Deep Neural Network Based Automatic Modulation Classification

Thesis submitted in partial fulfillment of the

requirements for the degree

of

Master of Engineering

in

Electronics and Tele-Communication Engineering

by

Pritam Sadhukhan

Registration No. : 160207 of 2021-2022

Examination Roll No. : M4ETC23002

Under the guidance of

Dr. Jaydeb Bhaumik

Professor

Department of Electronics and Tele-Communication Engineering

Jadavpur University

Kolkata – 700032

West Bengal, India

September 2023

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled "**Deep Neural Network Based Automatic Modulation Classification**" submitted by **Pritam Sadhukhan** (Class Roll No.: **002110702008**, Examination Roll No.: **M4ETC23002** and Registration No.: **160207 of 2021-2022**) of Jadavpur University, Kolkata is a record of bonafide research work carried out by him under my guidance and supervision and can be accepted in partial fulfillment of the requirements for the degree of **Master of Engineering in Electronics and Tele-Communication Engineering**, with specialisation in **Communication Engineering**, of the University. The results presented in this thesis have been verified and are found to be satisfactory. The results presented in this thesis are not included in any other paper submitted for the award of degree to any other University or Institute.

Prof. Jaydeb Bhaumik
Supervisor
Department of Electronics and Tele-Communication Engineering
Jadavpur University
Kolkata – 700032

Prof. Manotosh Biswas
Head of the Department
Department of Electronics and
Tele-Communication Engineering
Jadavpur University
Kolkata – 700032

Prof. Saswati Mazumdar
Dean
Faculty of Engineering &
Technology
Jadavpur University
Kolkata – 700032

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL*

The foregoing thesis entitled "**Deep Neural Network Based Automatic Modulation Classification**" is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval, the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

Committee on Final Examination

for Evaluation of the Thesis

____

Signature of the External Examiner

____

Signature of the Supervisor

Dr. Jaydeb Bhaumik

Supervisor
Department of Electronics and
Tele-Communication Engineering
Jadavpur University
Kolkata – 700032

*Only in the case the thesis is approved.

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

DECLARATION

I hereby declare that this thesis entitled "**Deep Neural Network Based Automatic Modulation Classification**" contains a literature survey and original research work by the undersigned, as a part of my degree of **Master of Engineering in Electronics and Tele-Communication Engineering**, with specialisation in **Communication Engineering.**

All information has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results which are not original to this work.

Name: **Pritam Sadhukhan**

Class Roll No.: **002110702008**

Registration No.: **160207 of 2021-2022**

Examination Roll No.: **M4ETC23002**

Thesis Title: **Deep Neural Network Based Automatic Modulation Classification**

____

Signature of the Candidate

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis supervisor, Dr. Jaydeb Bhaumik, Professor of Department of Electronics and Tele-Communication Engineering, Jadavpur University for guiding and motivating me throughout my thesis work.

I would like to express my gratitude to Prof. Manotosh Biswas, Head of the Department, Electronics and Tele-Communication Engineering, Jadavpur University. I would also like to express my thanks to Prof. Ananda Shankar Chowdhury and all other faculty members of Department of Electronics and Tele-Communication Engineering, Jadavpur University.

I would also like to give special thanks to my parents Mrs Aparna Sadhukhan & Mr. Pradip Kumar Sadhukhan and other family members for their continuous support throughout the research work and writing the thesis. Their blessings sustained me this far.

Pritam Sadhukhan
Jadavpur University
Kolkata- 700032
West Bengal
India

ABSTRACT

In recent times, Deep Neural Network (DNN) has drawn a lot of interest due to its exceptional performance in identifying complex structured data. In this thesis, the automatic classification of modulation types of analog and digital modulated signals is examined using the DNN method. Due to its crucial function in dynamic spectrum access, which can support fifth generation (5G) wireless communications, automatic modulation classification (AMC) is an unavoidable component of different intelligent communication systems. The AMC has been investigated for more than 25 years, but it has proven challenging to create a classifier that is effective in a variety of multipath fading scenarios and other limitations. AMC systems have recently embraced DNN or Deep Learning (DL) based approaches, and significant advancements have been noted. This thesis suggests AMC approaches based on Convolutional Neural Network (CNN), Residual Neural Network (ResNet), and Convolutional Long Short-Term Deep Neural Network (CLDNN). The RadioML2016.10a dataset has been used in this investigation. It comprises of synthetic signals with 11 modulation types: AM-DSB, AM-SSB, WBFM, GFSK, CPFSK, PAM-4, BPSK, QPSK, 8-PSK, 16-QAM, and 64-QAM. Google Colaboratory has been used as the foundation for all the simulations. Batch Normalization layers have been added after each convolutional layer and first dense layer in the CNN based AMC model and significant accuracy improvement is observed. An accuracy of 68.54% is achieved at high SNR for the CNN based AMC model with Batch Normalization. We have tuned the hyperparameter, dropout rate to 0.3 in ResNet based AMC model and achieved an accuracy of 73.773% at 6 dB SNR. The hyperparameter, batch size has been tuned to 32 and an accuracy of 30.682% has been achieved at 14 dB SNR in the CLDNN based AMC model, which is 12.876% better compared to the CLDNN model with batch size 128.

Keywords**:** Automatic Modulation Classification, Deep Neural Network, Deep Learning, Convolutional Neural Network, Residual Network, Convolutional Long Short-Term Deep Neural Network, Batch Normalization

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Overview

Wireless communications over long distances revolutionized the way world functions, works, and interacts. The technique that made all this possible is modulation. Signal modulation is an important process in wireless communication systems. Modulation is simply the process to modulate the amplitude, frequency, or phase of a carrier signal in accordance to the amplitude of the message or baseband signal. Some of the advantages of modulation are as follows:

- Antenna size gets reduced
- Communication range increases
- Multiplexing can be performed
- No signal mixing occurs
- Reception quality improves

Automatic modulation classification (AMC) is the process to detect, identify and classify modulation format of the received signals automatically. AMC is considered to be an important part of various civilian and military communications systems, such as electronic warfare, radio surveillance and spectrum awareness. AMC is an essential part of cognitive radio (CR) and the demand for cognitive radio in communications and electronic warfare is rapidly increasing.

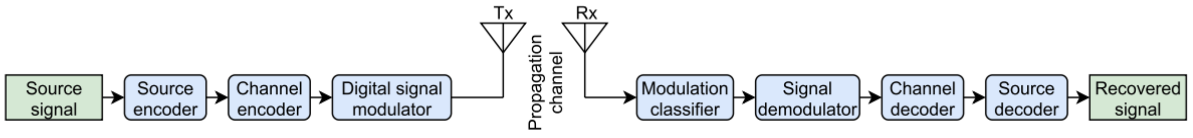


Fig. 1.1 Basic block diagram of digital communication system

The majority of conventional modulation recognition techniques fall under the likelihood-based (LB) and feature-based (FB) schemes categories. The AMC problem is approached by LB methods as a multi-hypothesis problem, where the received signal will be the modulation type with the highest likelihood among all options. Because likelihood-based methods need sophisticated computations, they perform significantly worse when there is phase or frequency offset or impulsive noise. FB approaches, on the other hand, manually extract statistical characteristics before applying a variety of machine learning techniques, such the support vector machine (SVM), decision tree, etc., to categorize and identify data. These currently used conventional procedures are labour and time-intensive, with low recognition rates. This method is essentially ineffective, especially when SNR is low and it is nearly hard to manually extract features.

Due to the explosive growth of raw data volume, the quick advancement of processing power, and the appearance of new deep learning (DL) models like GoogleNet, AlexNet, VGGNet, ResNet, etc., more and more people are becoming interested in artificial intelligence (AI)..

Recent research has demonstrated that end-to-end deep learning models may learn the distribution properties of complicated data and have higher classification performance when categorizing signal modulation by using their potent fitting capabilities. These techniques are more reliable than conventional techniques. For AMC tasks, a variety of deep learning models have been used, including Convolutional Neural Networks (CNN), Residual Networks (ResNet), and Densely Connected Convolutional Neural Networks (DenseNet).In this thesis work, we have solved the AMC problem using three different Deep Neural Network Models which are as follows:

- Convolutional Neural Network (CNN)
- Residual Network (ResNet)
- Convolutional Long Short-Term Deep Neural Network (CLDNN)

1.2 Contributions

The major contributions of this thesis work are as follows:

- Batch Normalization layer has been added after each convolution layer and first dense layer. This resulted in significant accuracy improvement. Addition of Batch Normalization layer in CNN based AMC model is novel in this field.
- Dropout rate has been tuned to 0.3 in ResNet based AMC and major improvement in accuracy is observed without any significant increase in overfitting.
- We have tuned another hyperparameter namely the Batch Size in CLDNN based AMC model to 32 and noted improvement in accuracy by a considerable margin.

1.3 Thesis Layout

The rest of the thesis is organized as follows:

Chapter 2 presents the necessary theoretical backgrounds and literature survey.

Chapter 3 presents the theoretical understanding of Convolutional Neural Network (CNN) and describes the model performance of CNN based AMC for various epochs.

Chapter 4 presents the theoretical understanding of Residual Network (ResNet) and describes the model performance of ResNet based AMC for various epochs

Chapter 5 presents the theoretical understanding of the LSTM layer and the Convolutional Long Short-Term Deep Neural Network (CLDNN) and describes the model performance of CLDNN based AMC for various epochs

Chapter 6 concludes the thesis with summary and future scopes.

CHAPTER 2

Background and Literature Survey

2.1 Modulation of Signals

Modulation refers to a process that moves the message signal into a specific frequency band. It is defined as the process by which some characteristic of a carrier is varied in accordance with a modulating signal. The baseband signal, which represents the original signal as delivered by a source of information, is referred to as the modulating signal. A high frequency sinusoid is a carrier. The modulated signal is the end product of the modulation process. This modulated signal is transmitted by the transmitter through the channel. To reconstruct the baseband signal at the receiver, the modulated signal must pass through a reversal process called demodulation.

Reasons for modulation:

- Ease of radiation/transmission
- Multiplexing, which is the simultaneous transmission of multiple signals

One of the carrier sinusoidal properties, such as amplitude, frequency, or phase, is changed proportionally to the baseband signal by modulation. As a result, we have phase modulation (PM), amplitude modulation (AM), or frequency modulation (FM). According to the type of baseband or modulating signal we have two different kinds of modulation – analog modulation and digital modulation. In this research work the following modulation types are considered:

- Analog modulation: AM-DSB, AM-SSB, WBFM
- Digital modulation: BPSK, QPSK, 8PSK, 16-QAM, 64-QAM, GFSK, CPFSK, PAM4

2.1.1 AM-DSB

The basic philosophy of DSBSC (Double Side Band Suppressed Carrier) is to multiply the message signal with a sinusoid say $\cos(\omega_c t)$ and then to obtain a signal which is simply the frequency shifted version of the message signal.

Consider a message signal *m(t),* then we have,

$$\phi_{DSBSC}(t) = m(t)\cos(\omega_c t) \tag{2.1}$$

But due to the problem of synchronisation i.e., to generate the same frequency f_c in receiver another modulation scheme is introduced namely Double Side Band Full Carrier (DSBFC) where an additional carrier is sent with the DSBSC signal. Hence,

$$\phi_{DSBFC}(t) = m(t)\cos(\omega_c t) + A\cos(\omega_c t)$$

$$\phi_{DSBFC}(t) = [A + m(t)]\cos(\omega_c t) \tag{2.2}$$

In this research work AM-DSB (Amplitude Modulation-Double Side Band) represents DSBFC signal.

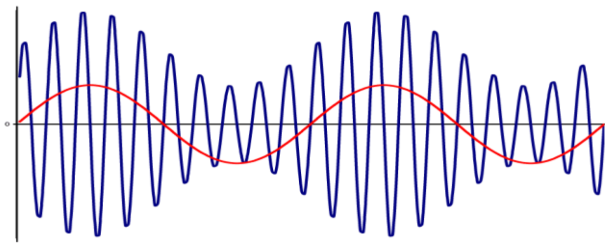


Fig. 2.1 AM-DSB modulated signal

2.1.2 AM-SSB

The upper sideband (USB) and the lower sideband (LSB), which each include the entire information of the baseband signal m(t), are the two sidebands that make up the DSB spectrum (including supressed carrier and full carrier). As a result, DSB modulations require twice as much radio frequency (RF) bandwidth to transmit for a baseband signal m(t) of bandwidth B Hz. The Single-sideband (SSB) modulation technique, which eliminates either the LSB or the USB that uses bandwidth of B Hz for one message signal m(t), is presented to increase the spectral efficiency of AM. The equations for AM-SSB modulated signals are as shown:

$$\phi_{USBSC}(t) = m(t)\cos(\omega_c t) - m_h(t)\sin(\omega_c t) \tag{2.3}$$

$$\phi_{LSBSC}(t) = m(t)\cos(\omega_c t) + m_h(t)\sin(\omega_c t) \tag{2.4}$$

Hence, combining equation (2.3) and (2.4) we get,

$$\phi_{SSBSC}(t) = m(t)\cos(\omega_c t) \pm m_h(t)\sin(\omega_c t) \tag{2.5}$$

where $m_h(t)$ is the Hilbert Transform of m(t).

2.1.3 WBFM

In angle modulation the angle (phase/frequency) of the carrier is varied with accordance to the amplitude of the message signal.

The instantaneous frequency in case of a frquency modulated signal for a message signal m(t) is given as,

$$\omega_i = \omega_c + K_f\, m(t) \tag{2.6}$$

where K_f is the frequency sensitivity measured in rad/volt.

Now the phase of the modulated signal is given as,

$$\theta(t) = \int_0^t \omega_i \, dt$$

$$= \int_0^t \omega_c + K_f\, m(t)\, dt$$

$$= \omega_c t + K_f \int_0^t m(t)\, dt$$

Hence the expression for the frequency modulated signal is given as,

$$\phi_{FM} = A_c \cos \theta(t)$$

$$\phi_{FM} = A_c \cos [\omega_c t + K_f \int_0^t m(t)\, dt] \qquad (2.7)$$

When the message is a single tone sinusoid then the FM obtained from an FM generator is known as a single tone FM.

$$m(t) = A_m \cos \omega_m t$$

So,

$$\phi_{FM} = A_c \cos [\omega_c t + K_f \int_0^t m(t)\, dt]$$

$$= A_c \cos [\omega_c t + K_f A_m \int_0^t cos\omega_m t \, dt]$$

$$= A_c \cos [\omega_c t + \frac{Kf\ Am}{\omega m} \sin \omega_m t]$$

$$= A_c \cos [\omega_c t + \frac{\Delta\omega}{\omega m} \sin \omega_m t]$$

$$= A_c \cos [\omega_c t + \beta \sin \omega_m t] \qquad (2.8)$$

where, β is the *modulation index*. If β is less, then the FM generated is called a Narrow Band FM (NBFM). If the β is high, then the FM generated is called Wide Band FM (WBFM). WBFM is considered here in this thesis work.

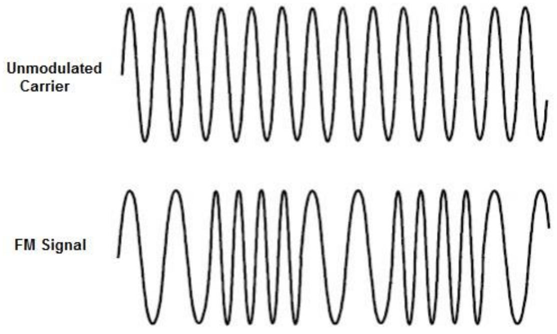


Fig. 2.2 Unmodulated carrier and an FM signal

2.1.4 BPSK

Binary Phase Shift Keying (BPSK) is a digital modulation method. In coherent BPSK system, the pair of signals, $s_1(t)$ and $s_2(t)$, used to represent binary symbols 1 and 0, respectively, are defined by

$$s_1(t) = \sqrt{(2E_b/T_b)}\ \cos (2\pi f_c t) \qquad (2.9)$$

$$s_2(t) = \sqrt{(2E_b/T_b)}\ \cos (2\pi f_c t + \pi)$$

$$= -\sqrt{(2E_b/T_b)}\ \cos (2\pi f_c t) \qquad (2.10)$$

where $0 \le t \le T_b$ and E_b is transmitted signal energy per bit.

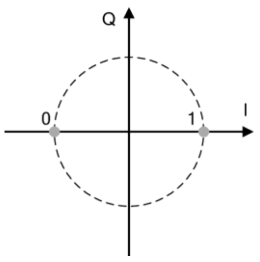


Fig. 2.3 Constellation diagram of BPSK modulation

2.1.5 QPSK

As with the BPSK, Quadrature Phase Shift Keying (QPSK) is characterized by the fact that the information carried by the transmitted signal is contained in the phase. In particular, in QPSK signal, the phase of the carrier takes one of the four possible values, such as π/4, 3π/4, 5π/4, 7π/4, as shown by

$$s_i(t) = \sqrt{\frac{2E}{T}}\cos\left[2\pi f_c t + (2i-1)\frac{\pi}{4}\right] \qquad 0 \le t \le T$$

$$= 0 \qquad \text{elsewhere} \qquad (2.11)$$

where i = 1, 2, 3, 4 and E is the transmitted signal energy per symbol, T is the symbol duration, and the carrier frequency f_c equals n_c/T for some fixed integer n_c. A unique pair of bits known as a dibit corresponds to each conceivable value of the phase. Thus, for example, we may choose the above set of phase values to represent the following set of dibits: 10, 00, 01 and 11.

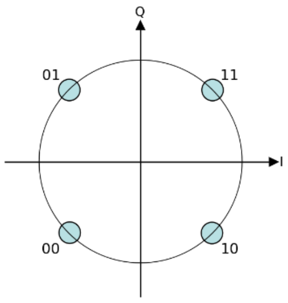


Fig. 2.4 Constellation diagram of QPSK modulation

2.1.6 8PSK

In 8PSK system, the phase of the carrier takes on one of the 8 possible values $\theta_i = 2i\pi/8$, where i = 1, 2, 3 … 8. Accordingly, during each signalling interval of duration T, one of the 8 possible signals:

$$s_i(t) = \sqrt{\frac{2E}{T}}\cos\left[2\pi f_c t + \frac{2i\pi}{8}\right] \qquad i = 1, 2, 3 \dots 8 \qquad (2.12)$$

is transmitted, where E denotes the energy of a signal per symbol. The carrier frequency $f_c = n_c/T$ for some fixed integer n_c.

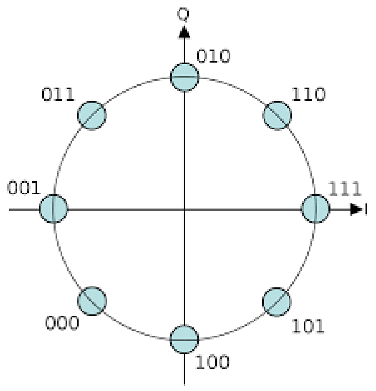


Fig. 2.5 Constellation diagram of 8PSK modulation

2.1.7 16-QAM

In QAM-16 (Quadrature Amplitude Modulation) the carrier signal of fixed frequency can exist in one of the sixteen different states. Each symbol (carrier signal) represents a sequence of 4 bits (0000 to 1111). The constellation diagram of 16-QAM is as shown in figure 2.6.

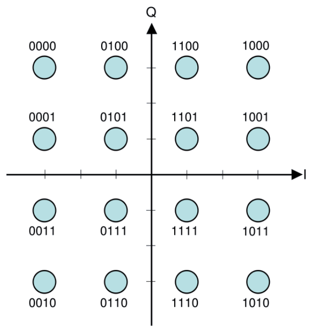


Fig. 2.6 Constellation diagram of 16-QAM

2.1.8 64-QAM

In QAM-64 the carrier signal of fixed frequency can exist in one of the sixty-four different states. Each symbol (carrier signal) represents a sequence of 6 bits (000000 to 111111). The constellation diagram of 64-QAM is as shown in figure 2.7.

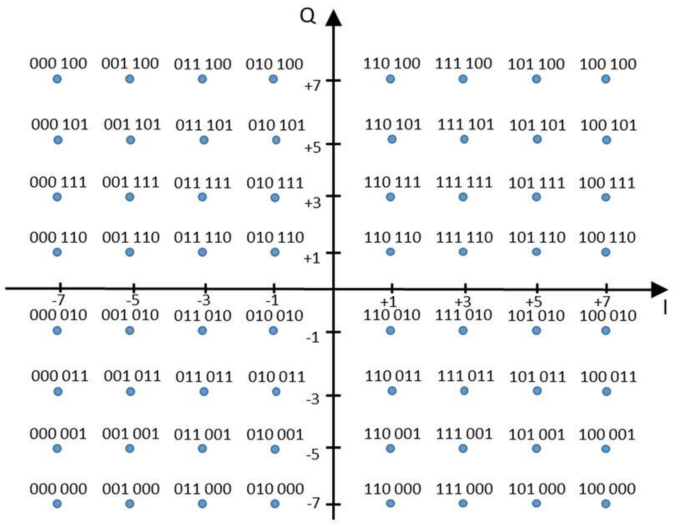


Fig. 2.7 Constellation diagram of 64-QAM

2.1.9 GFSK

In a binary FSK (Frequency Shift Keying) system symbol 1 and 0 are distinguished from each other by transmitting one of two sinusoidal signals that differ in frequency by a fixed amount. A typical pair of sinusoid signals is described by

$$s_i(t) = \sqrt{(2E_b/T_b)}\ \cos(2\pi f_i t) \qquad 0 \leq t \leq T$$

$$s_i(t) = 0 \qquad \text{elsewhere} \qquad (2.13)$$

where i = 1, 2 and E_b is the transmitted signal energy per bit, and the transmitted frequency equals

$$f_c = (n_c+i)/T_b \ \text{ for some fixed integer } n_c \text{ and } i = 1,2 \qquad (2.14)$$

Thus symbol 1 is represented by $s_1(t)$ and symbol 0 by $s_2(t)$.

Gaussian Frequency Shift Keying (GFSK) is a type of FSK modulation where the frequency of the modulated signal does not change instantaneously at the beginning of each symbol period of binary data. Thus, the transition from bit 0 to 1 and 1 to 0 becomes smoother.

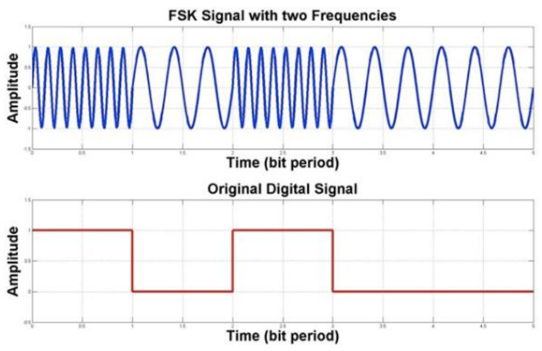


Fig. 2.8 FSK modulation

2.1.10 CPFSK

A typical frequency shift keyed (FSK) signal that is limited to preserve continuous phase at its symbol time boundaries is what Continuous Phase Frequency Shift Keying (CPFSK) is. Both the performance of the error rate and the containment of the signal spectrum are significantly improved by this constraint.

