Neural Network based Modulation and Channel Coding Identification for SATCOM Systems

3rd Milestone's Performance Evaluation

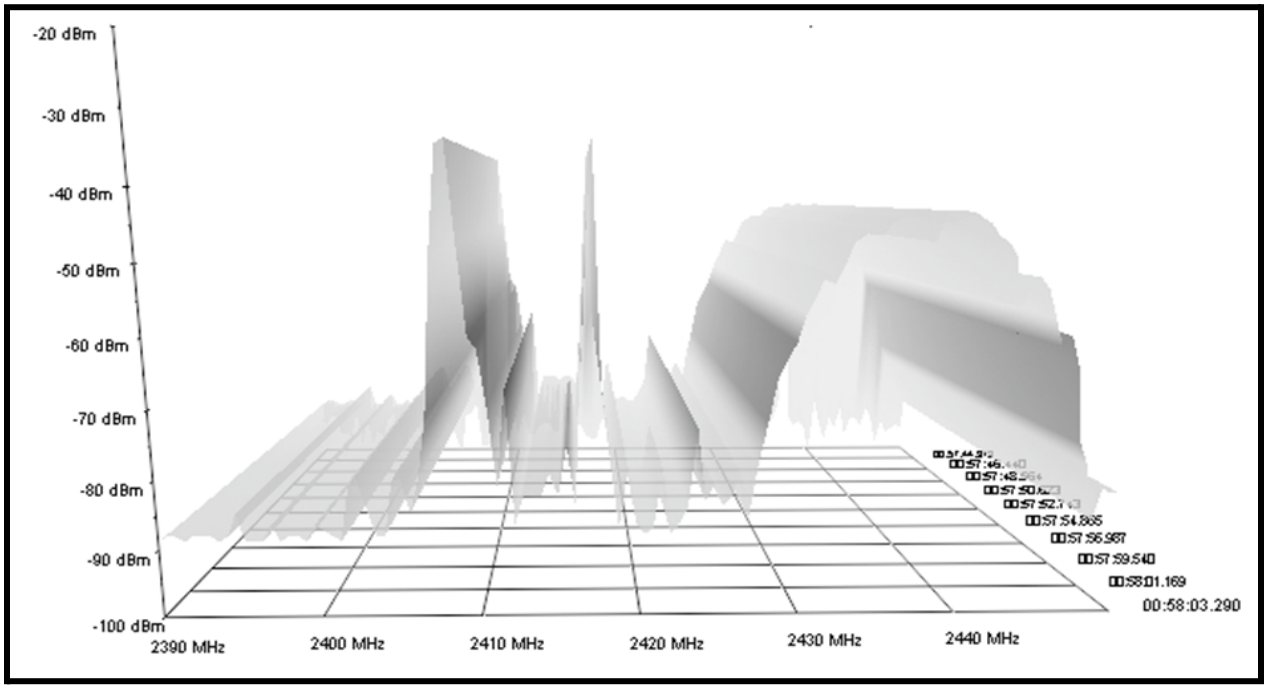
Principal Investigator:
Syed Ali Hassan, Ph.D.
Electrical Engineering

Department of Electrical Engineering
School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST)
Islamabad, Pakistan
2022

SDR-based Automatic Modulation Recognition (AMR) System Implementation

Project report

Neural Network based Modulation and 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 report is prepared to fulfill the requirement of the 3rd milestone “SDR-based Automatic Modulation Recognition (AMR) System Implementation” of a project titled “Neural Network (NN) based Modulation and Channel Coding Identification for SATCOM Systems”. Author has penned down every aspect encountered in designing SDR-based AMR for testing proposed NN models. The SDR based AMR are popular for their ability to apply modulation classification on the broadcast stream captured via RF antenna. Such a standalone NN-based AMR can act as cognitive radio in the field environments which is a very crucial requirement in modern SATCOM.

The document is divided into three subcategories. To ensure the reader has a thorough grasp, sections are connected together and are in chronological order, The first section discussed the complete system level design of SDR-based AMR. It provides a complete walkthrough for the motivated readers to develop their own AMR using off-the-shelf SDRs. The second section elaborated the generation of On-Air dataset under different scenarios i.e. variation in SNR and Rayleigh and Rician channel fadings. In the last section, the author has discussed the training and testing of the proposed NN-models using the extracted On-Air dataset. Out of many, the author has selected two significant NN-models for the discussion.

The report concludes with discussing the distinct results and findings made available after the stressful testing of the finalized NN-model. This document would serve as a comprehensive guide for creating one’s AMR based on SDR and will be regarded as essential reading for all upcoming AMR projects.

Introduction and Approach

Before transmission, all radio communication signals including those from radio, television, mobile phones, etc. are modulated. For proper demodulation to occur, modulation recognition is an essential task. Additionally, prior understanding of the coding scheme is necessary for signal interpretation if the transmission is employing the forward error correction (FEC) method. This is an overview of our effort, to have the received signal's modulation and coding scheme automatically recognised.

This report is associated with the third milestone of the SATCOM project, and it states:

1. Implementation of MODCOD Schemes on SDR.
2. Integration of developed classifier algorithm with SDR.

An indigenous SDR-based design of an AMR has been proposed and later, it has been implemented. The modular approach is being carried out for smooth execution of the project. As discussed in the 2nd report [2], recognition of coding is not feasible at all using ANN scheme unless few critical factors are constant i.e. . Such limitation is unjustified as it takes us away from real scenarios. Thus, coding schemes recognition has been dropped from the current discussion and the AMR design would be restricted to successful recognition of modulation schemes only.

The proposed approach for the development of the SDR-based AMR using state of the art SDRs elaborated below has flexibility and scalability which would prevent future researchers from reinventing the wheel. This strength has qualified this approach as the foundation for future AMR testbeds.

The approach in chronological order:

1. The hardware design for SDR-based testbed is carried out.
2. The GNU-radio toolkit is utilized for transmission and reception with multiple other features ensuring scalability and genuinity.
3. The generation of dataset tagged as On-Air at different scenarios for said modulation schemes.
4. The pre-processing of gathered dataset.
5. Re-evaluation of parameters of the proposed NN model for AMR i.e. the variation in the number and order of hidden layers in the model. Later, NN- Model training and testing to come up with a better finalized model under generic circumstances.

