical Engineering

Principal Investigator: **Syed Ali Hassan, Ph.D.**
Electrical Engineering
Department of Electrical Engineering
School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST)
Islamabad, Pakistan

Table of content

Preface

This document is penned in reference to the 2nd milestone of the research project titled "Neural Network-based Modulation and Channel Coding Identification for SATCOM Systems". The document contains the designing, training, and testing of proposed Neural Networks (NNs) over the generated dataset. Moreover, a detailed discussion on the results is attached to the document as a model's simulation report.

The report is sub-categorized into four sections. Sections are interlinked to ensure an accurate understanding of the reader. The first section discussed the dataset generation for the project. It provides a complete guide to re-generate a dataset from the basics for passionate readers. The second section has elaborated on salient features of the proposed NNs and talks about their implementation steps. The third section has discussed the training and testing of the proposed NNs. Around ~19 modulations and ~2 coding classifiers are mentioned in detail. The fourth section encompasses the debate on the results of the training and testing of NNs which has led to the designing of the efficient NN model for the classification of modulation and coding schemes. Later, the sequence diagram of the NN model for the classification of modulation and coding schemes is presented.

The report concludes with the future work, which talks about the next stage of the project. This document provides the complete recipe for the designing of NNs for the project and is considered a key document for the project to start with.

Introduction & Approach

All radio communication signals (radio, television, mobile phones, etc.) are modulated before transmission. Modulation recognition is fundamental for correct demodulation. Moreover, if the transmission is using the forward error correction (FEC) scheme then the prior knowledge of the coding scheme is essential for interpretation of the signal. This is the summary of our quest, to automatically recognize the modulation and coding scheme embedded in the received signal.

This report is associated with the second milestone of the project, which is as follow:

- Development and Simulation of Classifier Algorithm using NN
  - ☐ Formulation of Data packet library for MODCOD implementation i.e., M-PSK and M-APSK where M = 2, 4, 8, 16 and 32 with TC / LDPC or Polar Codes at different code rates i.e., 1/4, 1/3, 1/2, 3/4, 5/7, 7/8 etc.
  - ☐ Preparation of Data sets for training NN.
  - ☐ Implementation of classifier algorithm based on NN.
  - ☐ Simulation of proposed classifier algorithm in supporting software.

After the detailed literature review. It has been decided to keep the classification of modulation and channel coding separately. Therefore, the research project is splitted into two parts in its early stage i.e. "NN based classification of modulation scheme" and "NN based classification of Coding scheme". Almost the same modular approach has been devised for both sub-tasks. The approach was:

- ➔ Generation of DataSet
- ➔ Designing of the NN based Classifier models
- ➔ Training and Testing of every designed model over the dataset
- ➔ Inferences from the collected results
- ➔ Tune the proposed model with the best parameters obtained from Inference

This approach has been proved successful and the NN based models for both schemes are designed with high accuracy ~99%. However, coding scheme classifiers aren't the application of the NNs. This is discussed in detail in the heading "Simulation Results & Findings".

Dataset Generation

The dataset generation is the task of creating a dataset for the proposed Neural Network. Unbiased and uniform data has been generated in the MATLAB program for different modulation & coding schemes under variable scenarios. The "*randi([0 1],frame_len,1)*" MATLAB in-built function is utilized to ensure the randomization. The detailed data generation procedure is inscribed in the following subheadings.

DataSet for Modulation Schemes

Modulation is a technique for impressing information (voice, music, picture, or data) on a radio-frequency carrier wave by varying one or more characteristics of the wave in accordance with the intelligence signal. The key characteristics of any RF signal are its amplitude, frequency and phase[1] . The six digital modulation schemes are realized for the DataSet i.e. **BPSK, QPSK, PSK-8, PSK-16, QAM-16** and **QAM-32**.

Salient Features of Modulation Dataset

The signal transmitter is simulated in MATLAB for all six digital modulation schemes under a variable environment and its baseband output is collected as DataSet. Following are the salient features of data generation for modulation schemes:

- Sample Rate

The sample rate is termed as the number of samples transmitted per second. The fixed sample rate of *1024 Samples/sec* is devised for the modulation DataSet.

- Symbol Rate

The symbol rate is termed as the number of symbols transmitted per second. The three symbol rates i.e. *32, 64 and 128 symbols/sec* are used for the modulation DataSet.

- Pulse Shaping

In order to limit the bandwidth of the digital transmitted signal, the signal is passed through pulse shaping filter. Most of the transmitter utilizes a *root raised cosine filter* for pulse shaping. The *RRC filter with rolloff 0.35 spanning over 4 symbols* is selected for the modulation DataSet.

- EbNo

In data transmission, the receiver received a signal deteriorated by the noise. EbNo is defined as the signal to noise per bit. The EbNo *ranging from -4dB to 14dB with steps of 2dB* is realized for the modulation DataSet.

- Channel Fading

In data transmission, the channel fading occurs due to the effects of channel characteristics on the transmitted signal. The common fading types are *Rician and Rayleigh fading*. The modulation DataSet is gathered under the following environments:

- ➔ No fading environment
- ➔ Rician Fading environment with $k = 0.9$
- ➔ Rician Fading environment with $k = 0.7$
- ➔ Rician Fading environment with $k = 0.3$
- ➔ Rayleigh Fadingenvironment

Pre-processing of Modulation DataSet

The baseband signal is expressed using the 2-dimensional IQ data. It came to our understanding after a few trials that the IQ data is not sufficient enough for NN models to track variation patterns in phase and amplitude of the signal. Therefore, the amplitude and phase information is stacked with the IQ data in the modulation DataSet. Therefore, the 4-dimensional modulation DataSet is realized in the final version.

Normalization of Modulation DataSet

The finalized version of dataSet contains the IQ signal along with its phase and amplitude. The IQ signal value could be any from -ve infinity to +ve infinity. Similarly, phase value pounded between -pi to +pi and amplitude value will fluctuate between 0 to infinity. Keeping the nature of input for the neural network model, the scheme of normalization of the DataSet is accepted for trial.

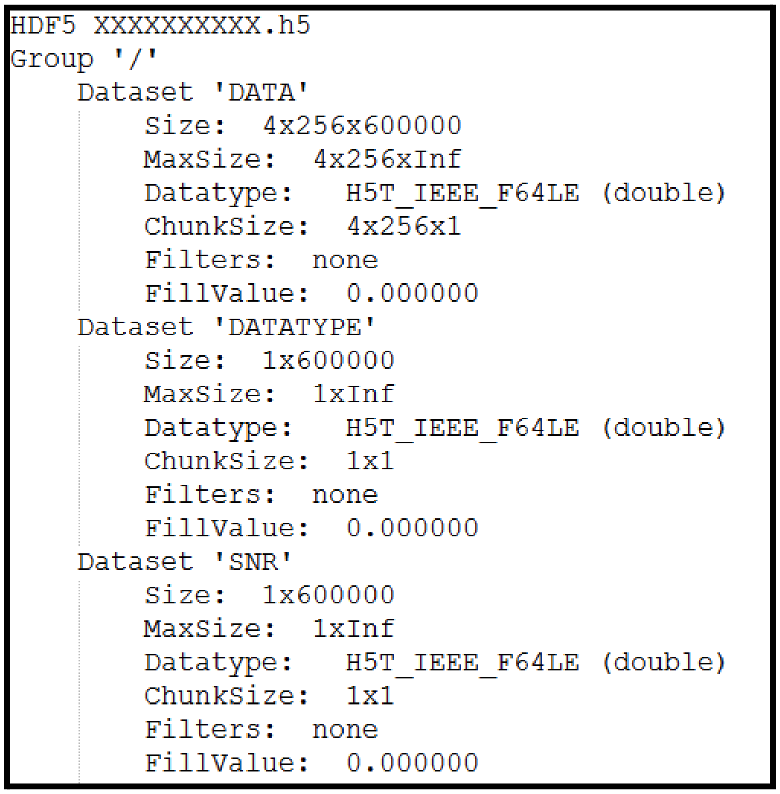
The Normalization is done through the conformal mapping tool, which is expressed as follow:

- ➔ Firstly, the IQ signal block is mapped in the half-radius circle. Afterwards, the circle is translated in the first quadrant. This ensures that IQ input for the NN will be limited between 0 to 1 with losing any of its valuable signatures.
- ➔ The absolute of the IQ signal block mapped on the unit circle would range between 0 to 1. This is considered as the amplitude of the input signal for NN.
- ➔ The phase of any IQ signal lies between 0 to 2pi. Therefore, the phase input for NN is the phase of the IQ signal block normalized by 2pi.

The procedure of normalization of the DataSet has ensured the input for the neural network would be between 0 to 1 without losing any integrity of the DataSet. The extensive research regarding neural networks suggests that normalization enhances the training and prediction time of the model.

Composition of Modulation DataSet

The multiple modulation dataset files were developed portraying different scenarios. The overall composition of the dataset is illustrated as in figure no. 1.



```
HDF5 XXXXXXXXXX.h5
Group '/'
    Dataset 'DATA'
        Size:  4x256x600000
        MaxSize:  4x256xInf
        Datatype:   H5T_IEEE_F64LE (double)
        ChunkSize:  4x256x1
        Filters:  none
        FillValue:  0.000000
    Dataset 'DATATYPE'
        Size:  1x600000
        MaxSize:  1xInf
        Datatype:   H5T_IEEE_F64LE (double)
        ChunkSize:  1x1
        Filters:  none
        FillValue:  0.000000
    Dataset 'SNR'
        Size:  1x600000
        MaxSize:  1xInf
        Datatype:   H5T_IEEE_F64LE (double)
        ChunkSize:  1x1
        Filters:  none
        FillValue:  0.000000
```

Figure No. 1 General Composition of Modulation DataSets

The 4-dimensional blocks sliced at the interval of 256 contain an IQ signal along with its amplitude and phase. The modulation scheme and EbNo of every block is labeled under the groups of ‘DataType’ and ‘SNR’ respectively in the dataset. The table no.1 enlist the all modulation dataSets generated for the project along with their respective attributes.

Table No.1 Summary of Modulation Datasets

S. No.	MATLAB files	DataSet Name	Normalization	fading	Symbol Rate
1	Mod_gen8x32.m	MOD_8Samp1.h5	True	Nil	128
2	Mod_gen16x16.m	MOD_16Samp1.h5	True	Nil	64
3	Mod_gen32x8.m	MOD_32Samp1.h5	True	Nil	32
4	RayleighFadded_Mod_gen16x16.m	MOD_Rayleighfadded_16Samp_N1.h5	True	Rayleigh	64
5	Rician_0p3_Fadded_Mod_gen16x16.m	MOD_Rician0p3fadded_16Samp_N1.h5	True	0.3 Rician	64
6	Rician_0p7_Fadded_Mod_gen16x16.m	MOD_Rician0p7fadded_16Samp_N1.h5	True	0.7 Rician	64
7	Rician_0p9_Fadded_Mod_gen16x16.m	MOD_Rician0p9fadded_16Samp_N1.h5	True	0.9 Rician	64
8	Mod_gen16x16_NN.m	MOD_16SampNN_1.h5	False	Nil	64
9	RayleighFadded_Mod_gen16x16_NN.m	MOD_Rayleighfadded_16Samp_N_NN1.h5	False	Rayleigh	64
10	Rician_0p3_Fadded_Mod_gen16x16_NN.m	MOD_Rician0p3fadded_16Samp_N_NN1.h5	False	0.3 Rician	64
11	Rician_0p7_Fadded_Mod_gen16x16_NN.m	MOD_Rician0p7fadded_16Samp_N_NN1.h5	False	0.7 Rician	64
12	Rician_0p9_Fadded_Mod_gen16x16_NN.m	MOD_Rician0p9fadded_16Samp_N_NN1.h5	False	0.9 Rician	64

DataSet for Coding Schemes

Channel coding, also known as forward error control coding (FECC), is a process of detecting and correcting bit errors in Digital communication systems. To ensure the data integrity in digital transmission, there are two prominent methods:

- ➔ Forward Error Control Coding (FECC)
- ➔ Automatic Repeat Request (ARQ)

This research project focuses on the identification of different FECCs embedded in the transmission. The popular coding schemes i.e. *LDPC, Turbo and polar codes* are selected and their encoders are simulated in the MATLAB. Following are the salient features of data generation for different coding schemes:

LDPC

LDPC are linear error-correcting block codes also known as sparse parity check matrix, suitable for error correction in large block sizes transmitted via very noisy channels. Consider a parity based code that operates on a block on **N bits**. Out of the **N bits** in the block, **M bits** carry data and **C bits** carry parity. Thus, code rate would be

$$Coderate = M/N$$

The classical approach is selected in generating the **LDPC Encoded Message**. Our selected code rates for LDPC $1/2,\ 2/3,\ 3/4,\ 5/6$ with fixed block length $N = 1296$. The LDPC encoder requires the Parity Check Matrix for encoding the message. The selected prototype matrices for each coding rate is given below.

For ½ coding Rate

```
H_1296_1_2 = [40 -1 -1 -1 22 -1 49 23 43 -1 -1 -1 1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1;
              50 1 -1 -1 48 35 -1 -1 13 -1 30 -1 -1 0 0 -1 -1 -1 -1 -1 -1 -1 -1 -1;
              39 50 -1 -1 4 -1 2 -1 -1 -1 -1 49 -1 -1 0 0 -1 -1 -1 -1 -1 -1 -1 -1;
              33 -1 -1 38 37 -1 -1 4 1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1 -1 -1;
              45 -1 -1 -1 0 22 -1 -1 20 42 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1 -1;
              51 -1 -1 48 35 -1 -1 -1 44 -1 18 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1;
              47 11 -1 -1 -1 17 -1 -1 51 -1 -1 -1 0 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1;
              5 -1 25 -1 6 -1 45 -1 13 40 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1;
              33 -1 -1 34 24 -1 -1 -1 23 -1 -1 46 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1;
              1 -1 27 -1 1 -1 -1 -1 38 -1 44 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1;
              -1 18 -1 -1 23 -1 -1 8 0 35 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0;
              49 -1 17 -1 30 -1 -1 -1 34 -1 -1 19 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0];
```

For ⅔ coding Rate

```
H_1296_2_3 = [39 31 22 43 -1 40 4 -1 11 -1 -1 50 -1 -1 -1 6 1 0 -1 -1 -1 -1 -1 -1;
              25 52 41 2 6 -1 14 -1 34 -1 -1 -1 24 -1 37 -1 -1 0 0 -1 -1 -1 -1 -1;
              43 31 29 0 21 -1 28 -1 -1 2 -1 -1 7 -1 17 -1 -1 -1 0 0 -1 -1 -1 -1;
              20 33 48 -1 4 13 -1 26 -1 -1 22 -1 -1 46 42 -1 -1 -1 -1 0 0 -1 -1 -1;
              45 7 18 51 12 25 -1 -1 -1 50 -1 -1 5 -1 -1 -1 0 -1 -1 -1 0 0 -1 -1;
              35 40 32 16 5 -1 -1 18 -1 -1 43 51 -1 32 -1 -1 -1 -1 -1 -1 -1 0 0 -1;
              9 24 13 22 28 -1 -1 37 -1 -1 25 -1 -1 52 -1 13 -1 -1 -1 -1 -1 -1 0 0;
              32 22 4 21 16 -1 -1 -1 27 28 -1 38 -1 -1 -1 8 1 -1 -1 -1 -1 -1 -1 0];
```

For ¾ coding Rate

```
H_1296_3_4 = [39 40 51 41 3 29 8 36 -1 14 -1 6 -1 33 -1 11 -1 4 1 0 -1 -1 -1 -1;
              48 21 47 9 48 35 51 -1 38 -1 28 -1 34 -1 50 -1 50 -1 -1 0 0 -1 -1 -1;
              30 39 28 42 50 39 5 17 -1 6 -1 18 -1 20 -1 15 -1 40 -1 -1 0 0 -1 -1;
              29 0 1 43 36 30 47 -1 49 -1 47 -1 3 -1 35 -1 34 -1 0 -1 -1 0 0 -1;
              1 32 11 23 10 44 12 7 -1 48 -1 4 -1 9 -1 17 -1 16 -1 -1 -1 -1 0 0;
              13 7 15 47 23 16 47 -1 43 -1 29 -1 52 -1 2 -1 53 -1 1 -1 -1 -1 -1 0];
```

For ⅚ coding Rate

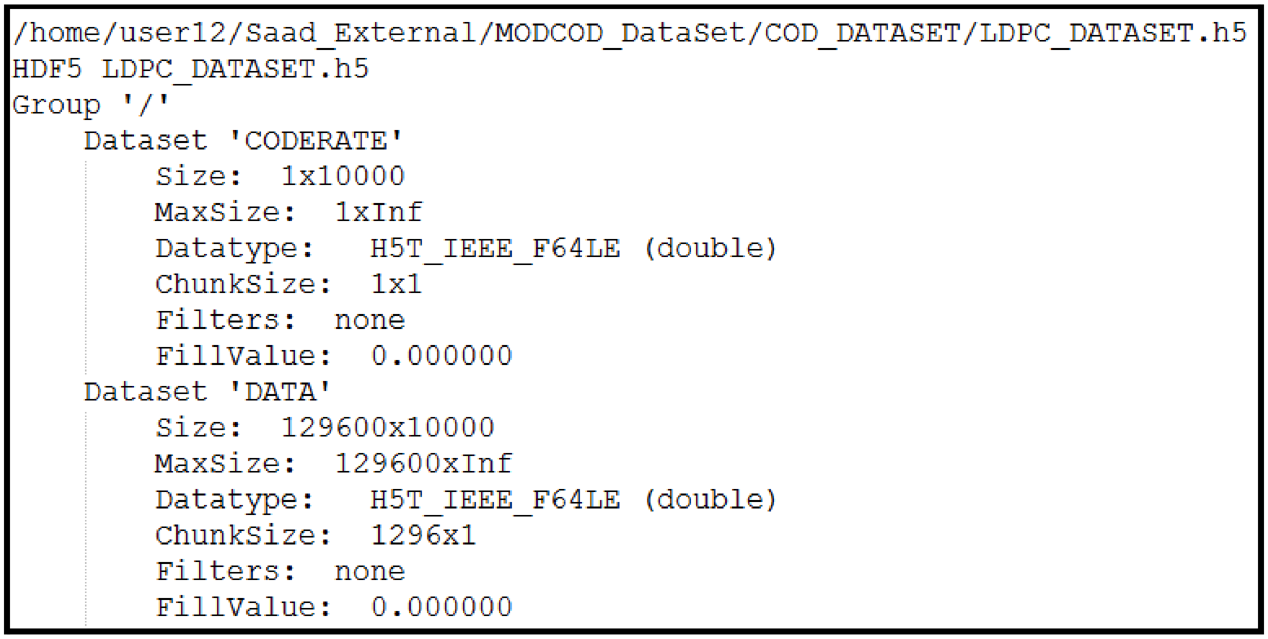
```
H_1296_5_6 = [48 29 37 52 2 16 6 14 53 31 34 5 18 42 53 31 45 -1 46 52 1 0 -1 -1;
              17 4 30 7 43 11 24 6 14 21 6 39 17 40 47 7 15 41 19 -1 -1 0 0 -1;
              7 2 51 31 46 23 16 11 53 40 10 7 46 53 33 35 -1 25 35 38 0 -1 0 0;
              19 48 41 1 10 7 36 47 5 29 52 52 31 10 26 6 3 2 -1 51 1 -1 -1 0 ];
```

The QC-LDPC is employed to generate the Parity Check Matrix from the prototype matrices. The built-in QC-LDPC function in tavildar's repository[2] has been utilized. For each coderate the Payload Size would be given as shown in table no. 2.

Table No. 2 composition of LPDC encoded message

Rate	**Payload Length**	**N**
1/2	648	1296
2/3	864	1296
3/4	972	1296
5/6	1080	1296

The LDPC dataSet Composition is shown in figure no. 2. The 100 LDPC encoded blocks of length 1296 are concatenated to form a mega block. Around 10,000 such mega blocks are created in the dataset and their code rates are labeled under the group “CODERATE”. Each coding rate (½,⅔,¾ & ⅚) has equal representation in the LDPC dataset.



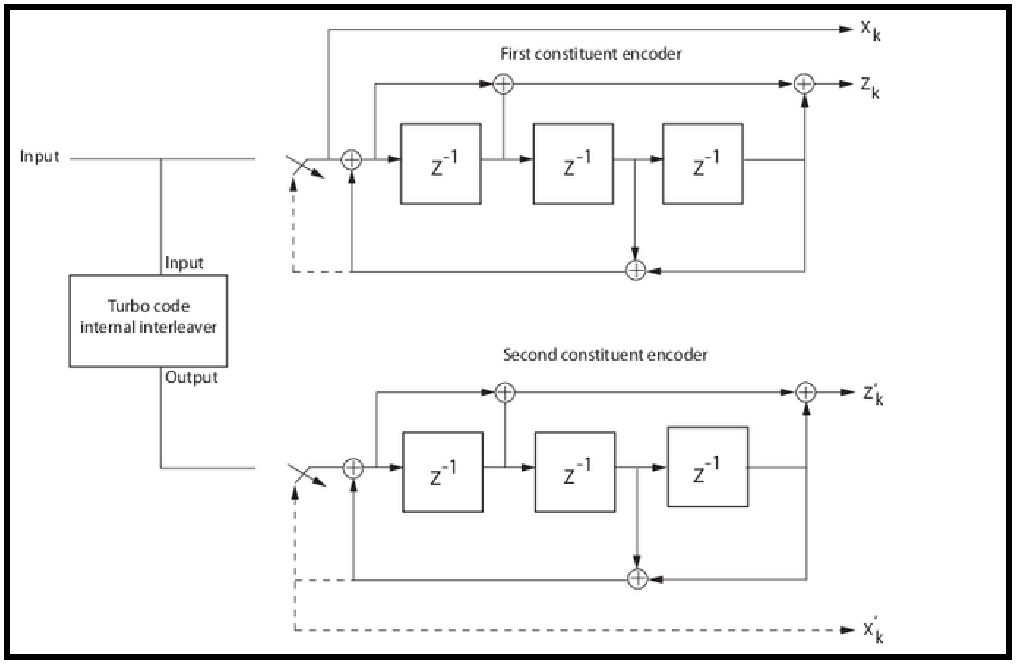
```
/home/user12/Saad_External/MODCOD_DataSet/COD_DATASET/LDPC_DATASET.h5
HDF5 LDPC_DATASET.h5
Group '/'
    Dataset 'CODERATE'
        Size:  1x10000
        MaxSize:  1xInf
        Datatype:   H5T_IEEE_F64LE (double)
        ChunkSize:  1x1
        Filters:  none
        FillValue:  0.000000
    Dataset 'DATA'
        Size:  129600x10000
        MaxSize:  129600xInf
        Datatype:   H5T_IEEE_F64LE (double)
        ChunkSize:  1296x1
        Filters:  none
        FillValue:  0.000000
```

Figure No. 2 Composition of LDPC DataSet

TURBO

Turbo codes were presented back in 1993 and are known as the most successful convolutional codes which have achieved performance very close to the capacity limit. In its common form, turbo encoding is done using two recursive convolutional encoders. The input stream is passed to the first encoder, and a permuted version is passed to the second one.

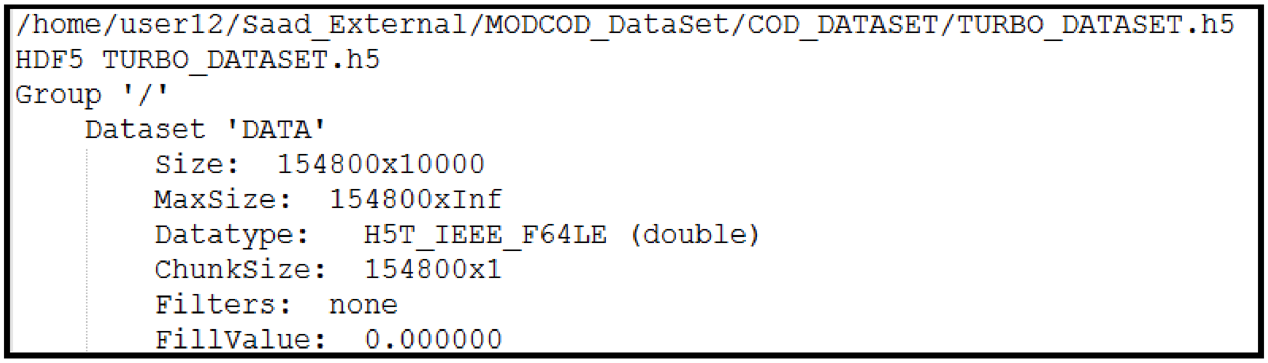
The ⅓ turbo encoder is selected with “*poly2trellis(4,[13 15],13)*” and internal interleaver as shown in figure no. 3.



Input
Input
Turbo code internal interleaver
Output
First constituent encoder
Second constituent encoder
z^{-1}
X_k
Z_k
Z'_k
X'_k

Figure No. 3 The Turbo Encoder Block Diagram[3]

The input message is composed of 512 bits, which resulted in coded output of size 1548. The 1M samples of Turbo encoded blocks were generated for the dataset as shown in figure no. 4.



```
/home/user12/Saad_External/MODCOD_DataSet/COD_DATASET/TURBO_DATASET.h5
HDF5 TURBO_DATASET.h5
Group '/'
    Dataset 'DATA'
        Size:  154800x10000
        MaxSize:  154800xInf
        Datatype:   H5T_IEEE_F64LE (double)
        ChunkSize:  154800x1
        Filters:  none
        FillValue:  0.000000
```

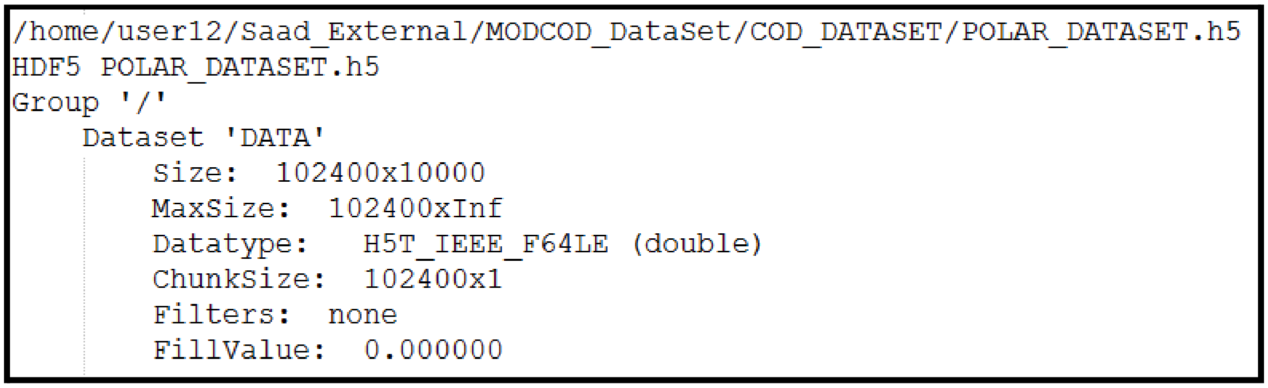
Figure No. 4 Composition of Turbo Dataset

POLAR

A fairly recent type of codes, called polar codes, were introduced by Arıkan in 2008. They are constructed using the channel polarization transform. These codes are error-correcting codes, which are able to achieve the capacity of binary-input memoryless symmetric (BMS) channels. This means that one can transmit at the highest possible rate over that class of channels.
The selected polar encoding is accomplished on the ½ code-rate. The polar encoder[4] is simulated in MATLAB with the following channel specs:

A. N: 1024
B. K: 512
C. n: 10
D. FZlookup: [1024×1 double]
E. Ec: 1
F. N0: 2
G. LLR: [1×2047 double]
H. BITS: [2×1023 double]
I. designSNRdB: 0

The block code is composed of 1024 bits. The 1M samples of polar encoded blocks were generated for the dataset as shown in figure no. 5.

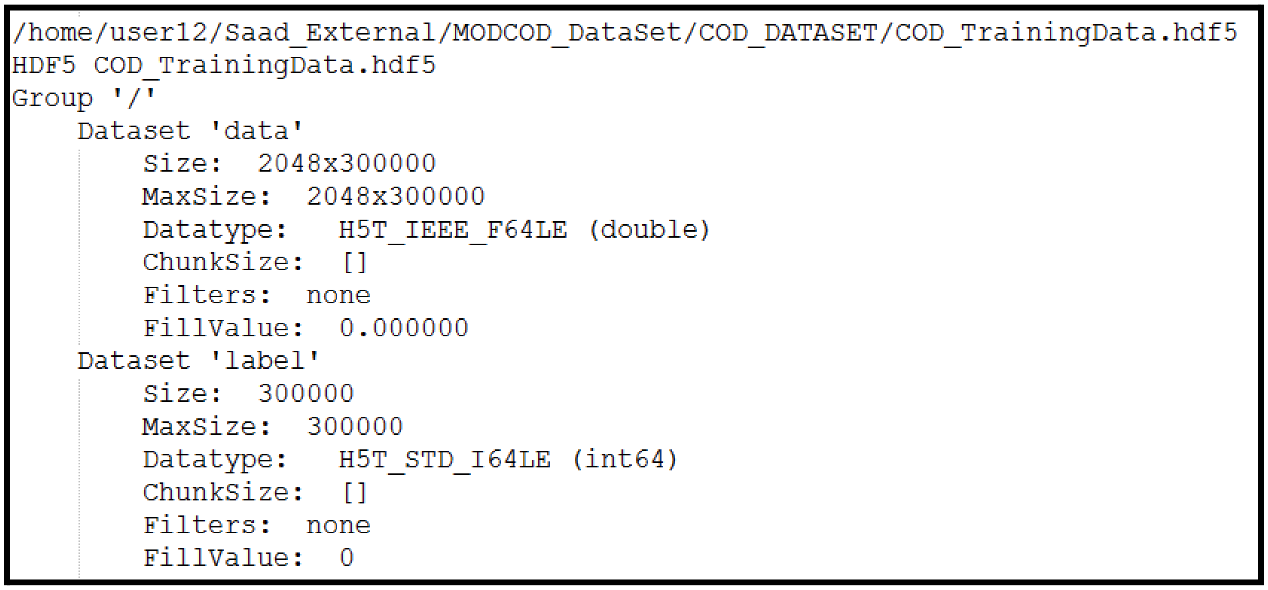


```
/home/user12/Saad_External/MODCOD_DataSet/COD_DATASET/POLAR_DATASET.h5
HDF5 POLAR_DATASET.h5
Group '/'
    Dataset 'DATA'
        Size:  102400x10000
        MaxSize:  102400xInf
        Datatype:   H5T_IEEE_F64LE (double)
        ChunkSize:  102400x1
        Filters:  none
        FillValue:  0.000000
```

Figure No. 5 Composition of the Polar Dataset

Collective Coding DataSet

The datasets of coding scheme i.e. LDPC, Turbo and Polar are required to be further processed for development of the single dataset with block size coherent to the input of the proposed Neural Network. The composition of the shared dataset is shown in figure no. 6. The *0.1 Million blocks of length 2048* for each coding scheme are introduced in the dataset and the label of the coding scheme is marked for each block under the group "label". The dataset is binary in nature containing raw coded messages. Therefore, the input of the NN model would be a binary coded block of length 2048.



```
/home/user12/Saad_External/MODCOD_DataSet/COD_DATASET/COD_TrainingData.hdf5
HDF5 COD_TrainingData.hdf5
Group '/'
    Dataset 'data'
        Size:  2048x300000
        MaxSize:  2048x300000
        Datatype:   H5T_IEEE_F64LE (double)
        ChunkSize:  []
        Filters:  none
        FillValue:  0.000000
    Dataset 'label'
        Size:  300000
        MaxSize:  300000
        Datatype:   H5T_STD_I64LE (int64)
        ChunkSize:  []
        Filters:  none
        FillValue:  0
```

Figure No. 6 Shared Dataset for Coding Scheme

Neural Network Model

Artificial Neural Networks (ANNs) mimic the human brain through a set of algorithms. At a basic level, a neural network consists of four main components:

- ➔ Inputs (x)
- ➔ Weights (w)
- ➔ Bias or threshold ($Bias$)
- ➔ Output (Y)

The algebraic formula would look something like this:

$$Y = w_1 x_1 + w_2 x_2 + w_3 x_3 + Bias$$

1-D convolution Neural Network (CNN) is proposed for the Modulation and Coding scheme classification. CNNs are a class of Artificial Neural Networks that can recognize and classify particular features from pictorial data and are widely acceptable for analyzing visual images. Recently, the 1-D CNN was introduced to fill the gap of incorporating CNN in the applications where the input stream is either text or 1D i.e. ECG, EEG, speech, music, RF signals etc.

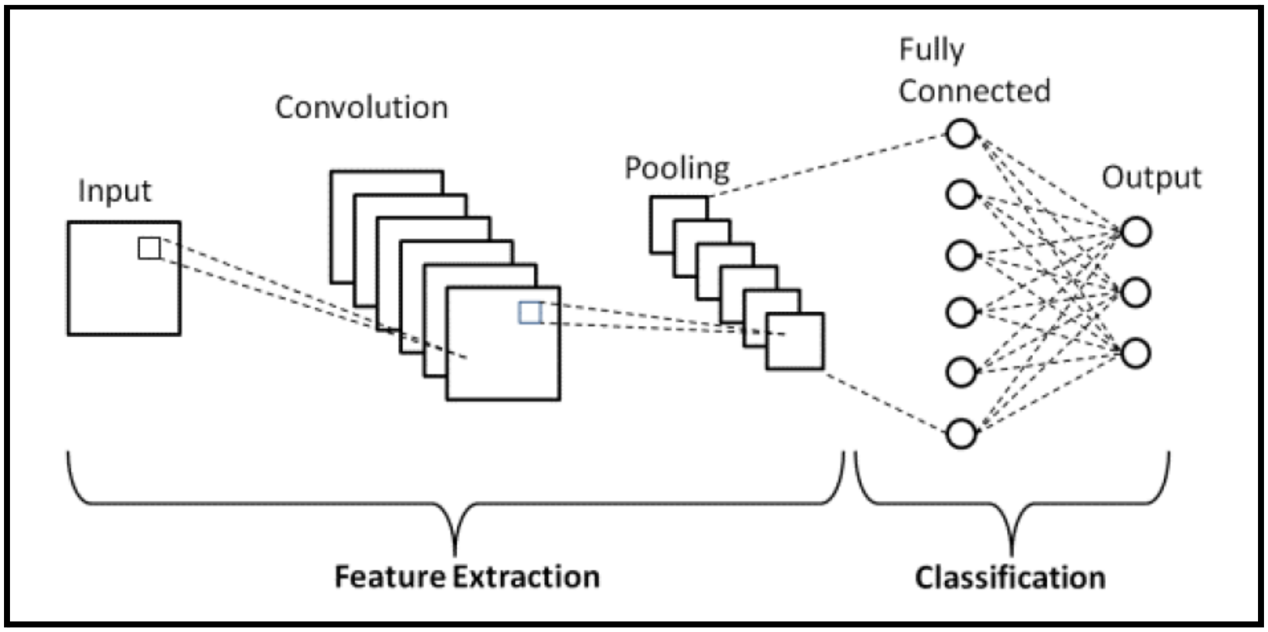


Figure No. 7 Block Diagram of Convolutional Neural Network (CNN)

Basics of Convolution Neural Network

The basic architecture of the CNN consists of two parts i.e. convolution tool and fully connected layers. A convolution tool separates and identifies the multiple features of the input for analysis in a process called *Feature Extraction*. A fully connected layer utilizes the output from the convolution process and *predicts the class of the input* based on the features extracted in previous stages as shown in figure no. 7. The basic building block of ConvNets is convolutional Layer.