2.1.11 PAM4

When using pulse-amplitude modulation (PAM), the instantaneous sample values of a continuous message signal are matched one-to-one with the amplitudes of regularly spaced rectangular pulses. A PAM signal is defined by:

$$s(t) = \sum_{n=-\infty}^{\infty} [\, 1 + k_a\, m(nT_s)]\; g(t - nT_s) \qquad (2.15)$$

where $m(nT_s)$ represents the nth sample of the message signal m(t), T_s is the sampling period, k_a is a constant called *amplitude sensitivity*, and g(t) denotes the pulse. In a PAM4 modulation the rectangular pulses can take one of the four possible values after modulation.

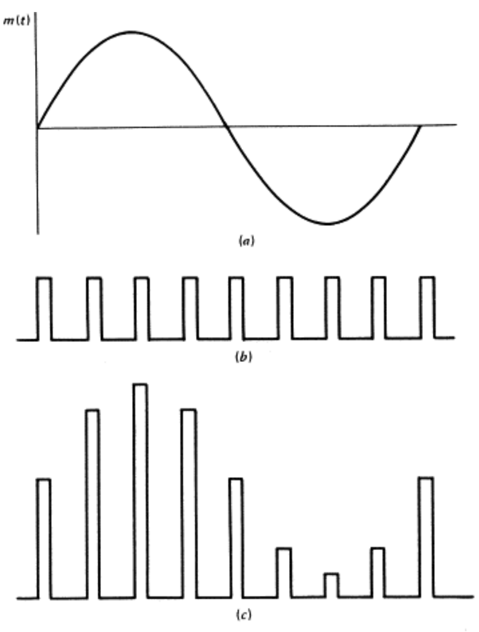


Fig. 2.9 (a) message signal m(t) (b) pulse carrier (c) PAM signal

2.2 Automatic Modulation Classification

2.2.1 Overview

Automatic modulation classification (AMC) is the technique to automatically detect the modulation type of the received signal from the received signal samples. It is a process often performed after signal detection and before signal demodulation.

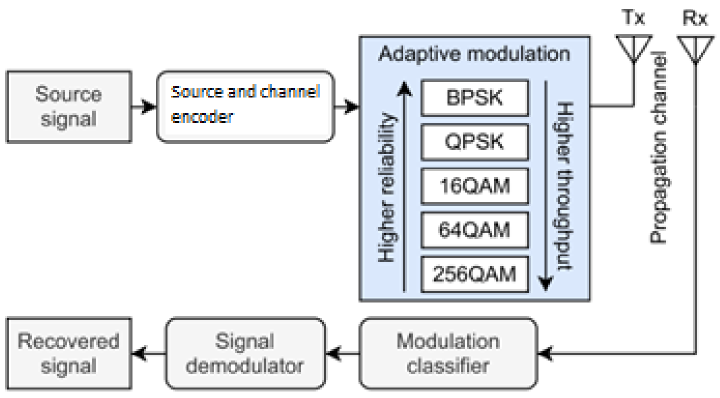


Fig. 2.10 Position of modulation classifier in a communication system with adaptive modulation

2.2.2 Automatic Modulation Classification Algorithms

Automatic modulation classification algorithms are classified into two broad categories: Likelihood based (LB) approach and Feature based (FB) approach.

The Likelihood based approach deals with the automatic modulation classification problem as a multi-hypothesis problem where the modulation type with the maximum likelihood among all the candidates will be assigned to the received signal. Many Likelihood based algorithms are suggested in the literature such as Generalized Likelihood Ratio Test (GLRT) [1], Average Likelihood Ratio Test (ALRT) [1], Hybrid Likelihood Ratio Test (HLRT) [2] and Quasi-Hybrid Likelihood Ratio Test (QHLRT) [2]. Generally, the computational complexity of the Likelihood based schemes is very high and they show poor performance in the existence of frequency or phase offset or impulsive noise.

On the other hand, Feature based schemes are easy to implement and can achieve great performance if the used features are chosen properly. The received signal features that are suggested in the literature are: instantaneous phase, amplitude and frequency [3], Fourier transform [4], wavelet transform, [5], high order moments (HOMs), high order cumulants (HOCs) [6, 7, 8], higher order cyclic cumulants [9], very high order statistics (VHOS) [10] and constellation diagram [11]. Generally, the proper features are selected depending upon the modulation type of interest. The selected features are used by the machine learning (ML) classifier to recognize the modulation type of the received signal. The types of ML classifiers used in automatic modulation recognition domain, as described in the literature, are: Artificial Neural Networks (ANN) [12], Support Vector Machines (SVM) [13], Polynomial Classifier (PC) [14], Clustering Algorithms, K-Nearest Neighbours (KNN) [15], Threshold-Based Classifiers [7,16] and Naïve Bayes Classifier [17]. Moreover, some researchers used optimization techniques such as Genetic Programming [15,18,19] and Particle Swarm Optimization (PSO) [13] in order to improve the classification features. The channel conditions set a limit in classifier's accuracy. Many researchers investigated AMC schemes in AWGN (Additive White Gaussian Noise) channels as in [17,18], whereas, other researchers have considered more realistic channel models that took into consideration multipath fading [20].

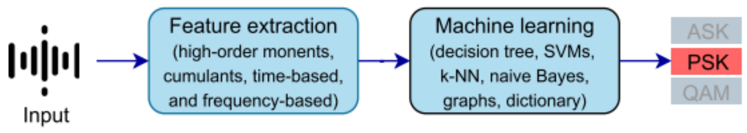

Input
Feature extraction
(high-order moments, cumulants, time-based, and frequency-based)
Machine learning
(decision tree, SVMs, k-NN, naive Bayes, graphs, dictionary)
ASK
PSK
QAM

Fig. 2.11 Overall processing flow of feature-based AMC [21]

2.2.3 Application Areas of Automatic Modulation Classification

The radio frequency (RF) spectrum is a crucial resource in radio communication technology. The RF spectrum is becoming more and more congested and scarce as the number of mobile devices increase. To address this issue of spectrum shortage Cognitive Radio (CR) proposes a dynamic spectrum access method where the idle band can be accessed dynamically. In order to ensure the normal usage of spectrum in cognitive radio automatic modulation classification (AMC) is a key technology to identify the modulation types of the target signals corrupted by noise and interference.

Apart from cognitive radio AMC is widely used in other civilian and military communication fields such as software radio, signal detection, spectrum detection and management, adaptive modulation transmission, threat analysis, interference identification, electronic reconnaissance, signal authentication, non-cooperative communication etc.

2.3 Neural Networks

2.3.1 Overview

A machine learning (ML) technique called a neural network is modelled after and inspired by the human nervous system and the structure of the human brain. Processing units are often arranged in input, hidden, and output layers. Each layer's nodes, units, or neurons are linked to the nodes or neurons of the layer before and after it. Every link has a weight value. At each unit, the weights are multiplied by the inputs and added. Depending on the activation function, the sum then goes through a transformation. The usage of these functions makes it simpler to compute the partial derivative of the delta of the error with regard to specific weights because of their mathematically advantageous derivative.

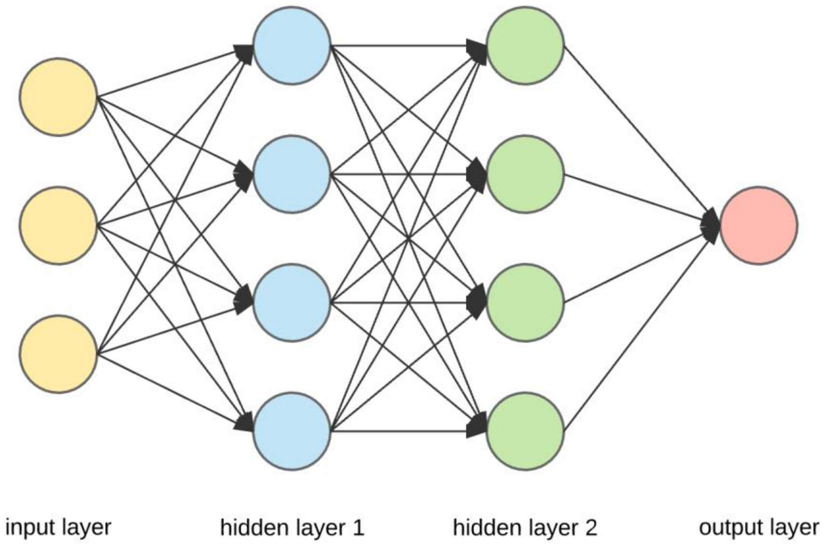


Fig. 2.12 Structure of a Neural Network with 2 hidden layers for binary classification

2.3.2 Training of Neural Network

Training a model consists of minimizing a loss function which reflects the performance of the predictor on a training set. Because the models are usually very complex, and their performance is directly related to how well the loss is minimized, the minimization is a key challenge which consists of both computational and mathematical difficulties.

2.3.2.1 <u>Losses</u>:

A loss, cost, or error function is a function used in decision theory and mathematical optimization that transfers an event or the values of one or more variables onto a real number that intuitively represents some "cost" connected to the occurrence. Loss is essentially the cost or inaccuracy for a single observation. Different types of Cost functions are used for different classification problems. The classification problems and the Cost function used for those problems are described below:

- Linear Regression: For linear regression problem the output can take any number. In such cases the error is given as,

$$\text{error} = a_i - y_i \tag{2.16}$$

  where,
  a_i = predicting value
  y_i = actual output label

  We generally take the absolute value. hence ,

$$\text{error} = |a_i - y_i| \tag{2.17}$$

  This is the error for only one observation. If we have m number of observations then we can take the average of all those errors and we will get the cost function.

$$\text{cost} = \frac{1}{m}\sum|a_i - y_i| \tag{2.18}$$

  This type of cost function is known as “Mean Absolute Error”. The other type of error namely “Mean Squared Error” is also used for linear regression problems.

$$\text{cost} = \frac{1}{2m}\sum|a_i - y_i|^2 \tag{2.19}$$

  Here, (2.17) is the loss for ith observation

- Binary Classification: In Binary Classification the output takes only two values : 0 &1. Error for one observation is gives as,

$$\text{error} = -[y_i \log(a_i) + (1-y_i)\log(1-a_i)] \tag{2.20}$$

  Cost function is obtained by taking the average of all these errors. Hence,

$$\text{cost} = \frac{-1}{m} \sum [y_i \log(a_i) + (1-y_i)\log(1-a_i)] \tag{2.21}$$

  This type of cost function is known as ‘Binary Cross Entropy’.

- Multi-Class Classification: For multi-class classification the output can take many categories. For multi-class classification the output label for the ith observation is given by a one-hot representation. One-hot representation means that output will be one for 1 position and 0 for other positions. As,

$$y_i = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad (2.22)$$

The cost function for multi-class classification is given as,

$$\text{cost} = - \sum_{j=0}^{M} \sum_{i=0}^{N} (y_{ij} * \log(a_{ij})) \quad (2.23)$$

Where,

N = total number of neurons in output layer

M = total number of observations

This cost function is known as “Categorical Cross Entropy” cost function. The indicator of the distinction between two probability distributions is categorical cross entropy. The probability distribution for deep learning classification tasks (like the one in this thesis) is often a Softmax. The error is estimated during the so-called forward pass, and during the so-called backward pass, weights are modified using the chain rule to identify errors for each parameter.

We will use this Categorical Cross Entropy as our cost function in this research work.

2.3.2.2 Gradient Descent:

Gradient descent is an optimization process that reduces the cost function by iteratively changing parameters in the direction of the negative gradient, with the goal of identifying the best set of parameters. It is used in machine learning.

The difference between the model's projected and actual outputs is represented by the cost function. Finding the set of parameters that minimizes this error and enhances the model's performance is the aim of gradient descent.

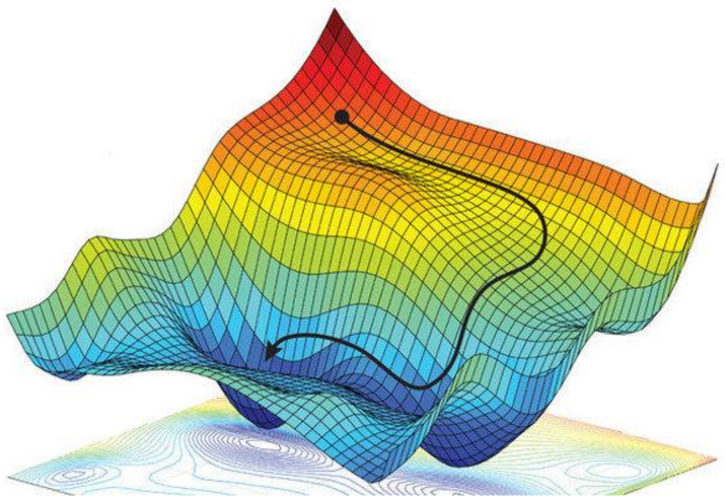


Fig. 2.12 Direction of the gradient flow in the Cost function graph plotted along z axis

2.3.2.3 Backpropagation:

One of a neural network's most important components is the backpropagation algorithm. Through the chain rule method, the algorithm is utilized to train a neural network efficiently. In other words, backpropagation makes a backward pass through a network after each forward run while modifying the model's parameters, such as the weights and biases.

2.3.3 Model components of Neural Network

2.3.3.1 The notion of layers:

Layers are typical complex compounded tensor operations that have been developed and empirically proven to be efficient and general. Modern models may take the shape of a complicated graph that takes into account numerous parallel paths.

2.3.3.2 Activation functions:

The weighted sum of the inputs and biases, which is used to decide whether or not a neuron can be activated in a neural network, is calculated using activation functions. It manages the displayed data and generates an output for the neural network that uses the data's parameters. Some texts also refer to the activation functions as transfer functions. These regulate the output of neural networks across several domains and can be either linear or nonlinear depending on the function they represent. Some of the popular activation functions are Sigmoid, hyperbolic tan (tanh), Rectified Linear Unit (ReLU), Leaky ReLU, Exponential Linear Unit (ELU), Softmax etc.

We have used ReLU and Softmax activation function in our Deep Learning model. The expression for ReLU is as shown:

$$f(x) = \text{ReLU}(x) = \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{if } x < 0, \end{cases} \tag{2.24}$$

Figure 2.13 shows the graph of ReLU function.

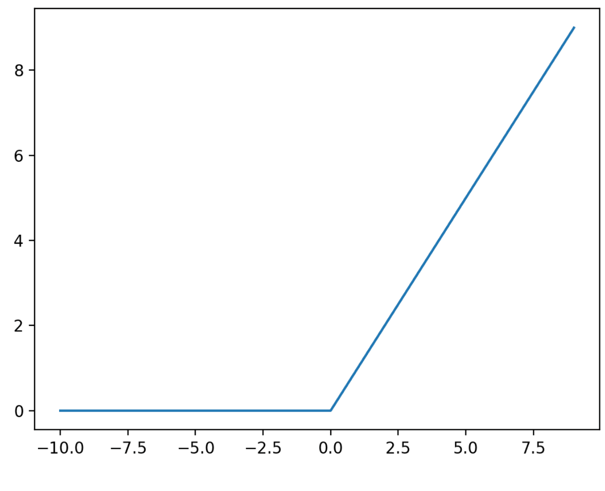


Fig. 2.13 Graph of the ReLU activation function

The expression for Softmax activation function is as follows:

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}},\ i,\ j = 1, 2, \cdots, N \tag{2.25}$$

Where y_i and N are any element of classes and the number of classes respectively.

2.3.3.3 **Pooling**:

One of the core components of convolutional neural networks is the pooling layer. Convolutional layers extract features from images, whereas pooling layers combine the features discovered by CNNs. Its guiding principle is to gradually reduce the spatial dimension of the representation in order to reduce the number of network parameters and computations.

2.3.3.4 **Dropout**:

The term "dropout" describes the removal of nodes from a neural network's input and hidden layers, as seen in figure 2.14. With a dropout probability of p, the nodes are dropped.

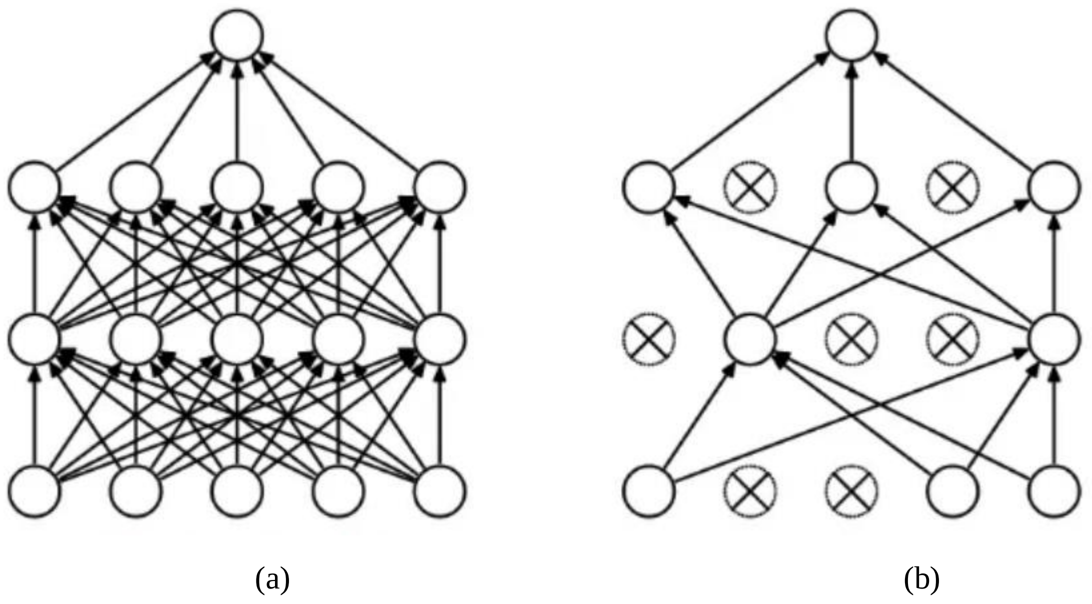


Fig. 2.14 (a) Parent Neural Network (b) Neural Network after applying dropout

2.4 Neural Network based Automatic Modulation Classification (Literature Survey)

Various Artificial Neural Network (ANN) based automatic modulation classification schemes have been suggested in the literature. Some of the schemes are described below:

- In [22], an improved automated modulation classification network (IC-AMCNet) based convolutional neural network (CNN) is suggested. To meet with the estimated latency criteria in communications beyond fifth generation, various levels have been changed and new types of layers have been added as compared to the current CNN architecture.
  Convolution, dropout, and Gaussian noise layers, among others, are used for regularization and to lessen system overfitting in order to improve the current architecture.
  To cut down on the amount of processing time needed by the system, each layer uses a small number of filters rather than many filters.

  The researchers concentrated on the computing time as well as the accuracy rate. To meet with beyond 5G communications standards, they tried to limit the system's processing time below 0.01 milliseconds.
  With a GPU 1xTesla K80, 2496 CUDA cores, and 12GB GDDR5 VRAM on top of Google Colaboratory, the system was simulated in Keras. RML2016.10b, the dataset utilized in this study, was created using GNU Radio. The RML2016.10b dataset includes 10 types of modulated signals, 1200000 data points, and 1024 samples. The dataset was created using the GNU Radio Dynamic Channel Model hierarchical block, which incorporates several effects including sampling rate offset, centre frequency offset, selective fading, and AWGN (Additive White Gaussian Noise) to replicate real-world conditions of the signal. The dataset was saved in Kaggle and imported to Google Colaboratory.

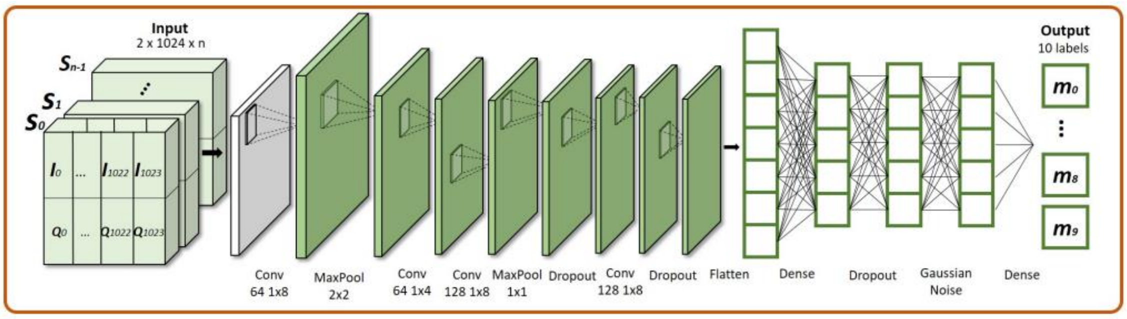


Fig. 2.15 IC-AMCNet architecture [22]

The IC-AMCNet's planned design is depicted in Figure 2.15 from [22]. Modulated signals with a 2 x 1024 IQ time-domain vector are the system's input. The input is

resized and the zero-padding function is used before applying it to the first convolution layer. ReLU activation functions with 64 filters are used in the first two convolution layers; the corresponding filter sizes are 1.8 and 1.4. ReLU activation function with 128 filters is used for the third and fourth convolutional layers, with filter sizes of 1.8 and 1.1, respectively. The findings are then normalized between 0 and 1 using the ReLU activation function.
Moreover, max pooling is applied following the first two and fourth convolution layers to minimize the number of parameters that must be learned without compromising the input's quality. In order to avoid overfitting and have a regularizing impact, dropout and Gaussian noise layers are also utilized. After the first dropout, the flatten layer is employed to convert the two-dimensional matrix into a vector. The last dense layer or completely linked network is subjected to the Softmax activation function in order to translate the outcome into a probability value. Here, cross entropy loss function and the Adam optimizer are employed.

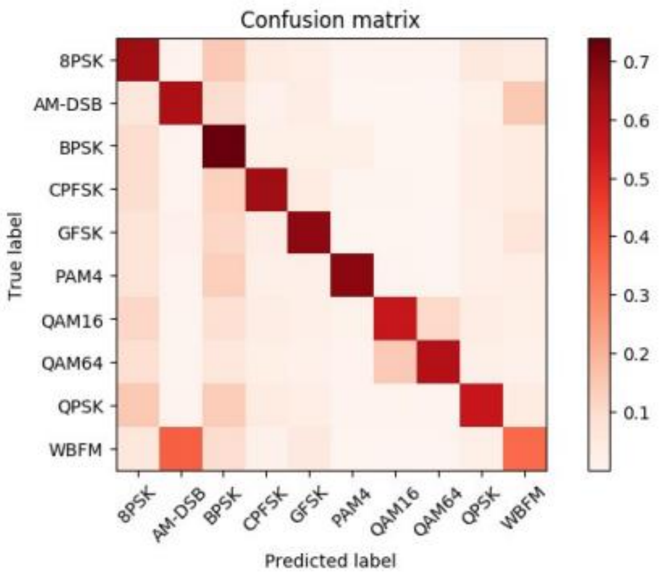


Fig. 2.16 Confusion matrix for the whole SNR of IC-AMCNet [22]

Figure 2.16 shows the confusion matrix for all data of the IC-AMCNet proposed in [22]. We find the distinct diagonal in the confusion matrix with some remaining discrepancies (such as WBFM is misclassified as AM-DSB). The highest accuracy of IC-AMCNet with 0.4 dropout rate, is about 83.40% at 18dB SNR.