System Level Design

General Overview

The hardware implementation of AMR generally requires the detection algorithm on the reception side. However, we are tasked to develop the complete testbed, which requires its own transmission side for ground truth. The hardware setup of the proposed AMR system is shown in figure no. 1**.** The two SDR-N200 kits are being utilized, one for transmission and other for reception. Both SDRs are inter-connected through MIMO cable for synchronization purposes. The IPT-machine (equipped with Gigabit NIC card) is interfaced with SDRs using a network switch for ensuring seamless data transmission among them. The TX and RX antennas are one meter apart from each other as per guidelines of the lab prototype.

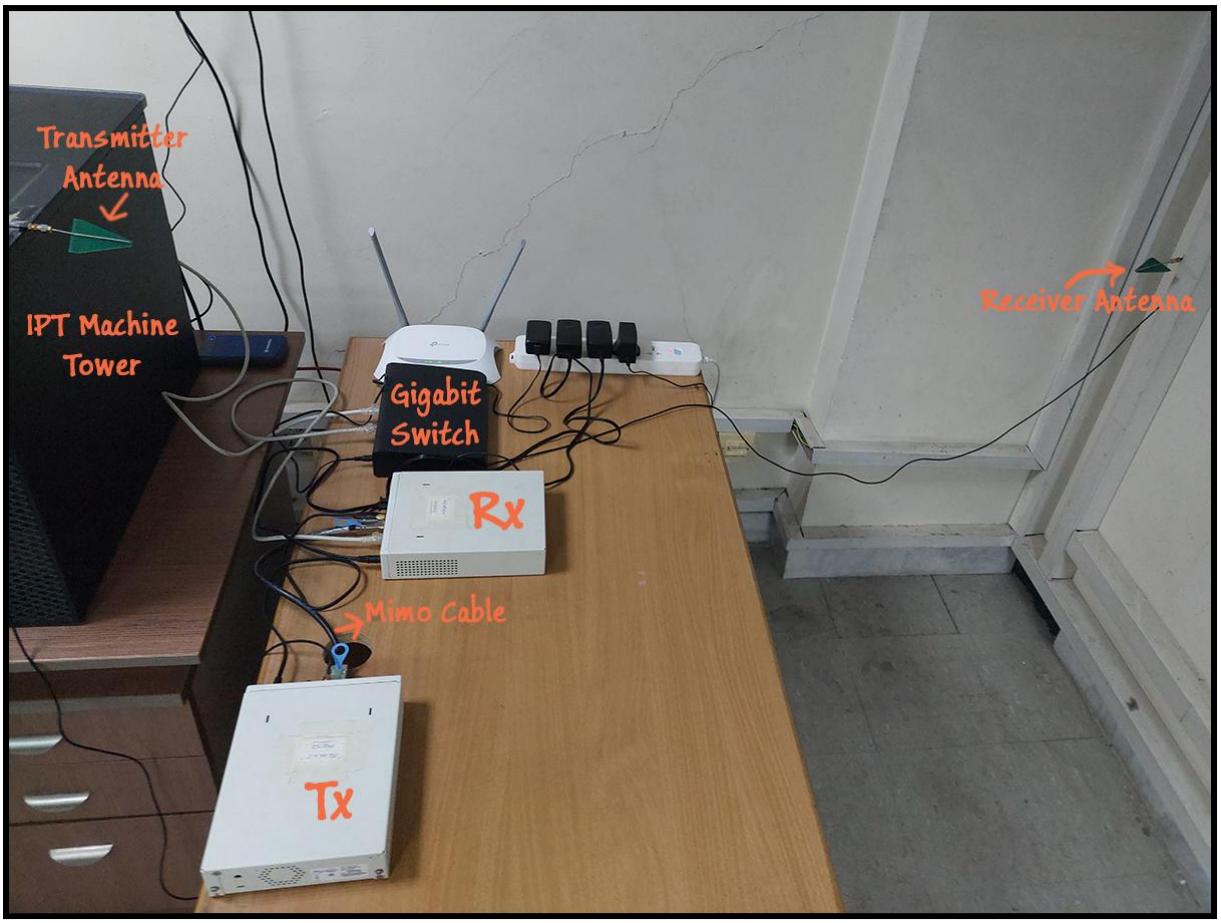


Figure No. 1 Hardware Setup of proposed SDR-based AMR

Hardware Resources

The complete equipment detail is given in table no. 1. The connectivity among hardware modules is checked and verified. Every related component is working properly.

Table No. 1 The hardware component list for AMR testbed.

S. No.	Equipment	Quantity
1.	SDR-N200	2
2.	MIMO cable	1
3.	Gigabit Network Switch	1
4.	IPT Tower (Computational Resource)	1
5.	1 TB HDD (Storage)	1
6.	Cat-6 Network Cable	2
7.	1 meter RF cable	2
8.	3dB gain directional Antenna	2
9.	Extension Board	1
10.	TP-link router	1

Software Application (GNU Radio)

At first, the analog transmission and reception of RF signals was carried out to validate the overall system. Signal processing and software side of SDRs for the SDR-based AMR has been designed using the GNU Radio software tool. The *GNU Radio* is an open source software, popular for designing the SDRs solutions and real-time signal processing. The complete software tool is being installed on the IPT machine to get the things working.