Convolutional layers are categorized into three main classes i.e. Convolutional Layers, Pooling Layers and Fully-Connected Layers. When these layers are stacked, a CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function. A little discussion for understanding of the reader is given below:

Convolutional Layer

Convolutional Layer is the first layer that performs the extraction of the various features from the pictorial input. In this layer, the convolution is performed between the input image and a filter of a predefined size MxM.

Polling Layer

Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. The Max Polling and Average Polling are the common pooling layers. It acts as a bridge between the Convolutional Layer and the FC Layer.

Fully Connected Layer

The Fully Connected (FC) layers comprises the weights and biases along with the neurons and is used to connect the neurons between two different layers. At this stage, all the inputs from the successive layer are mapped at the activation unit of the next layer. These layers are placed before the output layer of a CNN Architecture and are responsible for the classification process.

Dropout Layer

when all the features are connected to the FC layer, it would cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on new data. To mitigate this problem, a dropout layer is utilized wherein a few neurons are dropped from the neural network during the training process resulting in reduced size of the model.

Activation Function Layer

The activation function layers play the most important role in the CNN architecture. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. Commonly known activation functions are the ReLU, Softmax, tanH and the Sigmoid functions.

How do Conv1D and Conv2D in CCN differ from each other?

2-Dimensional CNN | Conv2D is the standard Convolution Neural Network which was first introduced in Lenet-5 architecture. Conv2D is generally used on Image data. It is called 2 dimensional CNN because the kernel slides along 2 dimensions on the data. However, **1-Dimensional CNN | Conv1D** is the Convolution Neural Network for the time-series data applications. The kernel slides along one dimension on the data in the Conv1D as shown in figure no. 8.

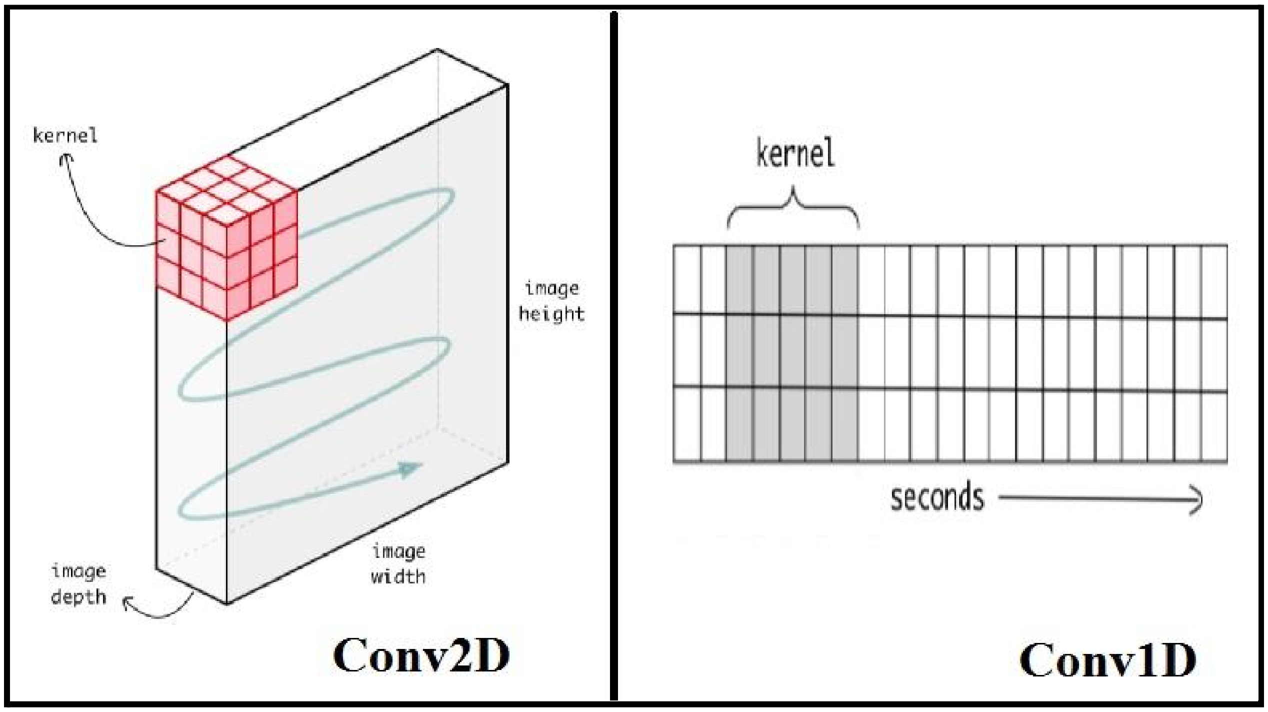


Figure No. 8 Kernel movement in Conv2D and Conv1D

Salient Features of Proposed Neural Network

The major characteristics of proposed Conv1D model devised for classification of modulation and coding schemes are provided below:

Software tools & library

The whole development and designing of the NN model for the MODCOD project is accomplished in Python language and Jupyter Notebook is selected as an IDE. The multiple in-built and data-science libraries are used to get the task done and few prominent among them are Numpy, H5py, Matplotlib, Scipy, Sklearn, etc.
The free and open-source Keras API is used to implement the 1D-CNN model running on top of TensorFlow which provides a modular approach to create deep networks. Following are the sub-libraries of Tensorflow incorporated for creating the proposed model for MODCOD project:

```
from tensorflow.keras.layers                  importReshape, Flatten, Dense
from tensorflow.keras.layers                  importConv1D ,Conv2D, MaxPooling2D
from tensorflow.keras.layers                  importMaxPooling1D,Dropout
from tensorflow.keras.models                  importModel, Sequential
from tensorflow.keras.optimizers              importSGD, Adam, Adamax
from tensorflow.keras.preprocessing.text      importTokenizer
from tensorflow.keras.preprocessing.sequence  importpad_sequences
```

Model's Input

The input of convolution neural networks (CNNs) is always in the form of a block and stricted to be of fixed shape due to its inherent architectural properties. The proposed model is Conv1D in nature. Therefore, the model has fixed shape input and its input is in the form of block rather than stream.
The NN model proposed to classify the six digital modulation schemes has the input shape of 256x4. The input block is 256 composed of IQ signal samples along with its phase and amplitude. Similarly, the NN model for the classification of coding schemes has the input shape of 2048x1. As stated earlier, the block lengths of LDPC, Turbo and Polar are 1296, 1548 and 1024 respectively. Therefore, the model's input is composed of more than one encoded binary message.

Convolutional layers

The proposed NN model is rigorously evaluated with varying the amount of Conv1D layers. The main purpose of the convolutional layer is to extract features from the block input. By increasing the layers in the NN model, the fine grained features can be extracted at the expense of computation complexity. Moreover, NN models encounter the vanishing gradient problem as the depth of the model is raised from the appropriate level.

The Proposed NN model for the classification of modulation scheme is evaluated with 2,3 and 4 Conv1D layers. Similarly, the model for the classification of coding schemes is thoroughly assessed with 3, 4 and 5 Conv1D layers.

Activation function

In general terms, an *Activation Function* decides whether a specific group of neurons in the layer should be activated or not. In the proposed neural network, the activation function is applied after every cascaded convolutional layer. The multiple intuitions for betterment of the NN model are tested by incorporating the different types of activation function i.e. relu, linear, sigmoid and tanh. In addition, the proposed model is also evaluated by employing more than one type of activation function in the same model.

Splitting DataSet

It’s common practice to split the dataset before initiating the training sequence of the NN model. The function “train_test_split” is provided in the library *sklearn*, which has been utilized for the splitting of the dataset further into the training and testing dataset. During whole training and testing of the multiple proposed NN models, 80% of the dataset is used for training of the model and the rest of it is utilized for testing of the model in the later stage.

Loss function

The loss function in a neural network quantifies the difference between the expected outcome and the outcome produced by the machine learning model. During training of the NN model, the loss function is torch bearer for the model to align itself in the right direction. It’s desirable to reduce the loss function’s output such that the model starts to predict everything accurately. After the rigorous training, the loss function value started to saturate to some point indicating the training complete or vanishing gradient problem. The “sparse_categorical_crossentropy” loss function is used for the training of the proposed Neural Network models of the MODCOD project.

Optimizer

The optimizers shape and mold neural network model into its most accurate possible form by futzing with the weights by the guidance of loss function. Because, loss function is telling the optimizer when it's moving in the right or wrong direction. Optimizers are related to model accuracy, a key component of AI/ML governance.
Adam optimizer is used during training of the proposed models of the MODCOD project. Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam optimizer is considered the best among the other adaptive optimizers because it is

really efficient when working with large problems involving a lot of data or parameters. Moreover, It requires less memory and is efficient

Epoch

The number of epochs is a hyperparameter that defines the number of times that the learning algorithm will work through the entire training dataset. The number of epochs for the training of the proposed model is selected as 20 for the classification of the modulation schemes. Whereas, it is selected as 10 for the classification of the coding schemes.

Transfer Learning

Transfer Learning is a machine learning method where we reuse a pre-trained model as the starting point for a model on a new task. In short, a neural network model trained on one task is repurposed on a second, related task as an optimization that allows rapid progress when modeling the second task.
Transfer learning is already adapted in various classification solutions where the fine-tuned classification is required with the limitation on dataset. Therefore, the pre-trained related models are used as a starting point for such solutions. Similarly, this approach has been used in the training phase for the fading models. The pre-trained models under the no-fading environment are used as baseline for the fading environments.

Training & Testing of NN

The general Conv1D model is being considered as the most suitable NN model for the classification of modulation and coding scheme. The salient features of the proposed Conv1D are discussed earlier which depict that the rigorous assessment of the NN model is appreciated before coming to any conclusion. For the same purpose, the training and testing of a few proposed Conv1D models will be discussed. Training and Testing of proposed NNs is carried out on the IPT computing machine equipped GPU resources.

The classification of modulation and coding schemes has been done individually. The further discussion is organized under subheadings of "*Conv1D model based classification of modulation schemes*" and "*Conv1D model based classification of coding schemes*".

Conv1D model based classification of modulation schemes

The classification of the modulation scheme has been performed using the classical 1D Convolutional Neural Network (Conv1D). Proposed model is tuned by the results and findings collected from implying the modifications in following areas:

- ➔ **Number of Hidden Layers** i.e. 2, 3 or 4 convolution layers for feature extraction
- ➔ **Activation Function** i.e. linear or nonlinear
- ➔ **Application of transfer function** enable the utilization of pre-trained models

And, Model's performance has been evaluated on the basis of it's agility and accuracy under :

- ➔ **Variation in symbol rate** in transmitted signal i.e. 128, 64, 32 symbols/sec
- ➔ **Variable environments** i.e. fading and non-fading scenarios
- ➔ **Different forms of model's input** i.e normalized and unnormalized input

The summary of ~19 unique Conv1D models for classification of modulation schemes is provided in table no.3. Generated modulation dataset discussed earlier in table no.01 used in the training and testing of these models.

Table No. 3 Summary of the devised Conv1D models for classification of modulation schemes

S. No.	Trained Model Name	Model type	hidden layers	Activation Function	Pre-trained Model Name	Dataset Name
1	Mod_3L_4D_8S32.h5	1D-CNN	3	relu	--	MOD_8Samp1.h5
2	Mod_3L_4D_16S16.h5	1D-CNN	3	relu	--	MOD_16Samp1.h5
3	Mod_3L_4D_32S8.h5	1D-CNN	3	relu	--	MOD_32Samp1.h5
4	Mod_2L_4D_16S1.h5	1D-CNN	2	relu	--	MOD_16Samp1.h5
5	Mod_3L_4D_16S16_TF_Rayleigh_N.h5	1D-CNN	--	--	Mod_3L_4D_16S16.h5	MOD_Rayleighfadded_16Samp_N1.h5
6	Mod_3L_4D_16S16_TF_0p3rician_N.h5	1D-CNN	--	--	Mod_3L_4D_16S16.h5	MOD_Rician0p3fadded_16Samp_N1.h5
7	Mod_3L_4D_16S16_TF_0p7rician_N.h5	1D-CNN	--	--	Mod_3L_4D_16S16.h5	MOD_Rician0p7fadded_16Samp_N1.h5
8	Mod_3L_4D_16S16_TF_0p9rician_N.h5	1D-CNN	--	--	Mod_3L_4D_16S16.h5	MOD_Rician0p9fadded_16Samp_N1.h5
9	Mod_3L_4D_NL_16S16.h5	1D-CNN	3	sigmoid	--	MOD_16SampNN_1.h5
10	Mod_3L_4D_NL_tan_16S16.h5	1D-CNN	3	tanh,relu	--	MOD_16SampNN_1.h5
11	Mod_3L_4D_NLnL_tan_16S16.h5	1D-CNN	3	tanh,linear	--	MOD_16SampNN_1.h5

12	Mod_3L_4D_16S16_TF_NL_Rayleighfadded_N.h5	1D-CNN	--	--	Mod_3L_4D_NL_tan_16S16.h5	MOD_Rayleighfadded_16Samp_N_NN1.h5
13	Mod_3L_4D_16S16_TF_NL_0p3ricianfadded_N	1D-CNN	--	--	Mod_3L_4D_NL_tan_16S16.h5	MOD_Rician0p3fadded_16Samp_N_NN1.h5
14	Mod_3L_4D_16S16_TF_NL_0p7ricianfadded_N.h5	1D-CNN	--	--	Mod_3L_4D_NL_tan_16S16.h5	MOD_Rician0p7fadded_16Samp_N_NN1.h5
15	Mod_3L_4D_16S16_TF_NL_0p9ricianfadded_N.h5	1D-CNN	--	--	Mod_3L_4D_NL_tan_16S16.h5	MOD_Rician0p9fadded_16Samp_N_NN1.h5
16	Mod_3L_4D_16S16_TF_NLnL_Rayleighfadded_N.h5	1D-CNN	--	--	Mod_3L_4D_NLnL_tan_16S16.h5	MOD_Rayleighfadded_16Samp_N_NN1.h5
17	Mod_3L_4D_16S16_TF_NLnL_0p3ricianfadded_N.h5	1D-CNN	--	--	Mod_3L_4D_NLnL_tan_16S16.h5	MOD_Rician0p3fadded_16Samp_N_NN1.h5
18	Mod_3L_4D_16S16_TF_NLnL_0p7ricianfadded_N.h5	1D-CNN	--	--	Mod_3L_4D_NLnL_tan_16S16.h5	MOD_Rician0p7fadded_16Samp_N_NN1.h5
19	Mod_3L_4D_16S16_TF_NLnL_0p9ricianfadded_N.h5	1D-CNN	--	--	Mod_3L_4D_NLnL_tan_16S16.h5	MOD_Rician0p9fadded_16Samp_N_NN1.h5

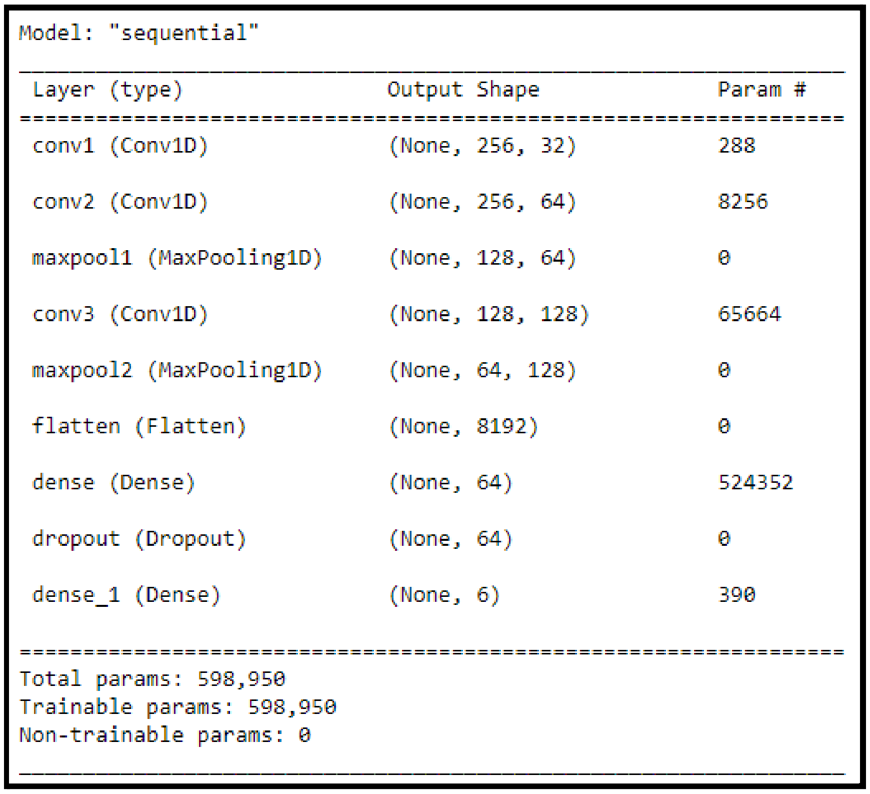
The detailed description of each model is provided below under their respective subheadings. It includes the actual model itself and its critical parameters along with its training and testing results i.e. computation complexity, training accuracy, training loss and the confusion matrix. Reader is encouraged to develop the intuition to evaluate the model best suited for his future research purposes.

Model # 01 “Mod_3L_4D_8S32.h5”

The “Mod_3L_4D_8S32.h5” Conv1D model as shown in figure no. 9 is trained on the dataset “MOD_8Samp1.h5”. Model’s further characteristics are tabulated in the table no. 4.

Table No. 4 Characteristics of Model #01

Model’s Characteristics		**DataSet Attributes**	
Input Layers	03	Symbol Rate	128
Activation Func	**relu** act. func. with all hidden layers except the last decision layer utilizing softmax.	Channel Nature	No fading
Transfer Learning	—	Input format	Normalized & shape 256x4



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1 (Conv1D)              (None, 256, 32)           288

 conv2 (Conv1D)              (None, 256, 64)           8256

 maxpool1 (MaxPooling1D)     (None, 128, 64)           0

 conv3 (Conv1D)              (None, 128, 128)          65664

 maxpool2 (MaxPooling1D)     (None, 64, 128)           0

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 64)                524352

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 6)                 390

=================================================================
Total params: 598,950
Trainable params: 598,950
Non-trainable params: 0
_________________________________________________________________
```

Figure No. 9 The model #01 “**Mod_3L_4D_8S32.h5”**

The model #01's training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 268s 18ms/step - loss: 0.2729 - accuracy: 0.8765 - val_loss: 0.1129 - val_accuracy: 0.9559
Epoch 2/20
15000/15000 [==============================] - 265s 18ms/step - loss: 0.0895 - accuracy: 0.9651 - val_loss: 0.0511 - val_accuracy: 0.9803
Epoch 3/20
15000/15000 [==============================] - 261s 17ms/step - loss: 0.0672 - accuracy: 0.9743 - val_loss: 0.0389 - val_accuracy: 0.9847
Epoch 4/20
15000/15000 [==============================] - 262s 17ms/step - loss: 0.0571 - accuracy: 0.9788 - val_loss: 0.0370 - val_accuracy: 0.9866
Epoch 5/20
15000/15000 [==============================] - 261s 17ms/step - loss: 0.0486 - accuracy: 0.9817 - val_loss: 0.0502 - val_accuracy: 0.9803
Epoch 6/20
15000/15000 [==============================] - 265s 18ms/step - loss: 0.0445 - accuracy: 0.9835 - val_loss: 0.0301 - val_accuracy: 0.9886
Epoch 7/20
15000/15000 [==============================] - 265s 18ms/step - loss: 0.0417 - accuracy: 0.9846 - val_loss: 0.0327 - val_accuracy: 0.9876
Epoch 8/20
15000/15000 [==============================] - 266s 18ms/step - loss: 0.0387 - accuracy: 0.9857 - val_loss: 0.0520 - val_accuracy: 0.9832
Epoch 9/20
15000/15000 [==============================] - 267s 18ms/step - loss: 0.0357 - accuracy: 0.9870 - val_loss: 0.0332 - val_accuracy: 0.9872
Epoch 10/20
15000/15000 [==============================] - 265s 18ms/step - loss: 0.0344 - accuracy: 0.9875 - val_loss: 0.0352 - val_accuracy: 0.9872
Epoch 11/20
15000/15000 [==============================] - 268s 18ms/step - loss: 0.0324 - accuracy: 0.9884 - val_loss: 0.0416 - val_accuracy: 0.9854
Epoch 12/20
15000/15000 [==============================] - 265s 18ms/step - loss: 0.0318 - accuracy: 0.9886 - val_loss: 0.0328 - val_accuracy: 0.9881
Epoch 13/20
15000/15000 [==============================] - 266s 18ms/step - loss: 0.0310 - accuracy: 0.9889 - val_loss: 0.0328 - val_accuracy: 0.9876
Epoch 14/20
15000/15000 [==============================] - 269s 18ms/step - loss: 0.0295 - accuracy: 0.9894 - val_loss: 0.0340 - val_accuracy: 0.9889
Epoch 15/20
15000/15000 [==============================] - 266s 18ms/step - loss: 0.0289 - accuracy: 0.9897 - val_loss: 0.0246 - val_accuracy: 0.9908
Epoch 16/20
15000/15000 [==============================] - 269s 18ms/step - loss: 0.0290 - accuracy: 0.9896 - val_loss: 0.0293 - val_accuracy: 0.9897
Epoch 17/20
15000/15000 [==============================] - 268s 18ms/step - loss: 0.0278 - accuracy: 0.9902 - val_loss: 0.0311 - val_accuracy: 0.9888
Epoch 18/20
15000/15000 [==============================] - 267s 18ms/step - loss: 0.0280 - accuracy: 0.9902 - val_loss: 0.0351 - val_accuracy: 0.9893
Epoch 19/20
15000/15000 [==============================] - 267s 18ms/step - loss: 0.0269 - accuracy: 0.9905 - val_loss: 0.0236 - val_accuracy: 0.9919
Epoch 20/20
15000/15000 [==============================] - 267s 18ms/step - loss: 0.0272 - accuracy: 0.9904 - val_loss: 0.0333 - val_accuracy: 0.9876
```

The "training & validation" loss and accuracy of model#01 is shown in figure no. 10 and 11 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 12.

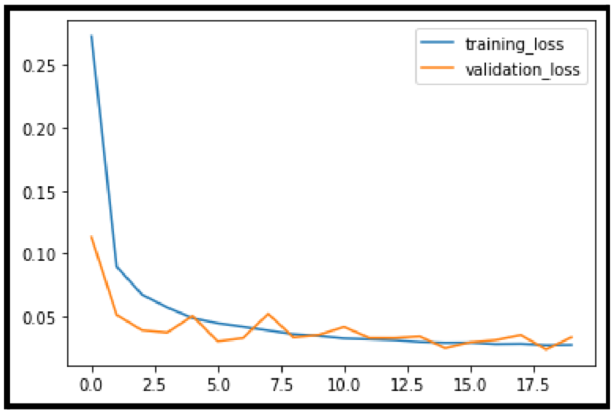


Figure No. 10 Training and validation loss of Model #01

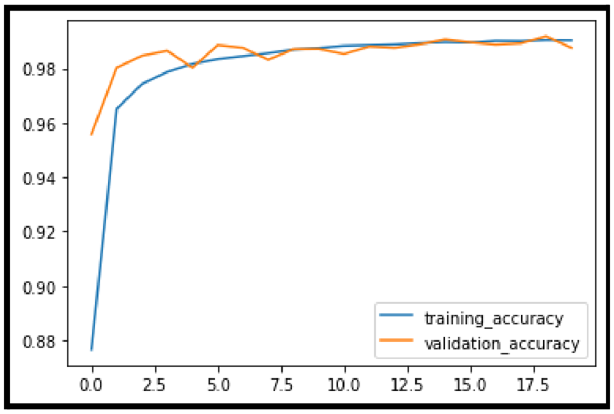
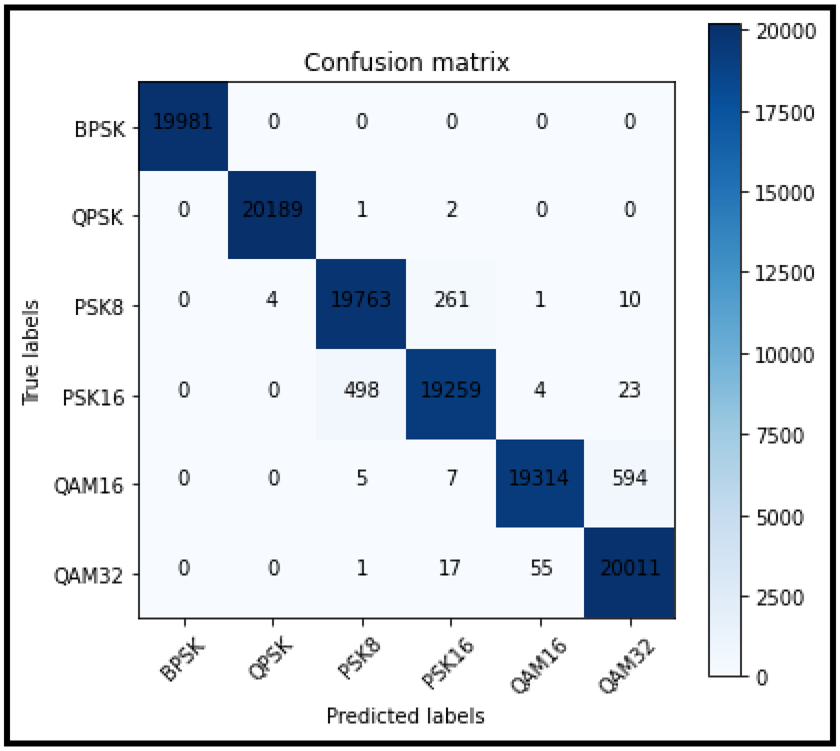


Figure No. 11 Training and validation accuracy of Model #01



	BPSK	QPSK	PSK8	PSK16	QAM16	QAM32
BPSK	19981	0	0	0	0	0
QPSK	0	20189	1	2	0	0
PSK8	0	4	19763	261	1	10
PSK16	0	0	498	19259	4	23
QAM16	0	0	5	7	19314	594
QAM32	0	0	1	17	55	20011

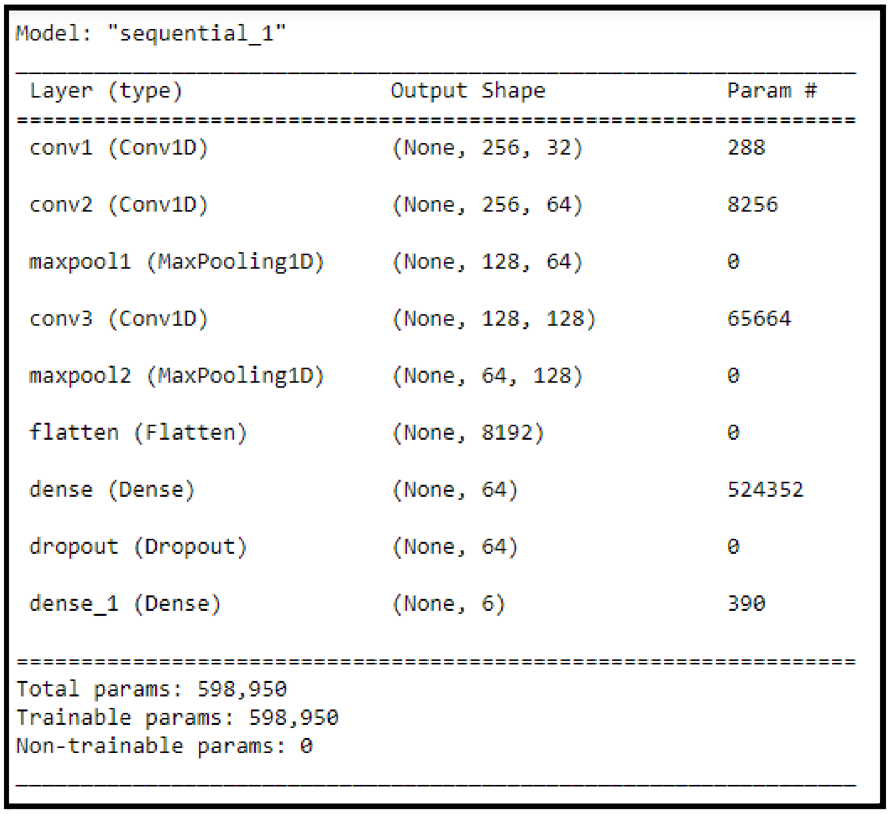
Figure No. 12 Performance result of model #01 is **98.76%** on testing dataset

Model # 02 "Mod_3L_4D_16S16.h5"

The "Mod_3L_4D_16S16.h5" Conv1D model as shown in figure no. 13 is trained on the dataset "MOD_16Samp1.h5". Model's further characteristics are tabulated in the table no. 5.

Table No. 5 Characteristics of Model #02

Model's Characteristics		**DataSet Attributes**	
Input Layers	03	Symbol Rate	64
Activation Func	**relu** act. func. with all hidden layers except the last decision layer utilizing softmax.	Channel Nature	No fading
Transfer Learning	—	Input format	Normalized & shape 256x4



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1 (Conv1D)              (None, 256, 32)           288

 conv2 (Conv1D)              (None, 256, 64)           8256

 maxpool1 (MaxPooling1D)     (None, 128, 64)           0

 conv3 (Conv1D)              (None, 128, 128)          65664

 maxpool2 (MaxPooling1D)     (None, 64, 128)           0

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 64)                524352

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 6)                 390

=================================================================
Total params: 598,950
Trainable params: 598,950
Non-trainable params: 0
_________________________________________________________________
```

Figure No. 13 The model #02 "Mod_3L_4D_16S16.h5"

The model #02's training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - ETA: 0s - loss: 0.2824 - accuracy: 0.8817
15000/15000 [==============================] - 238s 16ms/step - loss: 0.2824 - accuracy: 0.8817 - val_loss: 0.0849 - val_accuracy: 0.9691
Epoch 2/20
15000/15000 [==============================] - 241s 16ms/step - loss: 0.0989 - accuracy: 0.9646 - val_loss: 0.0531 - val_accuracy: 0.9806
Epoch 3/20
15000/15000 [==============================] - 244s 16ms/step - loss: 0.0721 - accuracy: 0.9746 - val_loss: 0.0474 - val_accuracy: 0.9830
Epoch 4/20
15000/15000 [==============================] - 244s 16ms/step - loss: 0.0581 - accuracy: 0.9799 - val_loss: 0.0336 - val_accuracy: 0.9881
Epoch 5/20
15000/15000 [==============================] - 244s 16ms/step - loss: 0.0506 - accuracy: 0.9828 - val_loss: 0.0344 - val_accuracy: 0.9890
Epoch 6/20
15000/15000 [==============================] - 239s 16ms/step - loss: 0.0453 - accuracy: 0.9844 - val_loss: 0.0360 - val_accuracy: 0.9883
Epoch 7/20
15000/15000 [==============================] - 239s 16ms/step - loss: 0.0416 - accuracy: 0.9863 - val_loss: 0.0326 - val_accuracy: 0.9891
Epoch 8/20
15000/15000 [==============================] - 242s 16ms/step - loss: 0.0391 - accuracy: 0.9871 - val_loss: 0.0315 - val_accuracy: 0.9895
Epoch 9/20
15000/15000 [==============================] - 242s 16ms/step - loss: 0.0381 - accuracy: 0.9875 - val_loss: 0.0290 - val_accuracy: 0.9907
Epoch 10/20
15000/15000 [==============================] - 241s 16ms/step - loss: 0.0363 - accuracy: 0.9880 - val_loss: 0.0649 - val_accuracy: 0.9811
Epoch 11/20
15000/15000 [==============================] - 247s 16ms/step - loss: 0.0356 - accuracy: 0.9884 - val_loss: 0.0282 - val_accuracy: 0.9911
Epoch 12/20
15000/15000 [==============================] - 247s 16ms/step - loss: 0.0336 - accuracy: 0.9890 - val_loss: 0.0311 - val_accuracy: 0.9906
Epoch 13/20
15000/15000 [==============================] - 247s 16ms/step - loss: 0.0326 - accuracy: 0.9893 - val_loss: 0.0289 - val_accuracy: 0.9910
Epoch 14/20
15000/15000 [==============================] - 246s 16ms/step - loss: 0.0325 - accuracy: 0.9895 - val_loss: 0.0280 - val_accuracy: 0.9910
Epoch 15/20
15000/15000 [==============================] - 248s 17ms/step - loss: 0.0312 - accuracy: 0.9899 - val_loss: 0.0338 - val_accuracy: 0.9897
Epoch 16/20
15000/15000 [==============================] - 247s 16ms/step - loss: 0.0313 - accuracy: 0.9900 - val_loss: 0.0311 - val_accuracy: 0.9911
Epoch 17/20
15000/15000 [==============================] - 246s 16ms/step - loss: 0.0313 - accuracy: 0.9901 - val_loss: 0.0297 - val_accuracy: 0.9913
Epoch 18/20
15000/15000 [==============================] - 246s 16ms/step - loss: 0.0306 - accuracy: 0.9903 - val_loss: 0.0349 - val_accuracy: 0.9894
Epoch 19/20
15000/15000 [==============================] - 246s 16ms/step - loss: 0.0295 - accuracy: 0.9907 - val_loss: 0.0272 - val_accuracy: 0.9916
Epoch 20/20
15000/15000 [==============================] - 245s 16ms/step - loss: 0.0297 - accuracy: 0.9908 - val_loss: 0.0379 - val_accuracy: 0.9903
```

The "training & validation" loss and accuracy of model#02 is shown in figure no. 14 and 15 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 16.