- In [23], an automatic modulation classification approach is put forth that is based on improved convolutional neural network (CNN) architecture and wavelet denoising pre-treatment.
  The article [23] makes a better CNN architecture proposal that logically adds pooling layers to remove distracting elements and simplify the architecture. When there is low SNR, it performs well.
  Wavelet denoising technique, which is widely employed in the wireless communication area, is used to reduce the high-frequency noise in signals. The recognition rate is further improved with the wise use of wavelet transformation order.

Additionally, simulation creates a signal dataset. Distractors such SNR ranging from -20 dB to 16 dB, Rayleigh/Rician fading channels, Doppler frequency shift, center frequency, and phase offset, among others, are taken into account.

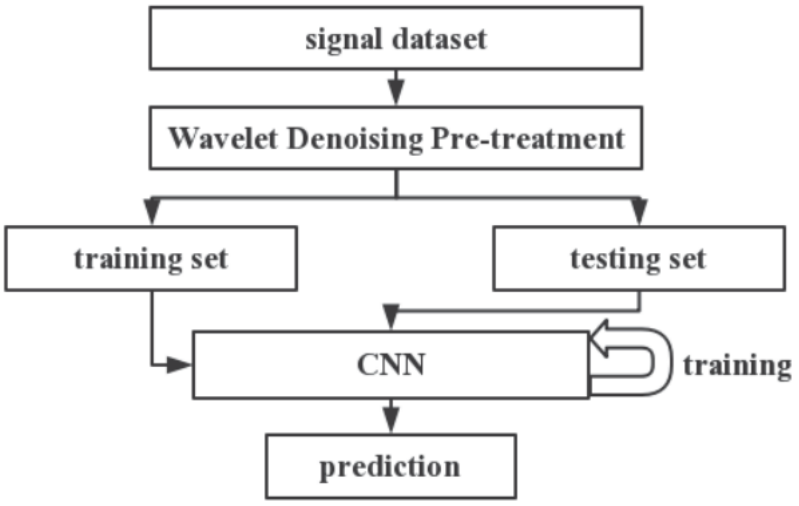


Fig. 2.17 CNN based classification scheme [23]

The wavelet denoising pre-treatment and classification using the modified CNN model make up the CNN classification methodology. Training and test sets are created from the pre-treated signal dataset. The updated CNN model is trained by sequentially feeding it the test and training datasets.

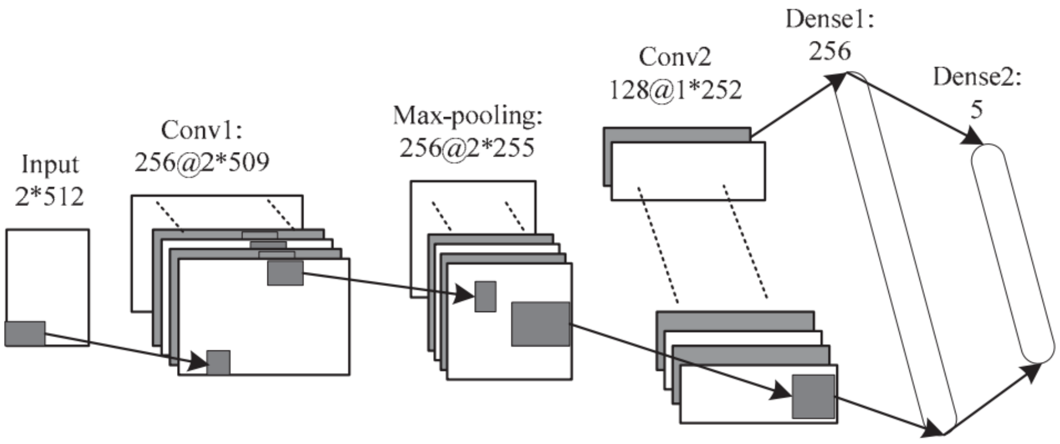
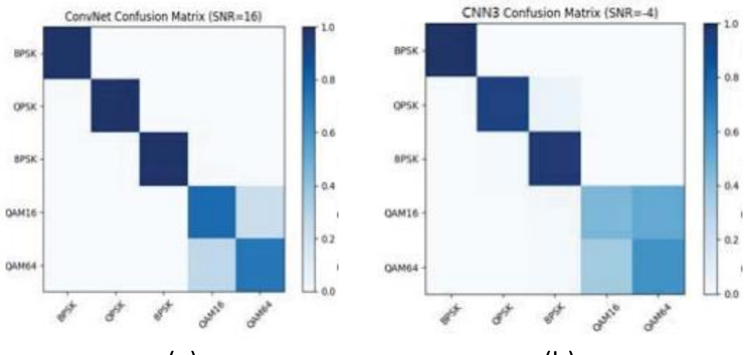


Fig. 2.18 Architecture of the proposed CNN model in [23]

Two convolution layers plus a pooling layer make up the convolution portion of the CNN model as stated. There are 256 convolution kernels in the first convolution layer, each with a size of 1 by 4 and a step of (1,1). The associated pooling layer uses a maximum pooling layer that is 1 by 2. With 128 convolution kernels, the second convolution is created. The step is (1,1), and the kernel size is 2 by 4. A pooling layer, however, is not introduced since it can lead to feature distortions behind the second convolution layer. The output layer and the hidden layer make up the fully connected layer component. The output layer has 5 neurons, whereas the hidden layer has 256 neurons. Rectified Linear Units (ReLU) are used as the activation function in the convolutional and fully connected learning layers as well as the output layer. The output layer uses the Softmax function as its activation function. Utilizing Dropout technology prevents over-fitting. Given that the Dropout factor in this case is set to 0.5, 50% of the neurons will be locked in one learning loop. Adam technique and categorical cross entropy cost function are employed in the training procedure to learn the modulations. The architecture is developed in Python 3.5 and implemented in the Google Tensorflow-based Keras deep learning toolkit. Utilizing the NVIDIA GTX 1060 GPU, accelerate the calculation. 100,000 signal samples were used to create the dataset, which includes 5 different forms of digital modulations (BPSK, QPSK, 8PSK, 16-QAM, and 64-QAM). The training set and testing set are created by randomly and fairly dividing the dataset. The testing set is used to assess the models' performance, while the training set is used to train new models.



(a) (b)

Fig. 2.19 Confusion matrix for the CNN model at (a) SNR = 16dB & (b) SNR = -4dB [23]

The performance of the CNN model developed in [23] is shown in Figure 2.19. This model recognizes QAMs within classes with an accuracy of 70%. Additionally, the wavelet denoising pre-treatment improves MPSK modulation accuracy by more than 90%.

- The authors in [24] proposes a brand-new convolutional neural network (CNN) classifier model to categorize modulation classes according to their families, or types. The classifier is resistant to actual wireless channel limitations, according to the authors.

  Furthermore, taking into account real-life application, a complete dataset called HisarMod2019.1 is also introduced in [24]. With 26 modulation classes, 5 fading types (Ideal (no fading, only AWGN), static, Rayleigh, Rician, (k=3), and Nakagami-m (m=2)), as well as a variety of taps for classification, HisarMod2019.1 comprises 26 modulation classes traveling across the channels. Table 2.1 displays the various modulation types found in the HisarMod2019.1 dataset. The authors have used two datasets - HisarMod2019.1 and RadioML2016.10b and two different models CNN and CLDNN. The CLDNN model used here is inspired by the model described in [25]. Table 2.2 shows the CNN model layout for the two different datasets.

  In multipath fading environment it is difficult to deal with a large dataset, thus it is handled in two steps: modulation family classification and modulation type classification. This process is shown in figure 2.15.

Table 2.1 Modulation types in HisarMod2019.1 dataset [24]

Modulation Family	Modulation Types
Analog	AM–DSB
	AM–SC
	AM–USB
	AM–LSB
	FM
	PM
FSK	2–FSK
	4–FSK
	8–FSK
	16–FSK
PAM	4–PAM
	8–PAM
	16–PAM
PSK	BPSK
	QPSK
	8–PSK
	16–PSK
	32–PSK
	64–PSK
QAM	4–QAM
	8–QAM
	16–QAM
	32–QAM
	64–QAM
	128–QAM
	256–QAM

Table 2.2 CNN layout [24]

Layer	Output Dimensions	
	HisarMod2019.1	RadioML2016.10a
Input	2×1024	2×128
Noise Layer	2×1024	–
Conv1	$2 \times 1024 \times 256$	$2 \times 128 \times 256$
Max_Pool1	$2 \times 512 \times 256$	$2 \times 64 \times 256$
Dropout1	$2 \times 512 \times 256$	$2 \times 64 \times 256$
Conv2	$2 \times 512 \times 128$	$2 \times 64 \times 128$
Max_Pool2	$2 \times 256 \times 128$	$2 \times 32 \times 128$
Dropout2	$2 \times 256 \times 128$	$2 \times 32 \times 128$
Conv3	$2 \times 256 \times 64$	$2 \times 32 \times 64$
Max_Pool3	$2 \times 128 \times 64$	$2 \times 16 \times 64$
Dropout3	$2 \times 128 \times 64$	$2 \times 16 \times 64$
Conv4	$2 \times 128 \times 64$	$2 \times 16 \times 64$
Max_Pool4	$2 \times 64 \times 64$	$2 \times 8 \times 64$
Dropout4	$2 \times 64 \times 64$	$2 \times 8 \times 64$
Flatten	8192	1024
Dense1	128	128
Dense2	5	10
Trainable Par.	$15,764,53$	$6,595,94$

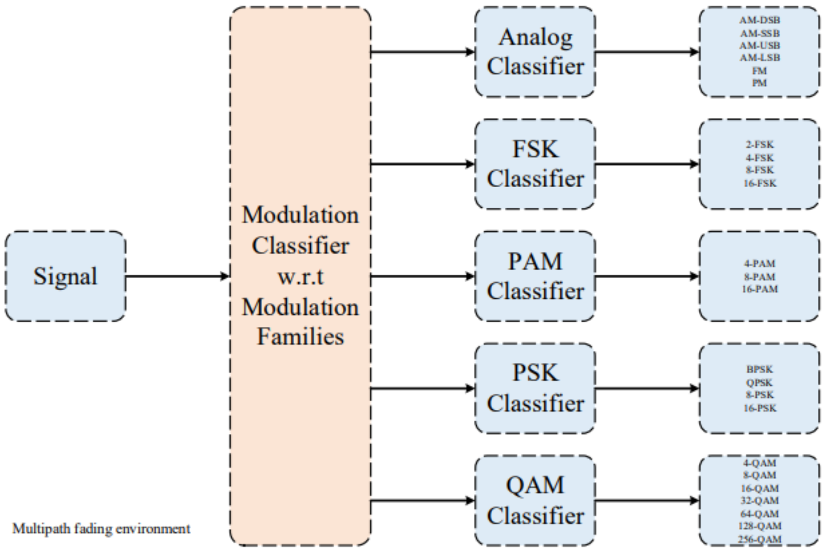


Fig. 2.20 Classification process in [24]

The maximum accuracy for the CNN model described in [23] and CLDNN model described in [24] are 94% and 85% respectively in HisarMod2019.1 dataset.

The maximum accuracy for the CNN model described in [23] and CLDNN model described in [24] are 90.7% and 88.5% respectively in RadioML2016.10b dataset. The confusion matrices for different SNR values are shown for two different datasets in figure 2.21, 2.22 and 2.23.

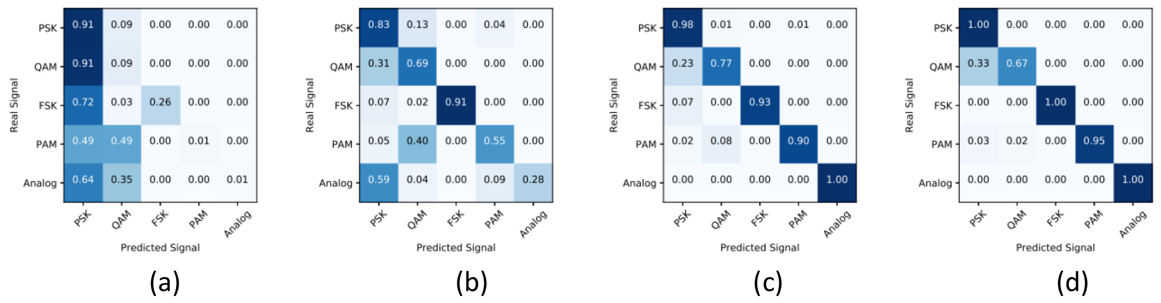


Fig 2.21 Confusion matrices for the CNN model in [24] at (a) 0dB, (b) 6dB, (c) 12dB, (d) 18dB using HisarMod2019.1 dataset

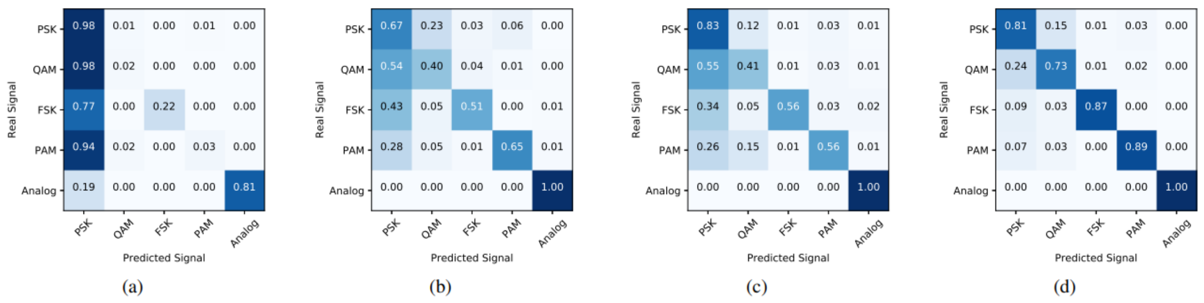


Fig. 2.22 Confusion matrices for the CLDNN model in [25] at (a) 0dB, (b) 6dB, (c) 12dB, (d) 18dB using HisarMod2019.1 dataset

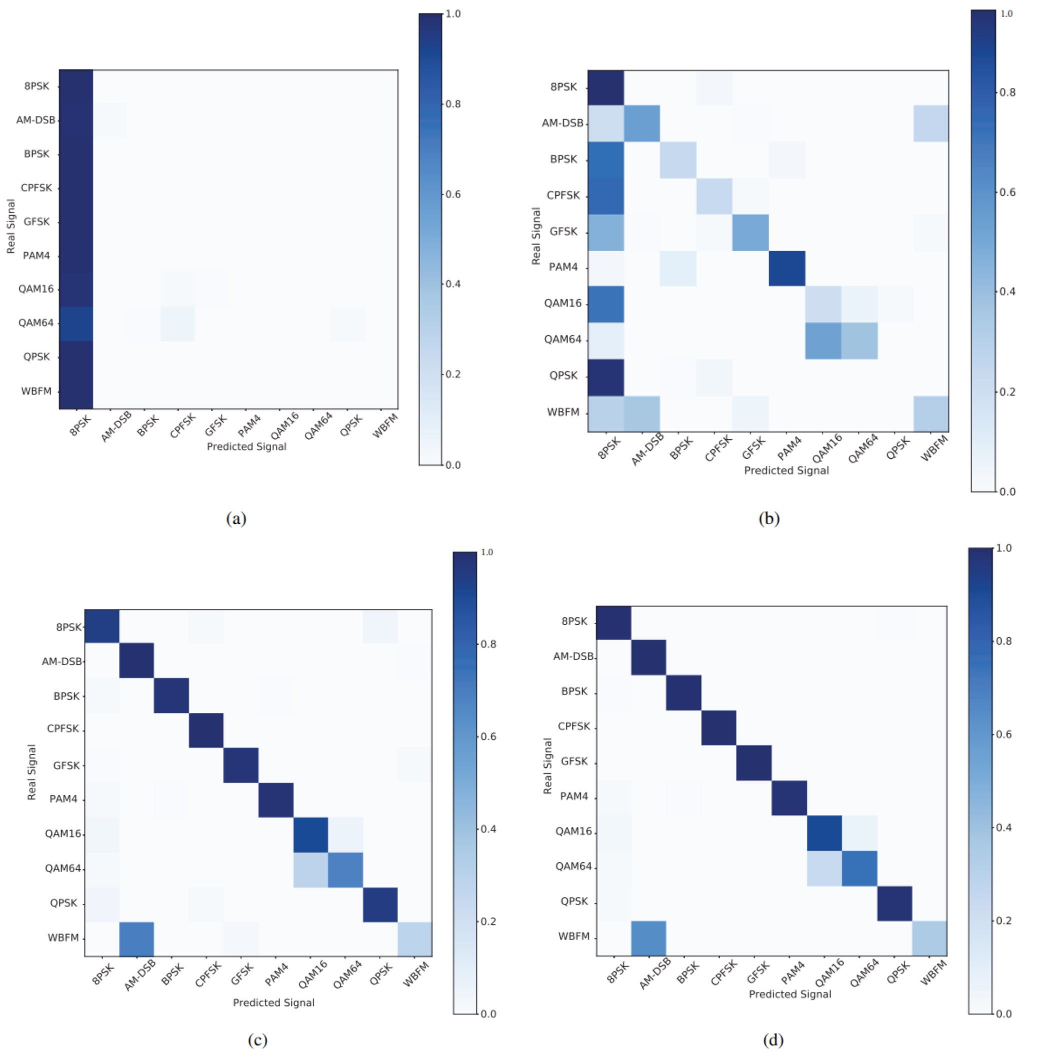


Fig. 2.23 Confusion matrices for the CNN model in [24] at (a)-12dB, (b) -6dB, (c) 0dB, (d) 6dB using RadioML2016.10b dataset

- The value of using deep learning for the problem of recognizing wireless signal modulation has been examined by the authors in [26]. The article [27] describes a framework that mimics the flaws in a real wireless channel by creating a dataset using GNU radio and ten distinct modulation schemes. A convolutional neural network (CNN) architecture was also created and demonstrated to provide performance that is superior to that of expert-based approaches. Following the framework of [27], the authors of [26] developed deep neural network topologies that offer greater accuracy than current standards. The architecture of [27] passed several tests with an accuracy rate of roughly 75% of identifying the modulation type. The authors adjust the CNN architecture of [27] first, and they suggested a configuration with four convolutional layers and two dense layers that provides an accuracy of roughly 83.8% at high SNR values. Then, to obtain high SNR accuracies of about 83.5% and 86.6%, respectively, they created architectures based on the concepts of Residual Networks (ResNet [28]) and Densely Connected Networks (DenseNet [29]). To finally reach an accuracy of about 88.5% at high SNR, the authors incorporated a Convolutional Long Short-term Deep Neural Network (CLDNN [30]).

- The authors of [31] looked at the automatic classification of modulation classes for digitally modulated signals using the DNN technique. First, the authors have chosen 21 statistical features that, for all modulation formats taken into consideration (BPSK, QPSK, 8PSK, 16QAM, and 64QAM), show good separation in empirical distributions. The fully connected DNN with three hidden layers receives these features as input from the signal samples of the received signal. The computer simulation produces training data with 25,000 feature vectors using both Additive White Gaussian Noise (AWGN) and Rician fading channels. According to the findings in [31], the suggested technique significantly outperforms the present classifier, especially for strong Doppler fading channels.

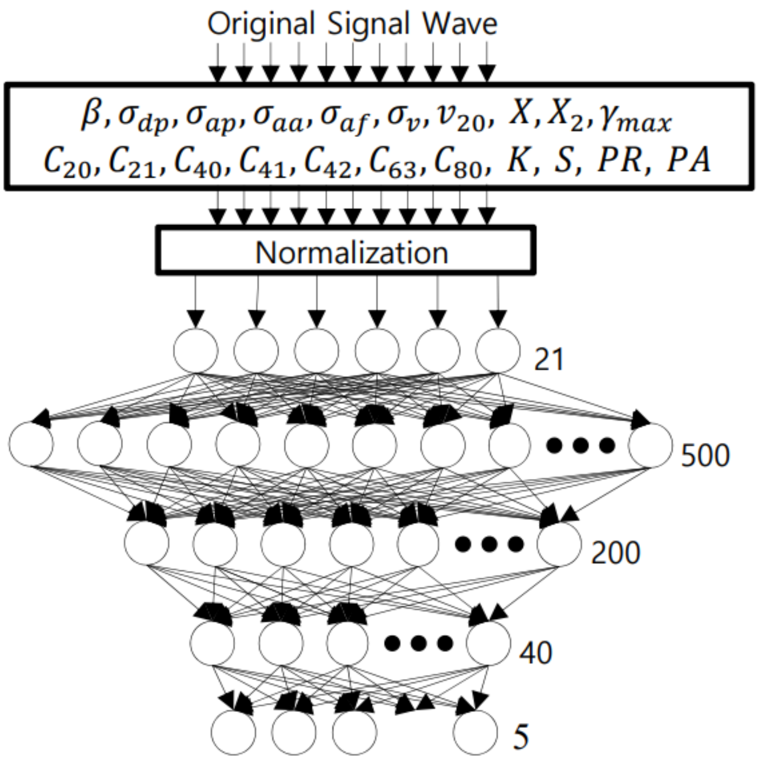

Original Signal Wave
$\beta, \sigma_{dp}, \sigma_{ap}, \sigma_{aa}, \sigma_{af}, \sigma_v, v_{20}, X, X_2, \gamma_{max}$
$C_{20}, C_{21}, C_{40}, C_{41}, C_{42}, C_{63}, C_{80}, K, S, PR, PA$
Normalization
21
500
200
40
5

Fig. 2.24 Structure of DNN-AMC proposed in [31]

Table 2.3 Configuration of the DNN-AMC proposed in [31]

Parameters	Value
Number of input nodes	21
Number of hidden layers	3
Number of nodes of 1st hidden layer	500
Number of nodes of 2nd hidden layer	200
Number of nodes of 3rd hidden layer	40
Number of output nodes	7
Activation function of hidden layer	ReLU
Activation function of output layer	Softmax
Max number of epochs	15000
Cost function	Negative-log-likelihood
Training method	SGD
Learning rate	< 0.01
Number of train data	15,000
Number of validation data	5,000
Number of test data	5,000
Batch size	50

- The authors of [32] have created and assessed a usable AMC system that can be quickly deployed to deliver reliable performance in a variety of real-time business settings. Therefore, their main objective is to create an AMC algorithm that is reliable and has a low computing complexity for practical deployment. To achieve this, they have made use of recently revived machine learning-based techniques that are employed for a variety of classification tasks. The authors of [32] offered certain statistics as features of the AMC signals in their proposed AMC architecture before designing an artificial neural network (ANN) based classifier that can conduct AMC over a wide range of SNRs. To enhance the classification performance of their ANN, they used the Nesterov accelerated adaptive moment (NADAM) estimate technique. They also put their proposed design into practice on an SDR (Software Defined Radio) testbed to determine whether it was practically feasible. The hybrid hierarchical AMC (HH-AMC) system is demonstrated to perform worse than the proposed ANN-based classifier, which is adaptable enough to readily expand the lexicon of modulation patterns for additional applications.