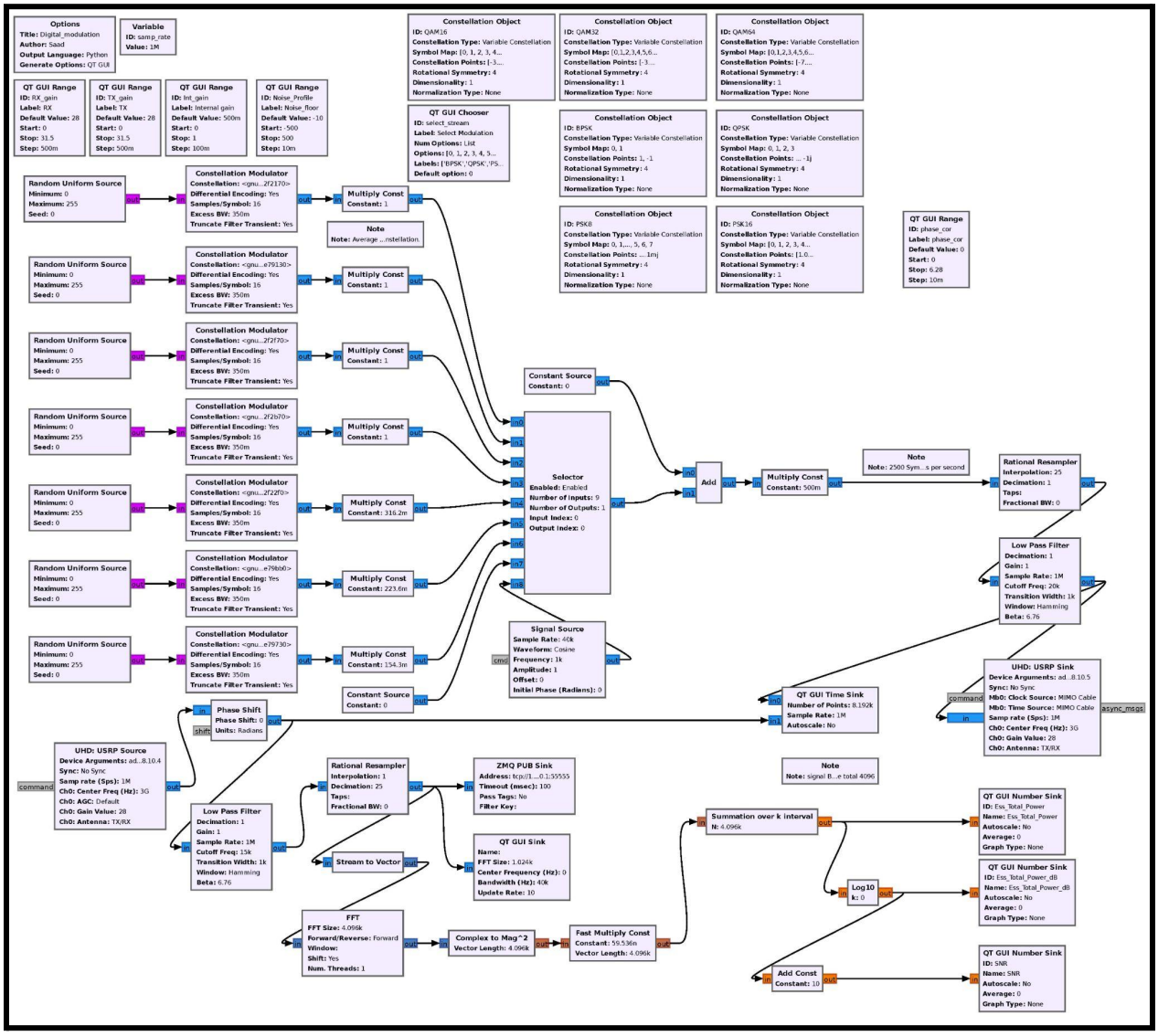


Figure No. 2 RF design for transmission and reception on GNU Radio for AMR system

The GNU process flow as shown in figure no.2 is interfaced with the SDRs, carrying out following six major functions:

1. Data broadcasting using modulation scheme among the selected ones i.e. BPSK, QPSK, PSK8, PSK16, QAM16 and QAM32.
2. Capturing the incoming signal and passing it to the data processing window via ZMQ protocol for training and testing of NN-models.
3. The real-time SNR monitoring.

4. Phase correcting at the receiver end.
5. Real-time spectrum monitoring using QT based GUI
6. The real-time controlling the Tx and Rx gains.

System Graphical Interface

All these six features are the core elements for assessment of good and reliable cognitive radio. The QT based GUI as shown in Figure no. 3 is being formulated via GNU Radio toolkit, and it acts as a controlling hub to pass-on I-Q received data along with its ground truth to processing tab. The processing tab (discussed later) is a window responsible to handle raw data fetch from the SDRs via ZMQ protocol.

Figure No. 3 QT GUI Screen of AMR system (in Action)

On-Air Dataset Generation

The dataset generation is to establish an impartial and consistent dataset for the NN models. For the On-Air dataset, this has been subdivided in following tasks,

- Storing and labeling the raw I-Q data feed from SDRs.
- Pre-processing the I-Q data.

The *processing tab* is responsible for carrying out both operations and it is linked with GNU radio via ZMQ protocol. The processing tab consists of two scripts i.e. ***data_retreive.ipynb*** *and* ***preprocessing.ipynb*** and is utilized in the fashion as shown in figure no. 4.

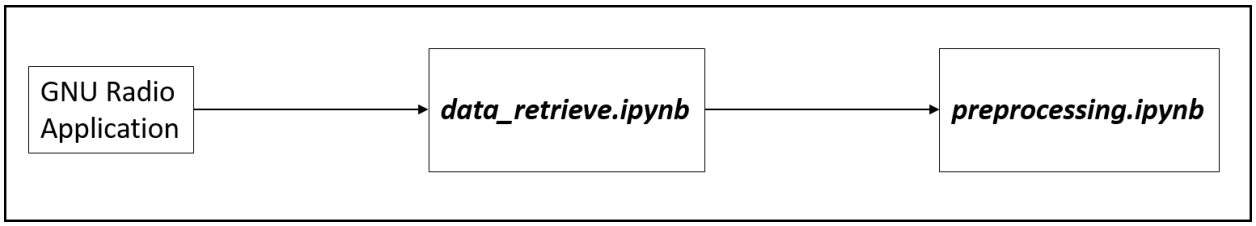


Figure No. 4 Flow diagram of *processing tab*

The I-Q data has been collected over the week on multiple channel configurations and different SNR (using *data_retreive.ipynb* script). The variations in the channel were employed via changing the relative distance and direction of the Tx and Rx antenna. Moreover, the SDR gain control is utilized to mimic the different SNR levels through GUI of GNU Radio application. Later, the ground truth is utilized for labeling the whole dataset.