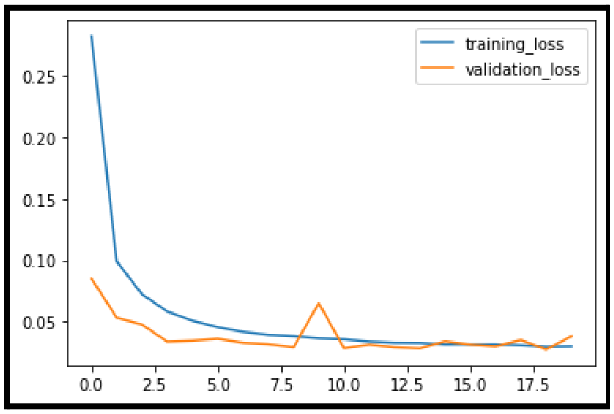


Figure No. 14 Training and validation loss of Model #02

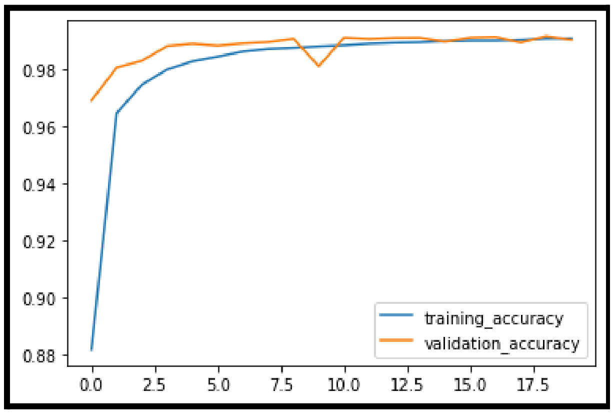


Figure No. 15 Training and validation accuracy of Model #02

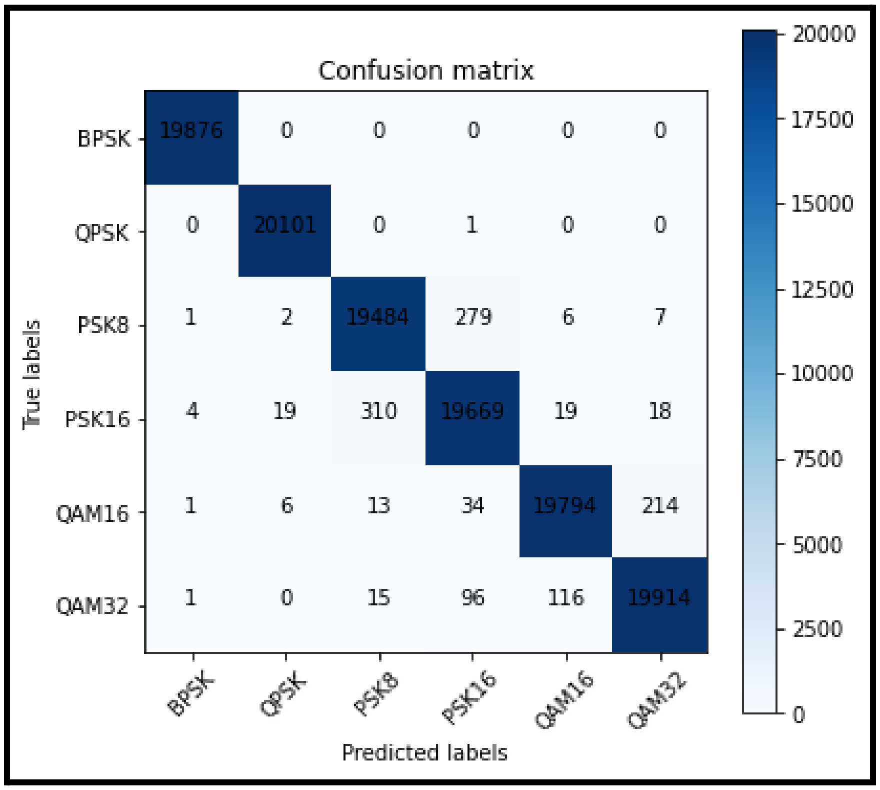


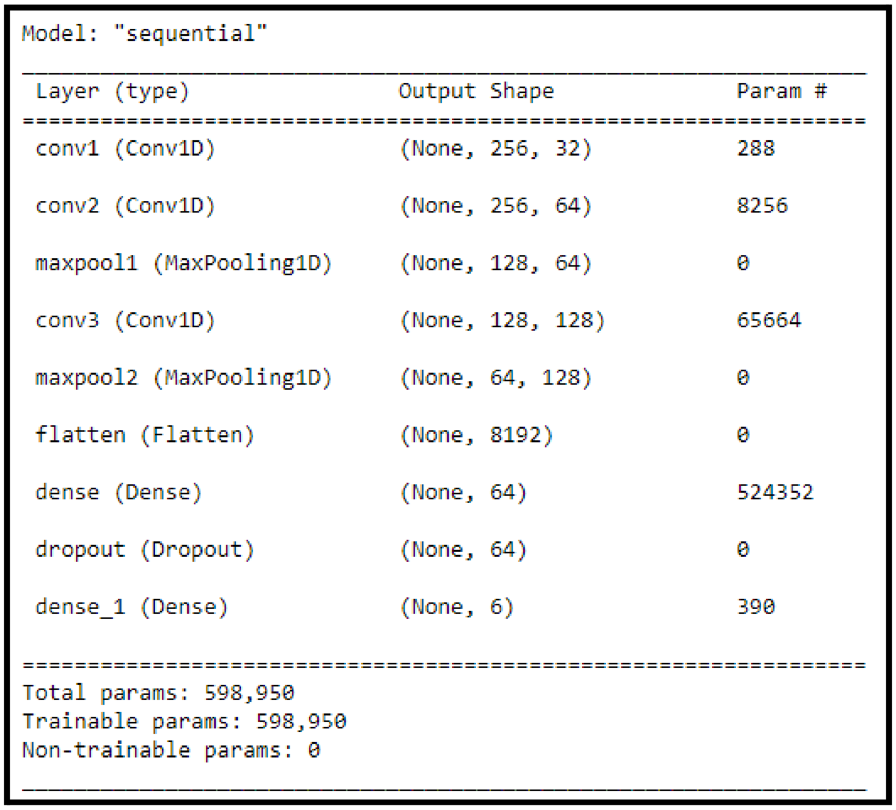
Figure No. 16 Performance result of model #02 is **99.03%** on testing dataset

Model # 03 “Mod_3L_4D_32S8.h5”

The “Mod_3L_4D_32S8.h5” Conv1D model is shown in figure no. 17 as trained on the dataset “MOD_32Samp1.h5”. Model’s further characteristics are tabulated in the table no. 6.

Table No. 6 Characteristics of Model #03

Model’s Characteristics		**DataSet Attributes**	
Input Layers	03	Symbol Rate	32
Activation Func	**relu** act. func. with all hidden layers except the last decision layer utilizing softmax.	Channel Nature	No fading
Transfer Learning	—	Input format	Normalized & shape 256x4



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1 (Conv1D)              (None, 256, 32)           288

 conv2 (Conv1D)              (None, 256, 64)           8256

 maxpool1 (MaxPooling1D)     (None, 128, 64)           0

 conv3 (Conv1D)              (None, 128, 128)          65664

 maxpool2 (MaxPooling1D)     (None, 64, 128)           0

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 64)                524352

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 6)                 390

=================================================================
Total params: 598,950
Trainable params: 598,950
Non-trainable params: 0
_________________________________________________________________
```

Figure No. 17 The model #03 “Mod_3L_4D_32S8.h5”

The model #03's training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 261s 17ms/step - loss: 0.4243 - accuracy: 0.8165 - val_loss: 0.1548 - val_accuracy: 0.9442
Epoch 2/20
15000/15000 [==============================] - 257s 17ms/step - loss: 0.1749 - accuracy: 0.9372 - val_loss: 0.1130 - val_accuracy: 0.9604
Epoch 3/20
15000/15000 [==============================] - 255s 17ms/step - loss: 0.1296 - accuracy: 0.9546 - val_loss: 0.1332 - val_accuracy: 0.9542
Epoch 4/20
15000/15000 [==============================] - 255s 17ms/step - loss: 0.1079 - accuracy: 0.9634 - val_loss: 0.0708 - val_accuracy: 0.9765
Epoch 5/20
15000/15000 [==============================] - 255s 17ms/step - loss: 0.0947 - accuracy: 0.9678 - val_loss: 0.0598 - val_accuracy: 0.9806
Epoch 6/20
15000/15000 [==============================] - 254s 17ms/step - loss: 0.0890 - accuracy: 0.9698 - val_loss: 0.0700 - val_accuracy: 0.9766
Epoch 7/20
15000/15000 [==============================] - 251s 17ms/step - loss: 0.0844 - accuracy: 0.9713 - val_loss: 0.0634 - val_accuracy: 0.9803
Epoch 8/20
15000/15000 [==============================] - 252s 17ms/step - loss: 0.0801 - accuracy: 0.9731 - val_loss: 0.0579 - val_accuracy: 0.9809
Epoch 9/20
15000/15000 [==============================] - 253s 17ms/step - loss: 0.0769 - accuracy: 0.9740 - val_loss: 0.0534 - val_accuracy: 0.9835
Epoch 10/20
15000/15000 [==============================] - 254s 17ms/step - loss: 0.0750 - accuracy: 0.9750 - val_loss: 0.0563 - val_accuracy: 0.9818
Epoch 11/20
15000/15000 [==============================] - 253s 17ms/step - loss: 0.0717 - accuracy: 0.9761 - val_loss: 0.0648 - val_accuracy: 0.9806
Epoch 12/20
15000/15000 [==============================] - 253s 17ms/step - loss: 0.0705 - accuracy: 0.9766 - val_loss: 0.0535 - val_accuracy: 0.9832
Epoch 13/20
15000/15000 [==============================] - 252s 17ms/step - loss: 0.0681 - accuracy: 0.9775 - val_loss: 0.0693 - val_accuracy: 0.9782
Epoch 14/20
15000/15000 [==============================] - 252s 17ms/step - loss: 0.0659 - accuracy: 0.9782 - val_loss: 0.0757 - val_accuracy: 0.9778
Epoch 15/20
15000/15000 [==============================] - 251s 17ms/step - loss: 0.0649 - accuracy: 0.9786 - val_loss: 0.0631 - val_accuracy: 0.9819
Epoch 16/20
15000/15000 [==============================] - 250s 17ms/step - loss: 0.0636 - accuracy: 0.9790 - val_loss: 0.0834 - val_accuracy: 0.9766
Epoch 17/20
15000/15000 [==============================] - 252s 17ms/step - loss: 0.0621 - accuracy: 0.9797 - val_loss: 0.0619 - val_accuracy: 0.9825
Epoch 18/20
15000/15000 [==============================] - 252s 17ms/step - loss: 0.0614 - accuracy: 0.9800 - val_loss: 0.0624 - val_accuracy: 0.9823
Epoch 19/20
15000/15000 [==============================] - 252s 17ms/step - loss: 0.0604 - accuracy: 0.9804 - val_loss: 0.0663 - val_accuracy: 0.9804
Epoch 20/20
15000/15000 [==============================] - 251s 17ms/step - loss: 0.0593 - accuracy: 0.9804 - val_loss: 0.1117 - val_accuracy: 0.9739
```

The "training & validation" loss and accuracy of model#03 is shown in figure no. 18 and 19 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 20.

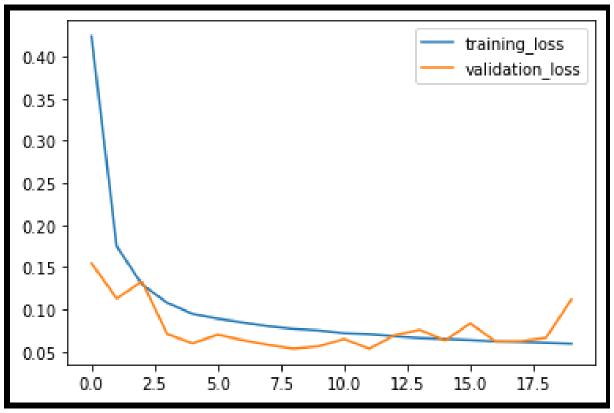


Figure No. 18 Training and validation loss of Model #02

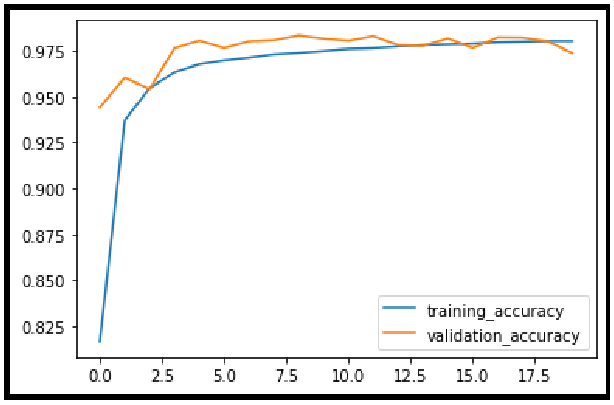
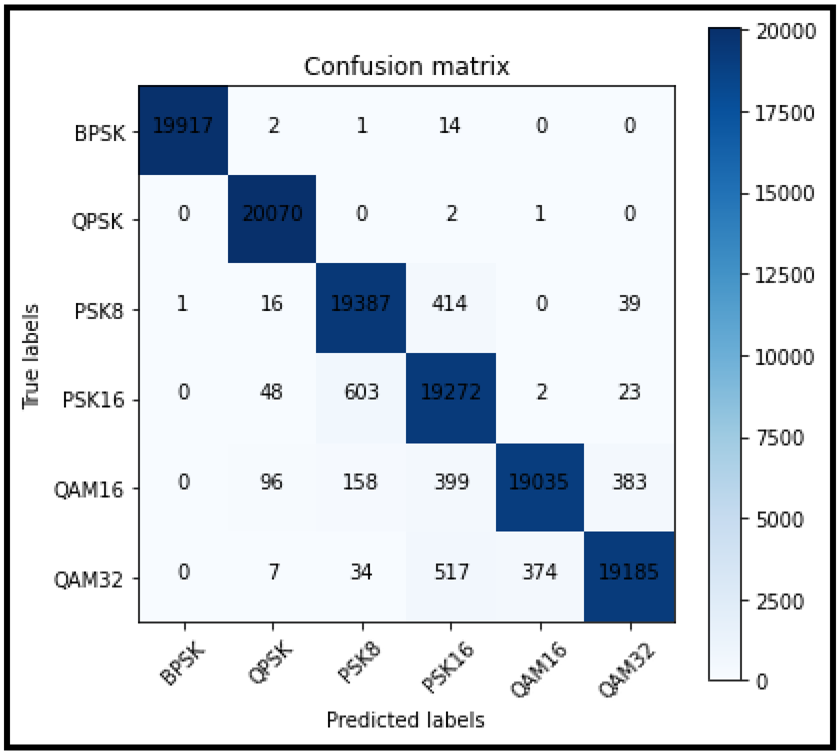


Figure No. 19 Training and validation accuracy of Model #02



True labels \ Predicted labels	BPSK	QPSK	PSK8	PSK16	QAM16	QAM32
BPSK	19917	2	1	14	0	0
QPSK	0	20070	0	2	1	0
PSK8	1	16	19387	414	0	39
PSK16	0	48	603	19272	2	23
QAM16	0	96	158	399	19035	383
QAM32	0	7	34	517	374	19185

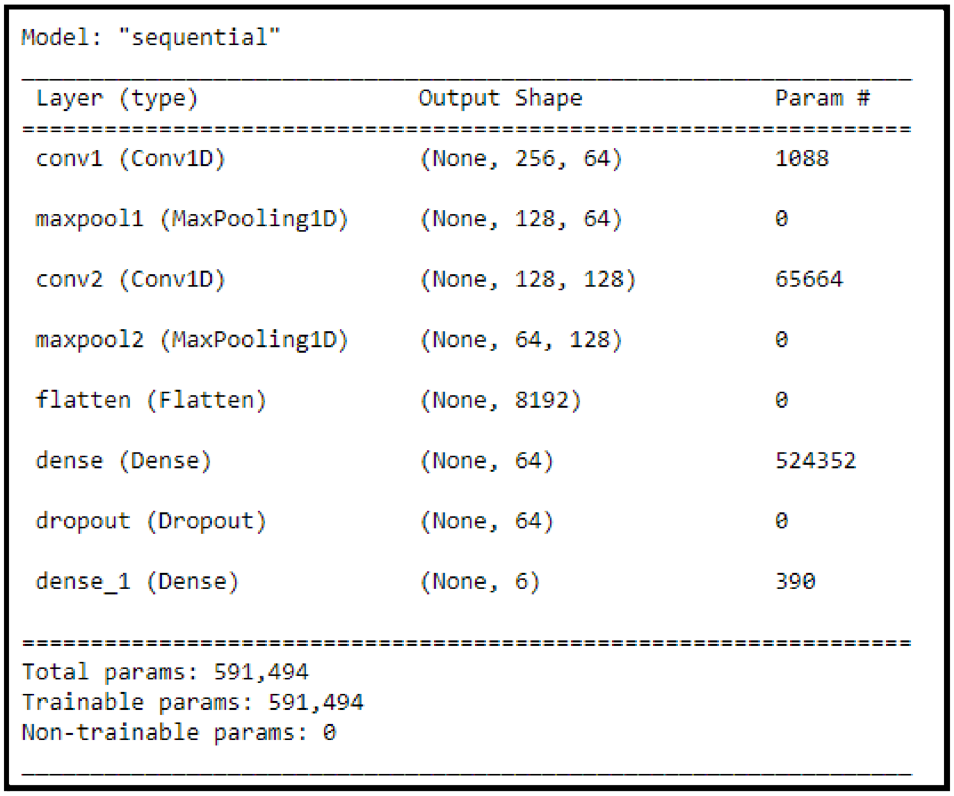
Figure No. 20 Performance result of model #03 is **97.39%** on testing dataset

Model # 04 “Mod_2L_4D_16S1.h5”

The “Mod_2L_4D_16S1.h5” Conv1D model is shown in figure no. 21 as trained on the dataset “MOD_16Samp1.h5”. Model’s further characteristics are tabulated in the table no. 7.

Table No. 7 Characteristics of Model #04

Model’s Characteristics		**DataSet Attributes**	
Input Layers	02	Symbol Rate	64
Activation Func	**relu** act. func. with all hidden layers except the last decision layer utilizing softmax.	Channel Nature	No fading
Transfer Learning	—	Input format	Normalized & shape 256x4



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1 (Conv1D)              (None, 256, 64)           1088

 maxpool1 (MaxPooling1D)     (None, 128, 64)           0

 conv2 (Conv1D)              (None, 128, 128)          65664

 maxpool2 (MaxPooling1D)     (None, 64, 128)           0

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 64)                524352

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 6)                 390

=================================================================
Total params: 591,494
Trainable params: 591,494
Non-trainable params: 0
_________________________________________________________________
```

Figure No. 21 The model #04 “Mod_2L_4D_16S1.h5”

The model #04's training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 225s 15ms/step - loss: 0.3839 - accuracy: 0.8217 - val_loss: 0.1315 - val_accuracy: 0.9469
Epoch 2/20
15000/15000 [==============================] - 229s 15ms/step - loss: 0.1435 - accuracy: 0.9446 - val_loss: 0.0911 - val_accuracy: 0.9651
Epoch 3/20
15000/15000 [==============================] - 230s 15ms/step - loss: 0.1118 - accuracy: 0.9577 - val_loss: 0.0816 - val_accuracy: 0.9691
Epoch 4/20
15000/15000 [==============================] - 231s 15ms/step - loss: 0.0971 - accuracy: 0.9640 - val_loss: 0.0669 - val_accuracy: 0.9758
Epoch 5/20
15000/15000 [==============================] - 235s 16ms/step - loss: 0.0865 - accuracy: 0.9684 - val_loss: 0.0711 - val_accuracy: 0.9738
Epoch 6/20
15000/15000 [==============================] - 232s 15ms/step - loss: 0.0782 - accuracy: 0.9716 - val_loss: 0.0573 - val_accuracy: 0.9795
Epoch 7/20
15000/15000 [==============================] - 235s 16ms/step - loss: 0.0734 - accuracy: 0.9735 - val_loss: 0.0587 - val_accuracy: 0.9792
Epoch 8/20
15000/15000 [==============================] - 237s 16ms/step - loss: 0.0678 - accuracy: 0.9754 - val_loss: 0.0532 - val_accuracy: 0.9804
Epoch 9/20
15000/15000 [==============================] - 236s 16ms/step - loss: 0.0643 - accuracy: 0.9769 - val_loss: 0.0604 - val_accuracy: 0.9779
Epoch 10/20
15000/15000 [==============================] - 238s 16ms/step - loss: 0.0607 - accuracy: 0.9781 - val_loss: 0.0671 - val_accuracy: 0.9767
Epoch 11/20
15000/15000 [==============================] - 239s 16ms/step - loss: 0.0589 - accuracy: 0.9789 - val_loss: 0.0492 - val_accuracy: 0.9830
Epoch 12/20
15000/15000 [==============================] - 240s 16ms/step - loss: 0.0567 - accuracy: 0.9801 - val_loss: 0.0753 - val_accuracy: 0.9746
Epoch 13/20
15000/15000 [==============================] - 242s 16ms/step - loss: 0.0560 - accuracy: 0.9803 - val_loss: 0.0439 - val_accuracy: 0.9849
Epoch 14/20
15000/15000 [==============================] - 243s 16ms/step - loss: 0.0534 - accuracy: 0.9811 - val_loss: 0.0499 - val_accuracy: 0.9824
Epoch 15/20
15000/15000 [==============================] - 241s 16ms/step - loss: 0.0520 - accuracy: 0.9818 - val_loss: 0.0603 - val_accuracy: 0.9796
Epoch 16/20
15000/15000 [==============================] - 240s 16ms/step - loss: 0.0510 - accuracy: 0.9823 - val_loss: 0.0384 - val_accuracy: 0.9865
Epoch 17/20
15000/15000 [==============================] - 243s 16ms/step - loss: 0.0494 - accuracy: 0.9828 - val_loss: 0.0518 - val_accuracy: 0.9825
Epoch 18/20
15000/15000 [==============================] - 245s 16ms/step - loss: 0.0487 - accuracy: 0.9830 - val_loss: 0.0401 - val_accuracy: 0.9864
Epoch 19/20
15000/15000 [==============================] - 243s 16ms/step - loss: 0.0461 - accuracy: 0.9842 - val_loss: 0.0355 - val_accuracy: 0.9881
Epoch 20/20
15000/15000 [==============================] - 242s 16ms/step - loss: 0.0456 - accuracy: 0.9841 - val_loss: 0.0405 - val_accuracy: 0.9866
```

The "training & validation" loss and accuracy of model#04 is shown in figure no. 22 and 23 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 24.

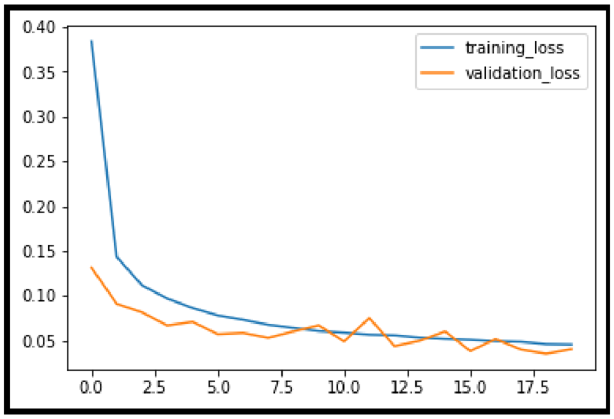


Figure No. 22 Training and validation loss of Model #04

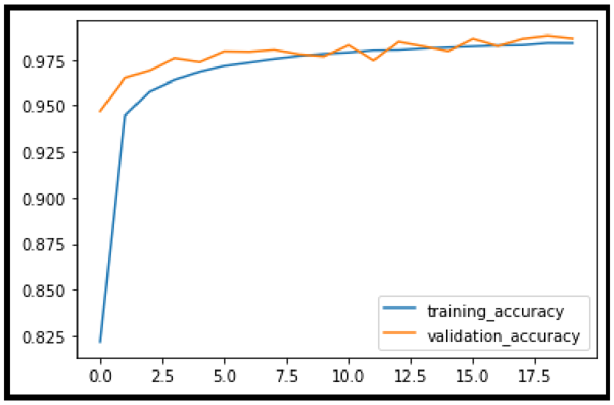


Figure No. 23 Training and validation accuracy of Model #04

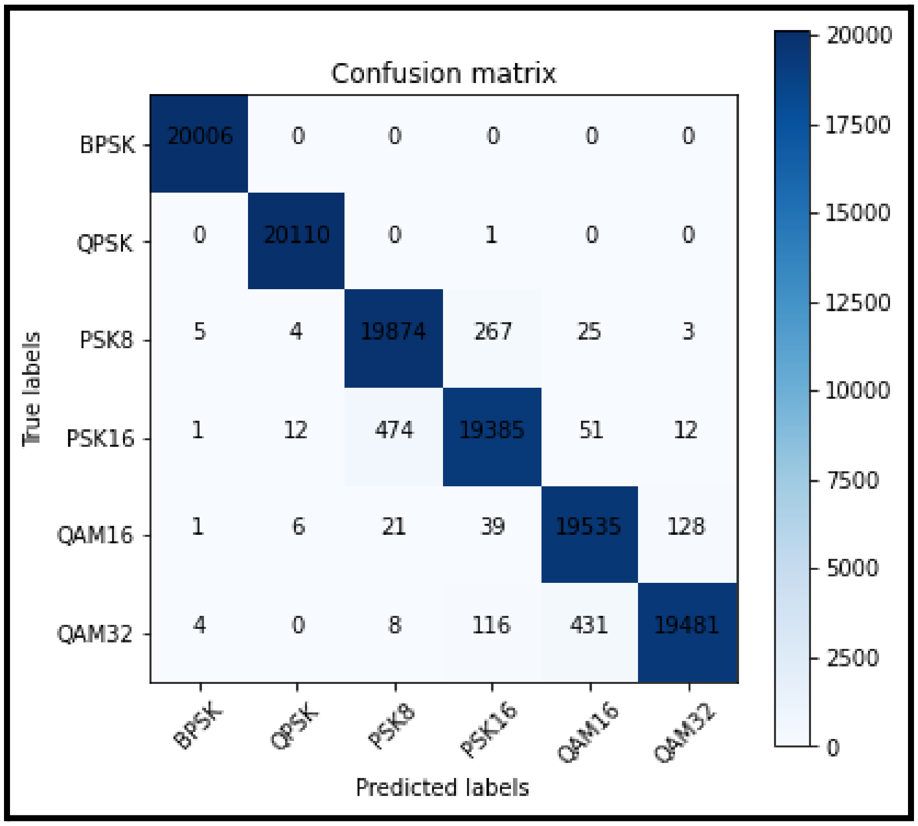


Figure No. 24 Performance result of model #04 is **98.66%** on testing dataset

Model # 05 “Mod_3L_4D_16S16_TF_Rayleigh_N.h5”

The “Mod_3L_4D_16S16_TF_Rayleigh_N.h5” Conv1D model utilized the pre-trained model #No.2 “Mod_3L_4D_16S16.h5” as the starting point and has been trained on the dataset “MOD_Rayleighfadded_16Samp_N1.h5”. Model’s further characteristics are tabulated in the table no. 8.

Table No. 8 Characteristics of Model #05

Model’s Characteristics		**DataSet Attributes**	
Input Layers	Same as model No#2 “Mod_3L_4D_16S16.h5”	Symbol Rate	64
Activation Func	Same as model No#2 “Mod_3L_4D_16S16.h5”	Channel Nature	Rayleigh fading
Transfer Learning	Trained on pre-trained model No#2 “Mod_3L_4D_16S16.h5”	Input format	Normalized & shape 256x4

The model #05’s training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - ETA: 0s - loss: 1.3061 - accuracy: 0.3944
2022-03-17 10:46:35.951018: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 491520000 exceeds 10% of free system memory.

15000/15000 [==============================] - 238s 16ms/step - loss: 1.3061 - accuracy: 0.3944 - val_loss: 1.0856 - val_accuracy: 0.4672
Epoch 2/20
15000/15000 [==============================] - 238s 16ms/step - loss: 1.0968 - accuracy: 0.4625 - val_loss: 1.0388 - val_accuracy: 0.4936
Epoch 3/20
15000/15000 [==============================] - 229s 15ms/step - loss: 1.0649 - accuracy: 0.4832 - val_loss: 1.0279 - val_accuracy: 0.4993
Epoch 4/20
15000/15000 [==============================] - 232s 15ms/step - loss: 1.0525 - accuracy: 0.4894 - val_loss: 1.0142 - val_accuracy: 0.5069
Epoch 5/20
15000/15000 [==============================] - 231s 15ms/step - loss: 1.0422 - accuracy: 0.4947 - val_loss: 1.0247 - val_accuracy: 0.5119
Epoch 6/20
15000/15000 [==============================] - 235s 16ms/step - loss: 1.0344 - accuracy: 0.4997 - val_loss: 1.0061 - val_accuracy: 0.5106
Epoch 7/20
15000/15000 [==============================] - 235s 16ms/step - loss: 1.0289 - accuracy: 0.5036 - val_loss: 1.0053 - val_accuracy: 0.5139
Epoch 8/20
15000/15000 [==============================] - 235s 16ms/step - loss: 1.0239 - accuracy: 0.5063 - val_loss: 0.9936 - val_accuracy: 0.5185
Epoch 9/20
15000/15000 [==============================] - 236s 16ms/step - loss: 1.0196 - accuracy: 0.5083 - val_loss: 0.9898 - val_accuracy: 0.5206
Epoch 10/20
15000/15000 [==============================] - 235s 16ms/step - loss: 1.0160 - accuracy: 0.5109 - val_loss: 0.9882 - val_accuracy: 0.5216
Epoch 11/20
15000/15000 [==============================] - 238s 16ms/step - loss: 1.0139 - accuracy: 0.5121 - val_loss: 0.9912 - val_accuracy: 0.5244
Epoch 12/20
15000/15000 [==============================] - 238s 16ms/step - loss: 1.0118 - accuracy: 0.5142 - val_loss: 0.9838 - val_accuracy: 0.5259
Epoch 13/20
15000/15000 [==============================] - 236s 16ms/step - loss: 1.0086 - accuracy: 0.5153 - val_loss: 0.9952 - val_accuracy: 0.5217
Epoch 14/20
15000/15000 [==============================] - 234s 16ms/step - loss: 1.0069 - accuracy: 0.5165 - val_loss: 0.9898 - val_accuracy: 0.5234
Epoch 15/20
```