- Several AMC approaches based on convolutional neural network (CNN) prototype and variant have been suggested in order to improve classification performance. However, the majority of AMC techniques based on CNNs now in use only use monomodal data from either the time domain or frequency domain. The complementary processing gain that results from combining multimodal data from various transformation domains is disregarded. The authors in [34] use a waveform-spectrum multimodal fusion (WSMF) method to implement AMC based on deep residual networks (Resnet) in order to solve the problem. The authors have proposed a feature fusion technique to combine multimodal aspects of signals in order to generate more discriminating features after extracting features from multimodal information using Resnet. Simulation findings show that their suggested WSMF method outperforms more established CNN-based AMC methods that just use single modality information. The approach suggested in [34] is capable of differentiating between sixteen different modulation signals and performs admirably even with higher-order digital modulation types like 256QAM and 1024QAM.

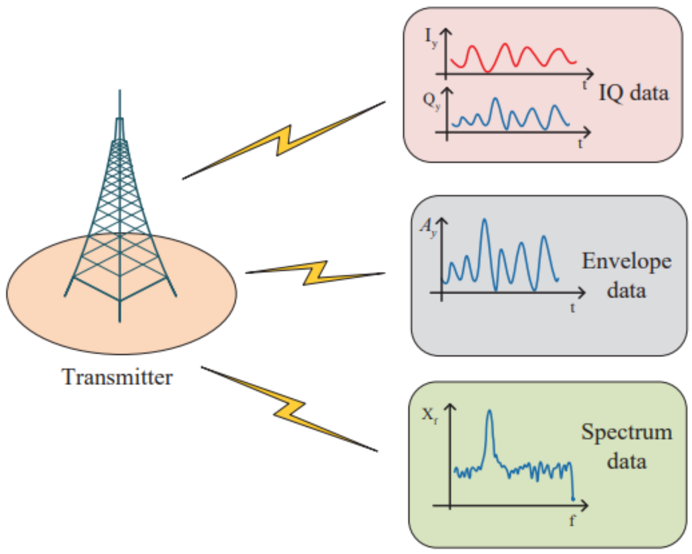


Fig. 2.25 Multimodal data of the received signal adopted in [34]

Several deep neural network models have been suggested in the literature. Mostly different variations of Convolutional Neural Network (CNN) with different optimizer and data preprocessing technique have been proposed. CNN based methods have significantly outperformed the traditional classification methods. That is why we have adopted the CNN based model in our research work. Along with CNN, ResNet and CLDNN models have proven to be very effective in various research works. Hence, we have also adopted these two models in this thesis work.

CHAPTER 3

Convolutional Neural Network Based Automatic Modulation Classification

3.1 Convolutional Neural Network

Computer Vision is a fascinating topic and it has many real world and highly useful applications such as self-driving cars, face recognition, object recognition and motion or movement detection etc. The foundation of computer vision is the ‘Convolutional Neural Network’ (CNN). In simple Neural Networks, the number of input features are generally very high for real world applications. Thus, the number of trainable parameters such as the weights and biases are extremely large in numbers. Normal CPUs and GPUs are not able to handle such large numbers of trainable parameters. Even if some processors with high computational capability handles such large number of tuneable parameters, the time required to train the model will be very high, and the model will tend to overfit the data. To overcome this limitation, deep learning community and researchers came up with an amazing concept called CNN, which is a special type of neural network. The main idea behind a CNN is that it uses ‘filters’. These filters are sliding windows in the one-dimensional vector of sample values of signals (these filters are ‘convolved’ with the vectors) that are responsible for detecting the features or patterns in the signal. Signals generally have these properties that it has amplitude, frequency, phase or combinedly it has different features. The filters of the CNN will be responsible for detecting these features associated with the signals. The number of parameters to train, is greatly reduced by using filters. Hence it also reduces the computation time. Convolution operation acts as a feature detector in CNN. That is why it is considered to be the heart of the CNN.

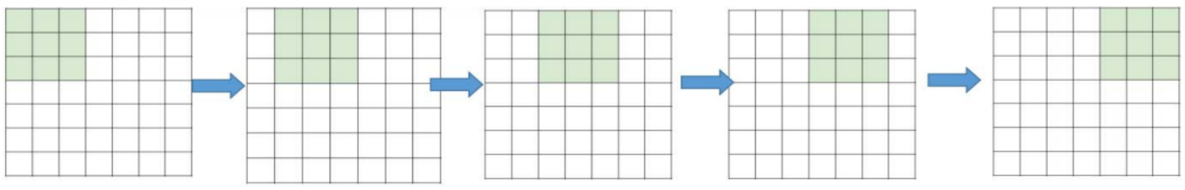
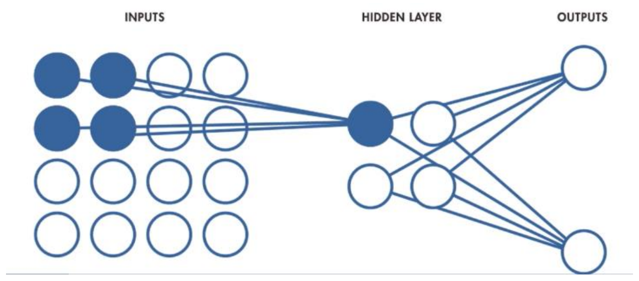
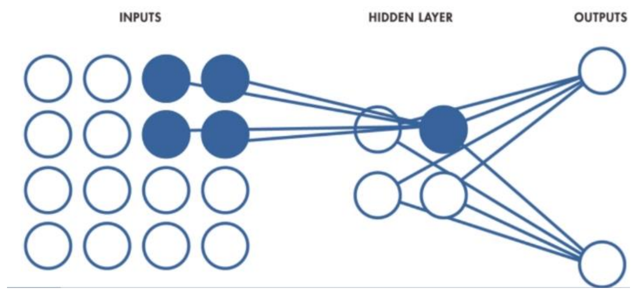


Fig. 3.1 Filter window moving only one pixel at a time (Stride 1)

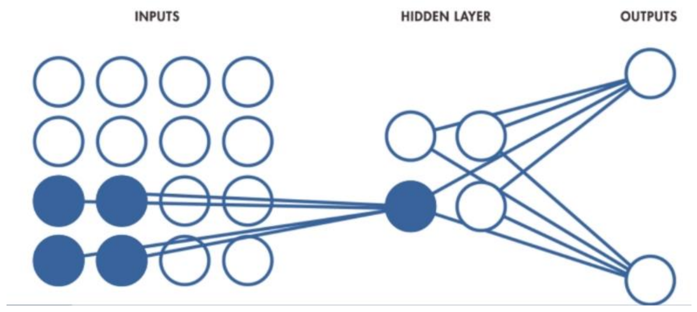
In a typical Neural Network, each neuron in the input layer is connected to a neuron in the hidden layer. However, in a CNN, only a small region in input layer neurons connect to neurons in the hidden layer. These regions are referred to as ‘Local Receptor Fields’. The Local Receptor Field is translated across an input vector to create a ‘Feature Map’ from the input layer to the hidden layer neurons as can be seen from figure 3.2. We use convolution to implement this process efficiently. That is why it is called a Convolutional Neural Network.



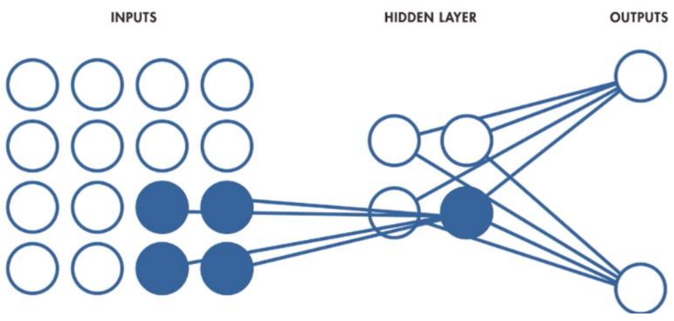
(a)



(b)



(c)



(d)

Fig. 3.2 (a) – (b) Translation of local receptor field from the input layer to the hidden layer neurons

When a 6x6 2D vector is convolved with 3x3 2D vector a 2D vector of size 4x4 is obtained. So, the size of the final image gets reduced by some amount. But if many such filters are used in a CNN, then it is possible that the final size of the image get reduced so much that some information are lost. Also, in case of convolution, the corner values do not get enough attention as compared to the values around the centre. To overcome these two issues, we pad the given 2D vector with a border of zeros (with one or more pixels). Now if we pad the original 6x6 with a single layer of zeros we obtain an 8x8 image. Now when it is convolved with 3x3 filter, an output vector of size 6x6 is obtained. This is known as ‘Padding’. Padding is of two types: ‘Valid’ and ‘Same’. The ‘Valid’ padding means that the convolution operation will be performed with no padding at all. In ‘Same’ padding, after performing the convolution operation the output vector will have the same size as the input vector. In our proposed model we have used ‘Same’ padding in each convolutional layer, whereas we have used ‘Valid’ padding in the maxpooling layers.

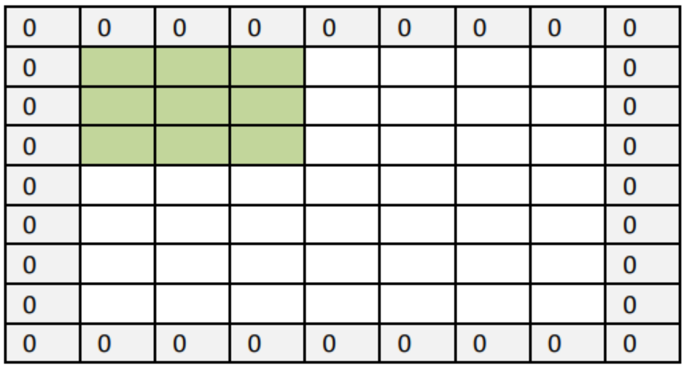


Fig. 3.3 Zero padding with a single border of zeros

After extracting the features from the input vector with sample values of the signals, they are passed on to the fully connected layer or dense layer. This dense layer is nothing but a regular artificial neural network. But before passing the features onto the fully connected layer, the input matrix is converted into a one-dimensional vector from a 2D vector. This process is known as ‘Flattening’ and it is shown in figure 3.4.

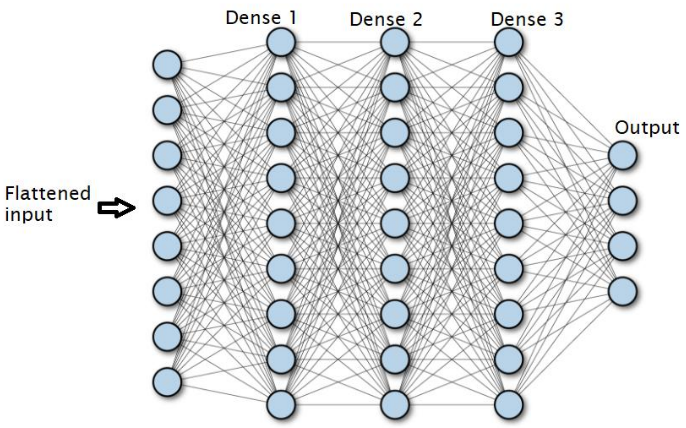


Fig. 3.4 Fully connected layer or dense layer

The number of neurons in the output layer is same as the number of classes in our given classification problem. We use Softmax activation function in the last dense layer.

3.2 CNN based Automatic Modulation Classification with 3 convolution layers

3.2.1 Model Specifications

The model specifications are shown in Table 3.1. We have taken the dropout rate to be 0.3.

Table 3.1 Model specifications of CNN based AMC with 3 convolution layers

Layer (type)	**Output Shape**	**Parameter**
conv2d_8 (Conv2D)	(None, 2, 128, 256)	2560
max_pooling2d_8 (Maxpooling 2D)	(None, 2, 64, 256)	0
dropout_8 (Dropout)	(None, 2, 64, 256)	0
Conv2d_9 (Conv2D)	(None, 2, 64, 128)	295040
max_pooling2d_9 (Maxpooling2D)	(None, 2, 32, 128)	0
dropout_9 (Dropout)	(None, 2, 32, 128)	0
Conv2d_10 (Conv2D)	(None, 2, 32, 64)	73792
max_pooling2d_10 (Maxpooling2D)	(None, 2, 16, 64)	0
dropout_10 (Dropout)	(None, 2, 16, 64)	0
flatten_2 (Flatten)	(None, 2048)	0
dense_4 (Dense)	(None, 128)	262272
dense_5 (Dense)	(None, 11)	1419

Total number of parameters are 635,083. Total number of trainable parameters are 635,083. The number of non-trainable parameters are 0.

3.2.2 Pseudocode

The code for the model function or definition is as shown:

```
def CNN_3_conv_layer():

    model = Sequential()

    model.add(Conv2D(256, (3, 3), activation='relu', padding='same', input_shape=(2,128,1)))
    model.add(MaxPooling2D(pool_size=(1, 2), padding='valid',  data_format=None))
    model.add(layers.Dropout(.3))

    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(1, 2), padding='valid', data_format=None))
    model.add(layers.Dropout(.3))

    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(1, 2), padding='valid', data_format=None))
    model.add(layers.Dropout(.3))

    model.add(Flatten())

    model.add(Dense(128, activation='relu'))
    model.add(Dense(11, activation='softmax'))

    model.summary()

    return model
```

3.2.3 Results and Analysis

We have run our model for two different epoch values: 1 and 10. The performance of the model in 1 epoch value is poor for obvious reasons. We have considered the model performance with epoch value 10 as it gives better results.

Epoch = 1 :

The model performance for 1 epoch is as shown in figure 3.5 to 3.8. The training duration is 30 minutes in this case.

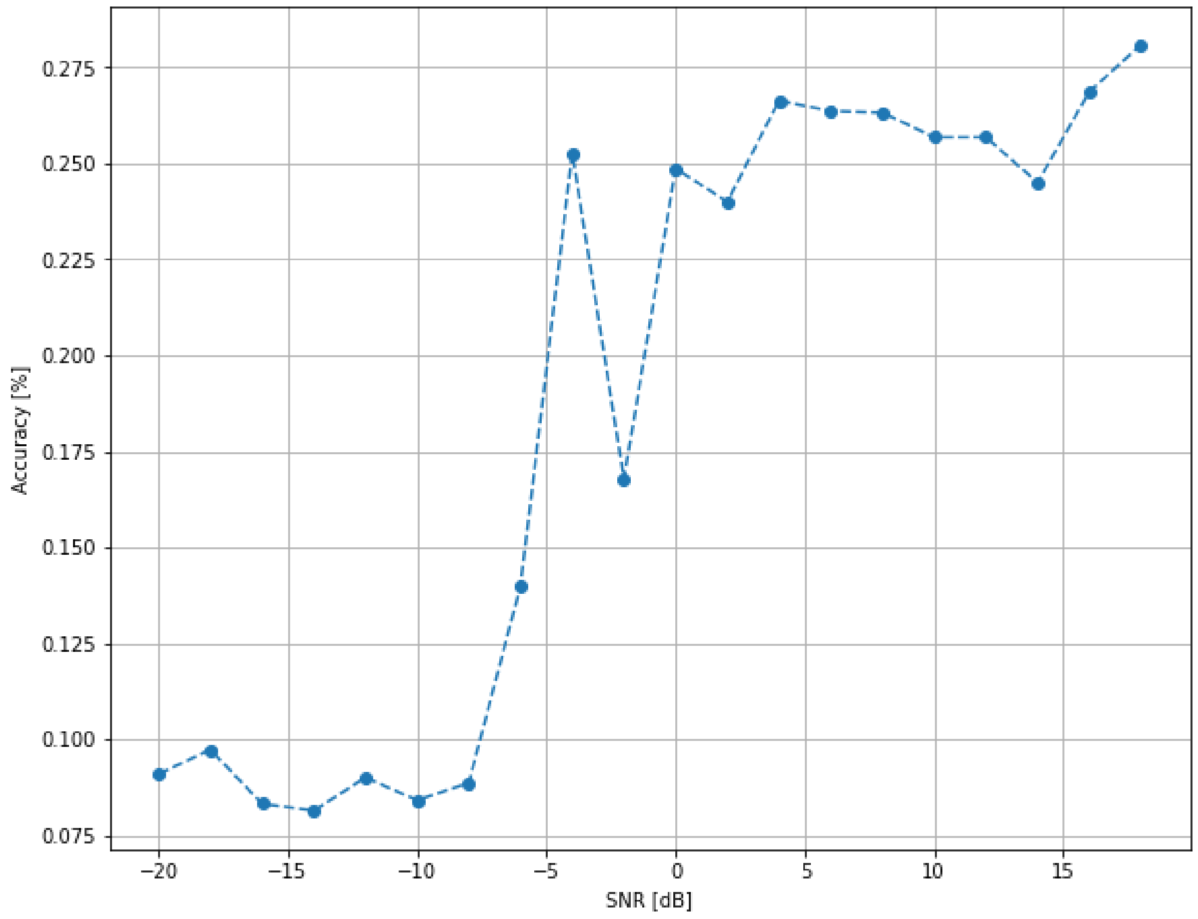


Fig. 3.5 Accuracy vs. SNR plot of CNN based AMC with 3 convolution layers in 1 epoch

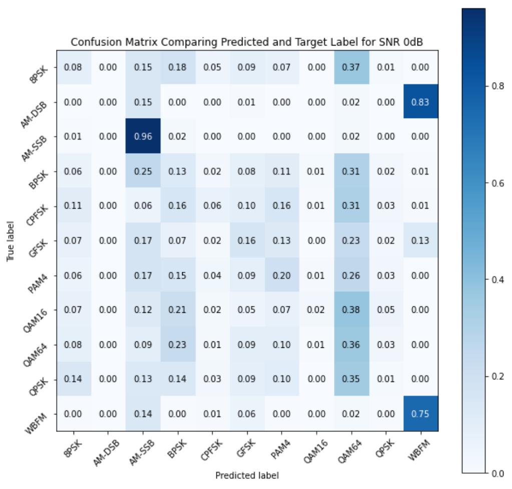


Fig. 3.6 Confusion matrix of CNN based AMC with 3 convolution layers for 0 dB SNR in 1 epoch

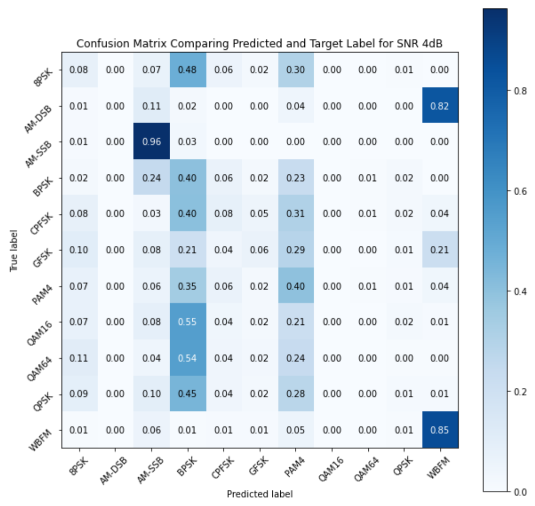


Fig. 3.7 Confusion matrix of CNN based AMC with 3 convolution layers for 4 dB SNR in 1 epoch

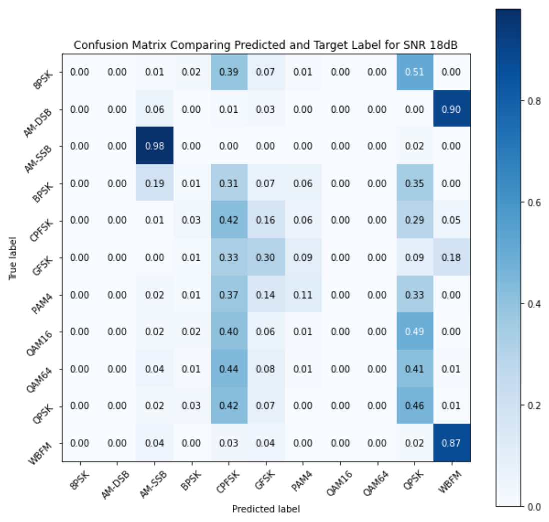


Fig. 3.8 Confusion matrix of CNN based AMC with 3 convolution layers for 18 dB SNR in 1 epoch

Epoch = 10 :

The model performance for 10 epoch is as shown in figure 3.9 and 3.11 to 3.22. The model shows best results in SNR 0 dB with an accuracy of 61.5%. The confusion matrix corresponding to 0 dB SNR, shown in figure 3.16, highlights a distinct diagonal. The training duration is 4 hours.