Preprocessing of labeled I-Q Dataset

The datastream from 2nd stage i.e. data-gathering(data_retrieve.ipynb) is injected into the *preprocessing.ipynb* for cleansing and preprocessing. Following are the effects that data has undergone during the pre-processing stage:

1. Data normalization
2. Parameters extraction i.e. magnitude and phase.
3. Windowing operation

4. Arbitrary stride

Data Normalization

Data normalization in machine learning is the technique of translating data into a specific range or simply transforming data onto the unit sphere. It eliminates the effects of certain gross influences and optimizes the gain of the data which allows comparison tools to perform better on the dataset. Both *normalized and original I-Q data streams* have been used for generation of On-Air datasets.

Parameters extraction

The variation in magnitude and phase information of the I-Q data stream are considered as significant parameters in modulation classification.

The magnitude of data stream is given as:

$$magnitude = \sqrt{a^2 + b^2}$$

And, phase of data stream is given as:

$$phase = tan^{-1}(b/a)$$

These two parameters are interleaved with the original I-Q stream and now the dimension of the output dataset becomes [4xN], where N is the number of samples.

Windowing operation

Windowing operation is used to split unbounded streams of data into finite sets, or windows, based on specified criteria, such as time and number of samples. Multiple window sizes i.e. 256, 512 and 1024 have been utilized in preparation of On-Air dataset. As the window grows, the number of symbols increases and it elevates the accuracy bar in detection at the expense of computation complexity.

Arbitrary stride

The count of samples between the beginnings of successive windows is called *stride*. Keeping the stride constant usually aids biases in the dataset. Therefore, it must be arbitrary during windowing operation. The stride has setted to be a random value between 128 to 384.

Detailed Summary of On-Air Dataset

The processing tab ingests raw I-Q data from GNU Radio application via ZMQ protocol at *40k samples per second* and samp/sys is fixed at *16 samp/sys*. The worth of ~5GB labeled raw-data is captured containing all the said modulations i.e BPSK, QPSK, PSK-8, PSK-16, QAM-16, and QAM-32. This has further passed through the processing phase as discussed before and its output has served the training and testing dataset for proposed NN-models. The table no. 2 provides a detailed summary of On-Air datasets devised for the project.

Table No. 02 Detail Summary of On-Air dataset

- The samp/sym is setted on 16.
- Modulation schemes are BPSK, QPSK, PSK-8, PSK-16, QAM-16, QAM-32 and QAM-64 (optional).
- Stride is setted on arbitrary.
- The magnitude and phase parameters are interleaved with the I-Q stream.
- Phase corr. refers to employing phase correction at the receiver end before storing the dataset.

Name	Phase Corr.	Window Size	Normalization	SNR (Tx,)(Rx)	Environment
D_01	No	256	True	(22-28),(22-28)	Multiple scenarios
D_02	No	512	True	(22-28),(22-28)	Multiple scenarios
D_03	No	1024	True	(22-28),(22-28)	Multiple scenarios
D_04	No	256	False	(22-28),(22-28)	Multiple scenarios
D_05	No	512	False	(22-28),(22-28)	Multiple scenarios

D_06	No	1024	False	(22-28),(22-28)	Multiple scenarios
D_07	Yes	256	True	(22-28),(22-28)	Multiple scenarios
D_08	Yes	512	True	(22-28),(22-28)	Multiple scenarios
D_09	Yes	1024	True	(22-28),(22-28)	Multiple scenarios
D_10	Yes	256	False	(22-28),(22-28)	Multiple scenarios
D_11	Yes	512	False	(22-28),(22-28)	Multiple scenarios
D_12	Yes	1024	False	(22-28),(22-28)	Multiple scenarios

Proposed NN Model for AMR

In the last report of the project [2], we have rigorously tested up to 19 NN-models and finalized Model#10 “Mod_3L_4D_NL_tan_16S16.h5” for future use. This 1D CNN model has beaten the rest of NN-models in accuracy. Its flow diagram is shown in figure No. 05.