```
15000/15000 [==============================] - 231s 15ms/step - loss: 1.0058 - accuracy: 0.5169 - val_loss: 0.9843 - val_accuracy: 0.5278
Epoch 16/20
15000/15000 [==============================] - 231s 15ms/step - loss: 1.0048 - accuracy: 0.5175 - val_loss: 0.9897 - val_accuracy: 0.5214
Epoch 17/20
15000/15000 [==============================] - 234s 16ms/step - loss: 1.0022 - accuracy: 0.5191 - val_loss: 0.9878 - val_accuracy: 0.5288
Epoch 18/20
15000/15000 [==============================] - 233s 16ms/step - loss: 1.0022 - accuracy: 0.5190 - val_loss: 0.9831 - val_accuracy: 0.5303
Epoch 19/20
15000/15000 [==============================] - 232s 15ms/step - loss: 1.0007 - accuracy: 0.5207 - val_loss: 0.9856 - val_accuracy: 0.5238
Epoch 20/20
15000/15000 [==============================] - 230s 15ms/step - loss: 0.9991 - accuracy: 0.5210 - val_loss: 0.9788 - val_accuracy: 0.5283
```

The “training & validation” loss and accuracy of model#05 is shown in figure no. 25 and 26 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 27.

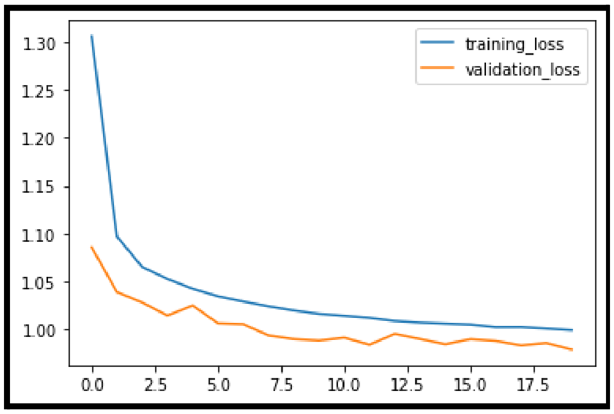


Figure No. 25 Training and validation loss of Model #05

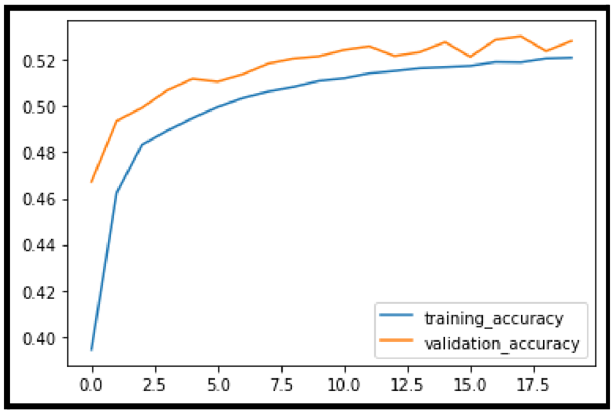


Figure No. 26 Training and validation accuracy of Model #05

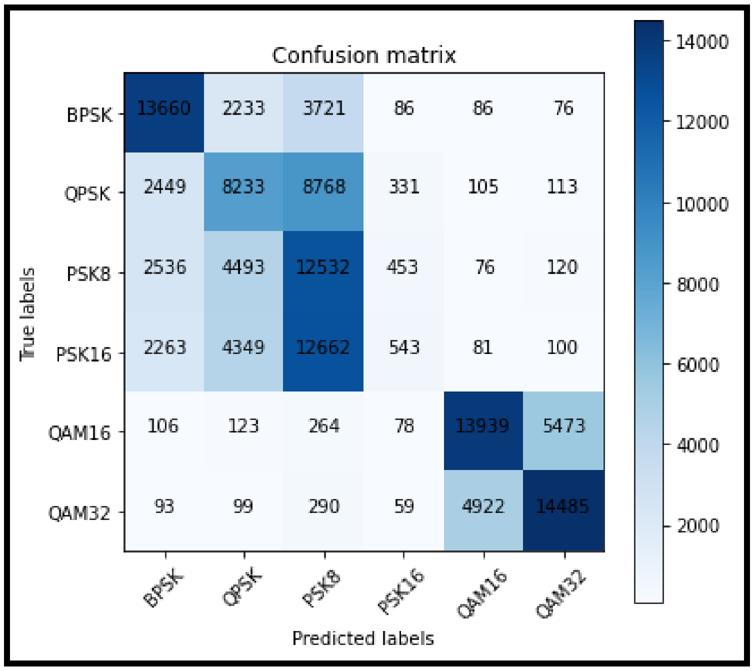


Figure No. 27 Performance result of model #05 is **53.83%** on testing dataset

Model # 06 "Mod_3L_4D_16S16_TF_0p3rician_N.h5"

The "Mod_3L_4D_16S16_TF_0p3rician_N.h5" Conv1D model utilized the pre-trained model #No.2 "Mod_3L_4D_16S16.h5" as the starting point and has been trained on the dataset "MOD_Rician0p3fadded_16Samp_N1.h5". Model's further characteristics are tabulated in the table no. 9.

Table No. 9 Characteristics of Model #06

Model's Characteristics		**DataSet Attributes**	
Input Layers	Same as model No#2 "Mod_3L_4D_16S16.h5"	Symbol Rate	64
Activation Func	Same as model No#2 "Mod_3L_4D_16S16.h5"	Channel Nature	Rician fading with k=0.3
Transfer Learning	Trained on pre-trained model No#2 "Mod_3L_4D_16S16.h5"	Input format	Normalized & shape 256x4

The model #06's training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 242s 16ms/step - loss: 0.8269 - accuracy: 0.5972 - val_loss: 0.5820 - val_accuracy: 0.7002
Epoch 2/20
15000/15000 [==============================] - 262s 17ms/step - loss: 0.5759 - accuracy: 0.7037 - val_loss: 0.5312 - val_accuracy: 0.7147
Epoch 3/20
15000/15000 [==============================] - 272s 18ms/step - loss: 0.5404 - accuracy: 0.7191 - val_loss: 0.5056 - val_accuracy: 0.7273
Epoch 4/20
15000/15000 [==============================] - 267s 18ms/step - loss: 0.5214 - accuracy: 0.7271 - val_loss: 0.5087 - val_accuracy: 0.7313
Epoch 5/20
15000/15000 [==============================] - 267s 18ms/step - loss: 0.5110 - accuracy: 0.7334 - val_loss: 0.5208 - val_accuracy: 0.7294
Epoch 6/20
15000/15000 [==============================] - 274s 18ms/step - loss: 0.5012 - accuracy: 0.7378 - val_loss: 0.4832 - val_accuracy: 0.7400
Epoch 7/20
15000/15000 [==============================] - 277s 18ms/step - loss: 0.4948 - accuracy: 0.7402 - val_loss: 0.4784 - val_accuracy: 0.7456
Epoch 8/20
15000/15000 [==============================] - 275s 18ms/step - loss: 0.4879 - accuracy: 0.7440 - val_loss: 0.4749 - val_accuracy: 0.7490
Epoch 9/20
15000/15000 [==============================] - 281s 19ms/step - loss: 0.4840 - accuracy: 0.7451 - val_loss: 0.4679 - val_accuracy: 0.7487
Epoch 10/20
15000/15000 [==============================] - 282s 19ms/step - loss: 0.4803 - accuracy: 0.7469 - val_loss: 0.4635 - val_accuracy: 0.7501
Epoch 11/20
15000/15000 [==============================] - 287s 19ms/step - loss: 0.4767 - accuracy: 0.7490 - val_loss: 0.4709 - val_accuracy: 0.7506
Epoch 12/20
15000/15000 [==============================] - 291s 19ms/step - loss: 0.4733 - accuracy: 0.7516 - val_loss: 0.4670 - val_accuracy: 0.7539
Epoch 13/20
15000/15000 [==============================] - 281s 19ms/step - loss: 0.4701 - accuracy: 0.7533 - val_loss: 0.4622 - val_accuracy: 0.7496
Epoch 14/20
15000/15000 [==============================] - 277s 18ms/step - loss: 0.4680 - accuracy: 0.7542 - val_loss: 0.4925 - val_accuracy: 0.7429
Epoch 15/20
15000/15000 [==============================] - 285s 19ms/step - loss: 0.4645 - accuracy: 0.7556 - val_loss: 0.4605 - val_accuracy: 0.7548
Epoch 16/20
15000/15000 [==============================] - 269s 18ms/step - loss: 0.4630 - accuracy: 0.7561 - val_loss: 0.4600 - val_accuracy: 0.7572
Epoch 17/20
15000/15000 [==============================] - 255s 17ms/step - loss: 0.4610 - accuracy: 0.7579 - val_loss: 0.4760 - val_accuracy: 0.7528
Epoch 18/20
15000/15000 [==============================] - 260s 17ms/step - loss: 0.4587 - accuracy: 0.7586 - val_loss: 0.4686 - val_accuracy: 0.7562
Epoch 19/20
15000/15000 [==============================] - 247s 16ms/step - loss: 0.4567 - accuracy: 0.7591 - val_loss: 0.4677 - val_accuracy: 0.7552
Epoch 20/20
15000/15000 [==============================] - 249s 17ms/step - loss: 0.4549 - accuracy: 0.7602 - val_loss: 0.4583 - val_accuracy: 0.7545
```

The "training & validation" loss and accuracy of model#06 is shown in figure no. 28 and 29 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 30.

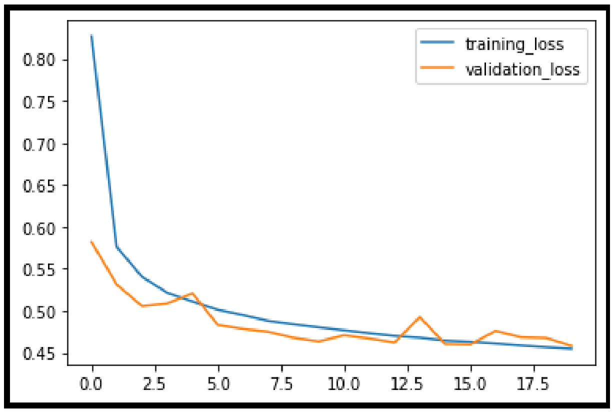


Figure No. 28 Training and validation loss of Model #06

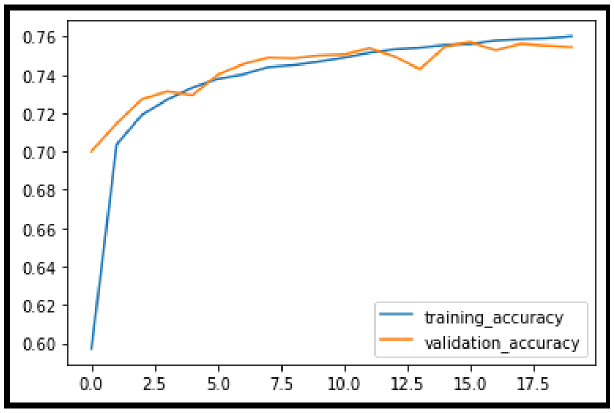
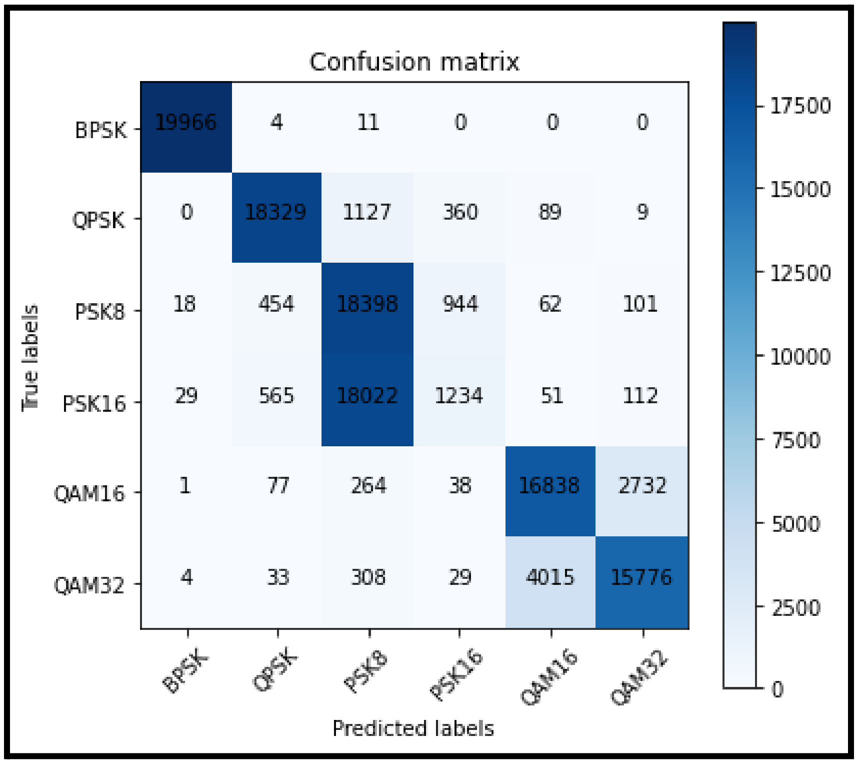


Figure No. 29 Training and validation accuracy of Model #06



	BPSK	QPSK	PSK8	PSK16	QAM16	QAM32
BPSK	19966	4	11	0	0	0
QPSK	0	18329	1127	360	89	9
PSK8	18	454	18398	944	62	101
PSK16	29	565	18022	1234	51	112
QAM16	1	77	264	38	16838	2732
QAM32	4	33	308	29	4015	15776

Figure No. 30 Performance result of model #06 is **75.45%** on testing dataset

Model # 07 “Mod_3L_4D_16S16_TF_0p7rician_N.h5”

The “Mod_3L_4D_16S16_TF_0p7rician_N.h5” Conv1D model utilized the pre-trained model #No.2 “Mod_3L_4D_16S16.h5” as the starting point and has been trained on the dataset “MOD_Rician0p7fadded_16Samp_N1.h5”. Model’s further characteristics are tabulated in the table no. 10.

Table No. 10 Characteristics of Model #07

Model’s Characteristics		**DataSet Attributes**	
Input Layers	Same as model No#2 “Mod_3L_4D_16S16.h5”	Symbol Rate	64
Activation Func	Same as model No#2 “Mod_3L_4D_16S16.h5”	Channel Nature	Rician fading with k=0.7
Transfer Learning	Trained on pre-trained model No#2 “Mod_3L_4D_16S16.h5”	Input format	Normalized & shape 256x4

The model #07’s training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 242s 16ms/step - loss: 0.5118 - accuracy: 0.7388 - val_loss: 0.3691 - val_accuracy: 0.7948
Epoch 2/20
15000/15000 [==============================] - 248s 17ms/step - loss: 0.3855 - accuracy: 0.7893 - val_loss: 0.3474 - val_accuracy: 0.8117
Epoch 3/20
15000/15000 [==============================] - 239s 16ms/step - loss: 0.3610 - accuracy: 0.8070 - val_loss: 0.3252 - val_accuracy: 0.8323
Epoch 4/20
15000/15000 [==============================] - 239s 16ms/step - loss: 0.3368 - accuracy: 0.8323 - val_loss: 0.2970 - val_accuracy: 0.8601
Epoch 5/20
15000/15000 [==============================] - 245s 16ms/step - loss: 0.3165 - accuracy: 0.8515 - val_loss: 0.2857 - val_accuracy: 0.8702
Epoch 6/20
15000/15000 [==============================] - 249s 17ms/step - loss: 0.3004 - accuracy: 0.8630 - val_loss: 0.2816 - val_accuracy: 0.8741
Epoch 7/20
15000/15000 [==============================] - 248s 17ms/step - loss: 0.2883 - accuracy: 0.8712 - val_loss: 0.2647 - val_accuracy: 0.8848
Epoch 8/20
15000/15000 [==============================] - 250s 17ms/step - loss: 0.2766 - accuracy: 0.8780 - val_loss: 0.2667 - val_accuracy: 0.8830
Epoch 9/20
15000/15000 [==============================] - 263s 18ms/step - loss: 0.2703 - accuracy: 0.8820 - val_loss: 0.2575 - val_accuracy: 0.8908
Epoch 10/20
15000/15000 [==============================] - 252s 17ms/step - loss: 0.2629 - accuracy: 0.8860 - val_loss: 0.2483 - val_accuracy: 0.8924
Epoch 11/20
15000/15000 [==============================] - 255s 17ms/step - loss: 0.2578 - accuracy: 0.8885 - val_loss: 0.2434 - val_accuracy: 0.8959
Epoch 12/20
15000/15000 [==============================] - 251s 17ms/step - loss: 0.2536 - accuracy: 0.8913 - val_loss: 0.2472 - val_accuracy: 0.8953
Epoch 13/20
15000/15000 [==============================] - 253s 17ms/step - loss: 0.2496 - accuracy: 0.8929 - val_loss: 0.2570 - val_accuracy: 0.8894
Epoch 14/20
15000/15000 [==============================] - 246s 16ms/step - loss: 0.2460 - accuracy: 0.8946 - val_loss: 0.2511 - val_accuracy: 0.8959
Epoch 15/20
15000/15000 [==============================] - 260s 17ms/step - loss: 0.2429 - accuracy: 0.8962 - val_loss: 0.2465 - val_accuracy: 0.8974
Epoch 16/20
15000/15000 [==============================] - 259s 17ms/step - loss: 0.2403 - accuracy: 0.8975 - val_loss: 0.2425 - val_accuracy: 0.8983
```

```
Epoch 17/20
15000/15000 [==============================] - 241s 16ms/step - loss: 0.2386 - accuracy: 0.8986 - val_loss: 0.2499 - val_accuracy: 0.8938
Epoch 18/20
15000/15000 [==============================] - 250s 17ms/step - loss: 0.2362 - accuracy: 0.8997 - val_loss: 0.2562 - val_accuracy: 0.8979
Epoch 19/20
15000/15000 [==============================] - 237s 16ms/step - loss: 0.2347 - accuracy: 0.9006 - val_loss: 0.2503 - val_accuracy: 0.8919
Epoch 20/20
15000/15000 [==============================] - 248s 17ms/step - loss: 0.2319 - accuracy: 0.9018 - val_loss: 0.2359 - val_accuracy: 0.9022
```

The “training & validation” loss and accuracy of model#07 is shown in figure no. 31 and 32 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 33.

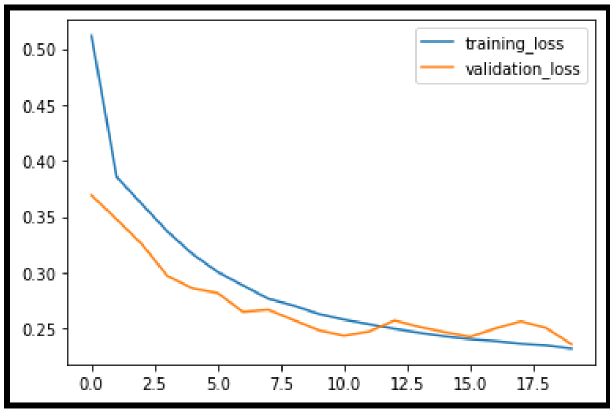


Figure No. 31 Training and validation loss of Model #07

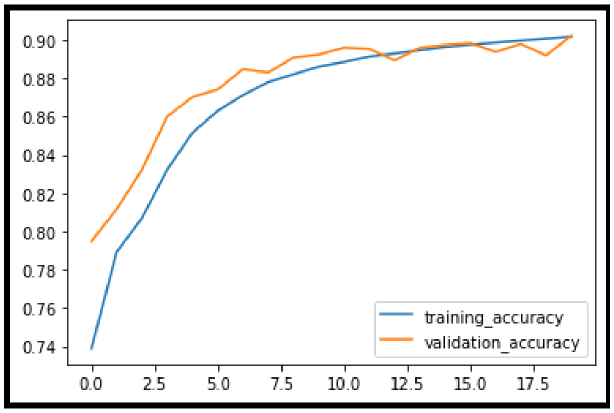


Figure No. 32 Training and validation accuracy of Model #07

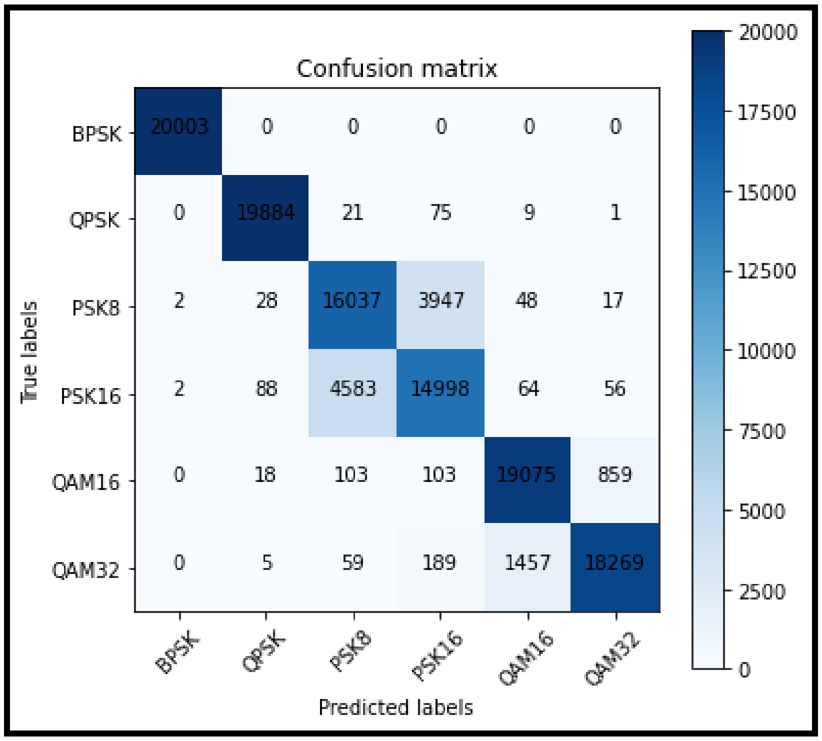


Figure No. 33 Performance result of model #07 is **90.22%** on testing dataset

Model # 08 "Mod_3L_4D_16S16_TF_0p9rician_N.h5"

The "Mod_3L_4D_16S16_TF_0p9rician_N.h5" Conv1D model utilized the pre-trained model #No.2 "Mod_3L_4D_16S16.h5" as the starting point and has been trained on the dataset "MOD_Rician0p9fadded_16Samp_N1.h5". Model's further characteristics are tabulated in the table no. 11.

Table No. 11 Characteristics of Model #08

Model's Characteristics		**DataSet Attributes**	
Input Layers	Same as model No#2 "Mod_3L_4D_16S16.h5"	Symbol Rate	64
Activation Func	Same as model No#2 "Mod_3L_4D_16S16.h5"	Channel Nature	Rician fading with k=0.9
Transfer Learning	Trained on pre-trained model No#2 "Mod_3L_4D_16S16.h5"	Input format	Normalized & shape 256x4

The model #08's training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 242s 16ms/step - loss: 0.4970 - accuracy: 0.7544 - val_loss: 0.3348 - val_accuracy: 0.8372
Epoch 2/20
15000/15000 [==============================] - 244s 16ms/step - loss: 0.3209 - accuracy: 0.8564 - val_loss: 0.2629 - val_accuracy: 0.8835
Epoch 3/20
15000/15000 [==============================] - 238s 16ms/step - loss: 0.2794 - accuracy: 0.8799 - val_loss: 0.2483 - val_accuracy: 0.8957
Epoch 4/20
15000/15000 [==============================] - 242s 16ms/step - loss: 0.2495 - accuracy: 0.8968 - val_loss: 0.2278 - val_accuracy: 0.9057
Epoch 5/20
15000/15000 [==============================] - 248s 17ms/step - loss: 0.2331 - accuracy: 0.9041 - val_loss: 0.1953 - val_accuracy: 0.9203
Epoch 6/20
15000/15000 [==============================] - 251s 17ms/step - loss: 0.2213 - accuracy: 0.9105 - val_loss: 0.1982 - val_accuracy: 0.9210
Epoch 7/20
15000/15000 [==============================] - 255s 17ms/step - loss: 0.2123 - accuracy: 0.9144 - val_loss: 0.2127 - val_accuracy: 0.9126
Epoch 8/20
15000/15000 [==============================] - 269s 18ms/step - loss: 0.2061 - accuracy: 0.9174 - val_loss: 0.1982 - val_accuracy: 0.9214
Epoch 9/20
15000/15000 [==============================] - 276s 18ms/step - loss: 0.2004 - accuracy: 0.9196 - val_loss: 0.1834 - val_accuracy: 0.9277
Epoch 10/20
15000/15000 [==============================] - 285s 19ms/step - loss: 0.1957 - accuracy: 0.9216 - val_loss: 0.1877 - val_accuracy: 0.9248
Epoch 11/20
15000/15000 [==============================] - 286s 19ms/step - loss: 0.1926 - accuracy: 0.9230 - val_loss: 0.2043 - val_accuracy: 0.9189
Epoch 12/20
15000/15000 [==============================] - 285s 19ms/step - loss: 0.1875 - accuracy: 0.9254 - val_loss: 0.1905 - val_accuracy: 0.9243
Epoch 13/20
15000/15000 [==============================] - 292s 19ms/step - loss: 0.1853 - accuracy: 0.9264 - val_loss: 0.2017 - val_accuracy: 0.9218
Epoch 14/20
15000/15000 [==============================] - 293s 20ms/step - loss: 0.1821 - accuracy: 0.9279 - val_loss: 0.1791 - val_accuracy: 0.9300
Epoch 15/20
15000/15000 [==============================] - 290s 19ms/step - loss: 0.1779 - accuracy: 0.9293 - val_loss: 0.1885 - val_accuracy: 0.9237
Epoch 16/20
15000/15000 [==============================] - 289s 19ms/step - loss: 0.1769 - accuracy: 0.9298 - val_loss: 0.1933 - val_accuracy: 0.9252
Epoch 17/20
15000/15000 [==============================] - 290s 19ms/step - loss: 0.1736 - accuracy: 0.9314 - val_loss: 0.1865 - val_accuracy: 0.9272
Epoch 18/20
15000/15000 [==============================] - 287s 19ms/step - loss: 0.1731 - accuracy: 0.9315 - val_loss: 0.1798 - val_accuracy: 0.9286
Epoch 19/20
15000/15000 [==============================] - 293s 20ms/step - loss: 0.1705 - accuracy: 0.9328 - val_loss: 0.2004 - val_accuracy: 0.9178
Epoch 20/20
15000/15000 [==============================] - 296s 20ms/step - loss: 0.1690 - accuracy: 0.9332 - val_loss: 0.1739 - val_accuracy: 0.9305
```

The "training & validation" loss and accuracy of model#08 is shown in figure no. 34 and 35 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 36.

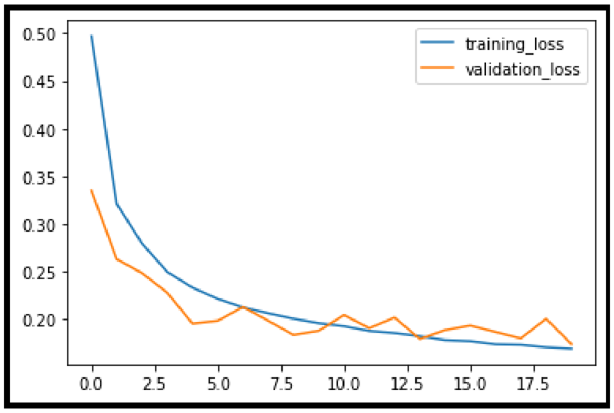


Figure No. 34 Training and validation loss of Model #08

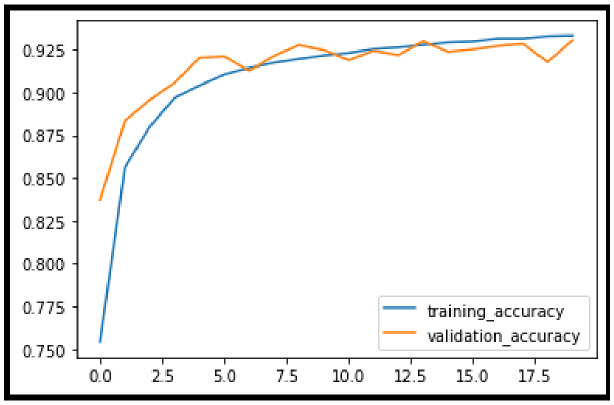
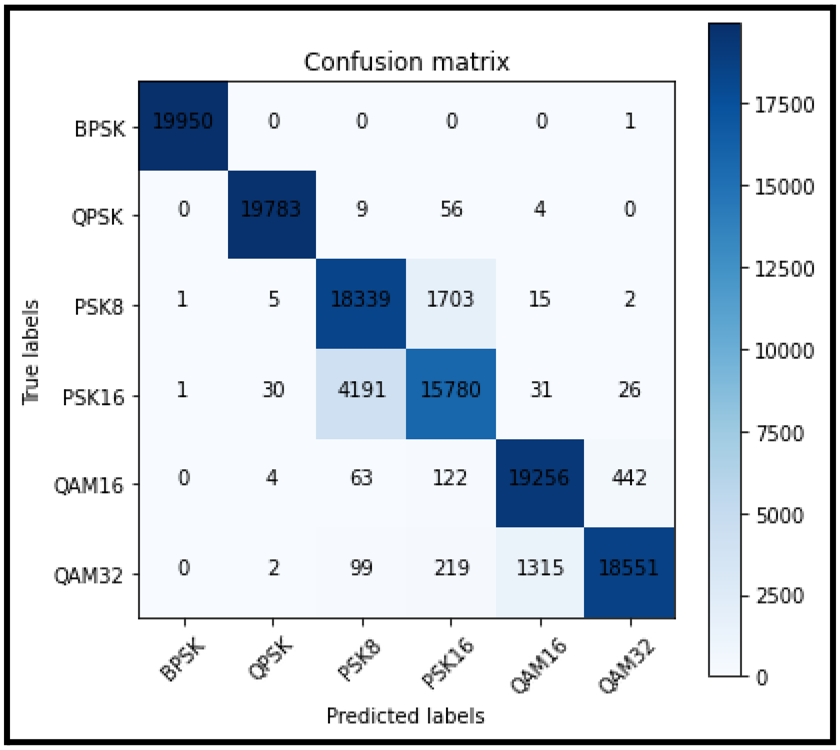


Figure No. 35 Training and validation accuracy of Model #08



	BPSK	QPSK	PSK8	PSK16	QAM16	QAM32
BPSK	19950	0	0	0	0	1
QPSK	0	19783	9	56	4	0
PSK8	1	5	18339	1703	15	2
PSK16	1	30	4191	15780	31	26
QAM16	0	4	63	122	19256	442
QAM32	0	2	99	219	1315	18551

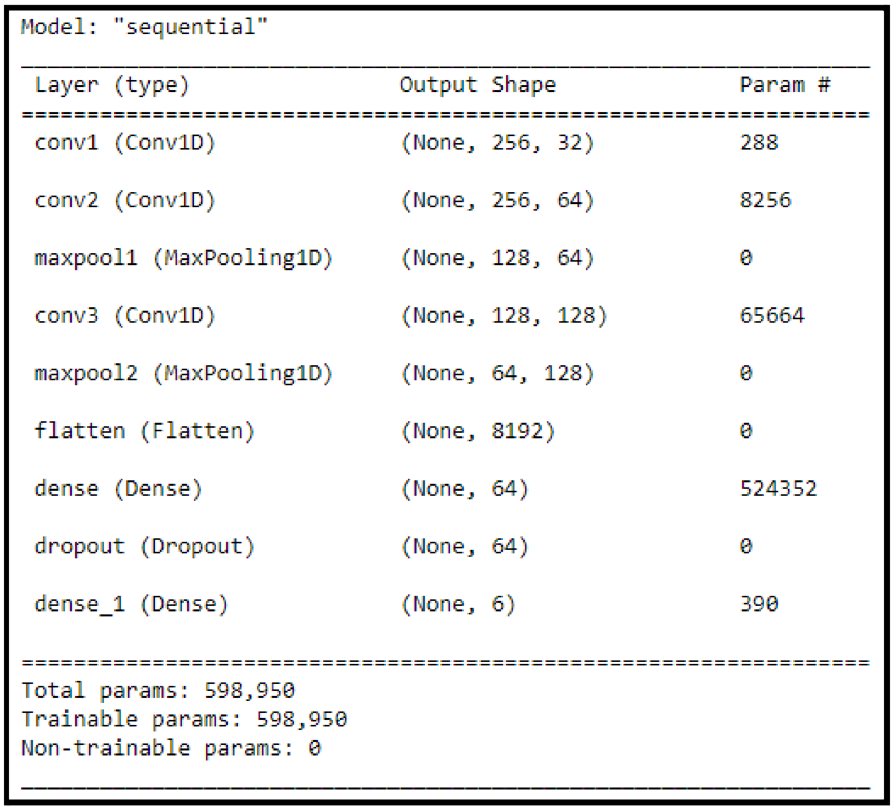
Figure No. 36 Performance result of model #08 is **93.05%** on testing dataset

Model # 09 “Mod_3L_4D_NL_16S16.h5”

The “Mod_3L_4D_NL_16S16.h5” Conv1D model is shown in figure no. 37 as trained on the dataset “MOD_16SampNN_1.h5”. Model’s further characteristics are tabulated in the table no. 12.

Table No. 12 Characteristics of Model #09

Model’s Characteristics		**DataSet Attributes**	
Input Layers	03	Symbol Rate	64
Activation Func	**sigmoid** act. func. with all hidden layers except the last decision layer utilizing softmax.	Channel Nature	No fading
Transfer Learning	—	Input format	Unnormalized & shape 256x4



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1 (Conv1D)              (None, 256, 32)           288

 conv2 (Conv1D)              (None, 256, 64)           8256

 maxpool1 (MaxPooling1D)     (None, 128, 64)           0

 conv3 (Conv1D)              (None, 128, 128)          65664

 maxpool2 (MaxPooling1D)     (None, 64, 128)           0

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 64)                524352

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 6)                 390

=================================================================
Total params: 598,950
Trainable params: 598,950
Non-trainable params: 0
_________________________________________________________________
```

Figure No. 37 The model #09 “Mod_3L_4D_NL_16S16.h5”

The model #09's training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 252s 17ms/step - loss: 1.7993 - accuracy: 0.1673 - val_loss: 1.7937 - val_accuracy: 0.1647
Epoch 2/20
15000/15000 [==============================] - 242s 16ms/step - loss: 1.7945 - accuracy: 0.1664 - val_loss: 1.7942 - val_accuracy: 0.1653
Epoch 3/20
15000/15000 [==============================] - 256s 17ms/step - loss: 1.7945 - accuracy: 0.1660 - val_loss: 1.7946 - val_accuracy: 0.1675
Epoch 4/20
15000/15000 [==============================] - 250s 17ms/step - loss: 1.7945 - accuracy: 0.1654 - val_loss: 1.7934 - val_accuracy: 0.1653
Epoch 5/20
15000/15000 [==============================] - 249s 17ms/step - loss: 1.7944 - accuracy: 0.1671 - val_loss: 1.7933 - val_accuracy: 0.1653
Epoch 6/20
15000/15000 [==============================] - 251s 17ms/step - loss: 1.7945 - accuracy: 0.1670 - val_loss: 1.7962 - val_accuracy: 0.1647
Epoch 7/20
15000/15000 [==============================] - 249s 17ms/step - loss: 1.7945 - accuracy: 0.1666 - val_loss: 1.7932 - val_accuracy: 0.1674
Epoch 8/20
15000/15000 [==============================] - 248s 17ms/step - loss: 1.7945 - accuracy: 0.1666 - val_loss: 1.7940 - val_accuracy: 0.1677
Epoch 9/20
15000/15000 [==============================] - 257s 17ms/step - loss: 1.7945 - accuracy: 0.1668 - val_loss: 1.7935 - val_accuracy: 0.1653
Epoch 10/20
15000/15000 [==============================] - 252s 17ms/step - loss: 1.7945 - accuracy: 0.1670 - val_loss: 1.7930 - val_accuracy: 0.1653
Epoch 11/20
15000/15000 [==============================] - 262s 17ms/step - loss: 1.7945 - accuracy: 0.1661 - val_loss: 1.7936 - val_accuracy: 0.1675
Epoch 12/20
15000/15000 [==============================] - 260s 17ms/step - loss: 1.7945 - accuracy: 0.1670 - val_loss: 1.7950 - val_accuracy: 0.1677
Epoch 13/20
15000/15000 [==============================] - 261s 17ms/step - loss: 1.7945 - accuracy: 0.1670 - val_loss: 1.7929 - val_accuracy: 0.1677
Epoch 14/20
15000/15000 [==============================] - 252s 17ms/step - loss: 1.7945 - accuracy: 0.1669 - val_loss: 1.7959 - val_accuracy: 0.1675
Epoch 15/20
15000/15000 [==============================] - 261s 17ms/step - loss: 1.7945 - accuracy: 0.1669 - val_loss: 1.7941 - val_accuracy: 0.1675
Epoch 16/20
15000/15000 [==============================] - 272s 18ms/step - loss: 1.7946 - accuracy: 0.1663 - val_loss: 1.7945 - val_accuracy: 0.1653
Epoch 17/20
15000/15000 [==============================] - 272s 18ms/step - loss: 1.7944 - accuracy: 0.1669 - val_loss: 1.7961 - val_accuracy: 0.1675
Epoch 18/20
15000/15000 [==============================] - 279s 19ms/step - loss: 1.7945 - accuracy: 0.1665 - val_loss: 1.7940 - val_accuracy: 0.1675
Epoch 19/20
15000/15000 [==============================] - 292s 19ms/step - loss: 1.7945 - accuracy: 0.1670 - val_loss: 1.7934 - val_accuracy: 0.1647
Epoch 20/20
15000/15000 [==============================] - 295s 20ms/step - loss: 1.7945 - accuracy: 0.1667 - val_loss: 1.7926 - val_accuracy: 0.1674
```

The "training & validation" loss and accuracy of model#09 is shown in figure no. 38 and 39 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 40.