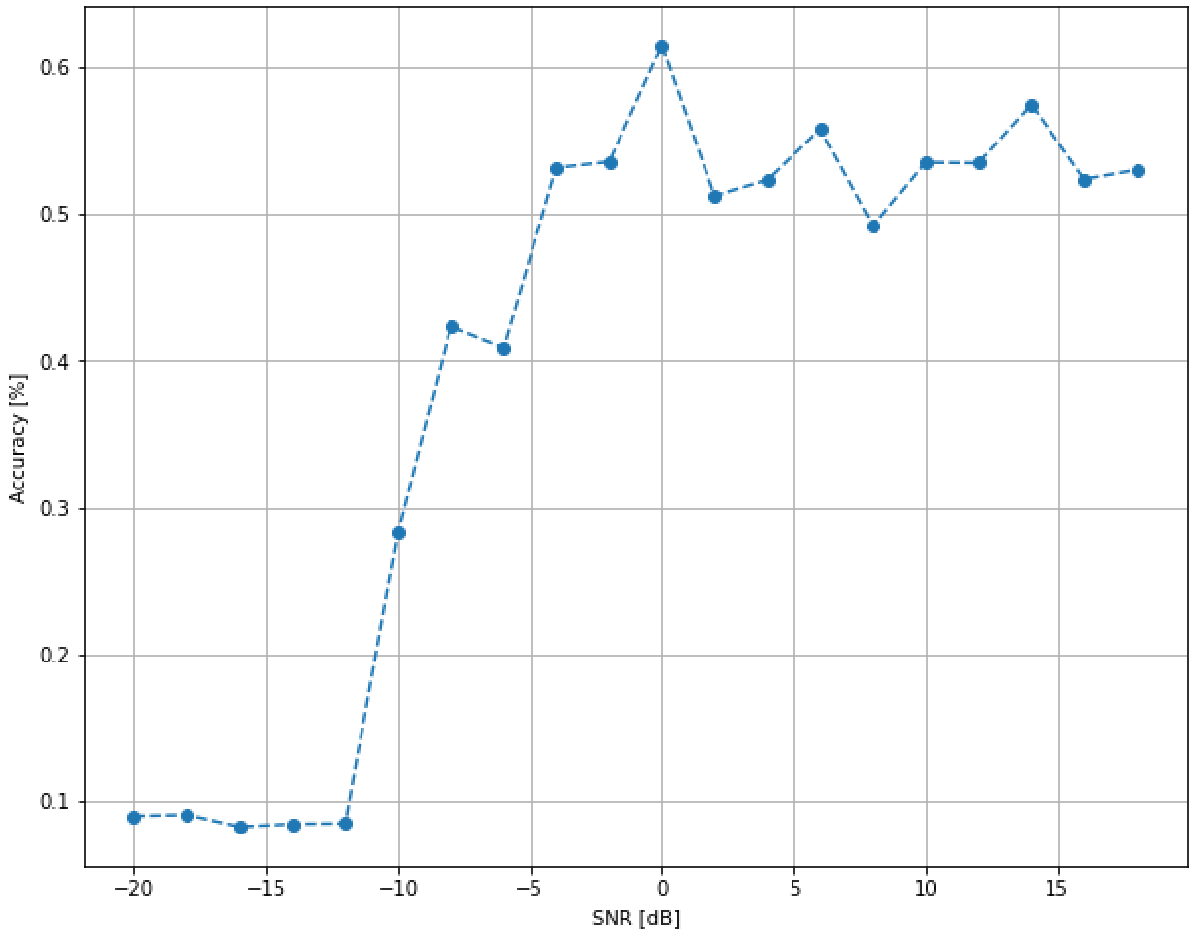


Fig. 3.9 Accuracy vs. SNR plot of CNN based AMC with 3 convolution layers in 10 epochs

The loss v/s epoch plot is shown in figure 3.11. It is to be noted that the loss value is greater than 1. It is so because we have used 'Categorical Cross Entropy' loss function with logit value being true, i.e., the loss can take any value between -∞ to +∞ (output of logit function). The Keras documentation can be found in [35]. The logit function is shown in figure 3.10.

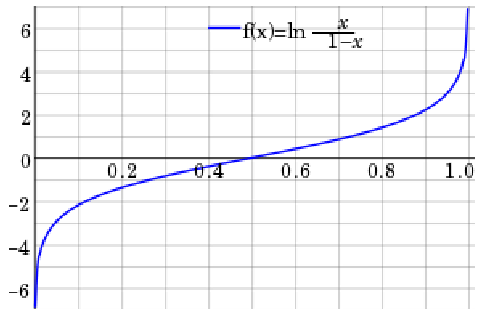


Fig. 3.10 Plot of logit(x) = f(x) in the domain of 0 to 1, where the base of the logarithm is e

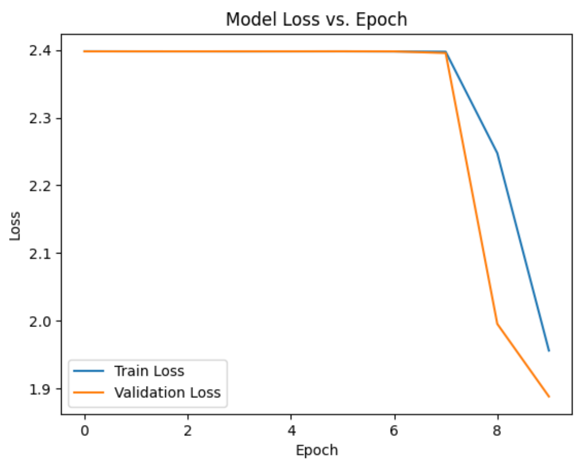


Fig. 3.11 Loss vs. epoch plot of CNN based AMC with 3 convolution layers for -10 dB SNR in 10 epochs

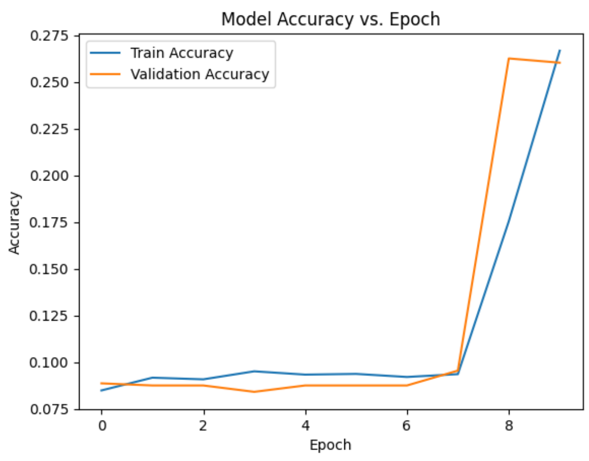
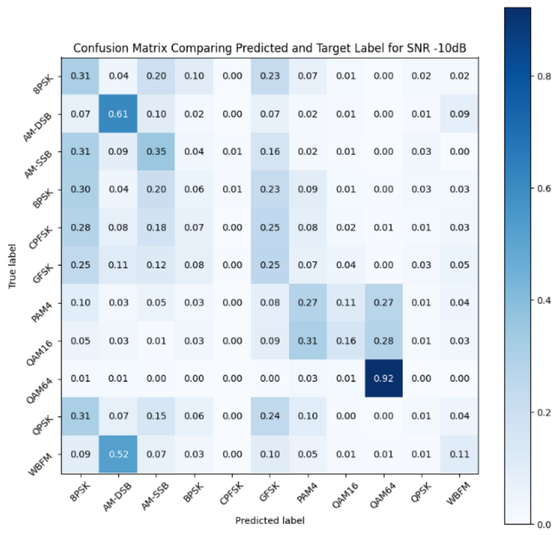


Fig. 3.12 Accuracy vs. epoch plot of CNN based AMC with 3 convolution layers for -10 dB SNR in 10 epochs



	8PSK	AM-DSB	AM-SSB	BPSK	CPFSK	GFSK	PAM4	QAM16	QAM64	QPSK	WBFM
8PSK	0.31	0.04	0.20	0.10	0.00	0.23	0.07	0.01	0.00	0.02	0.02
AM-DSB	0.07	0.61	0.10	0.02	0.00	0.07	0.02	0.01	0.00	0.01	0.09
AM-SSB	0.31	0.09	0.35	0.04	0.01	0.16	0.02	0.01	0.00	0.03	0.00
BPSK	0.30	0.04	0.20	0.06	0.01	0.23	0.09	0.01	0.00	0.03	0.03
CPFSK	0.28	0.08	0.18	0.07	0.00	0.25	0.08	0.02	0.01	0.01	0.03
GFSK	0.25	0.11	0.12	0.08	0.00	0.25	0.07	0.04	0.00	0.03	0.05
PAM4	0.10	0.03	0.05	0.03	0.00	0.08	0.27	0.11	0.27	0.01	0.04
QAM16	0.05	0.03	0.01	0.03	0.00	0.09	0.31	0.16	0.28	0.01	0.03
QAM64	0.01	0.01	0.00	0.00	0.00	0.00	0.03	0.01	0.92	0.00	0.00
QPSK	0.31	0.07	0.15	0.06	0.00	0.24	0.10	0.00	0.00	0.01	0.04
WBFM	0.09	0.52	0.07	0.03	0.00	0.10	0.05	0.01	0.01	0.01	0.11

Fig. 3.13 Confusion matrix of CNN based AMC with 3 convolution layers for -10 dB SNR in 10 epochs

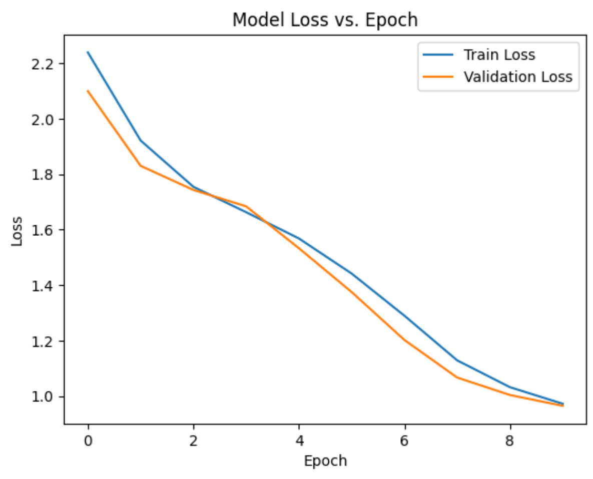


Fig. 3.14 Loss vs. epoch plot of CNN based AMC with 3 convolution layers for 0 dB SNR in 10 epochs

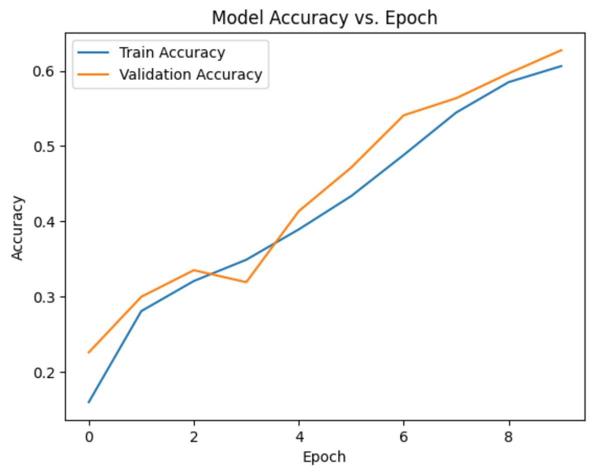


Fig. 3.15 Accuracy vs. epoch plot of CNN based AMC with 3 convolution layers for 0 dB SNR in 10 epochs

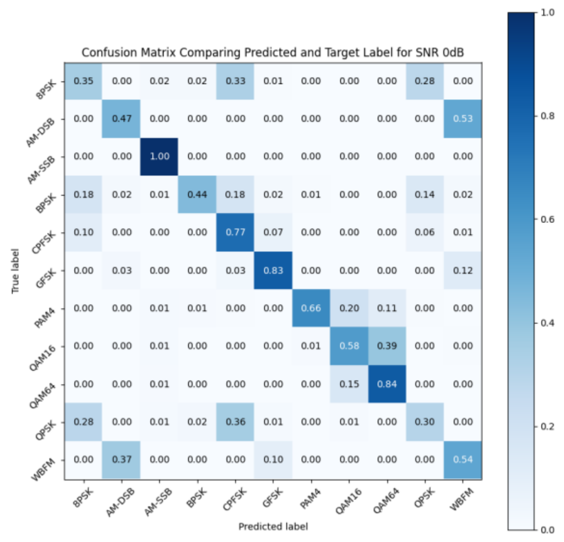


Fig. 3.16 Confusion matrix of CNN based AMC with 3 convolution layers for 0 dB SNR in 10 epochs

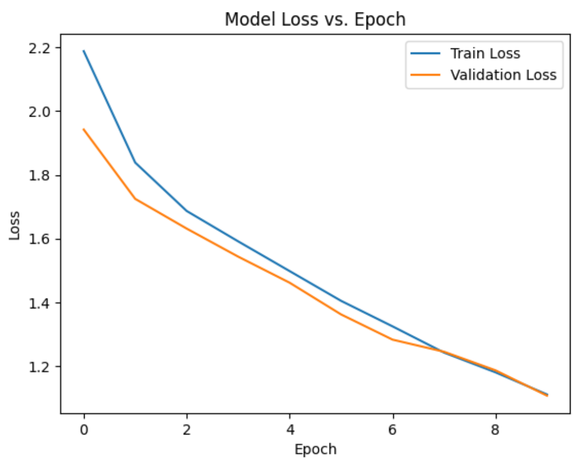


Fig. 3.17 Loss vs. epoch plot of CNN based AMC with 3 convolution layers for 6 dB SNR in 10 epochs

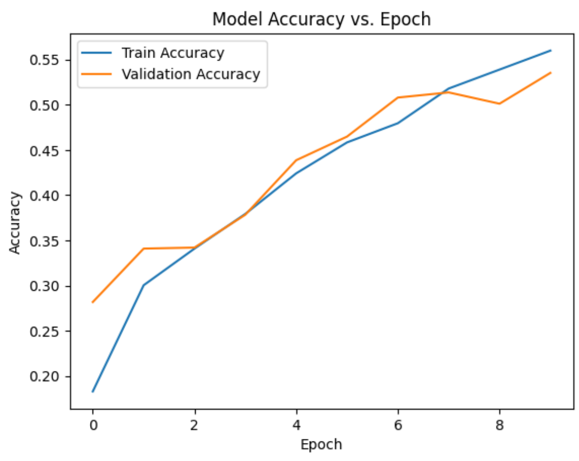


Fig. 3.18 Accuracy vs. epoch plot of CNN based AMC with 3 convolution layers for 6 dB SNR in 10 epochs

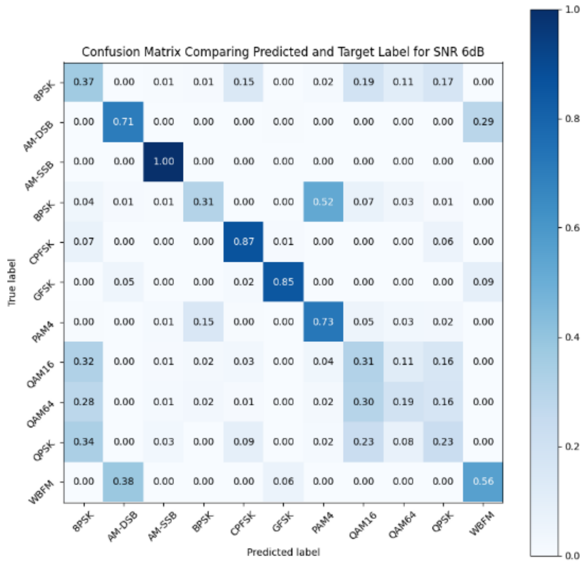


Fig. 3.19 Confusion matrix of CNN based AMC with 3 convolution layers for 6 dB SNR in 10 epochs

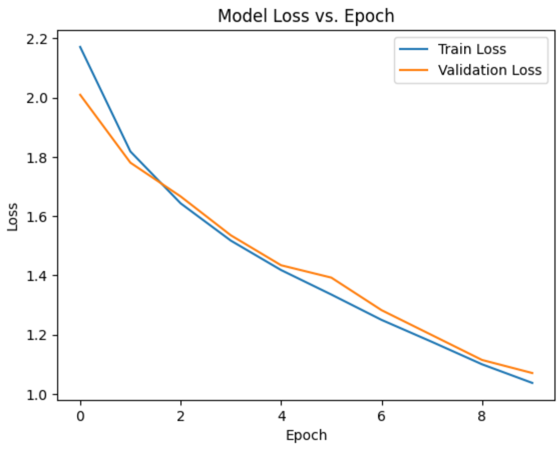


Fig. 3.20 Loss vs. epoch plot of CNN based AMC with 3 convolution layers for 14 dB SNR in 10 epochs

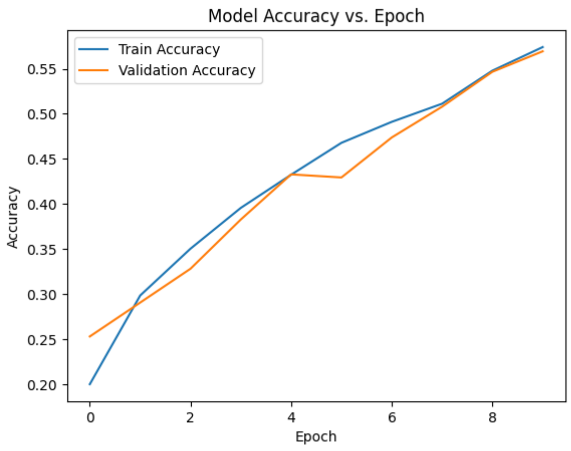
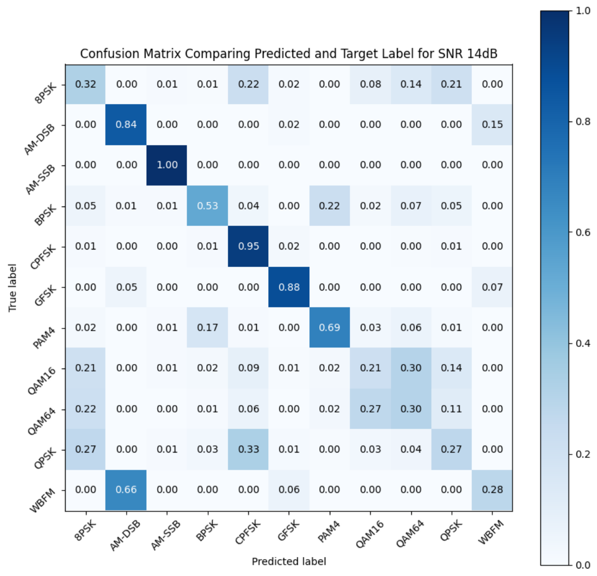


Fig. 3.21 Accuracy vs. epoch plot of CNN based AMC with 3 convolution layers for 14 dB SNR in 10 epochs



	8PSK	AM-DSB	AM-SSB	BPSK	CPFSK	GFSK	PAM4	QAM16	QAM64	QPSK	WBFM
8PSK	0.32	0.00	0.01	0.01	0.22	0.02	0.00	0.08	0.14	0.21	0.00
AM-DSB	0.00	0.84	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.15
AM-SSB	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
BPSK	0.05	0.01	0.01	0.53	0.04	0.00	0.22	0.02	0.07	0.05	0.00
CPFSK	0.01	0.00	0.00	0.01	0.95	0.02	0.00	0.00	0.00	0.01	0.00
GFSK	0.00	0.05	0.00	0.00	0.00	0.88	0.00	0.00	0.00	0.00	0.07
PAM4	0.02	0.00	0.01	0.17	0.01	0.00	0.69	0.03	0.06	0.01	0.00
QAM16	0.21	0.00	0.01	0.02	0.09	0.01	0.02	0.21	0.30	0.14	0.00
QAM64	0.22	0.00	0.00	0.01	0.06	0.00	0.02	0.27	0.30	0.11	0.00
QPSK	0.27	0.00	0.01	0.03	0.33	0.01	0.00	0.03	0.04	0.27	0.00
WBFM	0.00	0.66	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.28

Fig. 3.22 Confusion matrix of CNN based AMC with 3 convolution layers for 14 dB SNR 10 epochs

3.3 CNN based Automatic Modulation Classification with 4 convolution layers

3.3.1 Model Specifications

The model specifications are shown in table 3.2. This model is inspired by the work done in [24]. Nothing has been mentioned in the paper; hence we have taken dropout rate to be 0.3.

Table 3.2 Model specifications of CNN based AMC with 4 convolution layers

Layer (type)	**Output Shape**	**Parameter**
conv2d_4 (Conv2D)	(None, 2, 128, 256)	2560
max_pooling2d_4 (Maxpooling2D)	(None, 2, 64, 256)	0
dropout_4 (Dropout)	(None, 2, 64, 256)	0
Conv2d_5 (Conv2D)	(None, 2, 64, 128)	295040
max_pooling2d_5 (Maxpooling2D)	(None, 2, 32, 128)	0
dropout_5 (Dropout)	(None, 2, 32, 128)	0
Conv2d_6 (Conv2D)	(None, 2, 32, 64)	73792
max_pooling2d_6 (Maxpooling2D)	(None, 2, 16, 64)	0
dropout_6 (Dropout)	(None, 2, 16, 64)	0
Conv2d_7 (Conv2D)	(None, 2, 16, 64)	36928
max_pooling2d_7 (Maxpooling2D)	(None, 2, 8, 64)	0
dropout_7 (Dropout)	(None, 2, 8, 64)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_2 (Dense)	(None, 128)	131200
dense_3 (Dense)	(None, 11)	1419

Total number of parameters are 540,939. Total number of trainable parameters are 540,939. The number of non-trainable parameters are 0.

3.3.2 Pseudocode

The code for the model function or definition is as shown:

```
def CNN_4_conv_layer():

   model = Sequential()

   model.add(Conv2D(256, (3, 3), activation='relu', padding='same', input_shape=(2,128,1)))
   model.add(MaxPooling2D(pool_size=(1, 2), padding='valid',  data_format=None))
   model.add(layers.Dropout(.3))

   model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
   model.add(MaxPooling2D(pool_size=(1, 2), padding='valid', data_format=None))
   model.add(layers.Dropout(.3))

   model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
   model.add(MaxPooling2D(pool_size=(1, 2), padding='valid', data_format=None))
   model.add(layers.Dropout(.3))

   model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
   model.add(MaxPooling2D(pool_size=(1, 2), padding='valid', data_format=None))
   model.add(layers.Dropout(.3))

   model.add(Flatten())

   model.add(Dense(128, activation='relu'))
   model.add(Dense(11, activation='softmax'))

   model.summary()

   return model
```

3.3.3 Results and Analysis

We have run our model in two different epoch values: 1 and 10. The performance of the model in 1 epoch value is poor for obvious reasons. We have considered the model performance with epoch value 10 as it gives better results.

Epoch = 1 :

The model performance for 1 epoch is as shown in figure 3.23 to 3.26. The training duration is 30 minutes in this case also.