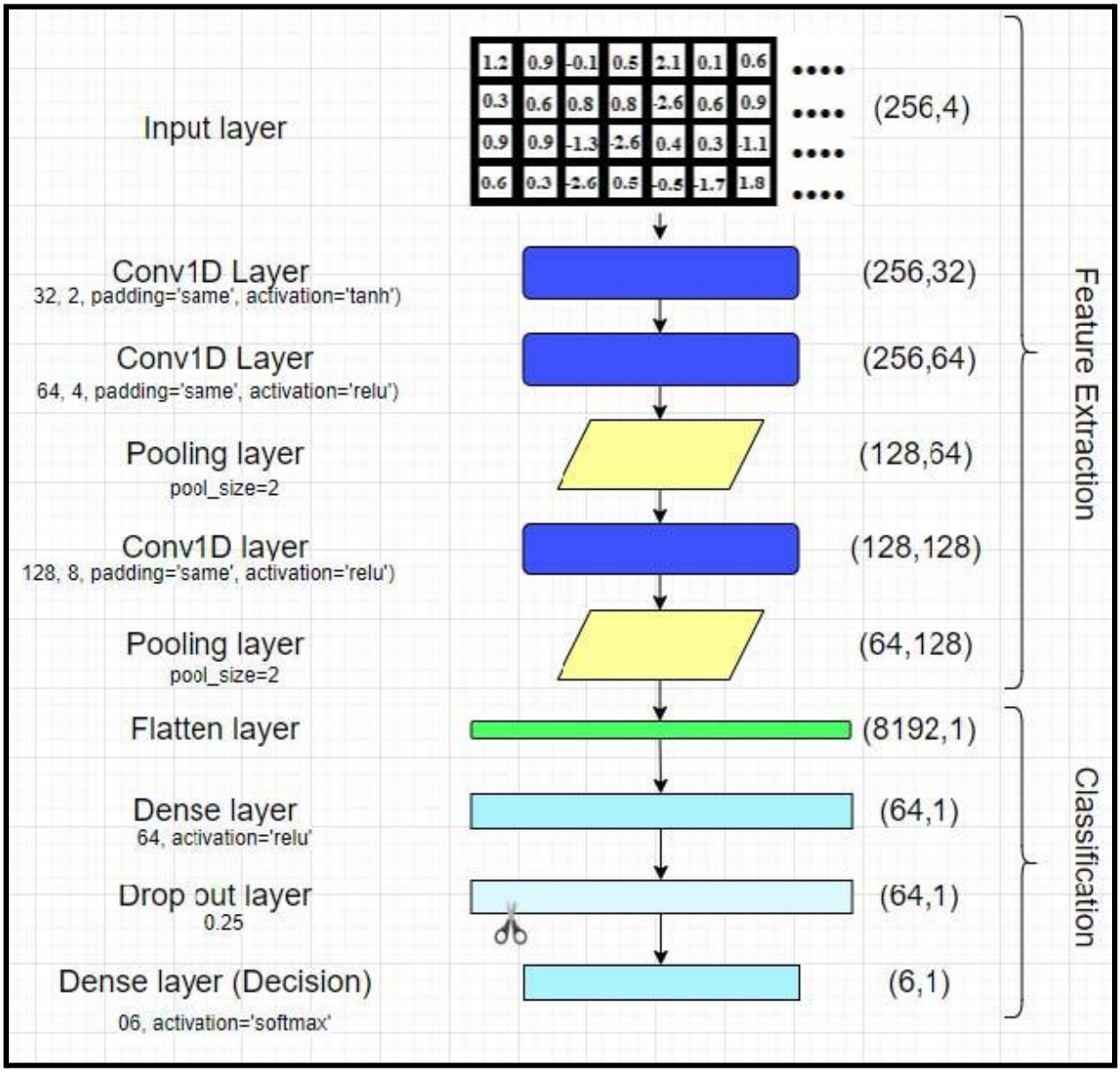


Figure No. 5 Flow Diagram of 1D CNN Model#10 “Mod_3L_4D_NL_tan_16S16.h5”[2]

1D CNN Model#10 “Mod_3L_4D_NL_tan_16S16.h5” is taken as the starting point and its performance is evaluated on the 4 batches of On-Air dataset, whose input dimension is 256xN (refer to table no. 02). The testing of the model has raised the need for tuning of the model. Following were the main aspects that were addressed on-fly:

- *Model performs better with the increase of symbols in the model's input*. This can be achieved by widening the window size of the model’s input layer. Therefore, the large dimensions i.e. 256, 512 and 1024 are introduced for the input layer to address this issue.
- Testing results with On-Air dataset, have depicted the need for more hidden layers in NN-models to increase prediction accuracy.
- Model has shown ease in scalability. The QAM64 was added for testing purposes and results were satisfactory.

After the rigorous testing and evaluation of several models, the two models were established i.e. **model#10** and **model#10a.** Input block dimension of model#10a is 4x512. It has four hidden layers and three fully connected layers. The detail discussion on model#10 and model#10a is given below.

Model#10 “Mod_3L_4D_NL_tan_16S16_256.h5”

The “Mod_3L_4D_NL_tan_16S16_256.h5” Conv1D model is shown in figure no. 06 as trained on the dataset “D_04.h5 (processed_256.h5)”. Model’s further characteristics are tabulated in the table no. 3.

Table no. 3 Characteristics of Model #10

Model’s characteristics		**Dataset Attributes**	
Model Layers	03	Symbol Rate	2500

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.	Data stream properties	● No phase Corr. ● Range of SNR ● Multiple Scenarios *Consult table no.02 for more info.
Output layer Neurons	07	Input format	Unnormalized and 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, 7)                 455

=================================================================
Total params: 599,015
Trainable params: 599,015
Non-trainable params: 0
_________________________________________________________________
```

Figure No. 06 The model #10 “Mod_3L_4D_NL_tan_16S16_256.h5”

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

```
Epoch 1/20
16275/16275 [==============================] - 299s 18ms/step - loss: 0.7577 - accuracy: 0.6362 - val_loss: 0.5953 - val_accuracy: 0.7245
Epoch 2/20
16275/16275 [==============================] - 300s 18ms/step - loss: 0.5966 - accuracy: 0.7393 - val_loss: 0.5413 - val_accuracy: 0.7667
Epoch 3/20
16275/16275 [==============================] - 295s 18ms/step - loss: 0.5370 - accuracy: 0.7730 - val_loss: 0.4920 - val_accuracy: 0.7936
Epoch 4/20
16275/16275 [==============================] - 267s 16ms/step - loss: 0.5063 - accuracy: 0.7895 - val_loss: 0.4680 - val_accuracy: 0.8039
Epoch 5/20
16275/16275 [==============================] - 263s 16ms/step - loss: 0.4832 - accuracy: 0.8011 - val_loss: 0.4612 - val_accuracy: 0.8158
Epoch 6/20
16275/16275 [==============================] - 260s 16ms/step - loss: 0.4674 - accuracy: 0.8087 - val_loss: 0.4293 - val_accuracy: 0.8297
Epoch 7/20
16275/16275 [==============================] - 263s 16ms/step - loss: 0.4551 - accuracy: 0.8149 - val_loss: 0.4093 - val_accuracy: 0.8378
Epoch 8/20
16275/16275 [==============================] - 264s 16ms/step - loss: 0.4445 - accuracy: 0.8196 - val_loss: 0.4173 - val_accuracy: 0.8341
Epoch 9/20
16275/16275 [==============================] - 265s 16ms/step - loss: 0.4349 - accuracy: 0.8254 - val_loss: 0.4032 - val_accuracy: 0.8434
Epoch 10/20
16275/16275 [==============================] - 265s 16ms/step - loss: 0.4282 - accuracy: 0.8278 - val_loss: 0.3982 - val_accuracy: 0.8431
Epoch 11/20
16275/16275 [==============================] - 268s 16ms/step - loss: 0.4218 - accuracy: 0.8316 - val_loss: 0.4087 - val_accuracy: 0.8393
Epoch 12/20
16275/16275 [==============================] - 268s 16ms/step - loss: 0.4153 - accuracy: 0.8338 - val_loss: 0.4211 - val_accuracy: 0.8400
Epoch 13/20
16275/16275 [==============================] - 267s 16ms/step - loss: 0.4117 - accuracy: 0.8362 - val_loss: 0.3842 - val_accuracy: 0.8485
Epoch 14/20
16275/16275 [==============================] - 268s 16ms/step - loss: 0.4066 - accuracy: 0.8377 - val_loss: 0.3883 - val_accuracy: 0.8481
```