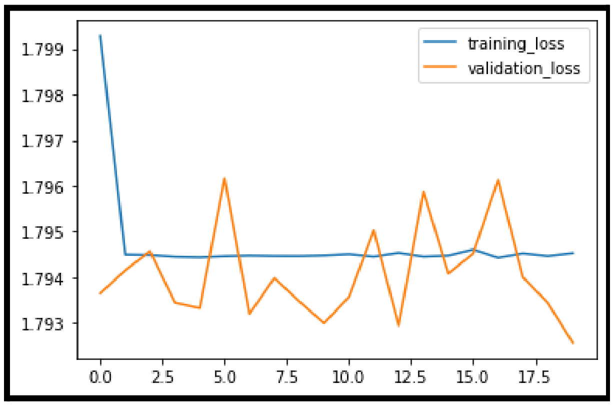


Figure No. 38 Training and validation loss of Model #09

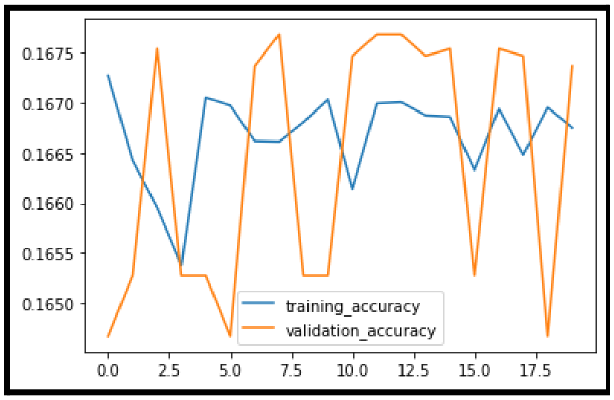


Figure No. 39 Training and validation accuracy of Model #09

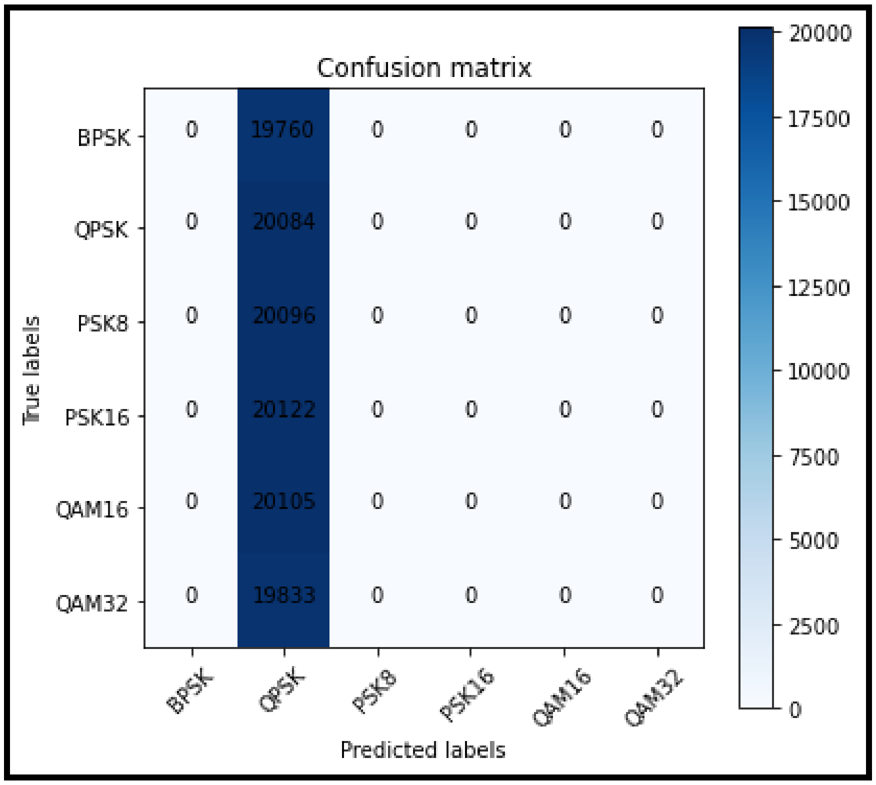


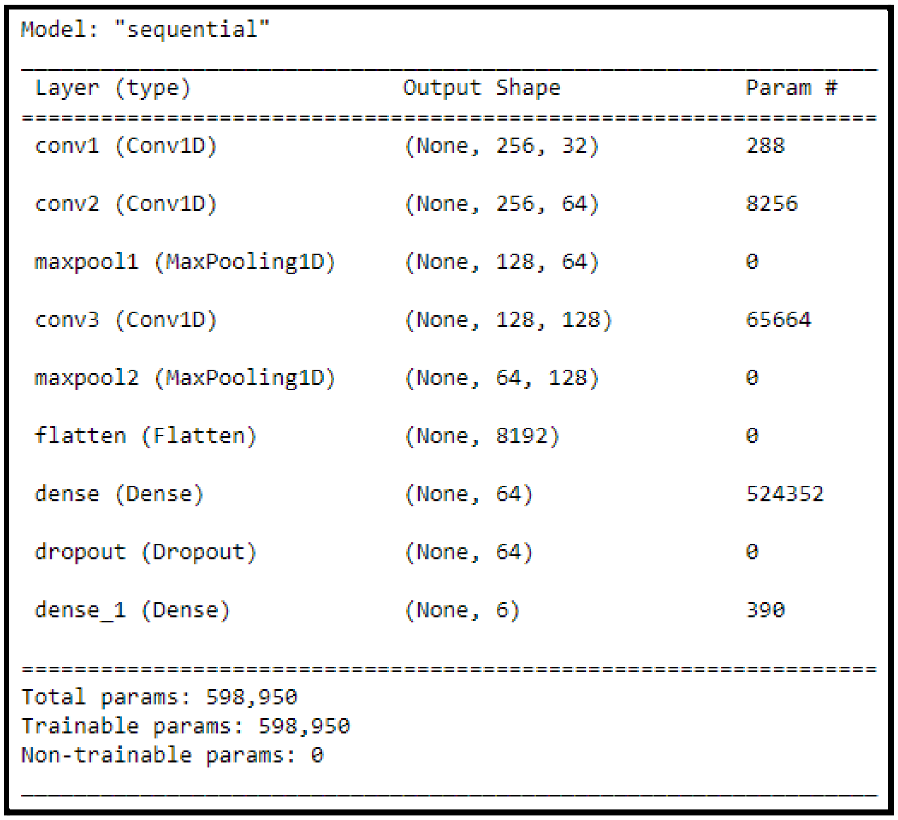
Figure No. 40 Performance result of model #09 is **16.73%** on testing dataset

Model # 10 “Mod_3L_4D_NL_tan_16S16.h5”

The “Mod_3L_4D_NL_tan_16S16.h5” Conv1D model is shown in figure no. 41 as trained on the dataset “MOD_16SampNN_1.h5”. Model’s further characteristics are tabulated in the table no. 13.

Table No. 13 Characteristics of Model #10

Model’s Characteristics		**DataSet Attributes**	
Input Layers	03	Symbol Rate	64
Activation Func	**tanh** act. func. employed in the first layer and **relu** act. func. On the rest of hidden layers except the last decision layer utilizing softmax.	Channel Nature	No fading
Transfer Learning	—	Input format	Unnormalized & shape 256x4



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1 (Conv1D)              (None, 256, 32)           288

 conv2 (Conv1D)              (None, 256, 64)           8256

 maxpool1 (MaxPooling1D)     (None, 128, 64)           0

 conv3 (Conv1D)              (None, 128, 128)          65664

 maxpool2 (MaxPooling1D)     (None, 64, 128)           0

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 64)                524352

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 6)                 390

=================================================================
Total params: 598,950
Trainable params: 598,950
Non-trainable params: 0
_________________________________________________________________
```

Figure No. 41 The model #10 “Mod_3L_4D_NL_tan_16S16.h5”

The model #10's training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 271s 18ms/step - loss: 0.0810 - accuracy: 0.9699 - val_loss: 0.0209 - val_accuracy: 0.9922
Epoch 2/20
15000/15000 [==============================] - 282s 19ms/step - loss: 0.0250 - accuracy: 0.9925 - val_loss: 0.0091 - val_accuracy: 0.9973
Epoch 3/20
15000/15000 [==============================] - 282s 19ms/step - loss: 0.0198 - accuracy: 0.9943 - val_loss: 0.0097 - val_accuracy: 0.9969
Epoch 4/20
15000/15000 [==============================] - 285s 19ms/step - loss: 0.0178 - accuracy: 0.9949 - val_loss: 0.0097 - val_accuracy: 0.9969
Epoch 5/20
15000/15000 [==============================] - 294s 20ms/step - loss: 0.0162 - accuracy: 0.9955 - val_loss: 0.0085 - val_accuracy: 0.9980
Epoch 6/20
15000/15000 [==============================] - 335s 22ms/step - loss: 0.0171 - accuracy: 0.9957 - val_loss: 0.0125 - val_accuracy: 0.9963
Epoch 7/20
15000/15000 [==============================] - 339s 23ms/step - loss: 0.0154 - accuracy: 0.9959 - val_loss: 0.0160 - val_accuracy: 0.9966
Epoch 8/20
15000/15000 [==============================] - 324s 22ms/step - loss: 0.0157 - accuracy: 0.9958 - val_loss: 0.0098 - val_accuracy: 0.9978
Epoch 9/20
15000/15000 [==============================] - 349s 23ms/step - loss: 0.0150 - accuracy: 0.9962 - val_loss: 0.0104 - val_accuracy: 0.9977
Epoch 10/20
15000/15000 [==============================] - 337s 22ms/step - loss: 0.0158 - accuracy: 0.9961 - val_loss: 0.0213 - val_accuracy: 0.9962
Epoch 11/20
15000/15000 [==============================] - 337s 22ms/step - loss: 0.0163 - accuracy: 0.9962 - val_loss: 0.0089 - val_accuracy: 0.9982
Epoch 12/20
15000/15000 [==============================] - 345s 23ms/step - loss: 0.0149 - accuracy: 0.9964 - val_loss: 0.0086 - val_accuracy: 0.9981
Epoch 13/20
15000/15000 [==============================] - 361s 24ms/step - loss: 0.0166 - accuracy: 0.9963 - val_loss: 0.0100 - val_accuracy: 0.9974
Epoch 14/20
15000/15000 [==============================] - 340s 23ms/step - loss: 0.0174 - accuracy: 0.9961 - val_loss: 0.0138 - val_accuracy: 0.9966
Epoch 15/20
15000/15000 [==============================] - 362s 24ms/step - loss: 0.0170 - accuracy: 0.9963 - val_loss: 0.0074 - val_accuracy: 0.9981
Epoch 16/20
15000/15000 [==============================] - 362s 24ms/step - loss: 0.0165 - accuracy: 0.9962 - val_loss: 0.0087 - val_accuracy: 0.9980
Epoch 17/20
15000/15000 [==============================] - 341s 23ms/step - loss: 0.0158 - accuracy: 0.9963 - val_loss: 0.0113 - val_accuracy: 0.9979
Epoch 18/20
15000/15000 [==============================] - 341s 23ms/step - loss: 0.0171 - accuracy: 0.9963 - val_loss: 0.0125 - val_accuracy: 0.9979
Epoch 19/20
15000/15000 [==============================] - 337s 22ms/step - loss: 0.0180 - accuracy: 0.9962 - val_loss: 0.0229 - val_accuracy: 0.9964
Epoch 20/20
15000/15000 [==============================] - 334s 22ms/step - loss: 0.0194 - accuracy: 0.9961 - val_loss: 0.0150 - val_accuracy: 0.9976
```

The "training & validation" loss and accuracy of model#10 is shown in figure no. 42 and 43 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 44.

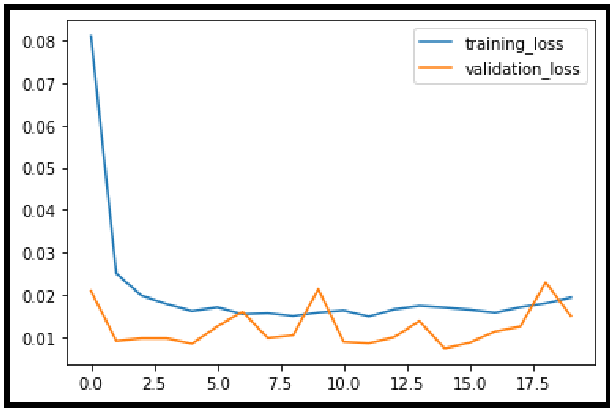


Figure No. 42 Training and validation loss of Model #10

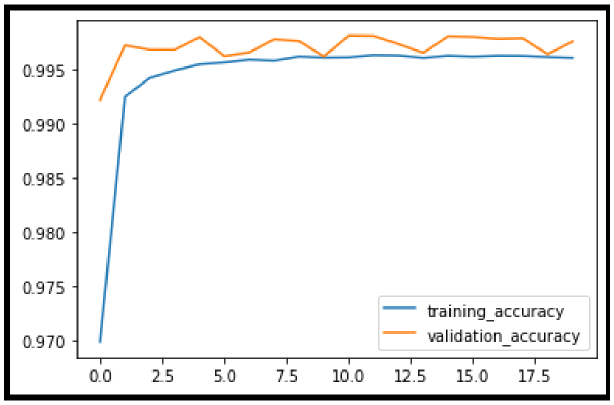


Figure No. 43 Training and validation accuracy of Model #10

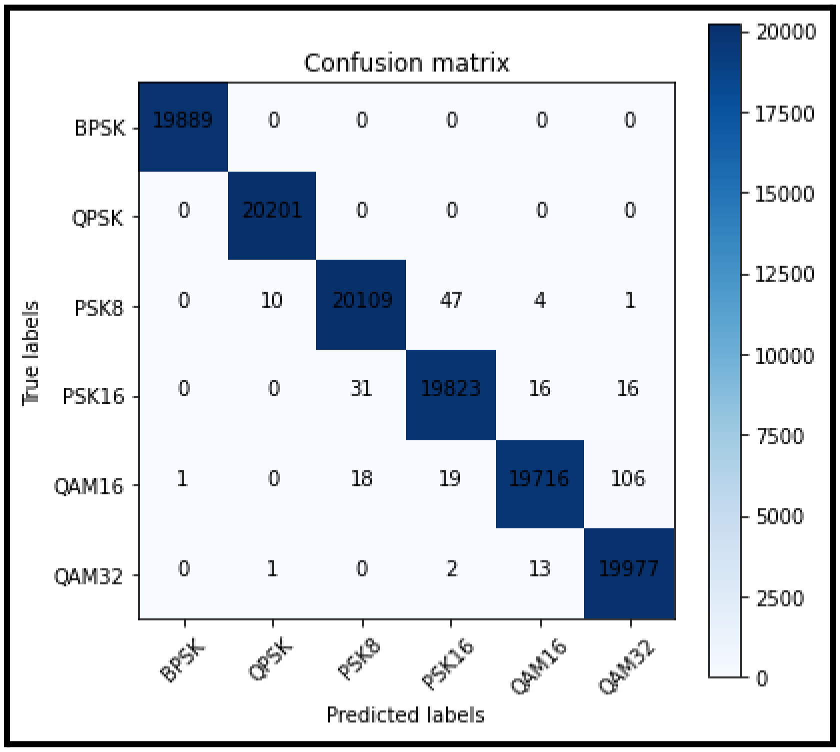


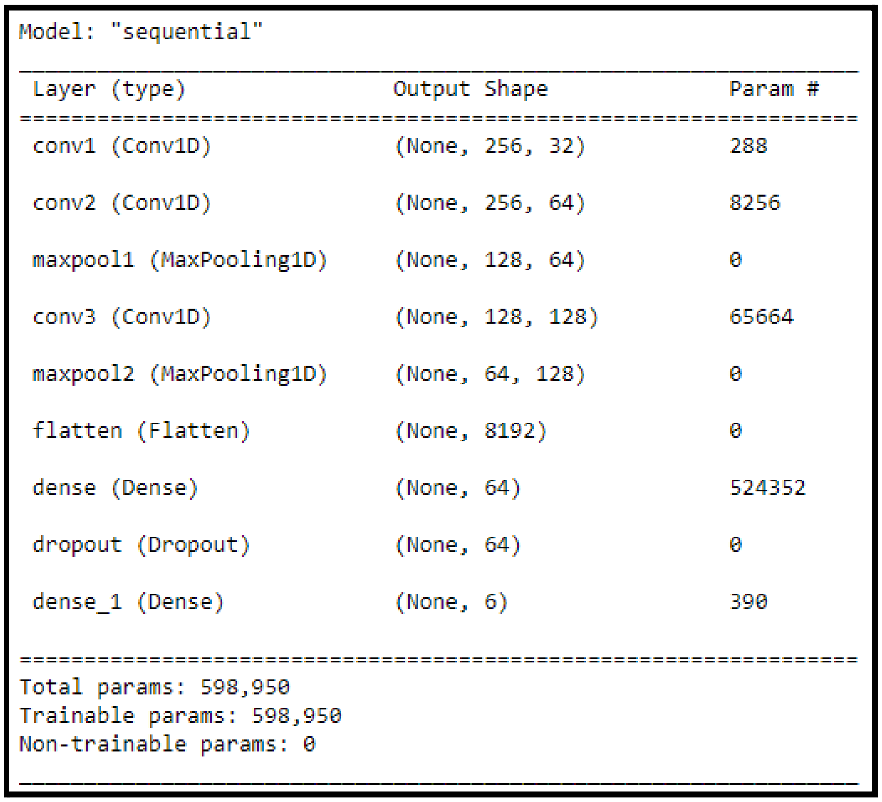
Figure No. 44 Performance result of model #10 is **99.76%** on testing dataset

Model # 11 “Mod_3L_4D_NLnL_tan_16S16.h5”

The “Mod_3L_4D_NLnL_tan_16S16.h5” Conv1D model is shown in figure no. 45 as trained on the dataset “MOD_16SampNN_1.h5”. Model’s further characteristics are tabulated in the table no. 14.

Table No. 14 Characteristics of Model #11

Model’s Characteristics		**DataSet Attributes**	
Input Layers	03	Symbol Rate	64
Activation Func	**tanh** act. func. employed in the first layer and **linear** act. func. On the rest of hidden layers except the last decision layer utilizing softmax.	Channel Nature	No fading
Transfer Learning	—	Input format	Unnormalized & shape 256x4



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1 (Conv1D)              (None, 256, 32)           288

 conv2 (Conv1D)              (None, 256, 64)           8256

 maxpool1 (MaxPooling1D)     (None, 128, 64)           0

 conv3 (Conv1D)              (None, 128, 128)          65664

 maxpool2 (MaxPooling1D)     (None, 64, 128)           0

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 64)                524352

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 6)                 390

=================================================================
Total params: 598,950
Trainable params: 598,950
Non-trainable params: 0
_________________________________________________________________
```

Figure No. 45 The model #11 “Mod_3L_4D_NLnL_tan_16S16.h5”

The model #11's training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 240s 16ms/step - loss: 0.2491 - accuracy: 0.9084 - val_loss: 0.0943 - val_accuracy: 0.9717
Epoch 2/20
15000/15000 [==============================] - 243s 16ms/step - loss: 0.1321 - accuracy: 0.9580 - val_loss: 0.0567 - val_accuracy: 0.9817
Epoch 3/20
15000/15000 [==============================] - 250s 17ms/step - loss: 0.1097 - accuracy: 0.9677 - val_loss: 0.0643 - val_accuracy: 0.9809
Epoch 4/20
15000/15000 [==============================] - 260s 17ms/step - loss: 0.1006 - accuracy: 0.9720 - val_loss: 0.0699 - val_accuracy: 0.9808
Epoch 5/20
15000/15000 [==============================] - 267s 18ms/step - loss: 0.0984 - accuracy: 0.9735 - val_loss: 0.0681 - val_accuracy: 0.9824
Epoch 6/20
15000/15000 [==============================] - 269s 18ms/step - loss: 0.0943 - accuracy: 0.9755 - val_loss: 0.0641 - val_accuracy: 0.9840
Epoch 7/20
15000/15000 [==============================] - 264s 18ms/step - loss: 0.0912 - accuracy: 0.9767 - val_loss: 0.0620 - val_accuracy: 0.9851
Epoch 8/20
15000/15000 [==============================] - 264s 18ms/step - loss: 0.0914 - accuracy: 0.9775 - val_loss: 0.0780 - val_accuracy: 0.9827
Epoch 9/20
15000/15000 [==============================] - 256s 17ms/step - loss: 0.0905 - accuracy: 0.9781 - val_loss: 0.0864 - val_accuracy: 0.9793
Epoch 10/20
15000/15000 [==============================] - 260s 17ms/step - loss: 0.0909 - accuracy: 0.9784 - val_loss: 0.0953 - val_accuracy: 0.9764
Epoch 11/20
15000/15000 [==============================] - 271s 18ms/step - loss: 0.0916 - accuracy: 0.9788 - val_loss: 0.0884 - val_accuracy: 0.9833
Epoch 12/20
15000/15000 [==============================] - 266s 18ms/step - loss: 0.0929 - accuracy: 0.9789 - val_loss: 0.0894 - val_accuracy: 0.9828
Epoch 13/20
15000/15000 [==============================] - 271s 18ms/step - loss: 0.0944 - accuracy: 0.9794 - val_loss: 0.1144 - val_accuracy: 0.9736
Epoch 14/20
15000/15000 [==============================] - 277s 18ms/step - loss: 0.0963 - accuracy: 0.9793 - val_loss: 0.0859 - val_accuracy: 0.9815
Epoch 15/20
15000/15000 [==============================] - 276s 18ms/step - loss: 0.0967 - accuracy: 0.9799 - val_loss: 0.1058 - val_accuracy: 0.9786
Epoch 16/20
15000/15000 [==============================] - 270s 18ms/step - loss: 0.0976 - accuracy: 0.9800 - val_loss: 0.0602 - val_accuracy: 0.9873
Epoch 17/20
15000/15000 [==============================] - 274s 18ms/step - loss: 0.0986 - accuracy: 0.9801 - val_loss: 0.0639 - val_accuracy: 0.9863
Epoch 18/20
15000/15000 [==============================] - 277s 18ms/step - loss: 0.1014 - accuracy: 0.9797 - val_loss: 0.0575 - val_accuracy: 0.9871
Epoch 19/20
15000/15000 [==============================] - 281s 19ms/step - loss: 0.1000 - accuracy: 0.9806 - val_loss: 0.0943 - val_accuracy: 0.9823
Epoch 20/20
15000/15000 [==============================] - 283s 19ms/step - loss: 0.1040 - accuracy: 0.9807 - val_loss: 0.0758 - val_accuracy: 0.9859
```

The "training & validation" loss and accuracy of model#11 is shown in figure no. 46 and 47 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 48.

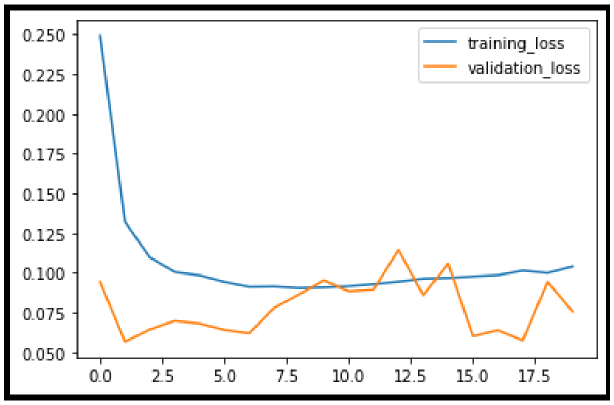


Figure No. 46 Training and validation loss of Model #11

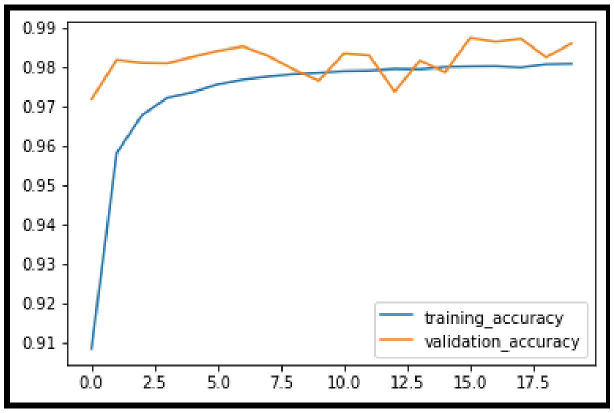


Figure No. 47 Training and validation accuracy of Model #11

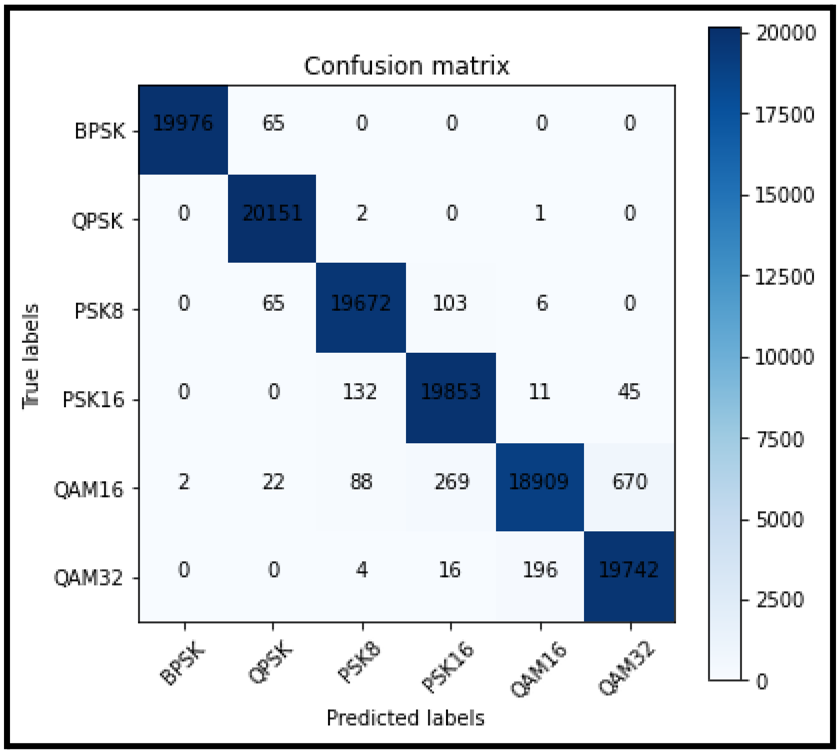


Figure No. 48 Performance result of model #11 is **98.56%** on testing dataset

Model # 12 “Mod_3L_4D_16S16_TF_NL_Rayleighfadded_N.h5”

The “Mod_3L_4D_16S16_TF_NL_Rayleighfadded_N.h5” Conv1D model utilized the pre-trained model #No.10 “Mod_3L_4D_NL_tan_16S16.h5” as the starting point and has been trained on the dataset “MOD_Rayleighfadded_16Samp_N_NN1.h5”. Model’s further characteristics are tabulated in the table no. 15.

Table No. 15 Characteristics of Model #12

Model’s Characteristics		**DataSet Attributes**	
Input Layers	Same as model No#10 “Mod_3L_4D_NL_tan_16S16.h5”	Symbol Rate	64
Activation Func	Same as model No#10 “Mod_3L_4D_NL_tan_16S16.h5”	Channel Nature	Rayleigh fading
Transfer Learning	Trained on pre-trained model No#10 “Mod_3L_4D_NL_tan_16S16.h5”	Input format	Unnormalized & shape 256x4

The model #12’s training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 273s 18ms/step - loss: 0.8742 - accuracy: 0.7241 - val_loss: 0.4442 - val_accuracy: 0.8087
Epoch 2/20
15000/15000 [==============================] - 290s 19ms/step - loss: 0.5039 - accuracy: 0.7860 - val_loss: 0.4144 - val_accuracy: 0.8249
Epoch 3/20
15000/15000 [==============================] - 298s 20ms/step - loss: 0.4644 - accuracy: 0.8019 - val_loss: 0.3881 - val_accuracy: 0.8319
Epoch 4/20
15000/15000 [==============================] - 299s 20ms/step - loss: 0.4473 - accuracy: 0.8096 - val_loss: 0.4106 - val_accuracy: 0.8246
Epoch 5/20
15000/15000 [==============================] - 302s 20ms/step - loss: 0.4367 - accuracy: 0.8135 - val_loss: 0.3896 - val_accuracy: 0.8330
Epoch 6/20
15000/15000 [==============================] - 314s 21ms/step - loss: 0.4307 - accuracy: 0.8167 - val_loss: 0.3703 - val_accuracy: 0.8435
Epoch 7/20
15000/15000 [==============================] - 318s 21ms/step - loss: 0.4236 - accuracy: 0.8204 - val_loss: 0.3659 - val_accuracy: 0.8451
Epoch 8/20
15000/15000 [==============================] - 316s 21ms/step - loss: 0.4183 - accuracy: 0.8219 - val_loss: 0.3873 - val_accuracy: 0.8304
Epoch 9/20
15000/15000 [==============================] - 321s 21ms/step - loss: 0.4124 - accuracy: 0.8247 - val_loss: 0.3580 - val_accuracy: 0.8486
Epoch 10/20
15000/15000 [==============================] - 325s 22ms/step - loss: 0.4092 - accuracy: 0.8263 - val_loss: 0.3631 - val_accuracy: 0.8407
Epoch 11/20
15000/15000 [==============================] - 330s 22ms/step - loss: 0.4059 - accuracy: 0.8269 - val_loss: 0.3598 - val_accuracy: 0.8475
Epoch 12/20
15000/15000 [==============================] - 320s 21ms/step - loss: 0.4027 - accuracy: 0.8286 - val_loss: 0.3751 - val_accuracy: 0.8377
Epoch 13/20
15000/15000 [==============================] - 331s 22ms/step - loss: 0.4006 - accuracy: 0.8291 - val_loss: 0.3515 - val_accuracy: 0.8480
Epoch 14/20
15000/15000 [==============================] - 337s 22ms/step - loss: 0.3968 - accuracy: 0.8315 - val_loss: 0.3560 - val_accuracy: 0.8453
```

```
Epoch 15/20
15000/15000 [==============================] - 324s 22ms/step - loss: 0.3958 - accuracy: 0.8319 - val_loss: 0.3601 - val_accuracy: 0.8474
Epoch 16/20
15000/15000 [==============================] - 334s 22ms/step - loss: 0.3938 - accuracy: 0.8324 - val_loss: 0.3527 - val_accuracy: 0.8502
Epoch 17/20
15000/15000 [==============================] - 346s 23ms/step - loss: 0.3912 - accuracy: 0.8338 - val_loss: 0.3516 - val_accuracy: 0.8506
Epoch 18/20
15000/15000 [==============================] - 342s 23ms/step - loss: 0.3916 - accuracy: 0.8334 - val_loss: 0.3489 - val_accuracy: 0.8526
Epoch 19/20
15000/15000 [==============================] - 332s 22ms/step - loss: 0.3901 - accuracy: 0.8342 - val_loss: 0.3644 - val_accuracy: 0.8435
Epoch 20/20
15000/15000 [==============================] - 343s 23ms/step - loss: 0.3881 - accuracy: 0.8359 - val_loss: 0.3570 - val_accuracy: 0.8505
```

The "training & validation" loss and accuracy of model#12 is shown in figure no. 49 and 50 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 51.

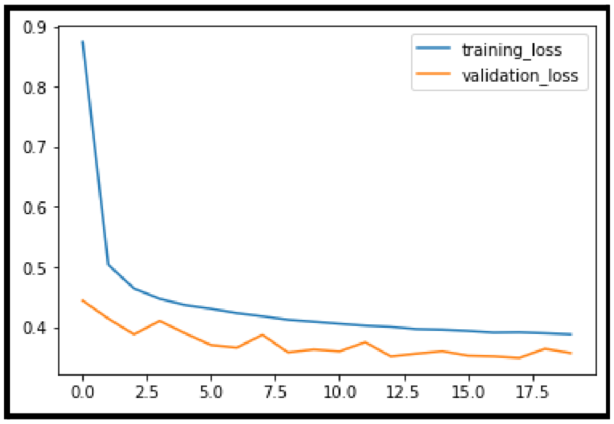


Figure No. 49 Training and validation loss of Model #12

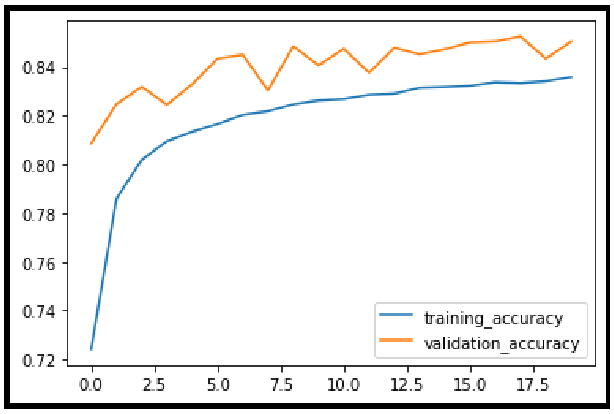


Figure No. 50 Training and validation accuracy of Model #12

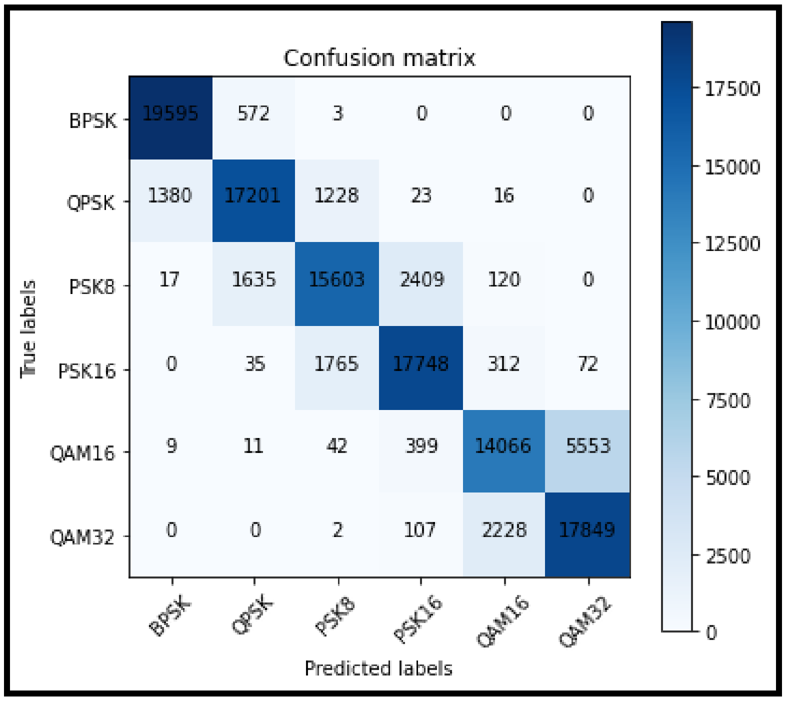


Figure No. 51 Performance result of model #12 is **85%** on testing dataset

Model # 13 "Mod_3L_4D_16S16_TF_NL_0p3ricianfadded_N"

The "Mod_3L_4D_16S16_TF_NL_0p3ricianfadded_N" Conv1D model utilized the pre-trained model #No.10 "Mod_3L_4D_NL_tan_16S16.h5" as the starting point and has been trained on the dataset "MOD_Rician0p3fadded_16Samp_N_NN1.h5". Model's further characteristics are tabulated in the table no. 16.

Table No. 16 Characteristics of Model #13

Model's Characteristics		**DataSet Attributes**	
Input Layers	Same as model No#10 "Mod_3L_4D_NL_tan_16S16.h5"	Symbol Rate	64
Activation Func	Same as model No#10 "Mod_3L_4D_NL_tan_16S16.h5"	Channel Nature	Rician fading with k=0.3
Transfer Learning	Trained on pre-trained model No#10 "Mod_3L_4D_NL_tan_16S16.h5"	Input format	Unnormalized & shape 256x4

The model #13's training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 240s 16ms/step - loss: 0.4234 - accuracy: 0.8298 - val_loss: 0.2806 - val_accuracy: 0.8805
Epoch 2/20
15000/15000 [==============================] - 245s 16ms/step - loss: 0.3042 - accuracy: 0.8796 - val_loss: 0.2609 - val_accuracy: 0.8951
Epoch 3/20
15000/15000 [==============================] - 244s 16ms/step - loss: 0.2741 - accuracy: 0.8932 - val_loss: 0.2471 - val_accuracy: 0.9047
Epoch 4/20
15000/15000 [==============================] - 253s 17ms/step - loss: 0.2587 - accuracy: 0.9006 - val_loss: 0.2139 - val_accuracy: 0.9179
Epoch 5/20
15000/15000 [==============================] - 256s 17ms/step - loss: 0.2498 - accuracy: 0.9044 - val_loss: 0.2151 - val_accuracy: 0.9184
Epoch 6/20
15000/15000 [==============================] - 254s 17ms/step - loss: 0.2414 - accuracy: 0.9083 - val_loss: 0.2934 - val_accuracy: 0.8941
Epoch 7/20
15000/15000 [==============================] - 256s 17ms/step - loss: 0.2355 - accuracy: 0.9097 - val_loss: 0.1995 - val_accuracy: 0.9226
Epoch 8/20
15000/15000 [==============================] - 258s 17ms/step - loss: 0.2305 - accuracy: 0.9125 - val_loss: 0.2080 - val_accuracy: 0.9202
Epoch 9/20
15000/15000 [==============================] - 262s 17ms/step - loss: 0.2272 - accuracy: 0.9138 - val_loss: 0.1973 - val_accuracy: 0.9237
Epoch 10/20
15000/15000 [==============================] - 261s 17ms/step - loss: 0.2262 - accuracy: 0.9141 - val_loss: 0.1913 - val_accuracy: 0.9265
Epoch 11/20
15000/15000 [==============================] - 257s 17ms/step - loss: 0.2224 - accuracy: 0.9154 - val_loss: 0.1853 - val_accuracy: 0.9278
Epoch 12/20
15000/15000 [==============================] - 261s 17ms/step - loss: 0.2219 - accuracy: 0.9159 - val_loss: 0.2071 - val_accuracy: 0.9168
Epoch 13/20
15000/15000 [==============================] - 258s 17ms/step - loss: 0.2166 - accuracy: 0.9174 - val_loss: 0.1987 - val_accuracy: 0.9213
Epoch 14/20
15000/15000 [==============================] - 257s 17ms/step - loss: 0.2170 - accuracy: 0.9173 - val_loss: 0.1977 - val_accuracy: 0.9205
Epoch 15/20
15000/15000 [==============================] - 257s 17ms/step - loss: 0.2143 - accuracy: 0.9184 - val_loss: 0.1964 - val_accuracy: 0.9234
Epoch 16/20
15000/15000 [==============================] - 260s 17ms/step - loss: 0.2137 - accuracy: 0.9189 - val_loss: 0.1799 - val_accuracy: 0.9304
Epoch 17/20
15000/15000 [==============================] - 262s 17ms/step - loss: 0.2122 - accuracy: 0.9194 - val_loss: 0.1892 - val_accuracy: 0.9282
Epoch 18/20
15000/15000 [==============================] - 263s 18ms/step - loss: 0.2132 - accuracy: 0.9189 - val_loss: 0.2038 - val_accuracy: 0.9223
Epoch 19/20
15000/15000 [==============================] - 262s 17ms/step - loss: 0.2113 - accuracy: 0.9198 - val_loss: 0.1834 - val_accuracy: 0.9287
Epoch 20/20
15000/15000 [==============================] - 262s 17ms/step - loss: 0.2107 - accuracy: 0.9201 - val_loss: 0.1869 - val_accuracy: 0.9259
```

The "training & validation" loss and accuracy of model#13 is shown in figure no. 52 and 53 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 54.