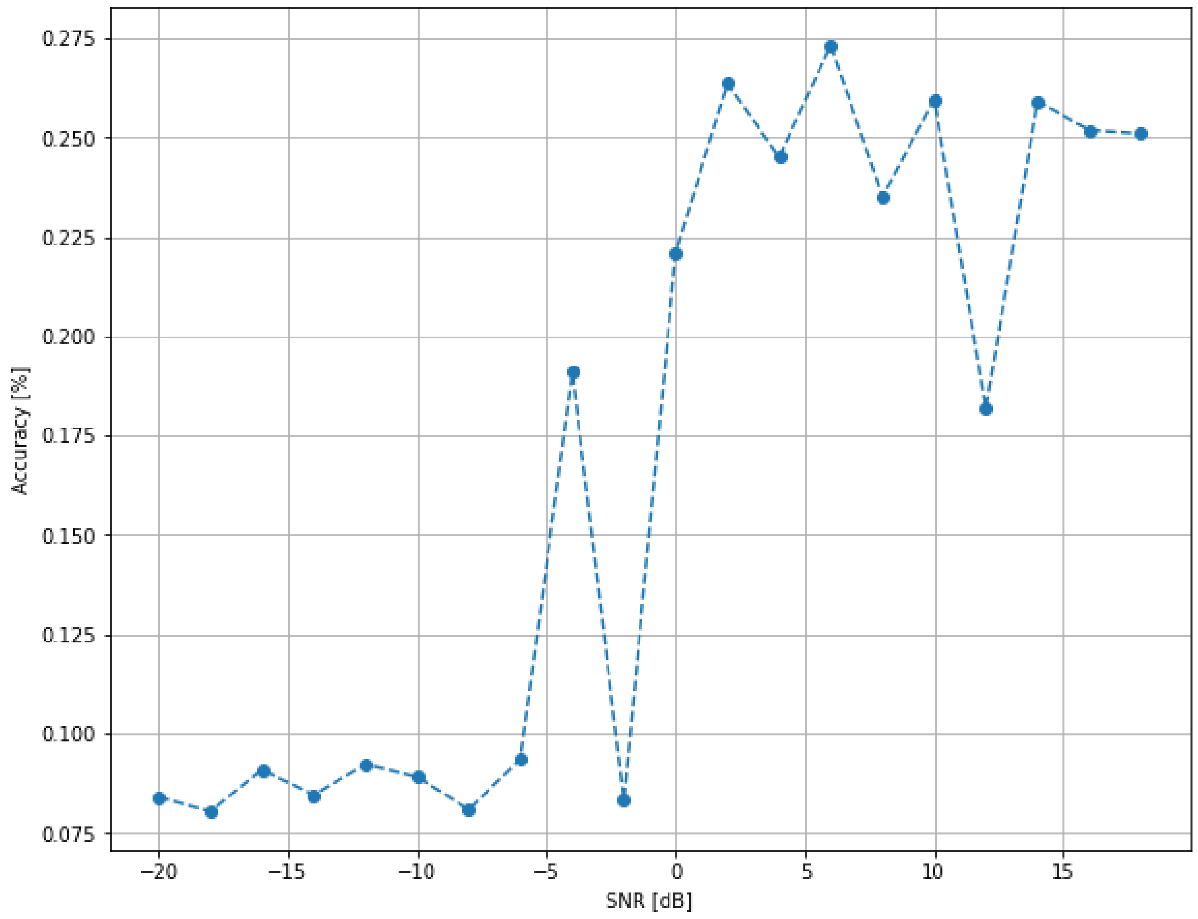


Fig. 3.23 Accuracy vs. SNR plot of CNN based AMC with 4 convolution layers in 1 epoch

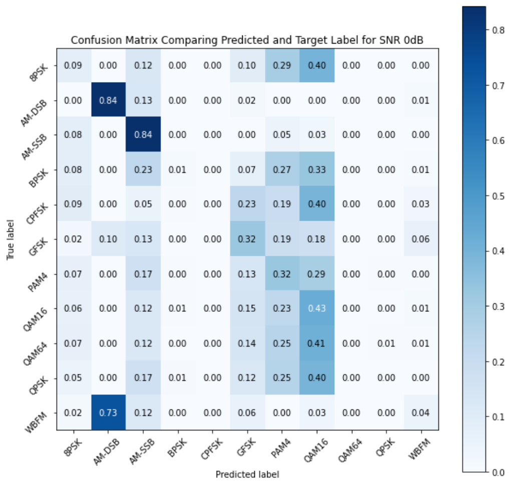


Fig. 3.24 Confusion matrix of CNN based AMC with 4 convolution layers for 0 dB SNR in 1 epoch

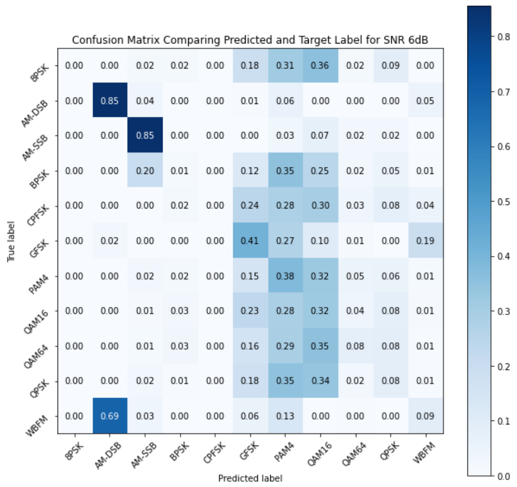


Fig. 3.25 Confusion matrix of CNN based AMC with 4 convolution layers for 6 dB SNR in 1 epoch

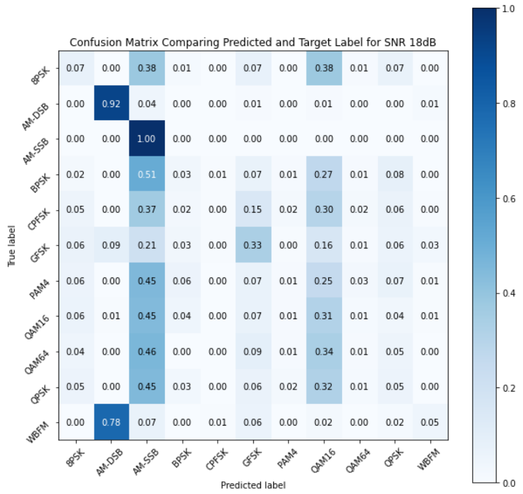


Fig. 3.26 Confusion matrix of CNN based AMC with 4 convolution layers for 18 dB SNR in 1 epoch

Epoch = 10 :

The model performance for 10 epoch is as shown in figure 3.27 to 3.39. The model shows best results in SNR 0 dB with an accuracy of 62.73%. The confusion matrix corresponding to 0 dB SNR, shown in figure 3.33, highlights a distinct diagonal. The training duration is 3 hours 30 minutes.

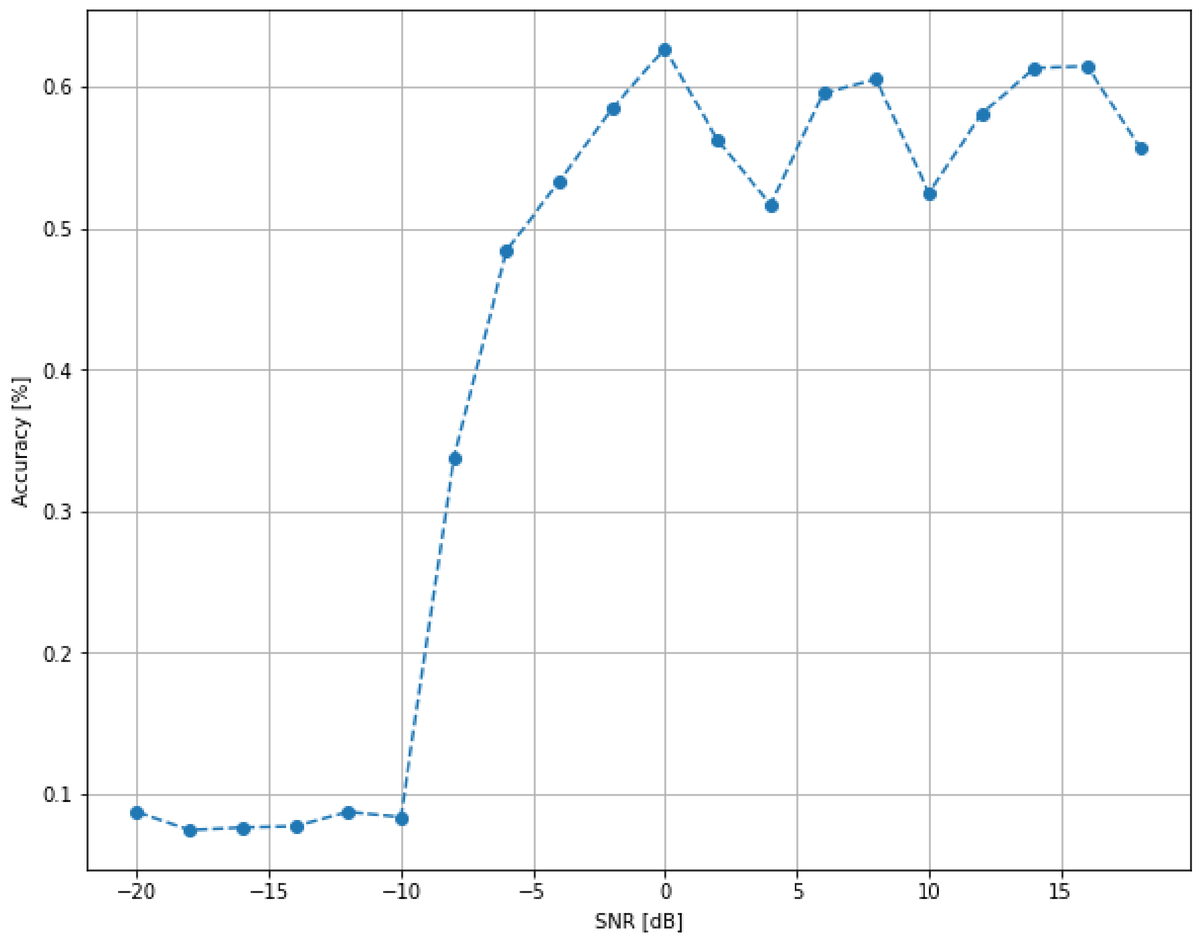


Fig. 3.27 Accuracy vs. SNR plot for CNN based AMC with 4 convolution layers in 10 epochs

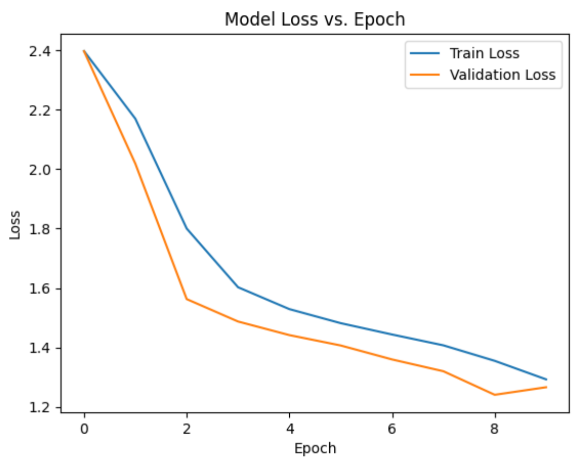


Fig. 3.28 Loss vs. epoch plot of CNN based AMC with 4 convolution layers for -6 dB SNR in 10 epochs

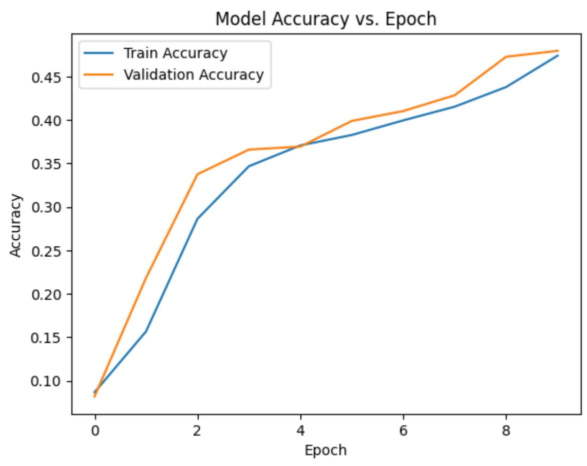
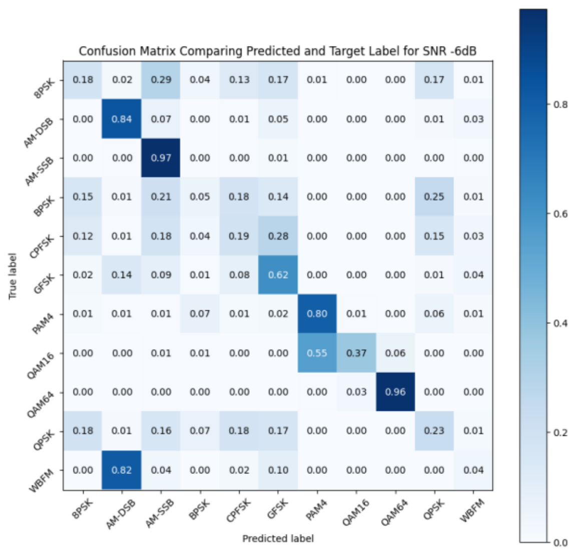


Fig. 3.29 Accuracy vs. epoch plot of CNN based AMC with 4 convolution layers for -6 dB SNR in 10 epochs



True \ Predicted	8PSK	AM-DSB	AM-SSB	BPSK	CPFSK	GFSK	PAM4	QAM16	QAM64	QPSK	WBFM
8PSK	0.18	0.02	0.29	0.04	0.13	0.17	0.01	0.00	0.00	0.17	0.01
AM-DSB	0.00	0.84	0.07	0.00	0.01	0.05	0.00	0.00	0.00	0.01	0.03
AM-SSB	0.00	0.00	0.97	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00
BPSK	0.15	0.01	0.21	0.05	0.18	0.14	0.00	0.00	0.00	0.25	0.01
CPFSK	0.12	0.01	0.18	0.04	0.19	0.28	0.00	0.00	0.00	0.15	0.03
GFSK	0.02	0.14	0.09	0.01	0.08	0.62	0.00	0.00	0.00	0.01	0.04
PAM4	0.01	0.01	0.01	0.07	0.01	0.02	0.80	0.01	0.00	0.06	0.01
QAM16	0.00	0.00	0.01	0.01	0.00	0.00	0.55	0.37	0.06	0.00	0.00
QAM64	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.96	0.00	0.00
QPSK	0.18	0.01	0.16	0.07	0.18	0.17	0.00	0.00	0.00	0.23	0.01
WBFM	0.00	0.82	0.04	0.00	0.02	0.10	0.00	0.00	0.00	0.00	0.04

Fig. 3.30 Confusion matrix of CNN based AMC with 4 convolution layers for -6 dB SNR in 10 epochs

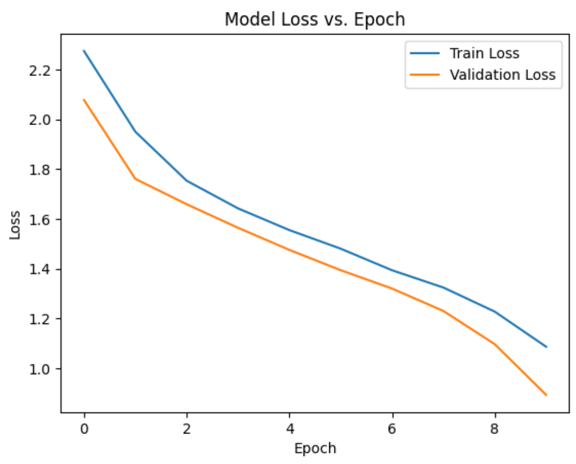


Fig. 3.31 Loss vs. epoch plot of CNN based AMC with 4 convolution layers for 0 dB SNR in 10 epochs

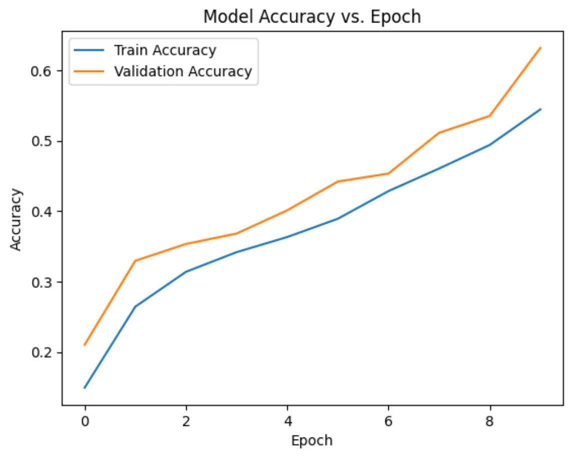
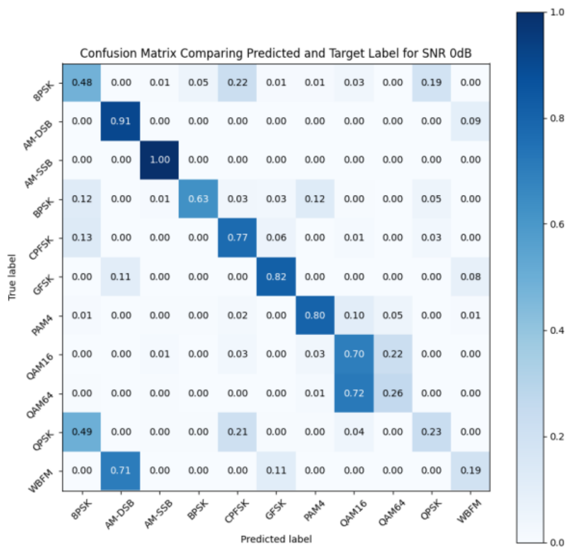


Fig. 3.32 Accuracy vs. epoch plot of CNN based AMC with 4 convolution layers for 0 dB SNR in 10 epochs



True label \ Predicted label	8PSK	AM-DSB	AM-SSB	BPSK	CPFSK	GFSK	PAM4	QAM16	QAM64	QPSK	WBFM
8PSK	0.48	0.00	0.01	0.05	0.22	0.01	0.01	0.03	0.00	0.19	0.00
AM-DSB	0.00	0.91	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09
AM-SSB	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
BPSK	0.12	0.00	0.01	0.63	0.03	0.03	0.12	0.00	0.00	0.05	0.00
CPFSK	0.13	0.00	0.00	0.00	0.77	0.06	0.00	0.01	0.00	0.03	0.00
GFSK	0.00	0.11	0.00	0.00	0.00	0.82	0.00	0.00	0.00	0.00	0.08
PAM4	0.01	0.00	0.00	0.00	0.02	0.00	0.80	0.10	0.05	0.00	0.01
QAM16	0.00	0.00	0.01	0.00	0.03	0.00	0.03	0.70	0.22	0.00	0.00
QAM64	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.72	0.26	0.00	0.00
QPSK	0.49	0.00	0.00	0.00	0.21	0.00	0.00	0.04	0.00	0.23	0.00
WBFM	0.00	0.71	0.00	0.00	0.00	0.11	0.00	0.00	0.00	0.00	0.19

Fig. 3.33 Confusion matrix of CNN based AMC with 4 convolution layers for 0 dB SNR in 10 epochs

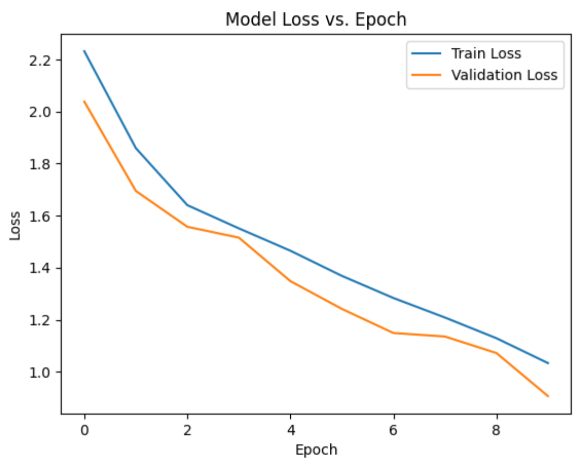


Fig. 3.34 Loss vs. epoch plot of CNN based AMC with 4 convolution layers for 8 dB SNR in 10 epochs

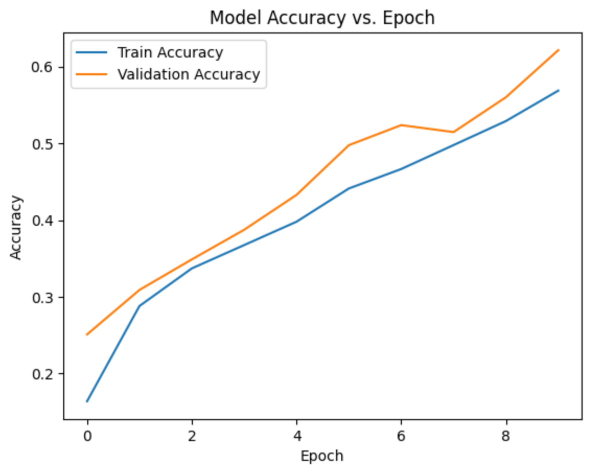


Fig. 3.35 Accuracy vs. epoch plot of CNN based AMC with 4 convolution layers for 8 dB SNR in 10 epochs

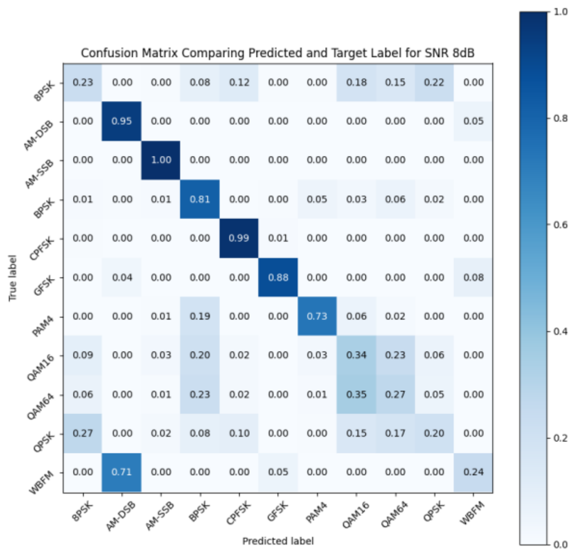


Fig. 3.36 Confusion matrix of CNN based AMC with 4 convolution layers for 8 dB SNR in 10 epochs

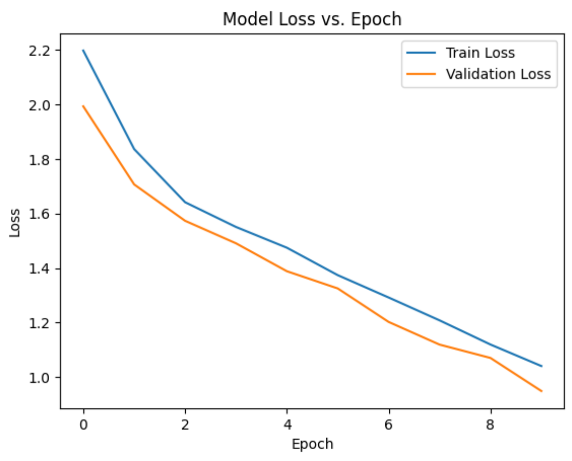


Fig. 3.37 Loss vs. epoch plot of CNN based AMC with 4 convolution layers for 16 dB SNR in 10 epochs

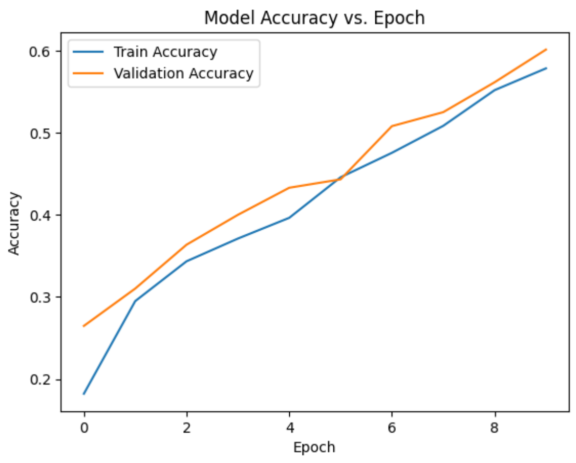


Fig. 3.38 Accuracy vs. epoch plot of CNN based AMC with 4 convolution layers for 16 dB SNR in 10 epochs

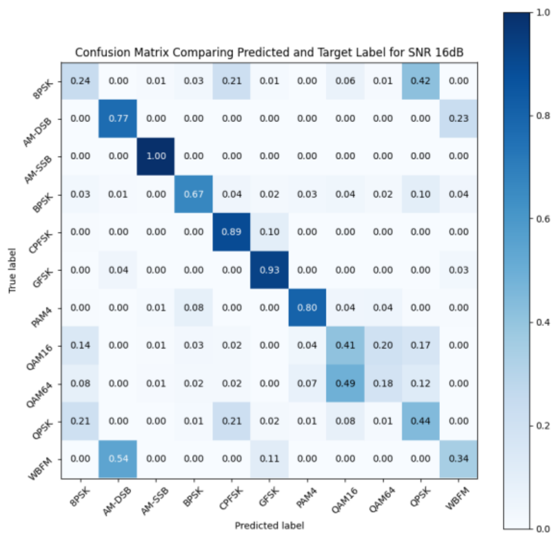


Fig. 3.39 Confusion matrix of CNN based AMC with 4 convolution layers for 16 dB SNR in 10 epochs

3.3.4 Comparison between 3 & 4 convolution layer CNN based AMC

The graph shown in figure 3.40 compares the performance of 3 and 4 convolution layer CNN based AMC. The model with 4 convolution layers shows 1.995% better performance than the model with 3 convolution layers.