```
Epoch 15/20
16275/16275 [==============================] - 268s 16ms/step - loss: 0.4036 - accuracy: 0.8403 - val_loss: 0.3889 - val_accuracy: 0.8475
Epoch 16/20
16275/16275 [==============================] - 267s 16ms/step - loss: 0.3980 - accuracy: 0.8426 - val_loss: 0.3983 - val_accuracy: 0.8442
Epoch 17/20
16275/16275 [==============================] - 266s 16ms/step - loss: 0.3967 - accuracy: 0.8432 - val_loss: 0.3794 - val_accuracy: 0.8503
Epoch 18/20
16275/16275 [==============================] - 267s 16ms/step - loss: 0.3939 - accuracy: 0.8449 - val_loss: 0.3791 - val_accuracy: 0.8536
Epoch 19/20
16275/16275 [==============================] - 267s 16ms/step - loss: 0.3899 - accuracy: 0.8471 - val_loss: 0.3750 - val_accuracy: 0.8539
Epoch 20/20
16275/16275 [==============================] - 267s 16ms/step - loss: 0.3877 - accuracy: 0.8481 - val_loss: 0.3919 - val_accuracy: 0.8532
```

The "training and validation" loss and accuracy of model#10 are shown in figure no. 07 and 08 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 09.

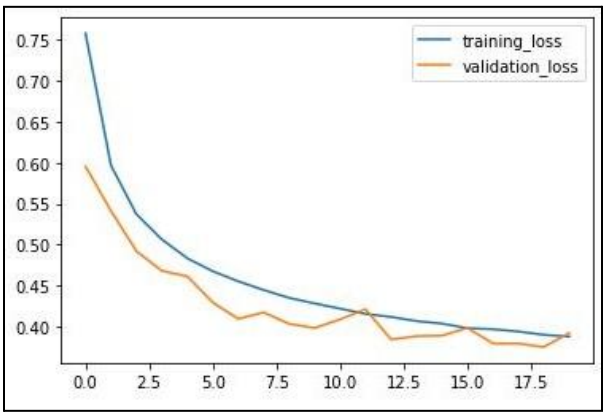


Figure No. 07 Training and validation loss of Model #10

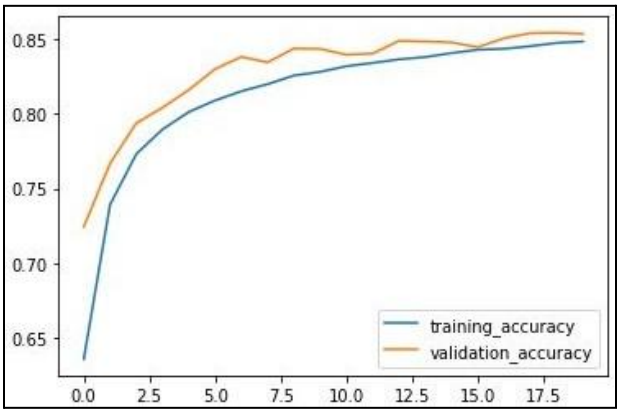


Figure No. 08 Training and validation accuracy of Model #10

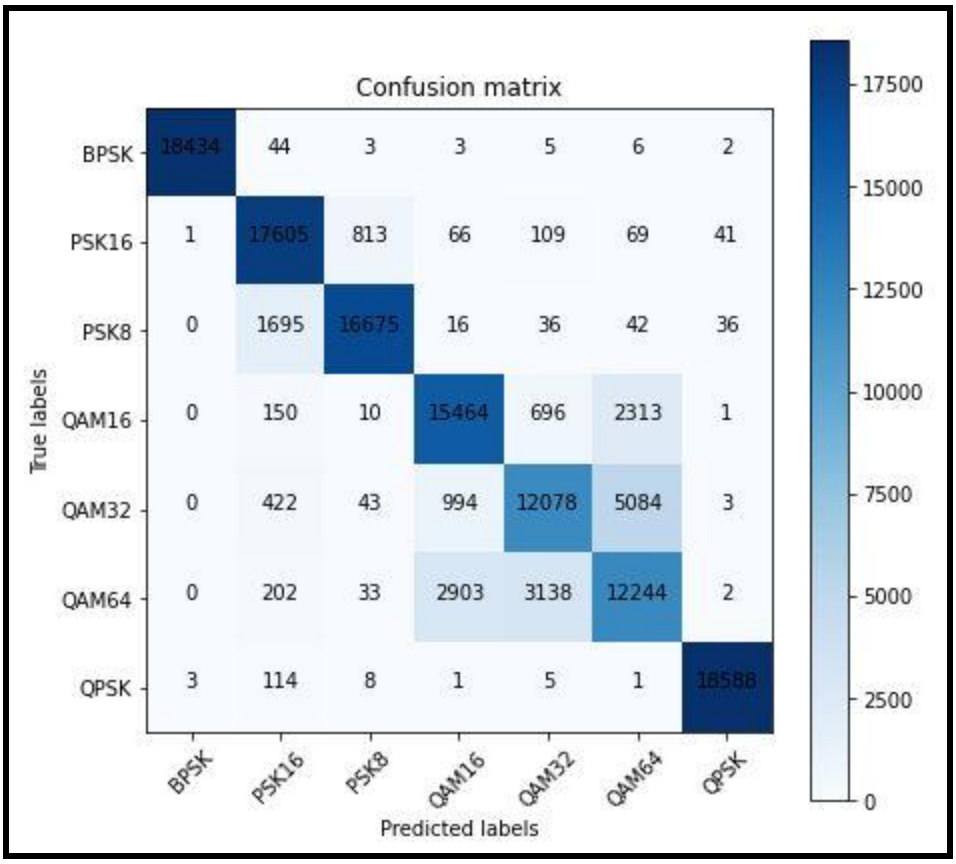


Figure No. 09 Performance result of model #10 is 85.32% on testing dataset

Model#10a "Mod_3L_4D_NL_tan_16S16_512.h5"

The "Mod_3L_4D_NL_tan_16S16_512.h5" Conv1D model is shown in figure no. 10 as trained on the dataset "D_11.h5 (processed_512.h5)". Model's further characteristics are tabulated in the table no. 4.

Table no. 4 Characteristics of Model #10a

Model's characteristics		**Dataset Attributes**	
Model Layers	Four hidden layers and three fully connected	Symbol Rate	2500

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.	Data stream properties	● Phase Corr. ● Range of SNR ● Multiple Scenarios *Consult table no.02 for more info.
Output layer Neurons	07	Input format	Unnormalized and shape 512x4

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1 (Conv1D)              (None, 512, 32)           288

 conv2 (Conv1D)              (None, 512, 64)           8256

 maxpool1 (MaxPooling1D)     (None, 256, 64)           0

 conv3 (Conv1D)              (None, 256, 96)           49248

 maxpool2 (MaxPooling1D)     (None, 128, 96)           0

 conv4 (Conv1D)              (None, 128, 128)          147584

 flatten (Flatten)           (None, 16384)             0

 dense (Dense)               (None, 128)               2097280

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 6)                 390

=================================================================
Total params: 2,311,302
Trainable params: 2,311,302
Non-trainable params: 0
_________________________________________________________________
```