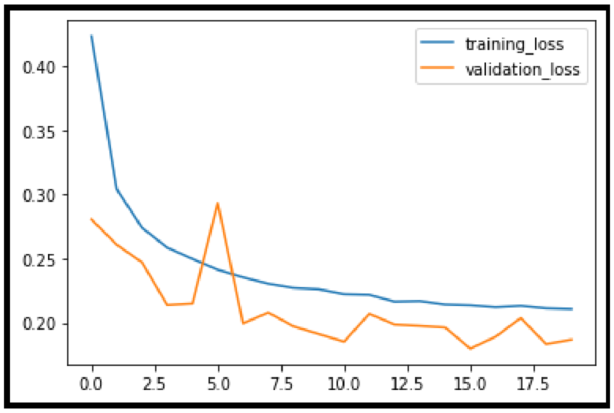


Figure No. 52 Training and validation loss of Model #13

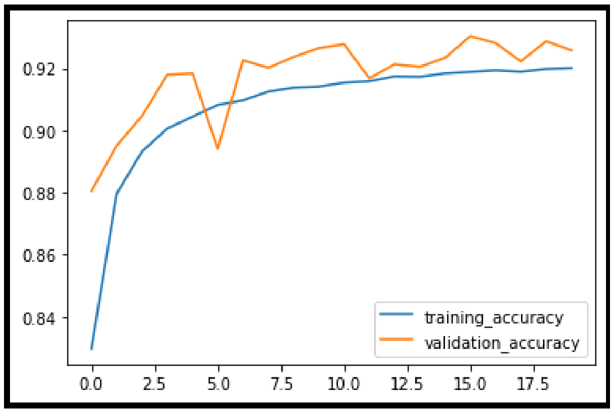


Figure No. 53 Training and validation accuracy of Model #13

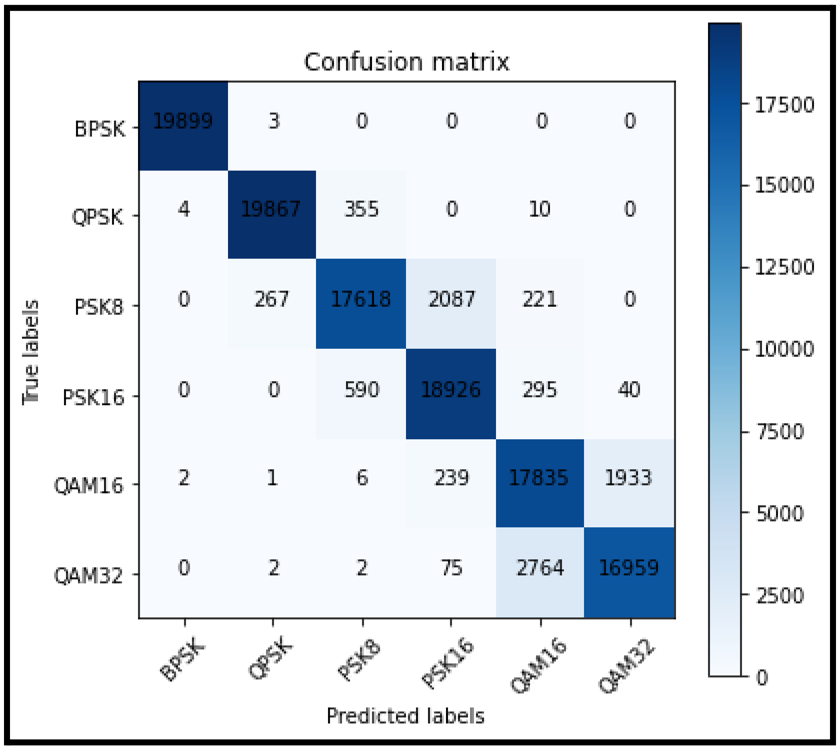


Figure No. 54 Performance result of model #13 is **92.59%** on testing dataset

Model # 14 “Mod_3L_4D_16S16_TF_NL_0p7ricianfadded_N.h5”

The “Mod_3L_4D_16S16_TF_NL_0p7ricianfadded_N” Conv1D model utilized the pre-trained model #No.10 “Mod_3L_4D_NL_tan_16S16.h5” as the starting point and has been trained on the dataset “MOD_Rician0p7fadded_16Samp_N_NN1.h5”. Model’s further characteristics are tabulated in the table no. 17.

Table No. 17 Characteristics of Model #14

Model’s Characteristics		**DataSet Attributes**	
Input Layers	Same as model No#10 “Mod_3L_4D_NL_tan_16S16.h5”	Symbol Rate	64
Activation Func	Same as model No#10 “Mod_3L_4D_NL_tan_16S16.h5”	Channel Nature	Rician fading with k=0.7
Transfer Learning	Trained on pre-trained model No#10 “Mod_3L_4D_NL_tan_16S16.h5”	Input format	Unnormalized & shape 256x4

The model #14’s training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 244s 16ms/step - loss: 0.2869 - accuracy: 0.8937 - val_loss: 0.1810 - val_accuracy: 0.9379
Epoch 2/20
15000/15000 [==============================] - 243s 16ms/step - loss: 0.1839 - accuracy: 0.9363 - val_loss: 0.1312 - val_accuracy: 0.9531
Epoch 3/20
15000/15000 [==============================] - 245s 16ms/step - loss: 0.1648 - accuracy: 0.9440 - val_loss: 0.1220 - val_accuracy: 0.9551
Epoch 4/20
15000/15000 [==============================] - 249s 17ms/step - loss: 0.1585 - accuracy: 0.9468 - val_loss: 0.1059 - val_accuracy: 0.9626
Epoch 5/20
15000/15000 [==============================] - 250s 17ms/step - loss: 0.1501 - accuracy: 0.9500 - val_loss: 0.1311 - val_accuracy: 0.9535
Epoch 6/20
15000/15000 [==============================] - 253s 17ms/step - loss: 0.1426 - accuracy: 0.9518 - val_loss: 0.1070 - val_accuracy: 0.9629
Epoch 7/20
15000/15000 [==============================] - 253s 17ms/step - loss: 0.1379 - accuracy: 0.9535 - val_loss: 0.1189 - val_accuracy: 0.9588
Epoch 8/20
15000/15000 [==============================] - 254s 17ms/step - loss: 0.1372 - accuracy: 0.9551 - val_loss: 0.0920 - val_accuracy: 0.9674
Epoch 9/20
15000/15000 [==============================] - 253s 17ms/step - loss: 0.1324 - accuracy: 0.9562 - val_loss: 0.1151 - val_accuracy: 0.9646
Epoch 10/20
15000/15000 [==============================] - 257s 17ms/step - loss: 0.1313 - accuracy: 0.9567 - val_loss: 0.0979 - val_accuracy: 0.9666
Epoch 11/20
15000/15000 [==============================] - 255s 17ms/step - loss: 0.1283 - accuracy: 0.9581 - val_loss: 0.1005 - val_accuracy: 0.9667
Epoch 12/20
15000/15000 [==============================] - 259s 17ms/step - loss: 0.1279 - accuracy: 0.9584 - val_loss: 0.1197 - val_accuracy: 0.9617
Epoch 13/20
15000/15000 [==============================] - 258s 17ms/step - loss: 0.1259 - accuracy: 0.9586 - val_loss: 0.1002 - val_accuracy: 0.9659
Epoch 14/20
15000/15000 [==============================] - 259s 17ms/step - loss: 0.1245 - accuracy: 0.9588 - val_loss: 0.0977 - val_accuracy: 0.9653
```

```
Epoch 15/20
15000/15000 [==============================] - 259s 17ms/step - loss: 0.1245 - accuracy: 0.9593 - val_loss: 0.1228 - val_accuracy: 0.9615
Epoch 16/20
15000/15000 [==============================] - 260s 17ms/step - loss: 0.1223 - accuracy: 0.9603 - val_loss: 0.1016 - val_accuracy: 0.9653
Epoch 17/20
15000/15000 [==============================] - 263s 18ms/step - loss: 0.1179 - accuracy: 0.9608 - val_loss: 0.0927 - val_accuracy: 0.9693
Epoch 18/20
15000/15000 [==============================] - 271s 18ms/step - loss: 0.1205 - accuracy: 0.9602 - val_loss: 0.0995 - val_accuracy: 0.9672
Epoch 19/20
15000/15000 [==============================] - 264s 18ms/step - loss: 0.1210 - accuracy: 0.9607 - val_loss: 0.1075 - val_accuracy: 0.9645
Epoch 20/20
15000/15000 [==============================] - 265s 18ms/step - loss: 0.1191 - accuracy: 0.9611 - val_loss: 0.1064 - val_accuracy: 0.9653
```

The "training & validation" loss and accuracy of model#14 is shown in figure no. 55 and 56 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 57.

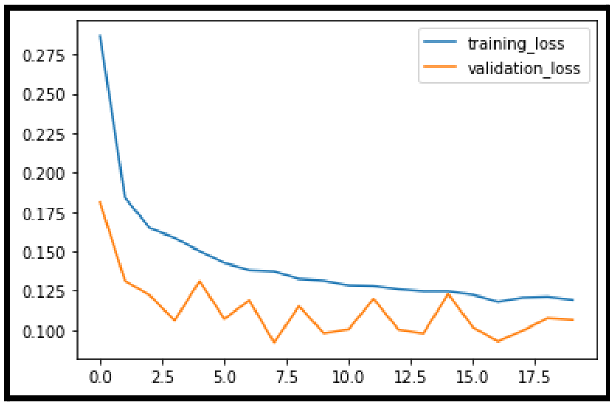


Figure No. 55 Training and validation loss of Model #14

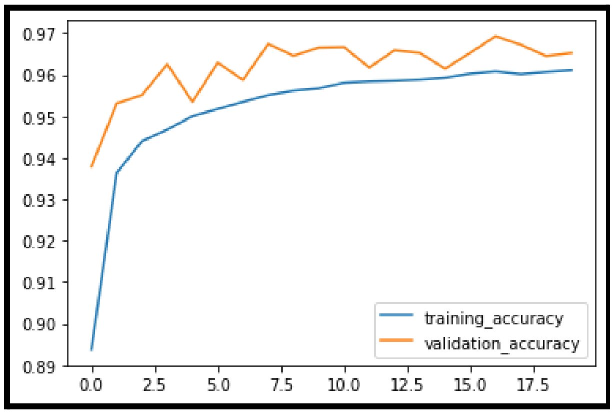


Figure No. 56 Training and validation accuracy of Model #14

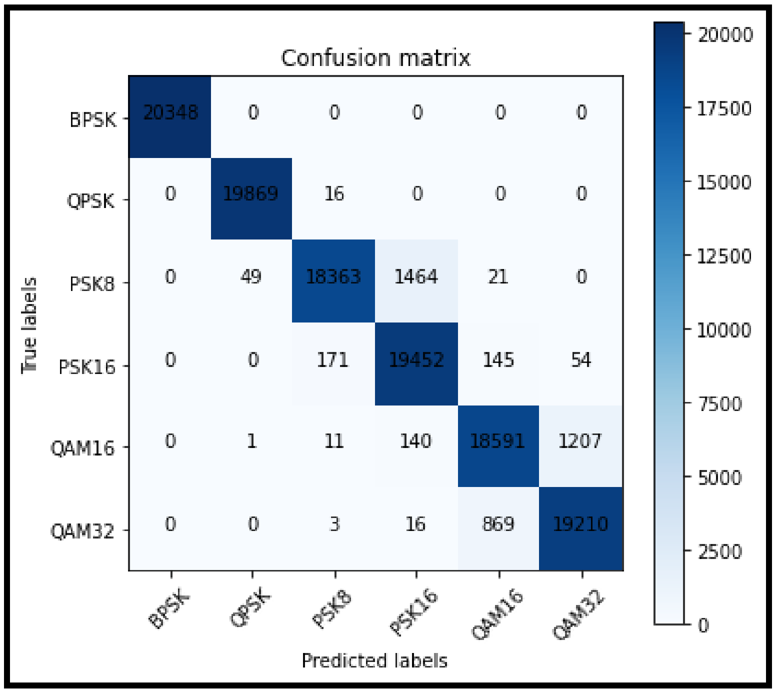


Figure No. 57 Performance result of model #14 is **96.53%** on testing dataset

Model # 15 "Mod_3L_4D_16S16_TF_NL_0p9ricianfadded_N.h5"

The "Mod_3L_4D_16S16_TF_NL_0p9ricianfadded_N" Conv1D model utilized the pre-trained model #No.10 "Mod_3L_4D_NL_tan_16S16.h5" as the starting point and has been trained on the dataset "MOD_Rician0p9fadded_16Samp_N_NN1.h5". Model's further characteristics are tabulated in the table no. 18.

Table No. 18 Characteristics of Model #15

Model's Characteristics		**DataSet Attributes**	
Input Layers	Same as model No#10 "Mod_3L_4D_NL_tan_16S16.h5"	Symbol Rate	64
Activation Func	Same as model No#10 "Mod_3L_4D_NL_tan_16S16.h5"	Channel Nature	Rician fading with k=0.9
Transfer Learning	Trained on pre-trained model No#10 "Mod_3L_4D_NL_tan_16S16.h5"	Input format	Unnormalized & shape 256x4

The model #15's training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 245s 16ms/step - loss: 0.2856 - accuracy: 0.9014 - val_loss: 0.1239 - val_accuracy: 0.9581
Epoch 2/20
15000/15000 [==============================] - 247s 16ms/step - loss: 0.1479 - accuracy: 0.9510 - val_loss: 0.1087 - val_accuracy: 0.9657
Epoch 3/20
15000/15000 [==============================] - 248s 17ms/step - loss: 0.1255 - accuracy: 0.9596 - val_loss: 0.0854 - val_accuracy: 0.9702
Epoch 4/20
15000/15000 [==============================] - 254s 17ms/step - loss: 0.1134 - accuracy: 0.9635 - val_loss: 0.0815 - val_accuracy: 0.9727
Epoch 5/20
15000/15000 [==============================] - 254s 17ms/step - loss: 0.1055 - accuracy: 0.9662 - val_loss: 0.0705 - val_accuracy: 0.9777
Epoch 6/20
15000/15000 [==============================] - 247s 16ms/step - loss: 0.1017 - accuracy: 0.9682 - val_loss: 0.1254 - val_accuracy: 0.9605
Epoch 7/20
15000/15000 [==============================] - 244s 16ms/step - loss: 0.0966 - accuracy: 0.9692 - val_loss: 0.0809 - val_accuracy: 0.9725
Epoch 8/20
15000/15000 [==============================] - 241s 16ms/step - loss: 0.0951 - accuracy: 0.9698 - val_loss: 0.0647 - val_accuracy: 0.9779
Epoch 9/20
15000/15000 [==============================] - 245s 16ms/step - loss: 0.0896 - accuracy: 0.9718 - val_loss: 0.0947 - val_accuracy: 0.9690
Epoch 10/20
15000/15000 [==============================] - 252s 17ms/step - loss: 0.0920 - accuracy: 0.9709 - val_loss: 0.0660 - val_accuracy: 0.9770
Epoch 11/20
15000/15000 [==============================] - 252s 17ms/step - loss: 0.0876 - accuracy: 0.9723 - val_loss: 0.0601 - val_accuracy: 0.9795
Epoch 12/20
15000/15000 [==============================] - 251s 17ms/step - loss: 0.0892 - accuracy: 0.9721 - val_loss: 0.0651 - val_accuracy: 0.9783
Epoch 13/20
15000/15000 [==============================] - 258s 17ms/step - loss: 0.0862 - accuracy: 0.9728 - val_loss: 0.0780 - val_accuracy: 0.9765
Epoch 14/20
15000/15000 [==============================] - 255s 17ms/step - loss: 0.0855 - accuracy: 0.9732 - val_loss: 0.0734 - val_accuracy: 0.9759
Epoch 15/20
15000/15000 [==============================] - 256s 17ms/step - loss: 0.0912 - accuracy: 0.9722 - val_loss: 0.0711 - val_accuracy: 0.9759
Epoch 16/20
15000/15000 [==============================] - 256s 17ms/step - loss: 0.0836 - accuracy: 0.9741 - val_loss: 0.0639 - val_accuracy: 0.9791
Epoch 17/20
15000/15000 [==============================] - 255s 17ms/step - loss: 0.0843 - accuracy: 0.9737 - val_loss: 0.0548 - val_accuracy: 0.9823
Epoch 18/20
15000/15000 [==============================] - 253s 17ms/step - loss: 0.0823 - accuracy: 0.9743 - val_loss: 0.0634 - val_accuracy: 0.9803
Epoch 19/20
15000/15000 [==============================] - 257s 17ms/step - loss: 0.0821 - accuracy: 0.9744 - val_loss: 0.0773 - val_accuracy: 0.9769
Epoch 20/20
15000/15000 [==============================] - 260s 17ms/step - loss: 0.0860 - accuracy: 0.9735 - val_loss: 0.0594 - val_accuracy: 0.9807
```

The "training & validation" loss and accuracy of model#15 is shown in figure no. 58 and 59 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 60.

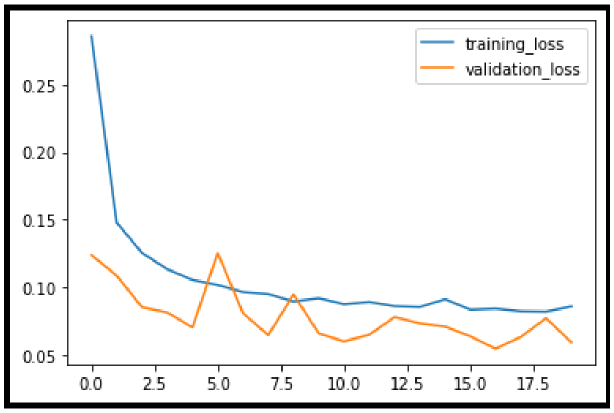


Figure No. 58 Training and validation loss of Model #15

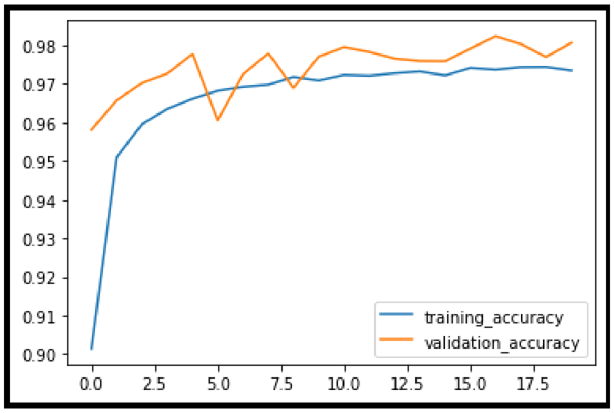
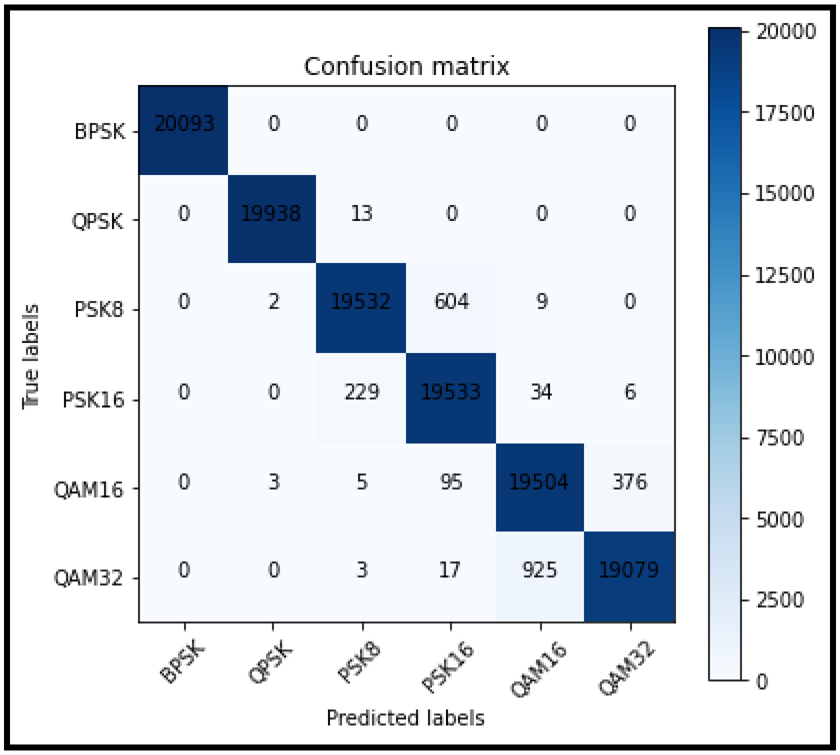


Figure No. 59 Training and validation accuracy of Model #15



	BPSK	QPSK	PSK8	PSK16	QAM16	QAM32
BPSK	20093	0	0	0	0	0
QPSK	0	19938	13	0	0	0
PSK8	0	2	19532	604	9	0
PSK16	0	0	229	19533	34	6
QAM16	0	3	5	95	19504	376
QAM32	0	0	3	17	925	19079

Figure No. 60 Performance result of model #15 is **98%** on testing dataset

Model # 16 “Mod_3L_4D_16S16_TF_NLnL_Rayleighfadded_N.h5”

The “Mod_3L_4D_16S16_TF_NLnL_Rayleighfadded_N.h5” Conv1D model utilized the pre-trained model #No.11 “Mod_3L_4D_NLnL_tan_16S16.h5” as the starting point and has been trained on the dataset “MOD_Rayleighfadded_16Samp_N_NN1.h5”. Model’s further characteristics are tabulated in the table no. 19.

Table No. 19 Characteristics of Model #16

Model’s Characteristics		**DataSet Attributes**	
Input Layers	Same as model No#11 “Mod_3L_4D_NLnL_tan _16S16.h5”	Symbol Rate	64
Activation Func	Same as model No#11 “Mod_3L_4D_NLnL_tan _16S16.h5”	Channel Nature	Rayleigh fading
Transfer Learning	Trained on pre-trained model No#11 “Mod_3L_4D_NLnL_tan _16S16.h5”	Input format	Unnormalized & shape 256x4

The model #16’s training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 231s 15ms/step - loss: 0.7436 - accuracy: 0.7147 - val_loss: 0.6104 - val_accuracy: 0.7339
Epoch 2/20
15000/15000 [==============================] - 223s 15ms/step - loss: 0.6181 - accuracy: 0.7399 - val_loss: 0.5670 - val_accuracy: 0.7567
Epoch 3/20
15000/15000 [==============================] - 220s 15ms/step - loss: 0.5933 - accuracy: 0.7471 - val_loss: 0.5812 - val_accuracy: 0.7326
Epoch 4/20
15000/15000 [==============================] - 218s 15ms/step - loss: 0.5799 - accuracy: 0.7517 - val_loss: 0.5246 - val_accuracy: 0.7702
Epoch 5/20
15000/15000 [==============================] - 219s 15ms/step - loss: 0.5684 - accuracy: 0.7559 - val_loss: 0.5144 - val_accuracy: 0.7784
Epoch 6/20
15000/15000 [==============================] - 219s 15ms/step - loss: 0.5595 - accuracy: 0.7591 - val_loss: 0.5257 - val_accuracy: 0.7787
Epoch 7/20
15000/15000 [==============================] - 224s 15ms/step - loss: 0.5461 - accuracy: 0.7636 - val_loss: 0.5170 - val_accuracy: 0.7789
Epoch 8/20
15000/15000 [==============================] - 230s 15ms/step - loss: 0.5411 - accuracy: 0.7655 - val_loss: 0.5042 - val_accuracy: 0.7796
Epoch 9/20
15000/15000 [==============================] - 229s 15ms/step - loss: 0.5359 - accuracy: 0.7686 - val_loss: 0.5046 - val_accuracy: 0.7802
Epoch 10/20
15000/15000 [==============================] - 229s 15ms/step - loss: 0.5337 - accuracy: 0.7685 - val_loss: 0.5041 - val_accuracy: 0.7796
Epoch 11/20
15000/15000 [==============================] - 230s 15ms/step - loss: 0.5319 - accuracy: 0.7692 - val_loss: 0.5114 - val_accuracy: 0.7775
Epoch 12/20
15000/15000 [==============================] - 229s 15ms/step - loss: 0.5299 - accuracy: 0.7709 - val_loss: 0.5190 - val_accuracy: 0.7780
Epoch 13/20
15000/15000 [==============================] - 230s 15ms/step - loss: 0.5304 - accuracy: 0.7699 - val_loss: 0.5278 - val_accuracy: 0.7693
Epoch 14/20
15000/15000 [==============================] - 318s 21ms/step - loss: 0.5289 - accuracy: 0.7705 - val_loss: 0.4896 - val_accuracy: 0.7871
```

```
Epoch 15/20
15000/15000 [==============================] - 318s 21ms/step - loss: 0.5275 - accuracy: 0.7708 - val_loss: 0.5306 - val_accuracy: 0.7711
Epoch 16/20
15000/15000 [==============================] - 331s 22ms/step - loss: 0.5268 - accuracy: 0.7718 - val_loss: 0.4939 - val_accuracy: 0.7790
Epoch 17/20
15000/15000 [==============================] - 329s 22ms/step - loss: 0.5269 - accuracy: 0.7711 - val_loss: 0.5440 - val_accuracy: 0.7614
Epoch 18/20
15000/15000 [==============================] - 340s 23ms/step - loss: 0.5246 - accuracy: 0.7727 - val_loss: 0.4883 - val_accuracy: 0.7865
Epoch 19/20
15000/15000 [==============================] - 327s 22ms/step - loss: 0.5243 - accuracy: 0.7727 - val_loss: 0.5020 - val_accuracy: 0.7839
Epoch 20/20
15000/15000 [==============================] - 331s 22ms/step - loss: 0.5232 - accuracy: 0.7732 - val_loss: 0.5670 - val_accuracy: 0.7443
```

The “training & validation” loss and accuracy of model#16 is shown in figure no. 61 and 62 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 63.

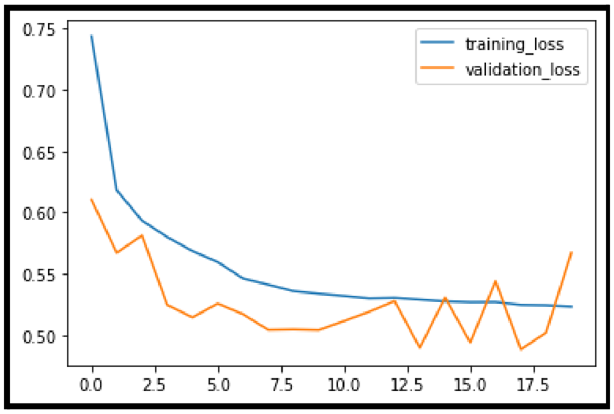


Figure No. 61 Training and validation loss of Model #16

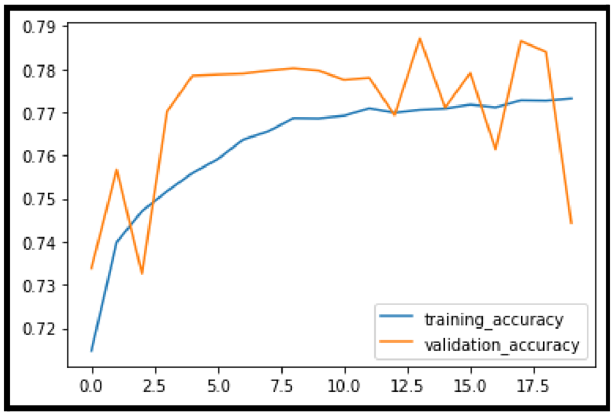


Figure No. 62 Training and validation accuracy of Model #16

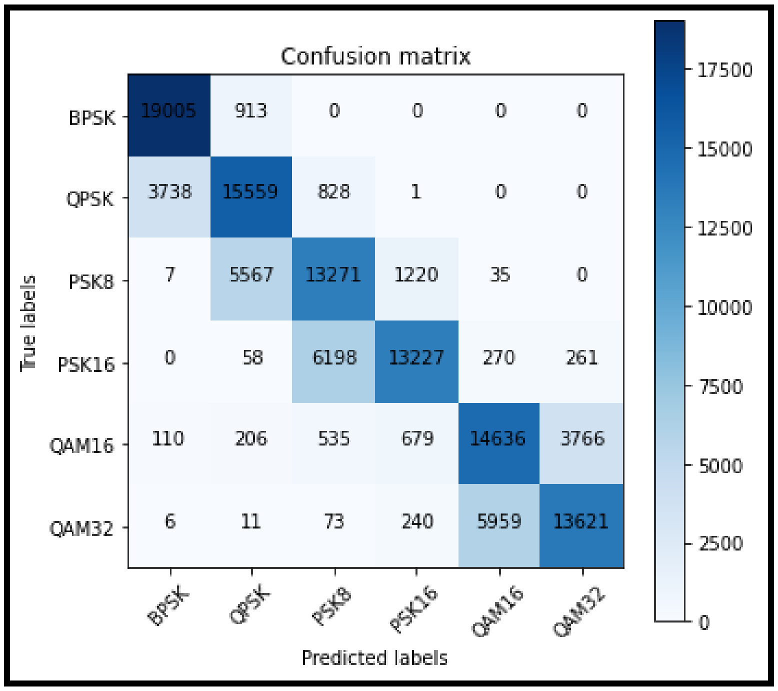


Figure No. 63 Performance result of model #16 is **74.43%** on testing dataset

Model # 17 "Mod_3L_4D_16S16_TF_NLnL_0p3ricianfadded_N.h5"

The "Mod_3L_4D_16S16_TF_NLnL_0p3ricianfadded_N.h5" Conv1D model utilized the pre-trained model #No.11 "Mod_3L_4D_NLnL_tan_16S16.h5" as the starting point and has been trained on the dataset "MOD_Rician0p3fadded_16Samp_N_NN1.h5". Model's further characteristics are tabulated in the table no. 20.

Table No. 20 Characteristics of Model #17

Model's Characteristics		**DataSet Attributes**	
Input Layers	Same as model No#11 "Mod_3L_4D_NLnL_tan _16S16.h5"	Symbol Rate	64
Activation Func	Same as model No#11 "Mod_3L_4D_NLnL_tan _16S16.h5"	Channel Nature	Rician Fading with k=0.3
Transfer Learning	Trained on pre-trained model No#11 "Mod_3L_4D_NLnL_tan _16S16.h5"	Input format	Unnormalized & shape 256x4

The model #17's training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 232s 15ms/step - loss: 0.5051 - accuracy: 0.8075 - val_loss: 0.3936 - val_accuracy: 0.8381
Epoch 2/20
15000/15000 [==============================] - 220s 15ms/step - loss: 0.4162 - accuracy: 0.8263 - val_loss: 0.3725 - val_accuracy: 0.8419
Epoch 3/20
15000/15000 [==============================] - 221s 15ms/step - loss: 0.3975 - accuracy: 0.8324 - val_loss: 0.3692 - val_accuracy: 0.8347
Epoch 4/20
15000/15000 [==============================] - 220s 15ms/step - loss: 0.3867 - accuracy: 0.8364 - val_loss: 0.3654 - val_accuracy: 0.8509
Epoch 5/20
15000/15000 [==============================] - 223s 15ms/step - loss: 0.3802 - accuracy: 0.8395 - val_loss: 0.3337 - val_accuracy: 0.8585
Epoch 6/20
15000/15000 [==============================] - 225s 15ms/step - loss: 0.3744 - accuracy: 0.8411 - val_loss: 0.3490 - val_accuracy: 0.8545
Epoch 7/20
15000/15000 [==============================] - 228s 15ms/step - loss: 0.3710 - accuracy: 0.8427 - val_loss: 0.3669 - val_accuracy: 0.8392
Epoch 8/20
15000/15000 [==============================] - 232s 15ms/step - loss: 0.3699 - accuracy: 0.8432 - val_loss: 0.3558 - val_accuracy: 0.8457
Epoch 9/20
15000/15000 [==============================] - 239s 16ms/step - loss: 0.3658 - accuracy: 0.8449 - val_loss: 0.3898 - val_accuracy: 0.8244
Epoch 10/20
15000/15000 [==============================] - 241s 16ms/step - loss: 0.3627 - accuracy: 0.8454 - val_loss: 0.3535 - val_accuracy: 0.8535
Epoch 11/20
15000/15000 [==============================] - 242s 16ms/step - loss: 0.3606 - accuracy: 0.8463 - val_loss: 0.3222 - val_accuracy: 0.8608
Epoch 12/20
15000/15000 [==============================] - 242s 16ms/step - loss: 0.3599 - accuracy: 0.8470 - val_loss: 0.3393 - val_accuracy: 0.8542
Epoch 13/20
15000/15000 [==============================] - 242s 16ms/step - loss: 0.3601 - accuracy: 0.8467 - val_loss: 0.3294 - val_accuracy: 0.8626
Epoch 14/20
15000/15000 [==============================] - 241s 16ms/step - loss: 0.3580 - accuracy: 0.8479 - val_loss: 0.3991 - val_accuracy: 0.8319
Epoch 15/20
15000/15000 [==============================] - 243s 16ms/step - loss: 0.3583 - accuracy: 0.8477 - val_loss: 0.3173 - val_accuracy: 0.8630
Epoch 16/20
15000/15000 [==============================] - 241s 16ms/step - loss: 0.3560 - accuracy: 0.8480 - val_loss: 0.3290 - val_accuracy: 0.8579
Epoch 17/20
15000/15000 [==============================] - 244s 16ms/step - loss: 0.3565 - accuracy: 0.8485 - val_loss: 0.3129 - val_accuracy: 0.8650
Epoch 18/20
15000/15000 [==============================] - 245s 16ms/step - loss: 0.3563 - accuracy: 0.8482 - val_loss: 0.3494 - val_accuracy: 0.8479
Epoch 19/20
15000/15000 [==============================] - 242s 16ms/step - loss: 0.3567 - accuracy: 0.8478 - val_loss: 0.3553 - val_accuracy: 0.8515
Epoch 20/20
15000/15000 [==============================] - 242s 16ms/step - loss: 0.3563 - accuracy: 0.8481 - val_loss: 0.3575 - val_accuracy: 0.8417
```

The "training & validation" loss and accuracy of model#17 is shown in figure no. 64 and 65 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 66.