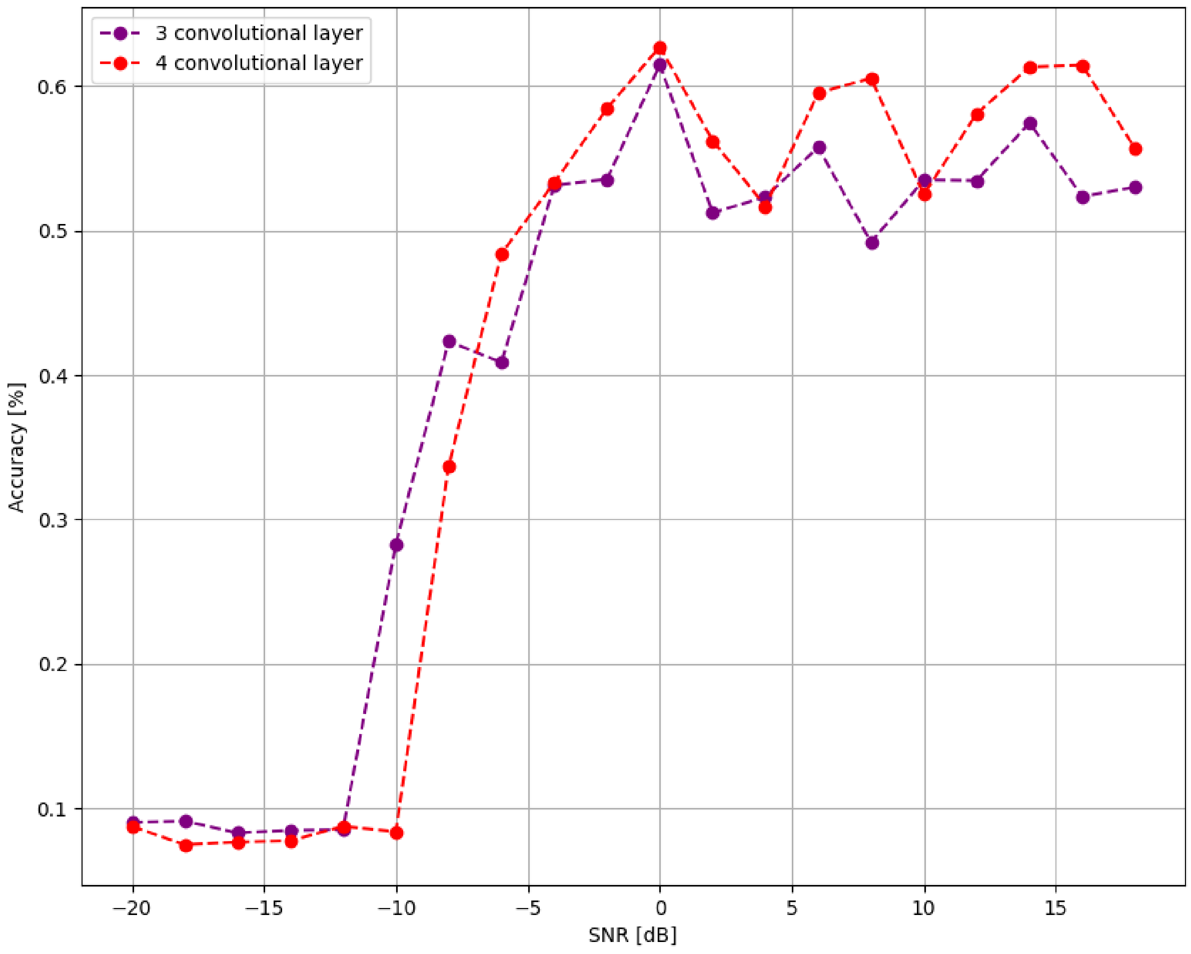


Fig. 3.40 Accuracy vs. SNR plot comparing 3 & 4 convolution layered CNN based AMC in 10 epochs

3.4 Modification in CNN based Automatic Modulation Classification

3.4.1 Batch Normalization

3.4.1.1 Overview

One of the most used techniques for accelerating network training and improving accuracy is batch normalization, often known as batch Norm or BN. It was initially presented in [36]. By minimizing the internal covariate shift in the data, it solves the problems encountered during the training of the deep learning models. Between training and inference, the batch normalization layer behaves differently. Each batch's mean and variance are determined separately during training, and during inference, the moving mean and variance of all the images, the network has seen, are employed.

3.4.1.2 Normalization

Data standardization is accomplished using the pre-processing method of normalization. Alternately, having many data sources within the same range and failing to normalize the data prior to training might lead to issues with deep neural networks, making it extremely difficult to train and slowing down its training rate**.** There are two main methods to normalize the data. The most straightforward method is to scale the data to a range from 0 to 1:

$$x_{normalized} = \frac{x - m}{x_{max} - x_{min}} \tag{3.1}$$

Where x is the data point to normalize, m is the mean of the data set, x_{max} is the highest value and x_{min} is the lowest value. This method is generally used in the inputs of the data. The data points which are not normalized with wide ranges, can cause instability in Neural Networks. Comparatively large inputs can cascade down to the layers, causing problems such as exploding gradients.

The other technique which is used to normalize data is to force the data points to have a mean of 0 and a standard deviation of 1, using the following formula:

$$x_{normalized} = \frac{x - m}{s} \tag{3.2}$$

Where, m is the mean of the data set, s is the standard deviation of the data set and x is the data point to be normalized. Now, each data point follows a standard normal distribution. Having all the features on this scale, none of them will have a bias, and therefore, the models will learn better.

In Batch Normalization, this last technique is used to normalize batches of data inside the network itself.

3.4.1.3 Working of Batch Normalization

Instead of applying it on the raw data, batch normalization uses normalizing between the layers of a neural network. Instead of using the entire data set, it is done in mini-batches. It uses faster learning rates and speeds up training, which facilitates learning**.** The batch normalization formula is as shown :

$$z^N = \left(\frac{z - m_z}{s_z}\right) \tag{3.3}$$

Where m_z is the mean of the output of the neuron and s_z is the standard deviation of the output of the neuron.

In figure 3.41, a regular feed-forward neural network with two hidden layers and batch normalization layer is shown. Here x_i is the input, z_i is the output of the neurons, a_i is the output of the activation functions, and y is the output of the network:

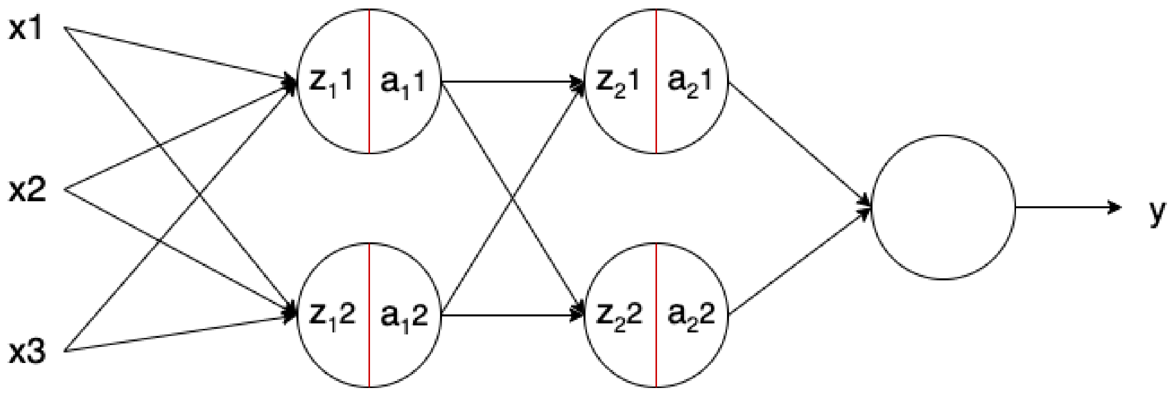


Fig. 3.41 Simple neural network with batch normalization layer shown in red line

Batch normalization (represented with a red line) is applied to the output of the neuron just before applying the activation function. Usually, a neuron without batch normalization is computed as follows:

$$z = g(w, x) + b; \qquad a = f(z) \tag{3.4}$$

g() is the linear transformation of the neuron, w is the weights of the neuron, f() is the activation function and b is the bias. The model learns the parameters w and b. Adding batch normalization, it looks as:

$$z = g(w, x); \qquad z^N = \left(\frac{z - m_z}{s_z}\right) \cdot \gamma + \beta; \qquad a = f(z^N) \tag{3.5}$$

Z^N is the output of batch normalization, m_z is the mean of the output of the neuron, s_z is the standard deviation of the output of the neurons, γ and β are the learning parameters of batch normalization.

The parameter, β shifts the mean and the parameter, γ shifts the standard deviation. Thus, the outputs of Batch Normalization over a layer result in a distribution with a mean β and a standard deviation of γ. These values are learned over epochs.

Deep neural network training is challenging because when the parameters of the earlier layers change, so do the input distributions of each layer. This makes it extremely difficult to train models with saturating nonlinearities and slows down the training, which suggests that lower learning rates should be utilized. The 'internal covariate shift' phenomenon solves the issue by levelling the inputs to the layer. The power of the strategy proposed in [36] comes from including normalization into the model architecture and carrying it out for each training mini-batch. We can use much larger learning rates with batch normalization, and that makes initialization less important. It also serves as a regularizer, sometimes removing the need for dropout.

3.4.2 Model Specifications

The model specifications are shown in table 3.3. The dropout rate has been taken to be 0.3. We have used batch normalization layer after each convolution layer and after the first dense layer in our model.

Table 3.3 Model specifications of CNN based AMC with 4 convolution layers with batch normalization layer

Layer (type)	**Output Shape**	**Parameter**
conv2d_4 (Conv2D)	(None, 2, 128, 256)	2560
batch_normalization (BatchNormalization)	(None, 2, 128, 256)	1024
max_pooling2d_4 (Maxpooling 2D)	(None, 2, 64, 256)	0
dropout_4 (Dropout)	(None, 2, 64, 256)	0
Conv2d_5 (Conv2D)	(None, 2, 64, 128)	295040
batch_normalization_1 (BatchNormalization)	(None, 2, 64, 128)	512
max_pooling2d_5 (Maxpooling2D)	(None, 2, 32, 128)	0
dropout_5 (Dropout)	(None, 2, 32, 128)	0
Conv2d_6 (Conv2D)	(None, 2, 32, 64)	73792
batch_normalization_2 (BatchNormalization)	(None, 2, 32, 64)	256
max_pooling2d_6 (Maxpooling2D)	(None, 2, 16, 64)	0
dropout_6 (Dropout)	(None, 2, 16, 64)	0
Conv2d_7 (Conv2D)	(None, 2, 16, 64)	36928
batch_normalization_3 (BatchNormalization)	(None, 2, 16, 64)	256
max_pooling2d_7 (Maxpooling2D)	(None, 2, 8, 64)	0
dropout_7 (Dropout)	(None, 2, 8, 64)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_2 (Dense)	(None, 128)	131200
batch_normalization_4 (BatchNormalization)	(None, 128)	512
dense_3 (Dense)	(None, 11)	1419

Total number of parameters are 543,499. Total number of trainable parameters are 542,219. The number of non-trainable parameters are 1,280.

3.4.3 Pseudocode

The code for the model function or definition is as shown:

```
def CNN_4_conv_layer_with_batch_norm():


   model = Sequential()

   model.add(Conv2D(256, (3, 3), activation='relu', padding='same', input_shape=(2,128,1)))
   model.add(BatchNormalization())
   model.add(MaxPooling2D(pool_size=(1, 2), padding='valid',  data_format=None))
   model.add(layers.Dropout(.3))

   model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
   model.add(BatchNormalization())
   model.add(MaxPooling2D(pool_size=(1, 2), padding='valid', data_format=None))
   model.add(layers.Dropout(.3))

   model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
   model.add(BatchNormalization())
   model.add(MaxPooling2D(pool_size=(1, 2), padding='valid', data_format=None))
   model.add(layers.Dropout(.3))

   model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
   model.add(BatchNormalization())
   model.add(MaxPooling2D(pool_size=(1, 2), padding='valid', data_format=None))
   model.add(layers.Dropout(.3))

   model.add(Flatten())

   model.add(Dense(128, activation='relu'))
   model.add(BatchNormalization())
   model.add(Dense(11, activation='softmax'))

   model.summary()

   return model
```

3.4.4 Results and Analysis

We have run our model in two different epoch values: 1 and 10. The performance of the model in 1 epoch value is poor for obvious reasons. We have considered the model performance with epoch value 10 as it gives better results.

Epoch = 1 :

The model performance for 1 epoch is as shown in figure 3.42 to 3.45 The training duration is 30 minutes in this case.

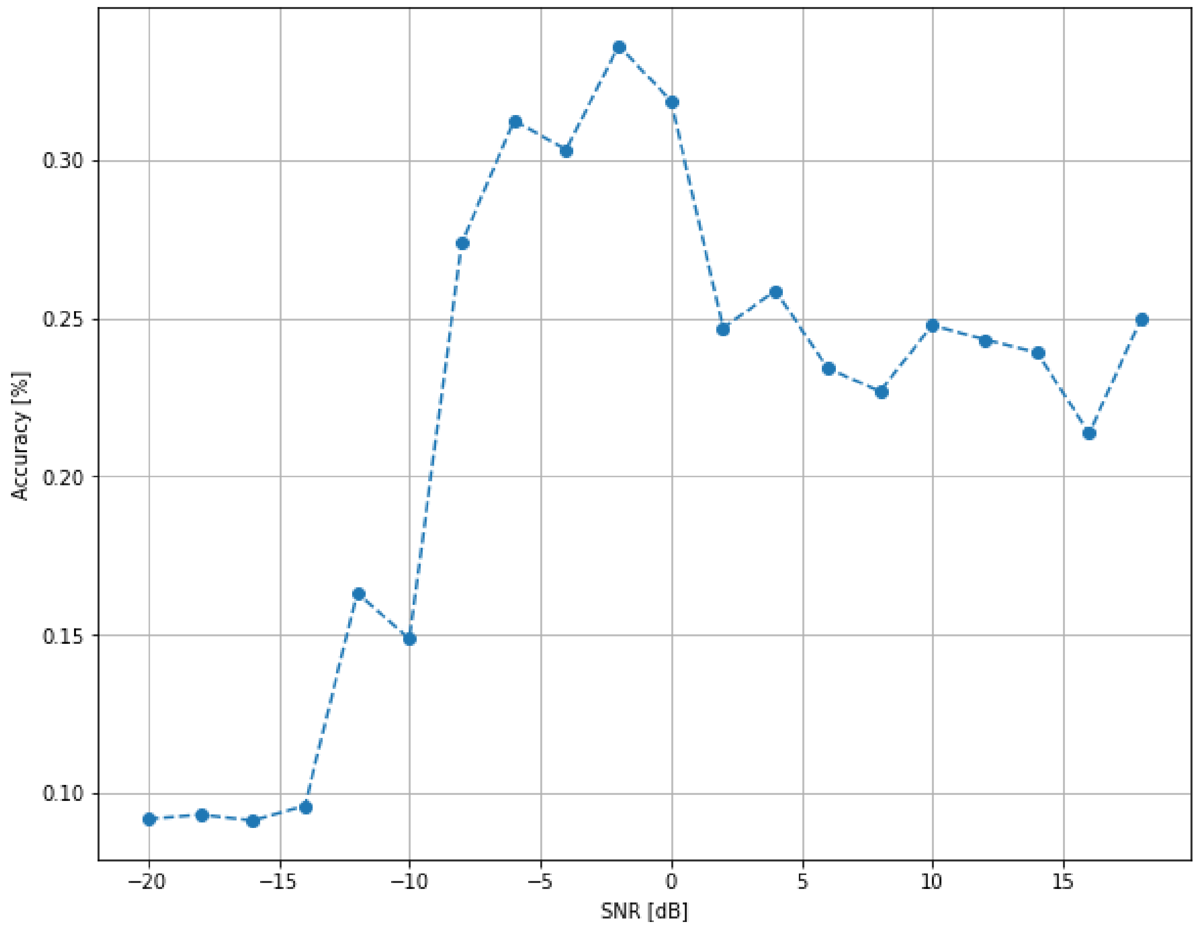


Fig. 3.42 Accuracy vs. SNR plot of CNN based AMC - 4 convolution layer with batch normalization in 1 epoch

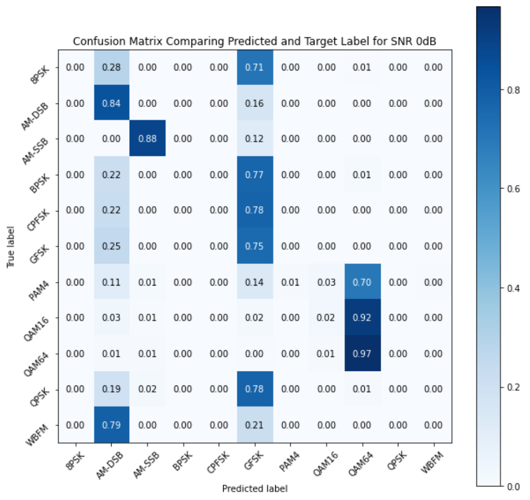


Fig. 3.43 Confusion matrix of CNN based AMC - 4 convolution layer with batch normalization for 0 dB SNR in 1 epoch

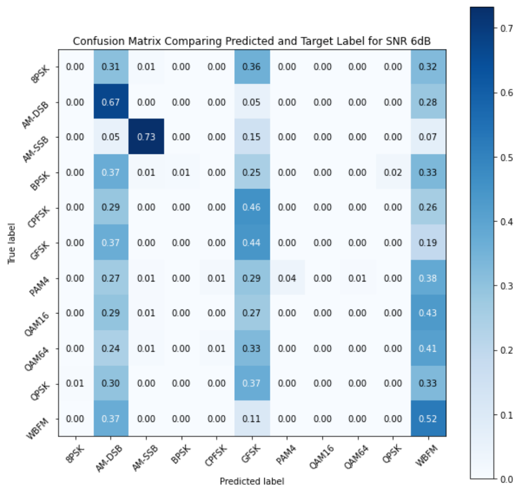


Fig. 3.44 Confusion matrix of CNN based AMC - 4 convolution layer with batch normalization for 6 dB SNR in 1 epoch

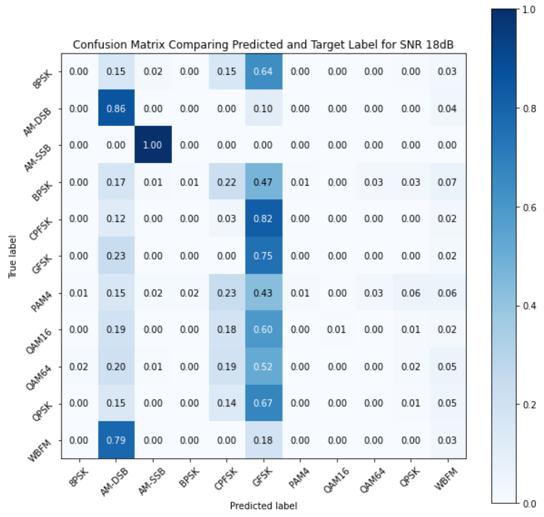


Fig. 3.45 Confusion matrix of CNN based AMC - 4 convolution layer with batch normalization for 18 dB SNR in 1 epoch

Epoch = 10 :

The model performance for 10 epoch is as shown in figure 3.46 to 3.58. The model shows best results in SNR -2 dB with an accuracy of 68.54%. The confusion matrix corresponding to -2 dB SNR, shown in figure 3.52, highlights a distinct diagonal. The training duration is 5 hours 30 minutes.