Figure no. 10 The model #10a “Mod_3L_4D_NL_tan_16S16_512.h5”

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

```
Epoch 1/10
7950/7950 [==============================] - 283s 36ms/step - loss:
0.3890 - accuracy: 0.8238 - val_loss: 0.1973 - val_accuracy: 0.9272
Epoch 2/10
7950/7950 [==============================] - 275s 35ms/step - loss:
0.1803 - accuracy: 0.9355 - val_loss: 0.1522 - val_accuracy: 0.9456
Epoch 3/10
7950/7950 [==============================] - 286s 36ms/step - loss:
0.1375 - accuracy: 0.9534 - val_loss: 0.1342 - val_accuracy: 0.9535
Epoch 4/10
7950/7950 [==============================] - 301s 38ms/step - loss:
0.1207 - accuracy: 0.9594 - val_loss: 0.1492 - val_accuracy: 0.9495
Epoch 5/10
7950/7950 [==============================] - 323s 41ms/step - loss:
0.1109 - accuracy: 0.9634 - val_loss: 0.1536 - val_accuracy: 0.9559
Epoch 6/10
7950/7950 [==============================] - 334s 42ms/step - loss:
0.0996 - accuracy: 0.9673 - val_loss: 0.1041 - val_accuracy: 0.9680
Epoch 7/10
7950/7950 [==============================] - 348s 44ms/step - loss:
0.0970 - accuracy: 0.9688 - val_loss: 0.1068 - val_accuracy: 0.9663
Epoch 8/10
7950/7950 [==============================] - 357s 45ms/step - loss:
0.0912 - accuracy: 0.9706 - val_loss: 0.1067 - val_accuracy: 0.9692
Epoch 9/10
7950/7950 [==============================] - 354s 44ms/step - loss:
0.0833 - accuracy: 0.9736 - val_loss: 0.1474 - val_accuracy: 0.9541
Epoch 10/10
7950/7950 [==============================] - 365s 46ms/step - loss:
0.0845 - accuracy: 0.9739 - val_loss: 0.1466 - val_accuracy: 0.9640
```

The "training and validation" loss and accuracy of model#10a are shown in figure no. 11 and 12 respectively. The model performance result on the test data after training can be visualized in a confusion matrix illustrated in figure no. 13.