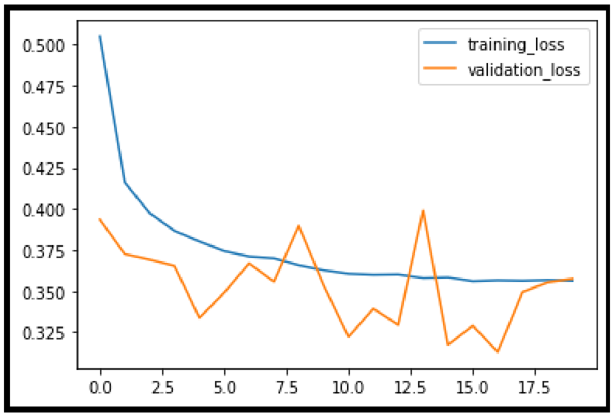


Figure No. 64 Training and validation loss of Model #17

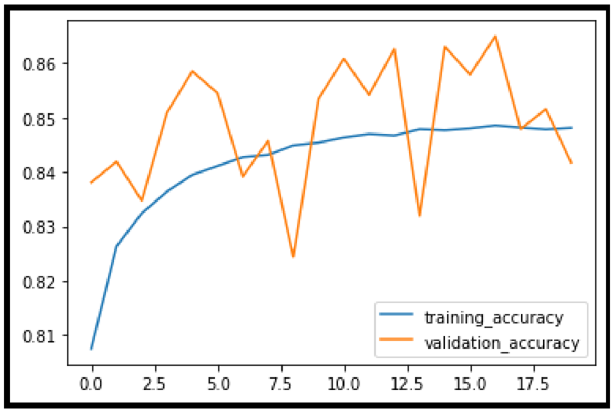


Figure No. 65 Training and validation accuracy of Model #17

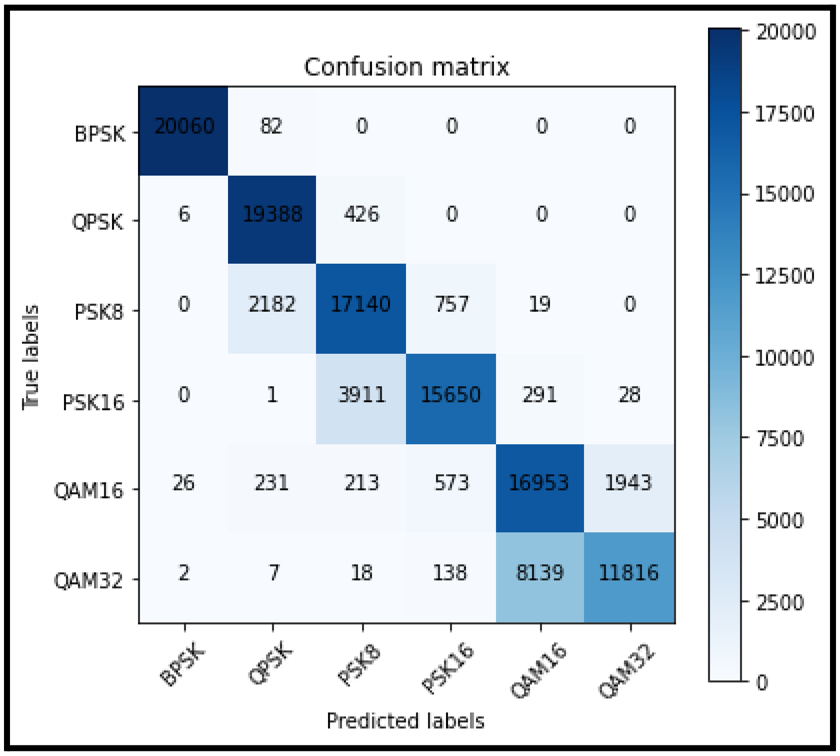


Figure No. 66 Performance result of model #17 is **84.17%** on testing dataset

Model # 18 “Mod_3L_4D_16S16_TF_NLnL_0p7ricianfadded_N.h5”

The “Mod_3L_4D_16S16_TF_NLnL_0p7ricianfadded_N.h5” Conv1D model utilized the pre-trained model #No.11 “Mod_3L_4D_NLnL_tan_16S16.h5” as the starting point and has been trained on the dataset “MOD_Rician0p7fadded_16Samp_N_NN1.h5”. Model’s further characteristics are tabulated in the table no. 21.

Table No. 21 Characteristics of Model #18

Model’s Characteristics		**DataSet Attributes**	
Input Layers	Same as model No#11 “Mod_3L_4D_NLnL_tan _16S16.h5”	Symbol Rate	64
Activation Func	Same as model No#11 “Mod_3L_4D_NLnL_tan _16S16.h5”	Channel Nature	Rician Fading with k=0.7
Transfer Learning	Trained on pre-trained model No#11 “Mod_3L_4D_NLnL_tan _16S16.h5”	Input format	Unnormalized & shape 256x4

The model #18’s training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 265s 18ms/step - loss: 0.3942 - accuracy: 0.8542 - val_loss: 0.3243 - val_accuracy: 0.8716
Epoch 2/20
15000/15000 [==============================] - 253s 17ms/step - loss: 0.3082 - accuracy: 0.8756 - val_loss: 0.2399 - val_accuracy: 0.8972
Epoch 3/20
15000/15000 [==============================] - 254s 17ms/step - loss: 0.2854 - accuracy: 0.8863 - val_loss: 0.2375 - val_accuracy: 0.9030
Epoch 4/20
15000/15000 [==============================] - 251s 17ms/step - loss: 0.2735 - accuracy: 0.8911 - val_loss: 0.2543 - val_accuracy: 0.8965
Epoch 5/20
15000/15000 [==============================] - 252s 17ms/step - loss: 0.2656 - accuracy: 0.8942 - val_loss: 0.2150 - val_accuracy: 0.9140
Epoch 6/20
15000/15000 [==============================] - 256s 17ms/step - loss: 0.2608 - accuracy: 0.8956 - val_loss: 0.2292 - val_accuracy: 0.9045
Epoch 7/20
15000/15000 [==============================] - 258s 17ms/step - loss: 0.2575 - accuracy: 0.8970 - val_loss: 0.2364 - val_accuracy: 0.9029
Epoch 8/20
15000/15000 [==============================] - 258s 17ms/step - loss: 0.2542 - accuracy: 0.8988 - val_loss: 0.2027 - val_accuracy: 0.9171
Epoch 9/20
15000/15000 [==============================] - 258s 17ms/step - loss: 0.2516 - accuracy: 0.8999 - val_loss: 0.2078 - val_accuracy: 0.9143
Epoch 10/20
15000/15000 [==============================] - 275s 18ms/step - loss: 0.2494 - accuracy: 0.9004 - val_loss: 0.2195 - val_accuracy: 0.9121
Epoch 11/20
15000/15000 [==============================] - 256s 17ms/step - loss: 0.2492 - accuracy: 0.9006 - val_loss: 0.1991 - val_accuracy: 0.9178
Epoch 12/20
15000/15000 [==============================] - 255s 17ms/step - loss: 0.2472 - accuracy: 0.9017 - val_loss: 0.2428 - val_accuracy: 0.9026
Epoch 13/20
15000/15000 [==============================] - 258s 17ms/step - loss: 0.2472 - accuracy: 0.9015 - val_loss: 0.2492 - val_accuracy: 0.9014
Epoch 14/20
15000/15000 [==============================] - 255s 17ms/step - loss: 0.2458 - accuracy: 0.9023 - val_loss: 0.3017 - val_accuracy: 0.8880
```

```
Epoch 15/20
15000/15000 [==============================] - 253s 17ms/step - loss: 0.2448 - accuracy: 0.9028 - val_loss: 0.2071 - val_accuracy: 0.9143
Epoch 16/20
15000/15000 [==============================] - 257s 17ms/step - loss: 0.2457 - accuracy: 0.9023 - val_loss: 0.2006 - val_accuracy: 0.9169
Epoch 17/20
15000/15000 [==============================] - 256s 17ms/step - loss: 0.2435 - accuracy: 0.9031 - val_loss: 0.2074 - val_accuracy: 0.9158
Epoch 18/20
15000/15000 [==============================] - 253s 17ms/step - loss: 0.2425 - accuracy: 0.9031 - val_loss: 0.2407 - val_accuracy: 0.9048
Epoch 19/20
15000/15000 [==============================] - 262s 17ms/step - loss: 0.2432 - accuracy: 0.9034 - val_loss: 0.2156 - val_accuracy: 0.9117
Epoch 20/20
15000/15000 [==============================] - 257s 17ms/step - loss: 0.2427 - accuracy: 0.9039 - val_loss: 0.2011 - val_accuracy: 0.9187
```

The "training & validation" loss and accuracy of model#18 is shown in figure no. 67 and 68 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 69.

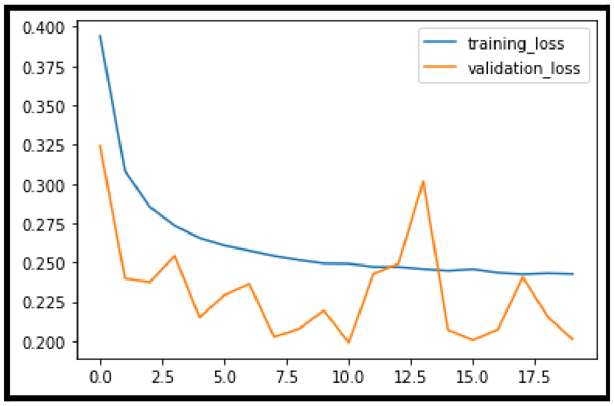


Figure No. 67 Training and validation loss of Model #18

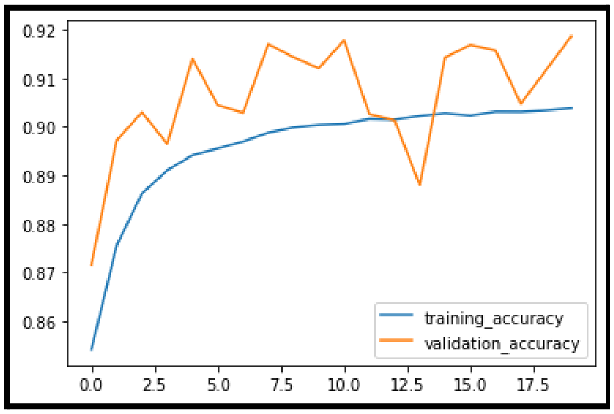


Figure No. 68 Training and validation accuracy of Model #18

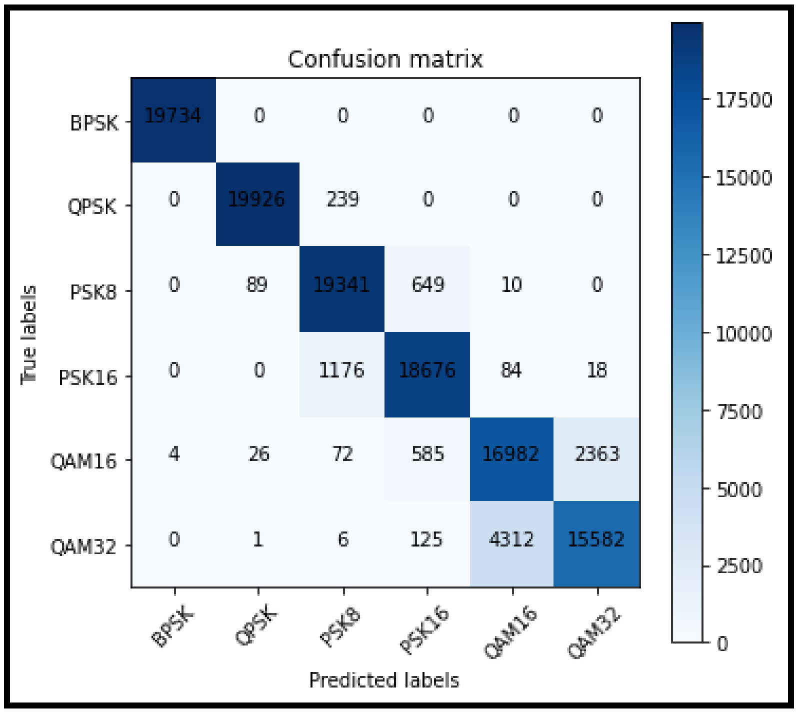


Figure No. 69 Performance result of model #18 is **91.87%** on testing dataset

Model # 19 "Mod_3L_4D_16S16_TF_NLnL_0p9ricianfadded_N.h5"

The "Mod_3L_4D_16S16_TF_NLnL_0p9ricianfadded_N.h5" Conv1D model utilized the pre-trained model #No.11 "Mod_3L_4D_NLnL_tan_16S16.h5" as the starting point and has been trained on the dataset "MOD_Rician0p9fadded_16Samp_N_NN1.h5". Model's further characteristics are tabulated in the table no. 22.

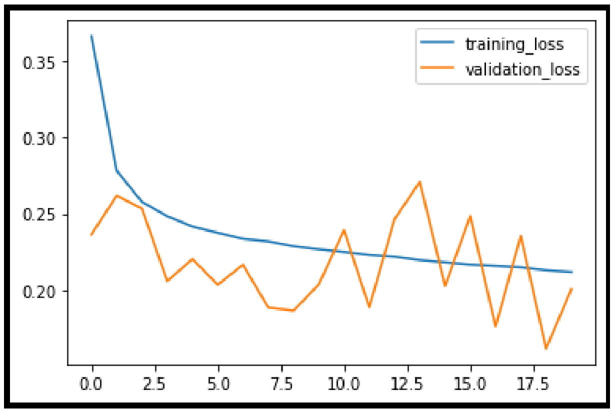
Table No. 22 Characteristics of Model #19

Model's Characteristics		**DataSet Attributes**	
Input Layers	Same as model No#11 "Mod_3L_4D_NLnL_tan _16S16.h5"	Symbol Rate	64
Activation Func	Same as model No#11 "Mod_3L_4D_NLnL_tan _16S16.h5"	Channel Nature	Rician Fading with k=0.9
Transfer Learning	Trained on pre-trained model No#11 "Mod_3L_4D_NLnL_tan _16S16.h5"	Input format	Unnormalized & shape 256x4

The model #19's training log is given below expressing the computational complexity:

```
Epoch 1/20
15000/15000 [==============================] - 313s 21ms/step - loss: 0.3667 - accuracy: 0.8675 - val_loss: 0.2364 - val_accuracy: 0.8992
Epoch 2/20
15000/15000 [==============================] - 315s 21ms/step - loss: 0.2783 - accuracy: 0.8899 - val_loss: 0.2619 - val_accuracy: 0.8906
Epoch 3/20
15000/15000 [==============================] - 328s 22ms/step - loss: 0.2578 - accuracy: 0.8969 - val_loss: 0.2534 - val_accuracy: 0.8917
Epoch 4/20
15000/15000 [==============================] - 327s 22ms/step - loss: 0.2484 - accuracy: 0.9009 - val_loss: 0.2058 - val_accuracy: 0.9163
Epoch 5/20
15000/15000 [==============================] - 339s 23ms/step - loss: 0.2416 - accuracy: 0.9034 - val_loss: 0.2202 - val_accuracy: 0.9121
Epoch 6/20
15000/15000 [==============================] - 327s 22ms/step - loss: 0.2375 - accuracy: 0.9051 - val_loss: 0.2034 - val_accuracy: 0.9188
Epoch 7/20
15000/15000 [==============================] - 330s 22ms/step - loss: 0.2336 - accuracy: 0.9064 - val_loss: 0.2165 - val_accuracy: 0.9113
Epoch 8/20
15000/15000 [==============================] - 230s 15ms/step - loss: 0.2319 - accuracy: 0.9077 - val_loss: 0.1884 - val_accuracy: 0.9215
Epoch 9/20
15000/15000 [==============================] - 229s 15ms/step - loss: 0.2288 - accuracy: 0.9086 - val_loss: 0.1864 - val_accuracy: 0.9234
Epoch 10/20
15000/15000 [==============================] - 233s 16ms/step - loss: 0.2268 - accuracy: 0.9099 - val_loss: 0.2036 - val_accuracy: 0.9156
Epoch 11/20
15000/15000 [==============================] - 235s 16ms/step - loss: 0.2250 - accuracy: 0.9108 - val_loss: 0.2392 - val_accuracy: 0.9029
Epoch 12/20
15000/15000 [==============================] - 235s 16ms/step - loss: 0.2231 - accuracy: 0.9116 - val_loss: 0.1886 - val_accuracy: 0.9222
Epoch 13/20
15000/15000 [==============================] - 237s 16ms/step - loss: 0.2218 - accuracy: 0.9122 - val_loss: 0.2465 - val_accuracy: 0.8955
Epoch 14/20
15000/15000 [==============================] - 239s 16ms/step - loss: 0.2196 - accuracy: 0.9131 - val_loss: 0.2710 - val_accuracy: 0.8889
Epoch 15/20
15000/15000 [==============================] - 237s 16ms/step - loss: 0.2181 - accuracy: 0.9142 - val_loss: 0.2027 - val_accuracy: 0.9198
Epoch 16/20
15000/15000 [==============================] - 229s 15ms/step - loss: 0.2166 - accuracy: 0.9147 - val_loss: 0.2485 - val_accuracy: 0.9007
Epoch 17/20
15000/15000 [==============================] - 231s 15ms/step - loss: 0.2158 - accuracy: 0.9150 - val_loss: 0.1759 - val_accuracy: 0.9305
Epoch 18/20
15000/15000 [==============================] - 228s 15ms/step - loss: 0.2149 - accuracy: 0.9155 - val_loss: 0.2356 - val_accuracy: 0.9072
Epoch 19/20
15000/15000 [==============================] - 232s 15ms/step - loss: 0.2129 - accuracy: 0.9162 - val_loss: 0.1614 - val_accuracy: 0.9345
Epoch 20/20
15000/15000 [==============================] - 233s 16ms/step - loss: 0.2118 - accuracy: 0.9174 - val_loss: 0.2004 - val_accuracy: 0.9185
```

The "training & validation" loss and accuracy of model#19 is shown in figure no. 70 and 71 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 72.



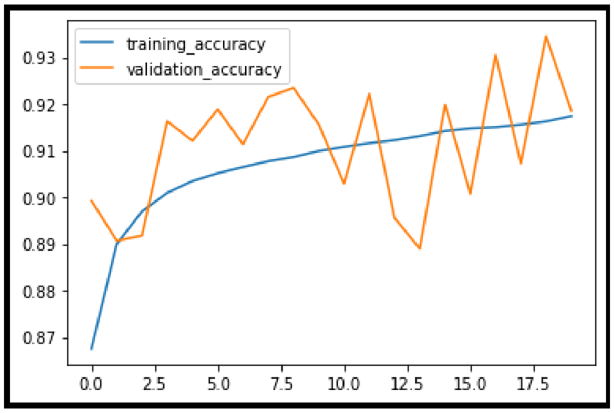


Figure No. 70 Training and validation loss of Model #19

Figure No. 71 Training and validation accuracy of Model #19

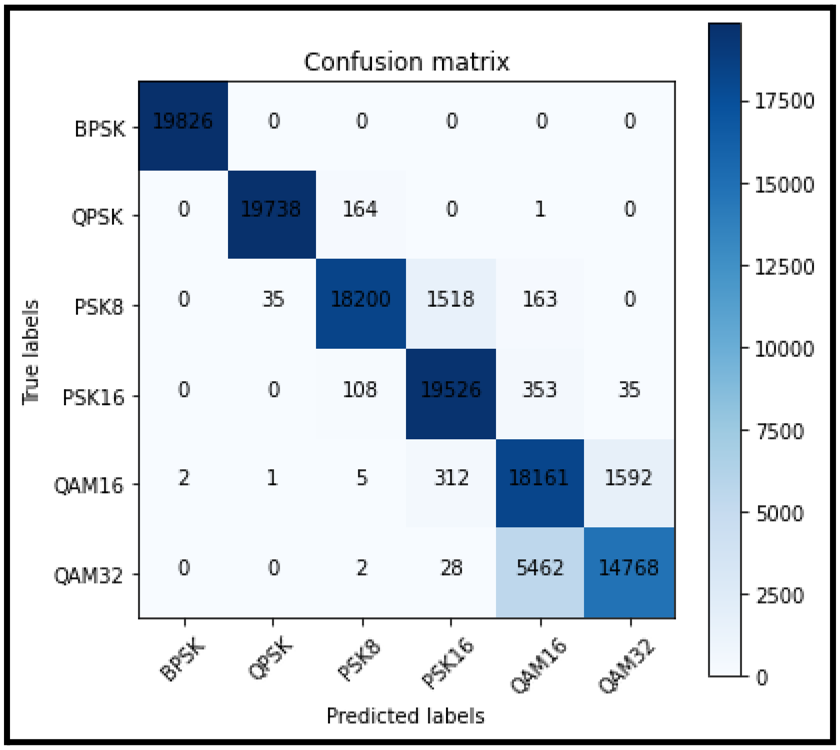


Figure No. 72 Performance result of model #19 is **91.85%** on testing dataset

Conv1D model based classification of Coding schemes

The similar methodology "classical 1D Convolutional Neural Network (Conv1D)" in classification of coding schemes has been worked out best. Proposed NN model is tuned by the results and findings collected from implying the modifications in following areas:

- ➔ **Number of Hidden Layers** i.e. 4, 5 or 6 convolution layers for feature extraction
- ➔ **Block Input size** i.e. 128, 256, 512, 1024 or 2048 block size for model's input
- ➔ **Model's Output window Size** i.e 3 or 6 output neurons with fixed label size of 3

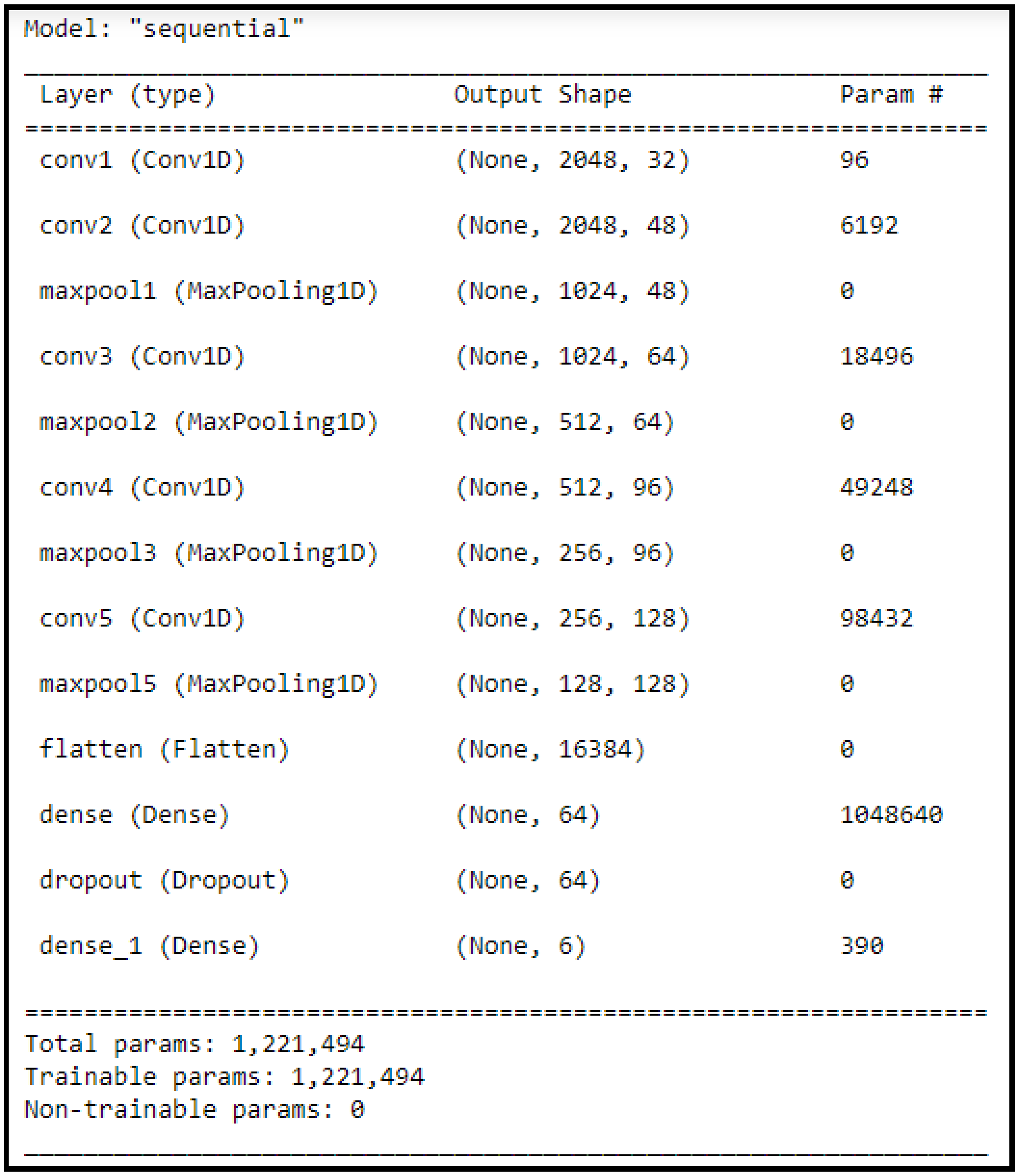
And, the Model's performance has been evaluated on the basis of its agility and accuracy. The best out of all Conv1D model for classification of coding schemes is discussed in detail.

Model # 01C "COD_TrainingData_5layer_1D_2048_06.h5"

The "COD_TrainingData_5layer_1D_2048_06.h5" Conv1D model is shown in figure no. 73 as trained on the dataset "COD_TrainingData.hdf5". Model's further characteristics are tabulated in the table no. 23.

Table No. 23 Characteristics of Model #01C

Model's Characteristics		**DataSet Attributes**	
Input Layers	05	Coding Schemes	LDPC, Turbo & Polar
Activation Func	**relu** act. func. with all hidden layers except the last decision layer utilizing softmax.	Encoded message length for Each Coding Scheme	LDPC → 1296 Turbo → 1584 Polar → 1024
Output neuron size	06	Input format	● Binary value data ● shape 2048x1 ● Block contain ~2 cascaded 2 encoded messages



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1 (Conv1D)              (None, 2048, 32)          96

 conv2 (Conv1D)              (None, 2048, 48)          6192

 maxpool1 (MaxPooling1D)     (None, 1024, 48)          0

 conv3 (Conv1D)              (None, 1024, 64)          18496

 maxpool2 (MaxPooling1D)     (None, 512, 64)           0

 conv4 (Conv1D)              (None, 512, 96)           49248

 maxpool3 (MaxPooling1D)     (None, 256, 96)           0

 conv5 (Conv1D)              (None, 256, 128)          98432

 maxpool5 (MaxPooling1D)     (None, 128, 128)          0

 flatten (Flatten)           (None, 16384)             0

 dense (Dense)               (None, 64)                1048640

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 6)                 390

=================================================================
Total params: 1,221,494
Trainable params: 1,221,494
Non-trainable params: 0
_________________________________________________________________
```

Figure No. 74 The model #01C "COD_TrainingData_5layer_1D_2048_06.h5"

The model #01C's training log is given below expressing the computational complexity:

```
Epoch 1/10
7500/7500 [==============================] - 569s 76ms/step - loss: 0.7027 - accuracy: 0.5544 - val_loss: 0.4872 - val_accuracy: 0.6688
Epoch 2/10
7500/7500 [==============================] - 567s 76ms/step - loss: 0.1205 - accuracy: 0.9246 - val_loss: 0.0036 - val_accuracy: 0.9991
Epoch 3/10
7500/7500 [==============================] - 639s 85ms/step - loss: 0.0084 - accuracy: 0.9978 - val_loss: 0.0118 - val_accuracy: 0.9980
Epoch 4/10
7500/7500 [==============================] - 671s 89ms/step - loss: 0.0063 - accuracy: 0.9984 - val_loss: 0.0026 - val_accuracy: 0.9994
Epoch 5/10
7500/7500 [==============================] - 672s 90ms/step - loss: 0.0056 - accuracy: 0.9986 - val_loss: 0.0025 - val_accuracy: 0.9994
Epoch 6/10
7500/7500 [==============================] - 730s 97ms/step - loss: 0.0046 - accuracy: 0.9990 - val_loss: 0.0020 - val_accuracy: 0.9996
Epoch 7/10
7500/7500 [==============================] - 765s 102ms/step - loss: 0.0040 - accuracy: 0.9990 - val_loss: 0.0023 - val_accuracy: 0.9996
Epoch 8/10
7500/7500 [==============================] - 758s 101ms/step - loss: 0.0038 - accuracy: 0.9991 - val_loss: 0.0022 - val_accuracy: 0.9995
Epoch 9/10
7500/7500 [==============================] - 771s 103ms/step - loss: 0.0040 - accuracy: 0.9991 - val_loss: 0.0076 - val_accuracy: 0.9991
Epoch 10/10
7500/7500 [==============================] - 756s 101ms/step - loss: 0.0031 - accuracy: 0.9992 - val_loss: 0.0028 - val_accuracy: 0.9992
```

The "training & validation" loss and accuracy of model#01C is shown in figure no. 75 and 76 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 77.

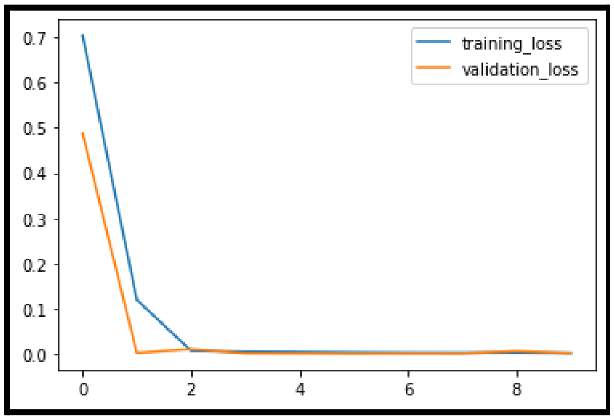


Figure No. 75 Training and validation loss of Model #01C

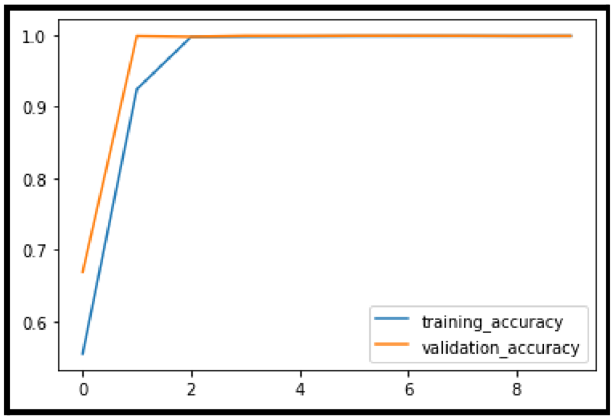


Figure No. 76 Training and validation accuracy of Model #01C

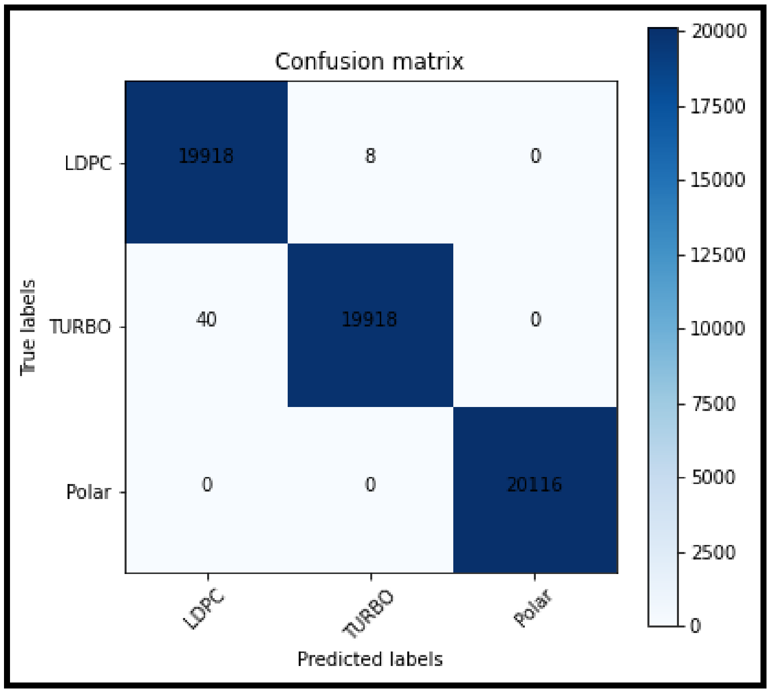


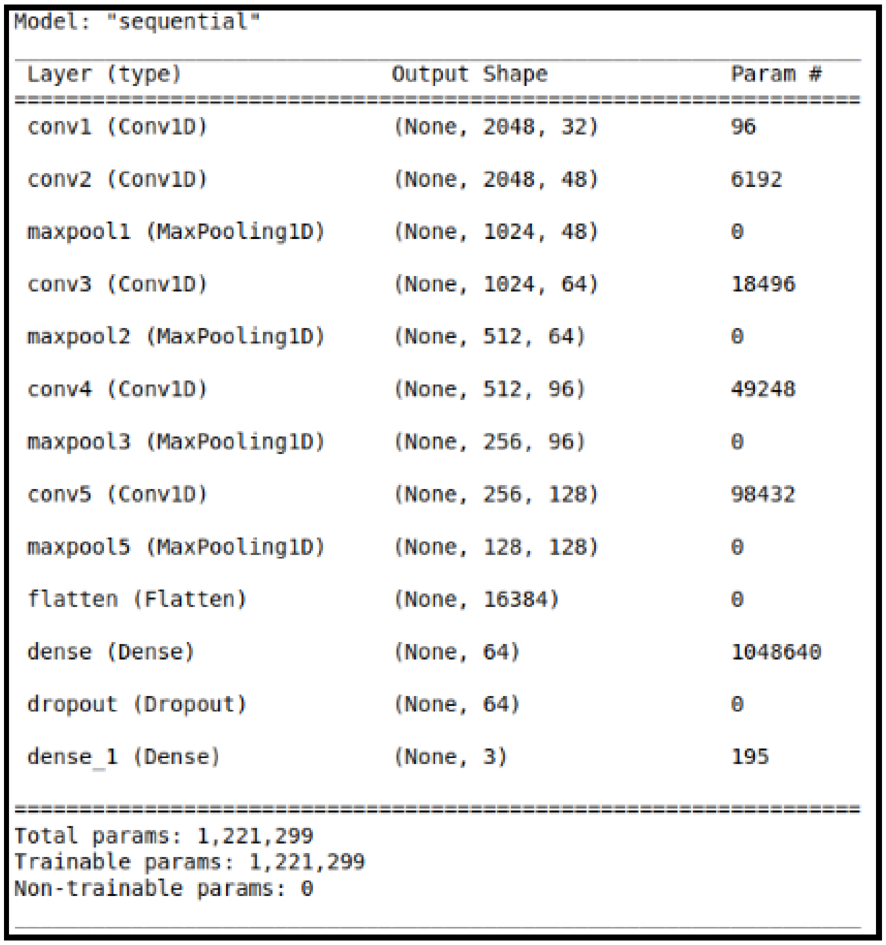
Figure No. 77 Performance result of model #01C is **99.92%** on testing dataset

Model # 02C "COD_TrainingData_5layer_1D_2048_03.h5"

The "COD_TrainingData_5layer_1D_2048_03.h5" Conv1D model is shown in figure no. 78 as trained on the dataset "COD_TrainingData.hdf5". Model's further characteristics are tabulated in the table no. 24.

Table No. 24 Characteristics of Model #02C

Model's Characteristics		**DataSet Attributes**	
Input Layers	05	Coding Schemes	LDPC, Turbo & Polar
Activation Func	**relu** act. func. with all hidden layers except the last decision layer utilizing softmax.	Encoded message length for Each Coding Scheme	LDPC → 1296 Turbo → 1584 Polar → 1024
Output neuron size	03	Input format	● Binary value data ● shape 2048x1 ● Block contain ~2 cascaded 2 encoded messages



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1 (Conv1D)              (None, 2048, 32)          96

 conv2 (Conv1D)              (None, 2048, 48)          6192

 maxpool1 (MaxPooling1D)     (None, 1024, 48)          0

 conv3 (Conv1D)              (None, 1024, 64)          18496

 maxpool2 (MaxPooling1D)     (None, 512, 64)           0

 conv4 (Conv1D)              (None, 512, 96)           49248

 maxpool3 (MaxPooling1D)     (None, 256, 96)           0

 conv5 (Conv1D)              (None, 256, 128)          98432

 maxpool5 (MaxPooling1D)     (None, 128, 128)          0

 flatten (Flatten)           (None, 16384)             0

 dense (Dense)               (None, 64)                1048640

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 3)                 195

=================================================================
Total params: 1,221,299
Trainable params: 1,221,299
Non-trainable params: 0
_________________________________________________________________
```

Figure No. 78 The model #02C "COD_TrainingData_5layer_1D_2048_03.h5"

The model #02C's training log is given below expressing the computational complexity:

```
Epoch 1/10
7500/7500 [==============================] - 601s 80ms/step - loss: 1.0987 - accuracy: 0.3325 - val_loss: 1.0987 - val_accuracy: 0.3314
Epoch 2/10
7500/7500 [==============================] - 573s 76ms/step - loss: 1.0987 - accuracy: 0.3328 - val_loss: 1.0986 - val_accuracy: 0.3335
Epoch 3/10
7500/7500 [==============================] - 579s 77ms/step - loss: 1.0987 - accuracy: 0.3330 - val_loss: 1.0987 - val_accuracy: 0.3314
Epoch 4/10
7500/7500 [==============================] - 584s 78ms/step - loss: 1.0986 - accuracy: 0.3348 - val_loss: 1.0986 - val_accuracy: 0.3351
Epoch 5/10
7500/7500 [==============================] - 582s 78ms/step - loss: 1.0987 - accuracy: 0.3334 - val_loss: 1.0986 - val_accuracy: 0.3335
Epoch 6/10
7500/7500 [==============================] - 585s 78ms/step - loss: 1.0987 - accuracy: 0.3308 - val_loss: 1.0986 - val_accuracy: 0.3351
Epoch 7/10
7500/7500 [==============================] - 585s 78ms/step - loss: 1.0987 - accuracy: 0.3329 - val_loss: 1.0986 - val_accuracy: 0.3314
Epoch 8/10
7500/7500 [==============================] - 589s 79ms/step - loss: 1.0987 - accuracy: 0.3327 - val_loss: 1.0986 - val_accuracy: 0.3335
Epoch 9/10
7500/7500 [==============================] - 588s 78ms/step - loss: 1.0987 - accuracy: 0.3314 - val_loss: 1.0986 - val_accuracy: 0.3351
Epoch 10/10
7500/7500 [==============================] - 577s 77ms/step - loss: 1.0987 - accuracy: 0.3326 - val_loss: 1.0987 - val_accuracy: 0.3314
```

The “training & validation” loss and accuracy of model#02C is shown in figure no. 79 and 80 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 81.

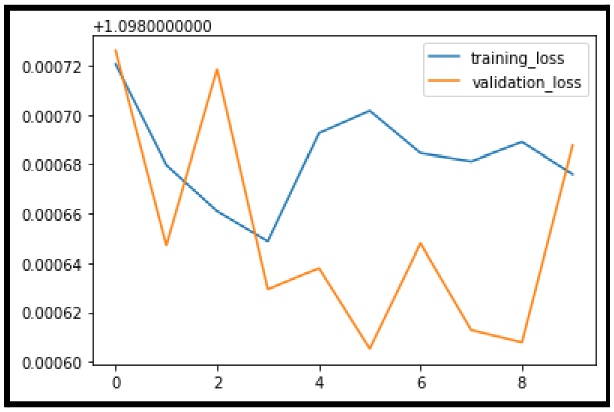


Figure No. 79 Training and validation loss of Model #02C

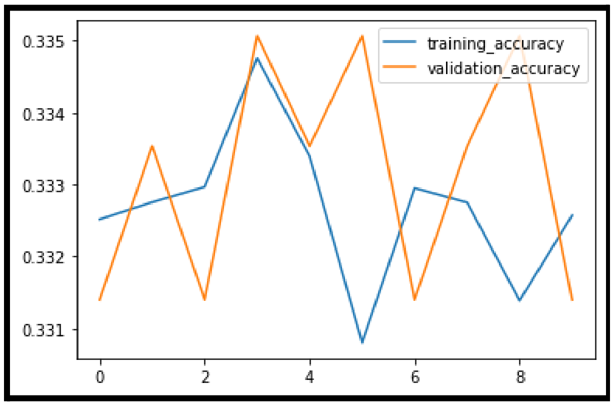


Figure No. 80 Training and validation accuracy of Model #02C

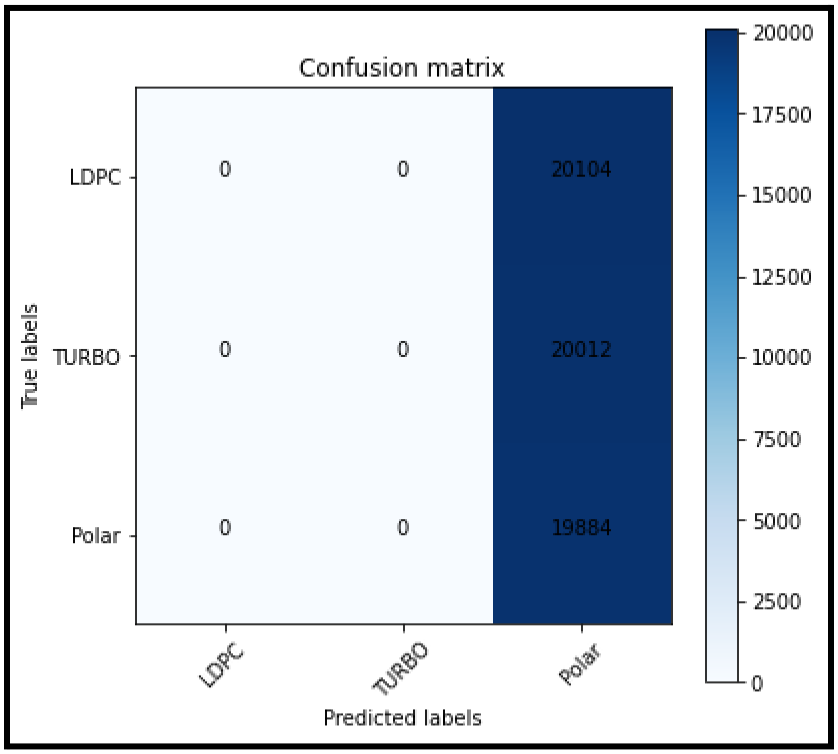


Figure No. 81 Performance result of model #02C is 33.14% on testing dataset

Simulation Results & Findings

In the last discussion, all proposed Conv1D models are trained and rigorously verified over the generated dataset. This whole activity helped to obtain the best model out of all. The overall summary regarding the accuracy of the trained models is provided in table no. 25.

Table No. 25 Accuracy summary of trained NN models (Model under Test)

S. No.	Classification Scheme	Model Name	Accuracy %
1.	Modulation	Mod_3L_4D_8S32.h5	98.8
2.	Modulation	Mod_3L_4D_16S16.h5	99.05
3.	Modulation	Mod_3L_4D_32S8.h5	97.4
4.	Modulation	Mod_2L_4D_16S1.h5	98.7
5.	Modulation	Mod_3L_4D_16S16_TF_Rayleigh_N.h5	52.85
6.	Modulation	Mod_3L_4D_16S16_TF_0p3rician_N.h5	75.5
7.	Modulation	Mod_3L_4D_16S16_TF_0p7rician_N.h5	90.25
8.	Modulation	Mod_3L_4D_16S16_TF_0p9rician_N.h5	93.05
9.	Modulation	Mod_3L_4D_NL_16S16.h5	16.75
10.	Modulation	Mod_3L_4D_NL_tan_16S16.h5	99.8
11.	Modulation	Mod_3L_4D_NLnL_tan_16S16.h5	98.6
12.	Modulation	Mod_3L_4D_16S16_TF_NL_Rayleighfadded_N.h5	85.1
13.	Modulation	Mod_3L_4D_16S16_TF_NL_0p3ricianfadded_N	92.6
14.	Modulation	Mod_3L_4D_16S16_TF_NL_0p7ricianfadded_N.h5	96.55
15.	Modulation	Mod_3L_4D_16S16_TF_NL_0p9ricianfadded_N.h5	98.1
16.	Modulation	Mod_3L_4D_16S16_TF_NLnL_Rayleighfadded_N.h5	74.45
17.	Modulation	Mod_3L_4D_16S16_TF_NLnL_0p3ricianfadded_N.h5	84.2
18.	Modulation	Mod_3L_4D_16S16_TF_NLnL_0p7ricianfadded_N.h5	91.9
19.	Modulation	Mod_3L_4D_16S16_TF_NLnL_0p9ricianfadded_N.h5	91.85

20.	Coding	COD_TrainingData_5layer_1D_2048_06.h5	99.92
21.	Coding	COD_TrainingData_5layer_1D_2048_03.h5	33.14

The classification of modulation and coding schemes is carried out individually. Therefore, our further analysis would be subdivided into “Analysis of Modulation Classifiers” and “Analysis of Coding Classifiers”.

Analysis of Modulation Classifiers

The ~19 unique Conv1D models for classification of modulation schemes are designed. The figure no. 82 illustrates the Bird’s eye view of every model’s impact in overall case study. Let’s cross-examine the performance of models based on variation in activation function, model’s input, input’s symbol rate and model’s depth with each other.

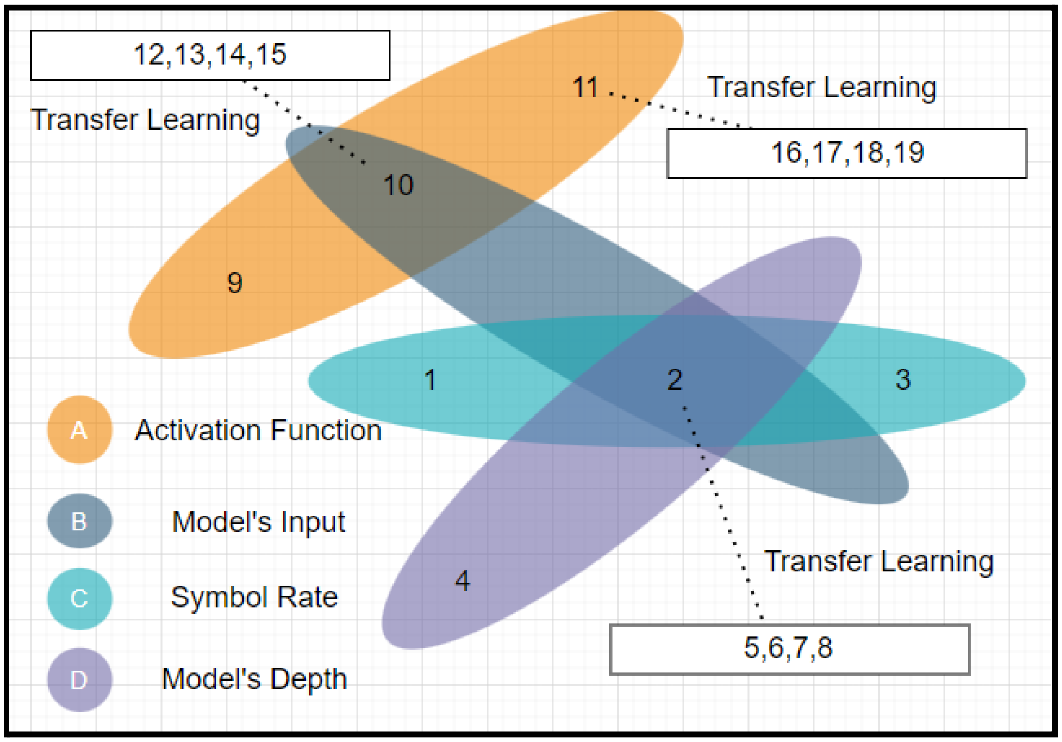


Figure No. 82 Analysis of proposed NN models for modulation classification

A. Variation in Activation Function

The performance of model 9, 10 and 11 would shed light on the impact of using the different types of activation function as Illustrated in figure no. 82. The model #09 is using the **sigmoid** as activation function and has accuracy of 16.75%. The model #10 is using the **tanh** as an activation function in its first hidden layer and **relu** activation function for its later hidden layers.

This combination resulted in accuracy of 99.8%. The model #11 is also using **tanh** as an activation function in its first hidden layer but **linear** activation function for its later hidden layers. The model #11's accuracy is 98.6%. The summary is depicted in table no. 26. The model #10 stands out among its peer models.

Table No. 26 Impact of different activation function on model's accuracy

Model No.	Activation Function	Accuracy
Model #9	sigmoid, sigmoid, sigmoid, sigmoid, softmax	16.75%
Model #10	tanh, relu, relu, relu, softmax	99.8%
Model #11	tanh, linear, linear, linear, softmax	98.6%

B. Variation in Model's input

The model 2 and 10 takes the normalized and unnormalized input respectively. Unnormalized input isn't pre-processed and can fluctuate even upto -10 to 10 as per literature study. However, the normalized input bounded between 0 to 1 on the expense of computational complexity of the system.

The model #10 contains the first hidden layer with activation function '**tanh**', this acts as a normalizer in the model and bounds the input of the next cascaded layer between -1 to 1. This alteration in model#10 has superseded the model 2 in accuracy. The accuracy of model#2 is 99.05% where as model#10 has accuracy 99.8%.

C. Variation in Symbol Rate

The NN model 1, 2 and 3 are trained on a dataset with different symbol rates i.e. 128, 64 and 32 respectively. The variation in symbol rate provides the intuition of opting for the favorable symbol rate which leads to high model's accuracy. Table no. 27 provides the impact of variation in symbol rate on model's accuracy.

Table No. 27 Impact of different activation function on model's accuracy

Model No.	Symbol Rate	Accuracy
Model #1	128	98.8%
Model #2	64	99.05%
Model #3	32	97.4%

The symbol rate of 64 Sym/sec turned out to be the best for the proposed NN model's input. Therefore, this symbol rate has been used in all later models including model# 10.

D. Variation in Model's Depth

The model 2 and 4 gives insight on selecting the appropriate depth of the proposed NN model as illustrated in figure no. 82. The model 2 has three hidden layers whereas model 4 has two hidden layers as mentioned in table No. 28. As the number of layers increases, the accuracy of the model will rise. Therefore, the accuracy of model 2 is 99.05% greater than the model 4's accuracy. However, the model's depth enhances the complexity of the model which restricts developers to incorporate hidden layers unless their impact is significant. The depth of all the rest of models is selected as 03 including model# 10.

Table No. 28 Impact of Model's depth on model's accuracy

Model No.	Model's Depth	Accuracy
Model #2	03	99.05%
Model #4	02	98.7%

E. Model's Strength in Fading Environments

The transfer learning is carried out on the pre-trained NN model 2, 10 and 11. Those models are evaluated for their performance under a variety of fading environments as shown in table no 29.

Table No. 29 Model's accuracy under Fading environment

Model No.	Fading Environment	Transfer Learning (Model's Starting Point)	Accuracy
Model #5	Rayleigh fading	02	52.85%
Model #6	Rician fading with k=0.3		75.5%
Model #7	Rician fading with k=0.7		90.25%
Model #8	Rician fading with k=0.9		93.05%
Model #12	Rayleigh fading	10	85.1%
Model #13	Rician fading with k=0.3		92.6%
Model #14	Rician fading with k=0.7		96.55%
Model #15	Rician fading with k=0.9		98.1%

Model #16	Rayleigh fading	11	74.45%
Model #17	Rician fading with k=0.3		84.2%
Model #18	Rician fading with k=0.7		91.9%
Model #19	Rician fading with k=0.9		91.85%

From the above table, model #10 stands out among the other models. This indicates that model #10 has more feature learning capability even under the fading environments.

Model# 10 — Proposed Model for classification of modulation schemes

The above discussion concluded that the Model#10 "Mod_3L_4D_NL_tan_16S16.h5" is the most appropriate choice for the classification of modulation schemes. The Figure No. 83 provides the complete sequence diagram of the Model# 10.

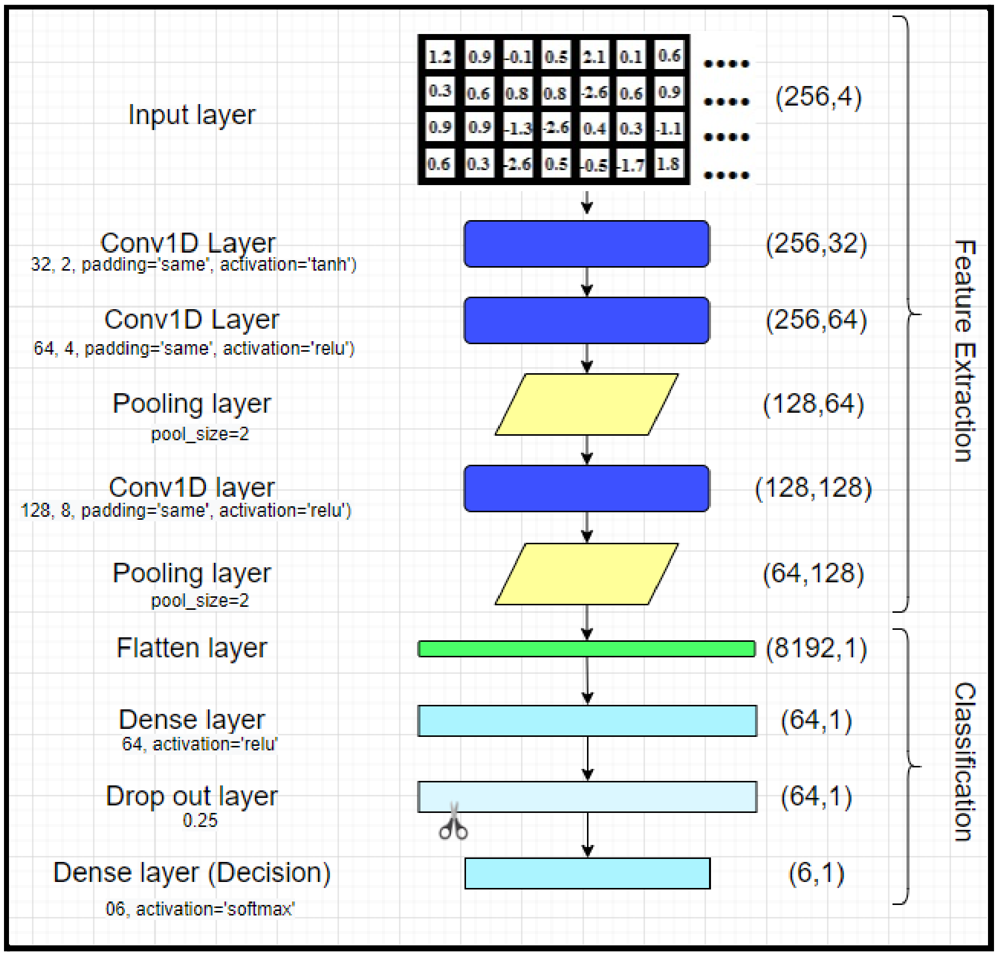


Figure No. 83 Proposed NN for classification of Modulation Scheme

Analysis of Coding Classifiers

The Conv1D based models for classification of coding schemes are developed in the MODCOD project as discussed earlier. The research has brought few reservations due to which it would not be wrong to state that "blind classification of coding schemes isn't the application of Neural Network". Neural networks can help partially to deduce results but can't be realized as a complete solution for coding scheme classifiers.

Why is NN not suitable for classification of channel coding Schemes?

Following are the main characteristics of channel encoded data which lead hindrance in the path of the NN model:

A. Nature of Encoders

Every channel coding scheme has been developed to enhance reliability and security of the data packet during transmission. Therefore, channel encoders add the redundancy along with the actual data stream which can later help the receiver to perform error correction.
In the case of LDPC encoder, the encoded data has a group of parity bits cascaded in the end of the message packet. However, the Turbo encoder is convolutional in nature and has memory as well. Encoded output is latched parallel to the message packet i.e. after each message bit, there are two or more encoded bits as per code rate of the encoder. Lastly, Polar codes are sensitive to the channel condition. The message packet relayed on reliable channels and intermittent channels are forced to zero. This results in the encoded output sensitive to the channel's state.
From the above discussion, it's concluded that all three encoders are generating output bearing different properties. In other words, consider a hypothetical item that has color, shape and taste. Now, In which category would you classify the item? Answer: Can't distinguish because the item has all three properties i.e. color, shape or taste.

B. Absence of Uniqueness

The channel encoded output can further split into two parts i.e. message packet and redundancy overhead. The message packet is considered as the purely random signal. Moreover, the redundancy overhead generated by encoder isn't mapped onto unique values/points. In actuality, it's the logical shadow of the whole message packet.
This leads to an absence of uniqueness in the encoded output. Therefore, it can be considered as just a random noise.

The above discussion helps us to understand the results & findings obtained from the proposed Conv1D models of coding schemes classification. The models are tested on the input, depth and output neuron layer of the model.

Variation in Model's input

The variation in the model's input length of the proposed Conv1D model has directly affected the overall accuracy. The probable reason is that at least one complete encoded message should be present in the model's input. Therefore, the accuracy of the model #01C with the input block length of 2048 is 99.92%.

Variation in no. of neurons in output layer

The model 01C and 02C has only one difference i.e. number of neurons in the output layer of the model. However, it has significantly changed the accuracy of the model as shown in table no.30.

Table No. 30 Accuracy vs no.of neuron in output layer

Model Name	No. of Neuron in Output layer	Accuracy
Model # 01C	06	99.92%
Model # 02C	03	33.14%

From the above results, the first question which comes to mind is **"What will happen if we have more output neurons than the labels?"** This has been debated over the Online Machine Learning forum. The link is provided below:

➔ https://datascience.stackexchange.com/questions/55774/more-output-neurons-than-labels

The probable cause is that, model #01C is overfitted on the dataset "COD_TrainingData.hdf5". Therefore, the model's accuracy is quite good. But, this claim needs to be rigorously tested in trials.

Model# 01C — Proposed Model for classification of Coding schemes

The Model#01C "COD_TrainingData_5layer_1D_2048_06.h5" is the most appropriate choice for the classification of coding schemes However, it is leading to model's overfitting problems. The Figure No. 84 provides the complete sequence diagram of the Model# 01C.

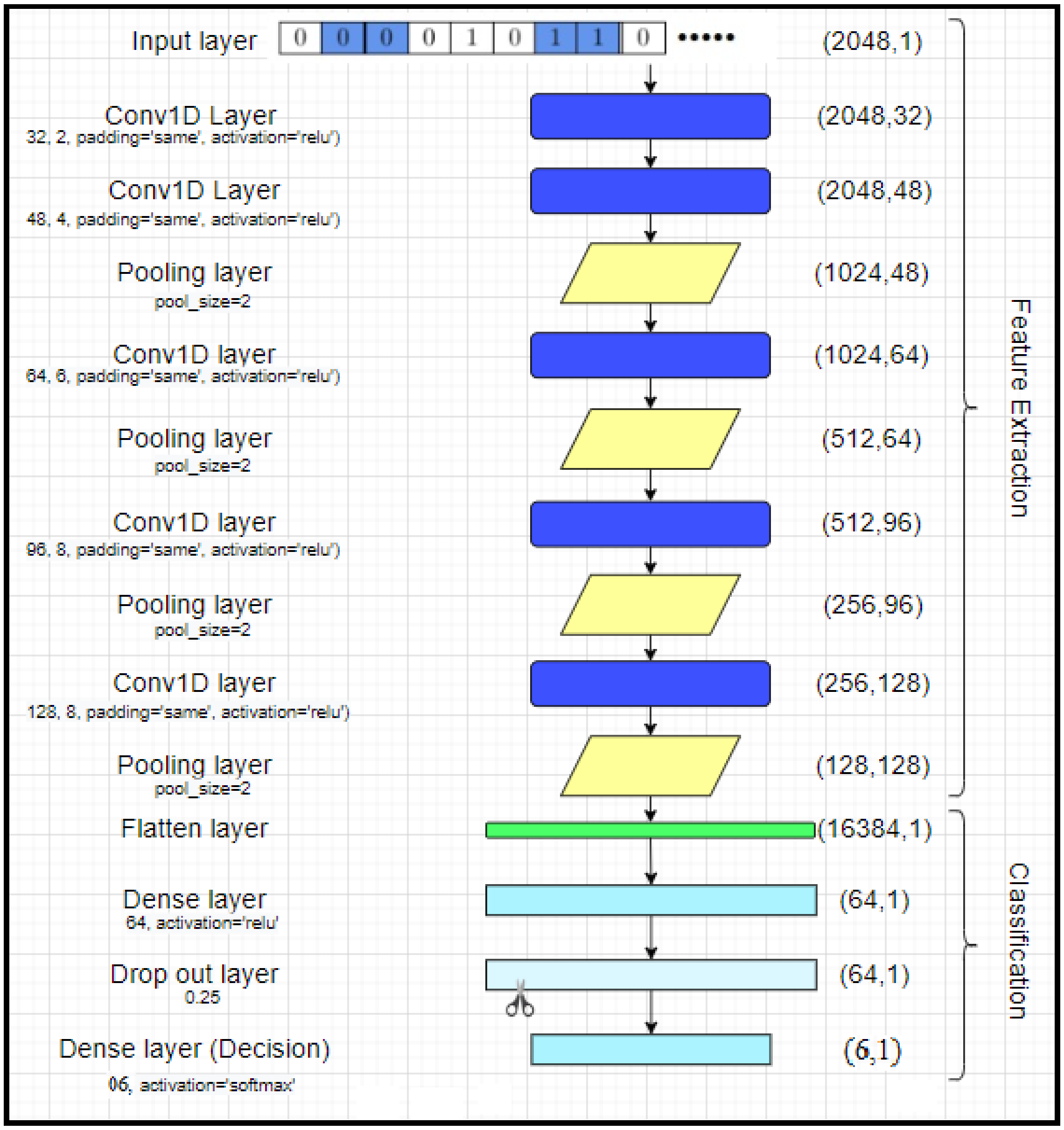


Figure No. 84 Proposed NN for classification of Coding Scheme

Conclusion & Future Directions

This detailed report encompasses everything related to the 2nd milestone of the research project titled "Neural Network based Modulation & Channel Coding Identification for SATCOM Systems". The all four sub-tasks given below of the 2nd milestone are achieved.

- ➔ Formulation of Data packet library for MODCOD implementation i.e., M-PSK and M-APSK where M = 2, 4, 8, 16 and 32 with TC / LDPC or Polar Codes at different code rates i.e., $1/4,\ 1/3,\ 1/2,\ 3/4,\ 5/7,\ 7/8$ etc.
- ➔ Preparation of Data sets for training NN.
- ➔ Implementation of classifier algorithm based on NN.
- ➔ Simulation of proposed classifier algorithm in supporting software

Moreover, **figure No. 83 & 84** represents the sequence diagram of the Conv1D based modulation and channel coding classifier respectively. According to the simulation results, both model's accuracy is above 99% in classification under no fading environment.

Following are the **potential research gaps** which are needed to be addressed:

- ➔ The designed Conv1D models evaluation under the SDRs Test-bed (Cognitive radio based on Conv1D model).
- ➔ The exploration of a new intuitive approach of classification of coding schemes.
- ➔ Adding of more modulation schemes i.e. QAM-64, OQPSK, etc. in the dataset to enhance the versatility of the model.
- ➔ Development of Modulation/coding schemes which can't be easily recognizable using existing Automatic Modulation & Coding Recognition (AMR) systems.
- ➔ Real-time identification of modulation schemes in the SATCOM systems.

References

1. Saad Iqbal and Syed Ali Hassan, "Literature Review & Feasibility Report of Neural Network based Modulation and Channel Coding Identification for SATCOM Systems", December, 2021.
2. Tavildar's repository, https://github.com/tavildar/LDPC/tree/master/LdpcM
3. MATLAB Reference for implementation of Turbo encoder, https://www.Mathworks.com/ help/comm/ref/comm.turboencoder-system-object.html#d123e230903
4. The MATLAB implementation of Polar encoder, https://ecse.monash.edu/staff/eviterbo/polarcodes.html