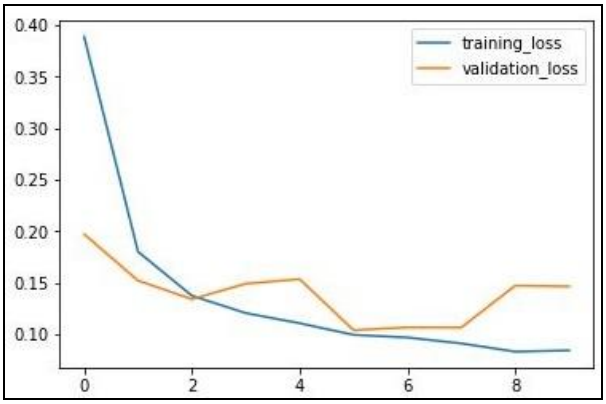


Figure No. 11 Training and validation loss of Model #10a

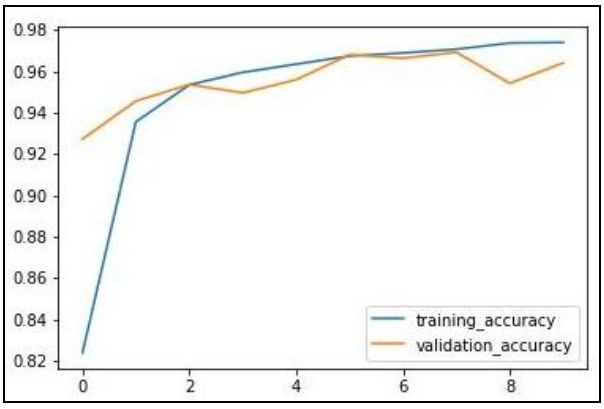


Figure No. 12 Training and validation accuracy of Model #10a

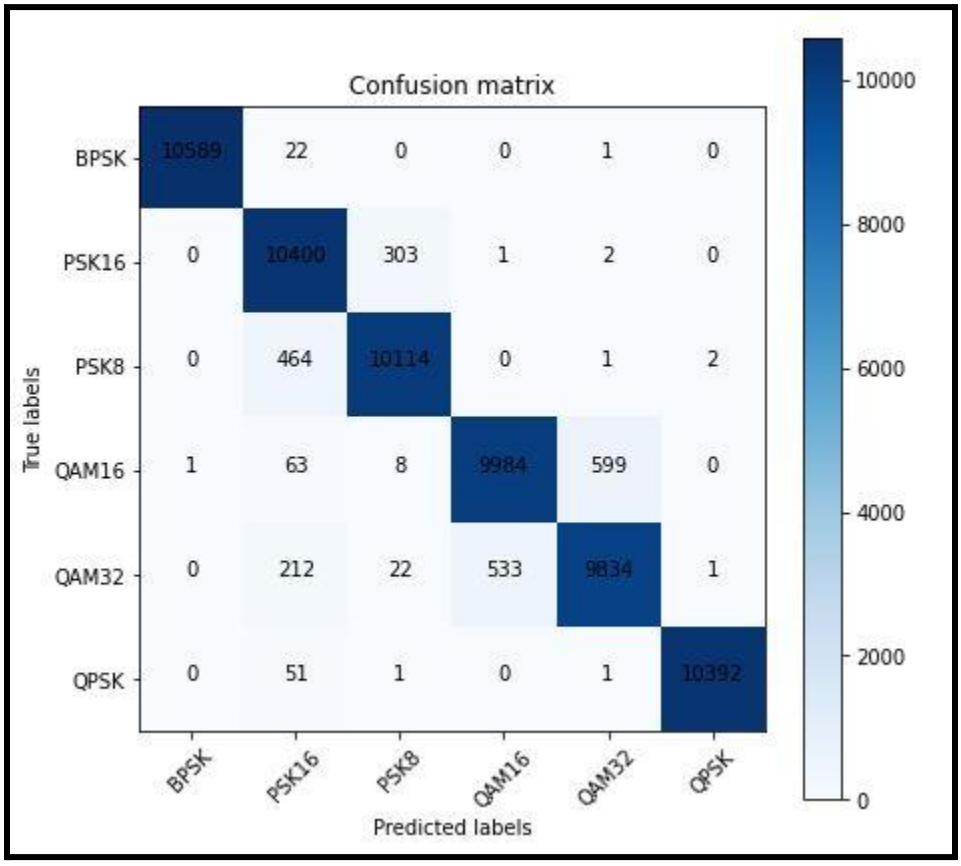


Figure No. 13 Performance result of model #10a is 96.40% on testing dataset

Result and Findings

General Overview

In the last discussion, the training and testing of the two significant NN-models has been elaborated out of ~24 models. This whole activity helped in obtaining the best model. The overall summary regarding the accuracy of these two models is provided in table no. 05.

Table No. 05 Summary of training and testing of respective NN-models

S.No.	Model Name	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1.	Model#10	84.81	38.77	85.32	39.19
2.	Model#10a	97.39	8.45	96.40	14.66

Salient Features

The following are the salient features of the NN-models trained with OnAir dataset of AMR:

Increment in symbol's count for Model's Input Layers

It has been noticed during models' training and testing(T&T) phase that the accuracy of modulation classification improves with the increase in the number of symbols in the model's Input layer. This phenomena isn't strange because *the intersection of the constellation points among digital modulations is not empty*.

For example, PSK modulation schemes are subsets of each other i.e. QPSK contains both constellation points of BPSK modulation. In the worst scenario of QPSK, the incoming datastream could produce symbols identical to BPSK and can confuse the AMR.

This baseline problem can only be eliminated via having a large number of symbols in the Model's input layer. The model#10 has **16 symbols/input-block** whereas model#10a has **32 symbols/input-block.** This is the one of the contributing factor towards the better accuracy of model#10a in comparison of model#10.

M-PSK and M-QAM bifurcation

From the figure no. 9 & figure no. 13, it is clarified that *the 1D-CNN models can accurately bifurcate the PSK and QAM modulation schemes even in the worst scenarios* i.e. no phase correction and limited symbols per model's input block.

Scalability(Adding more modulation schemes)

The neurons of output layer of model#10 discussed in the 2nd report[2] is incremented by one and mapped it on **QAM64** as shown in figure no. 06. *This experiment has qualified the scalability of the proposed model.*

Role of phase correction(At Rx)

The received I-Q stream always has phase error which is responsible for rotation of symbols in constellation diagrams. This can badly affect the AMR performance because *PSK modulation symbols are only distinguishable on the basis of their absolute phase*.

Therefore, it has been observed that AMR works efficiently if the cognitive radios are phase locked and the proper phase correction is implemented on the receiving end. The same phenomena has been observed in the performance metrics of model#10 and model#10a.

Tx and Rx gains

SDRs have internal gain provision, which can be used by researchers to manage the SNR of the communication link under test. The "SBX 400-4400 MHz Rx/Tx (40 MHz)" daughter board of Ettus family is being used in the USRP to enable the transmission at ~2GHz carrier frequency. It provides the accessibility of tuning gain of Tx and Rx via GNU Radio application in real-time. Each side i.e. Tx and Rx can contribute 0-31.5 dB gain in the communication link in our case. *The 22 dB to 28 dB window has been used to mimic the variation in SNR during data gathering procedure* and NN-models have shown comparable better performance at high gain settings of Tx and Rx.

Conclusion and Future Work

This detailed report encompasses everything related to the 3rd milestone of the research project titled “Neural Network based Modulation and Channel Coding Identification for SATCOM Systems”. Both sub-tasks given below of the 3rd milestone are achieved.

- ☑ ~~Implementation of MODCOD Schemes on SDR.~~
- ☑ ~~Integration of developed classifier algorithm with SDR.~~

In summary, the main aim of the whole effort was to achieve satisfactory modulation classification results on the SDR-based test-bed. Moreover, the secondary task was to ensure the provision of a road-map for future researchers to scale the system implementation rather than reinvent it from basics.

The model#10a discussed earlier has shown ***96.40%*** accuracy in testing results on On-Air dataset gathered from SDR-based testbed. This model would be utilized in the 4th milestone for real-time modulation classification. Moreover, the applied modular approach in system design has proven the overall scalability and reliability. The pipeline for On-Air dataset gathering is smooth and self-explanatory. Along with that, the training and testing scripts are automated to allow the future researchers to train and test new custom models without any big fuss.

The following are some potential research gaps that should be filled:

- The investigation of a novel, logical method for classification of coding schemes.
- QAM-64, OQPSK, and other more modulation schemes could be added to the dataset to increase the model's adaptability.
- The creation of modulation and coding schemes that are difficult for automatic modulation and coding recognition (AMR) systems to recognise.
- Real-time identification of modulation schemes (part of next milestone) in the SATCOM systems.

References

1. Saad Iqbal and Syed Ali Hassan, "Literature Review and Feasibility Report of Neural Network based Modulation and Channel Coding Identification for SATCOM Systems", December, 2021.
2. Saad Iqbal and Syed Ali Hassan, "Proposed Neural Network for MODCOD report of Neural Network based Modulation and Channel Coding Identification for SATCOM Systems", May, 2022.