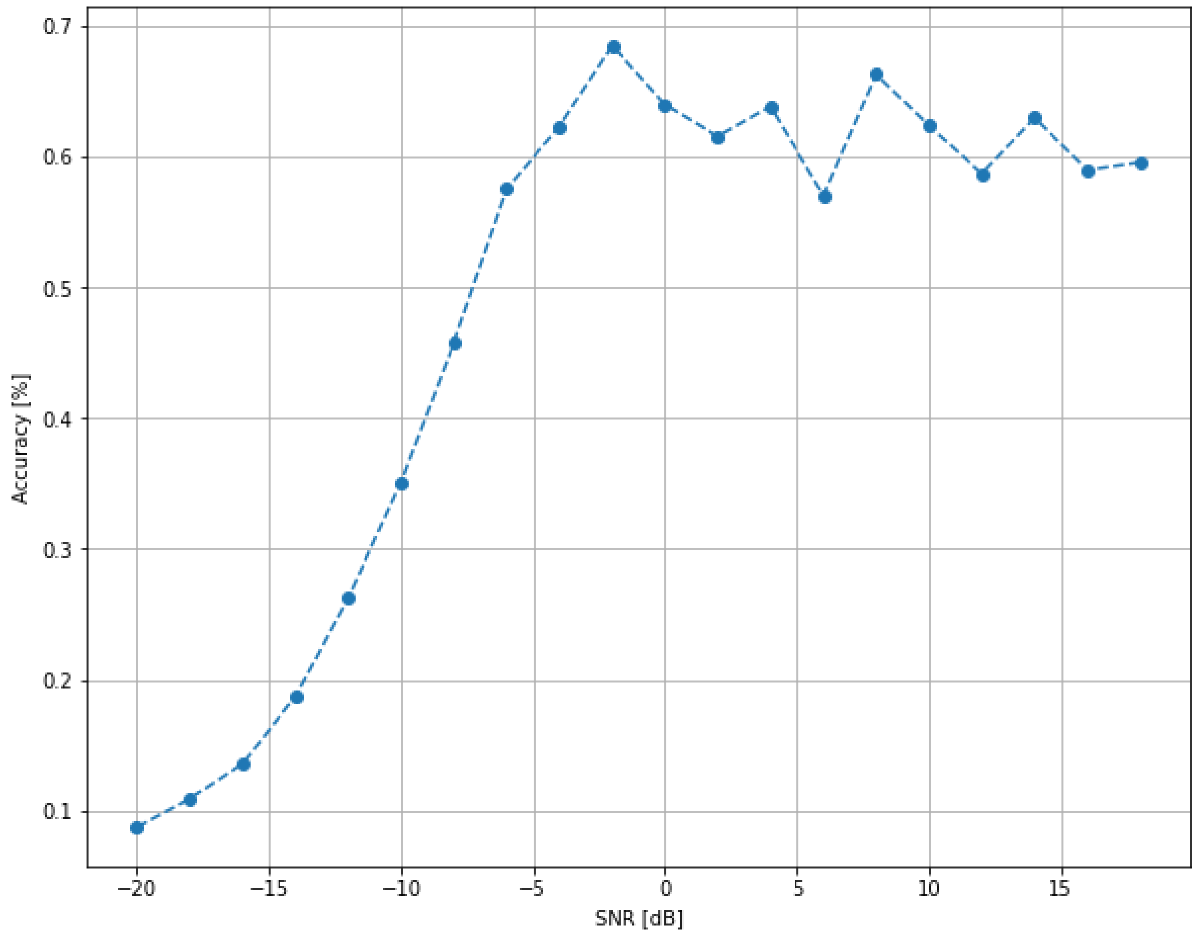


Fig. 3.46 Accuracy vs. SNR plot of CNN based AMC - 4 convolution layer with batch normalization in 10 epochs

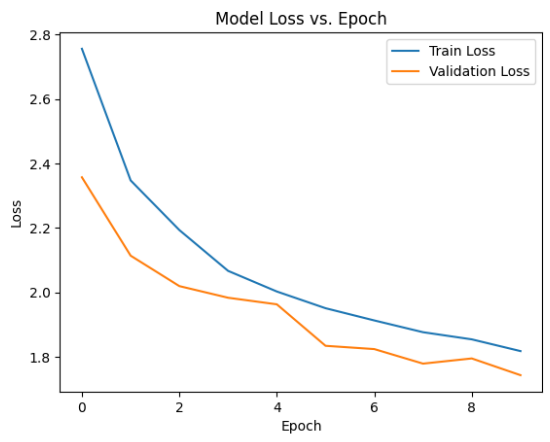


Fig. 3.47 Loss vs. epoch plot of CNN based AMC - 4 convolution layer with batch normalization for -10 dB SNR in 10 epochs

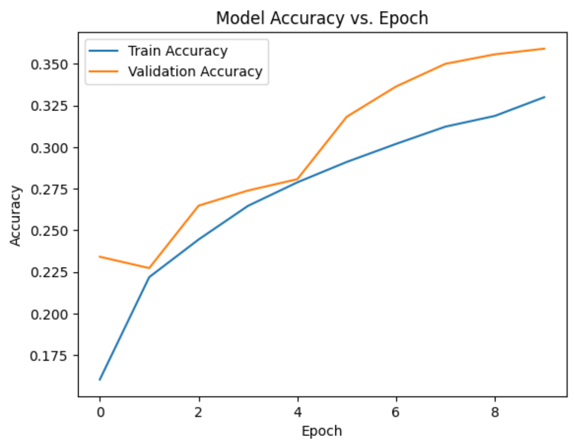


Fig. 3.48 Accuracy vs. epoch plot of CNN based AMC - 4 convolution layer with batch normalization for -10 dB SNR in 10 epochs

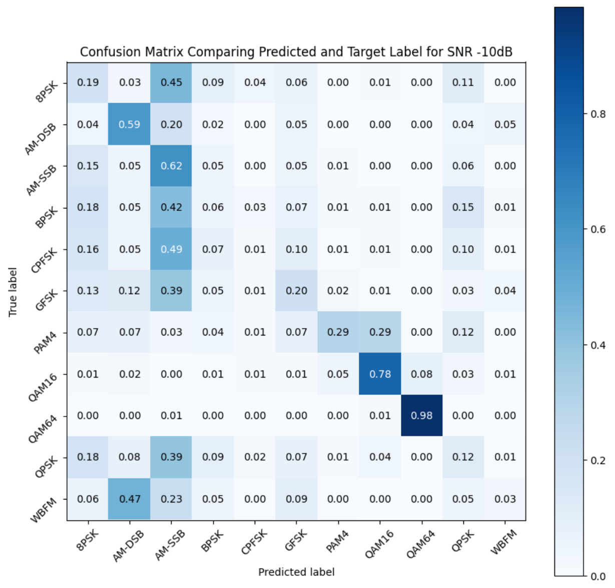


Fig. 3.49 Confusion matrix of CNN based AMC - 4 convolution layer with batch normalization for -10 dB SNR in 10 epochs

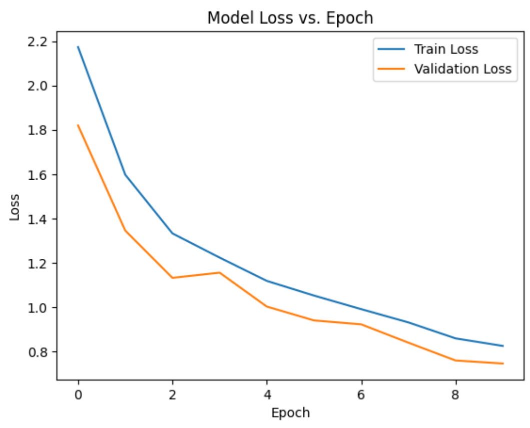


Fig. 3.50 Loss vs. epoch plot of CNN based AMC - 4 convolution layer with batch normalization for -2 dB SNR in 10 epochs

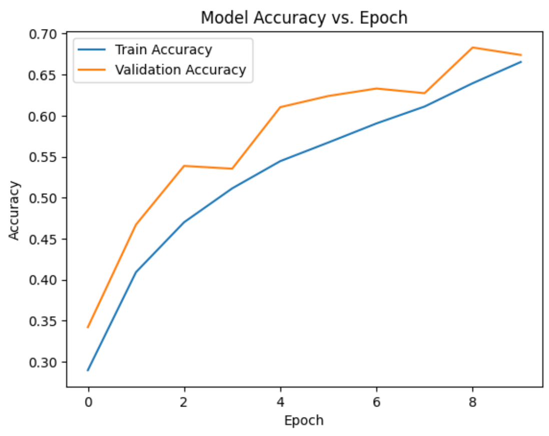
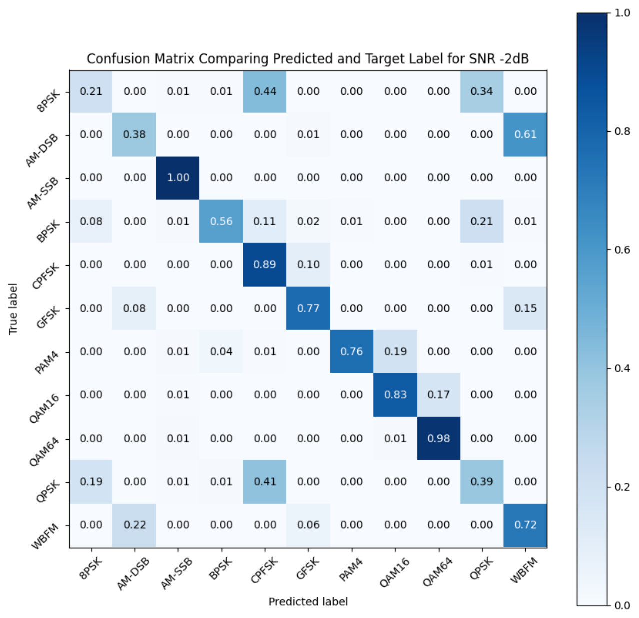


Fig. 3.51 Accuracy vs. epoch plot of CNN based AMC - 4 convolution layer with batch normalization for -2 dB SNR in 10 epochs



	8PSK	AM-DSB	AM-SSB	BPSK	CPFSK	GFSK	PAM4	QAM16	QAM64	QPSK	WBFM
8PSK	0.21	0.00	0.01	0.01	0.44	0.00	0.00	0.00	0.00	0.34	0.00
AM-DSB	0.00	0.38	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.61
AM-SSB	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
BPSK	0.08	0.00	0.01	0.56	0.11	0.02	0.01	0.00	0.00	0.21	0.01
CPFSK	0.00	0.00	0.00	0.00	0.89	0.10	0.00	0.00	0.00	0.01	0.00
GFSK	0.00	0.08	0.00	0.00	0.00	0.77	0.00	0.00	0.00	0.00	0.15
PAM4	0.00	0.00	0.01	0.04	0.01	0.00	0.76	0.19	0.00	0.00	0.00
QAM16	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.83	0.17	0.00	0.00
QAM64	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.98	0.00	0.00
QPSK	0.19	0.00	0.01	0.01	0.41	0.00	0.00	0.00	0.00	0.39	0.00
WBFM	0.00	0.22	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.72

Fig. 3.52 Confusion matrix of CNN based AMC - 4 convolution layer with batch normalization for -2 dB SNR in 10 epochs

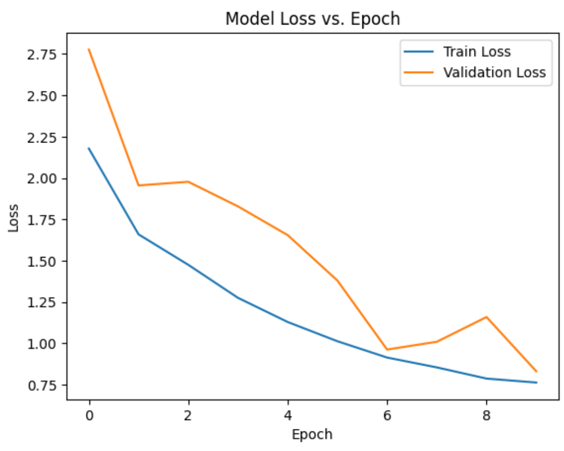


Fig. 3.53 Loss vs. epoch plot of CNN based AMC - 4 convolution layer with batch normalization for 8 dB SNR in 10 epochs

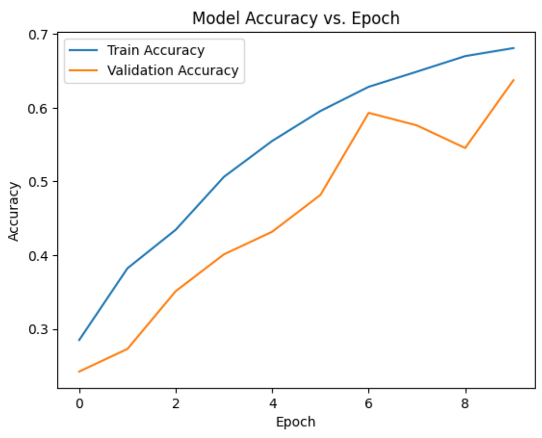


Fig. 3.54 Accuracy vs. epoch plot of CNN based AMC - 4 convolution layer with batch normalization in 10 epochs

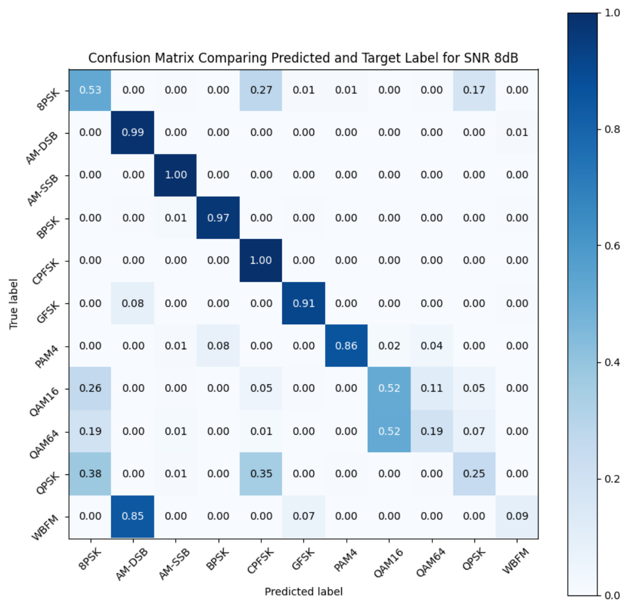


Fig. 3.55 Confusion matrix of CNN based AMC - 4 convolution layer with batch normalization for 8 dB SNR in 10 epochs

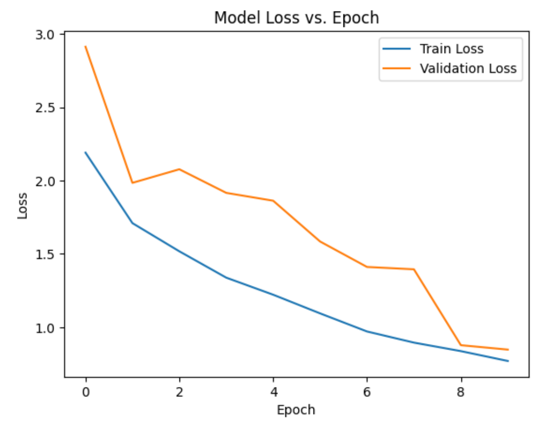


Fig. 3.56 Loss vs. epoch plot of CNN based AMC - 4 convolution layer with batch normalization for 14 dB SNR in 10 epochs

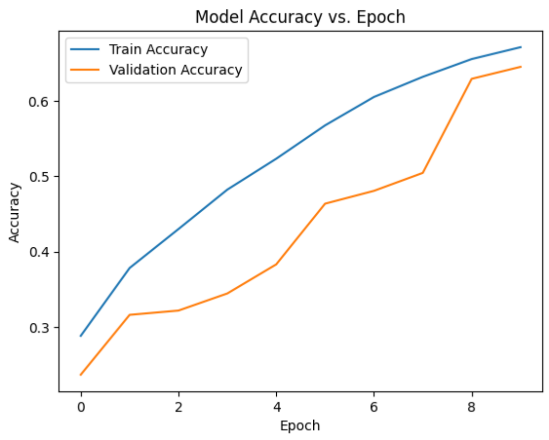


Fig. 3.57 Accuracy vs. epoch plot of CNN based AMC - 4 convolution layer with batch normalization for 14 dB SNR in 10 epochs

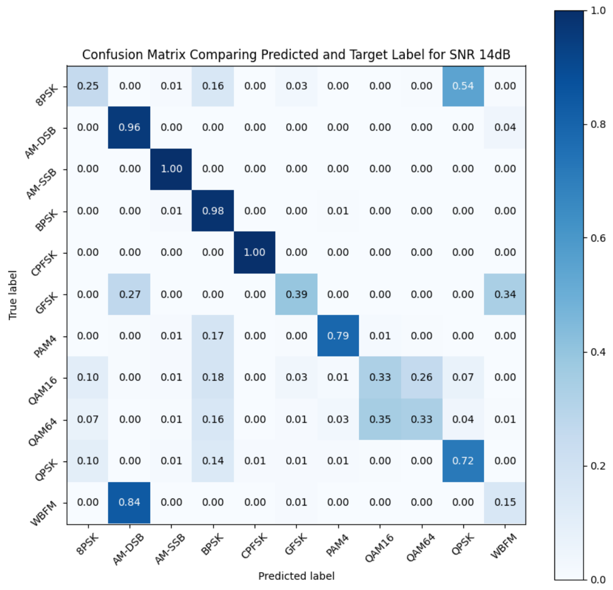


Fig. 3.58 Confusion matrix of CNN based AMC - 4 convolution layer with batch normalization for 14 dB SNR in 10 epochs

3.4.5 Comparison between CNN based AMC (4 convolution layer) with and without Batch Normalization

The graph shown in figure 3.58 compares the performance of CNN based AMC without batch normalization layer (paper model [24]) and with batch normalization layer (proposed model). The model with batch normalization layer (proposed model) shows **9.275% better** performance than the model with no batch normalization layer (paper model).

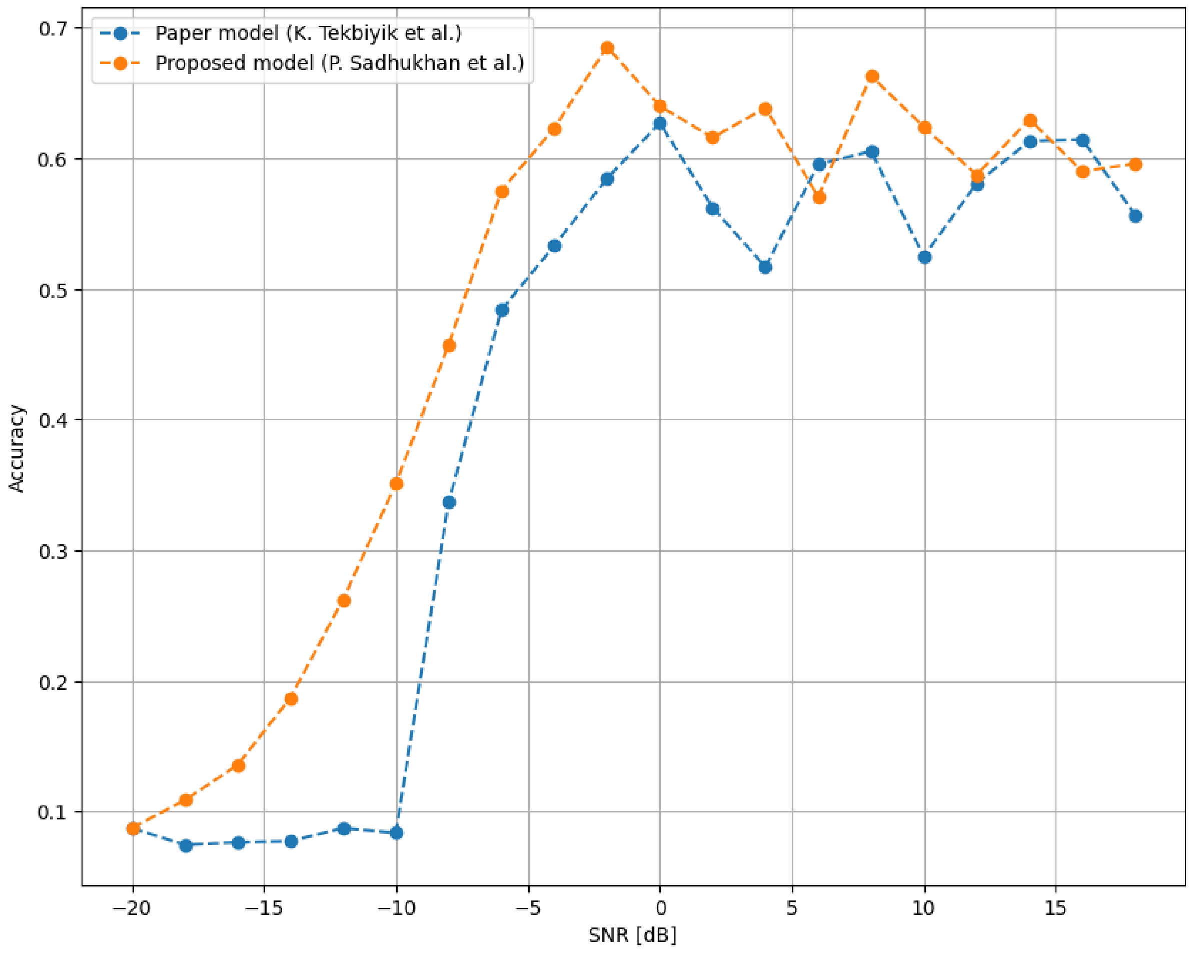


Fig. 3.59 Accuracy vs. SNR plot comparing CNN based AMC – 4 convolution layers with and without batch normalization layer in 10 epochs

3.5 Summary

The CNN has proven to work very well in signal processing domain. In our application of automatic modulation classification, it is observed that 4 convolution layer model shows best results among all other layer settings. It is because a greater number of layers leads to more complexity and takes larger training time due to its huge number of parameters, which in turn leads to overfitting, when the accuracy degrades and loss increases. Similarly, CNN model with small number of convolutional layers fails to classify complex datasets like the one we have used (RML2016.10a which contains 220000 samples). That is why the 3-convolution layer model has shown best performance in SNR 0 dB with an accuracy of 61.5%. The confusion matrix corresponding to 0 dB SNR, shown in figure 3.16, highlights a distinct diagonal. On the other hand, the 4-convolution layer model has shown best performance in SNR 0 dB with an accuracy of 62.73%. The confusion matrix corresponding to 0 dB SNR, shown in figure 3.33, highlights a distinct diagonal. The 3-convolution layer model has taken 4 hours as training time, whereas the 4-convolution layered model has taken 3 hours 30 minutes. As the number of parameters is less in 4 convolution layered model than that of 3-convolution layered model, it has taken 30 minutes less training time. Overall, we see that 4-convolution layered model shows almost 2% better performance than a 3 convolutional layered model. Hence, we have taken the 4 convolutional layered model and added batch normalization layer after each convolution layer and first dense layer. Adding batch normalization layer has significantly improved the accuracy but at the cost of training time. It takes almost 5.5 hours to train a model with 4 convolutional layers with batch normalization. However, it shows almost 9.3% better accuracy than a regular model. By minimizing the internal covariate shift in the data, the model with batch normalization can improve the accuracy of the model.

CHAPTER 4

Residual Network Based Automatic Modulation Classification

4.1 Residual Neural Network

4.1.1 Overview

The depth of a neural network can range from a few layers (AlexNet) to more than one hundred layers. A deep neural network's key advantage is that it can represent extremely complex functions. However, the 'vanishing gradients' provide a significant challenge to their training. Gradient descent learning is impractically slow in very deep neural networks because the gradient signal frequently drops to zero quickly. More specifically, during gradient descent, we multiply by the weight matrix at each step as we backpropagate from the top layer to the bottom layer. Because of the numerous multiplications while the gradients are tiny, they can either develop exponentially quickly and "explode" to take very large values, or they can rapidly fall exponentially to zero. The term "plain" networks refer to the standard networks (such as VGG-16). Even after tens of thousands of iterations, the training error for 56-layer networks in plain networks was greater than it was for 20-layer networks as the number of layers increased from 20 to 56 (as shown in figure 4.1). Although a deeper network should in theory enhance performance, in practice it has more test and training error.

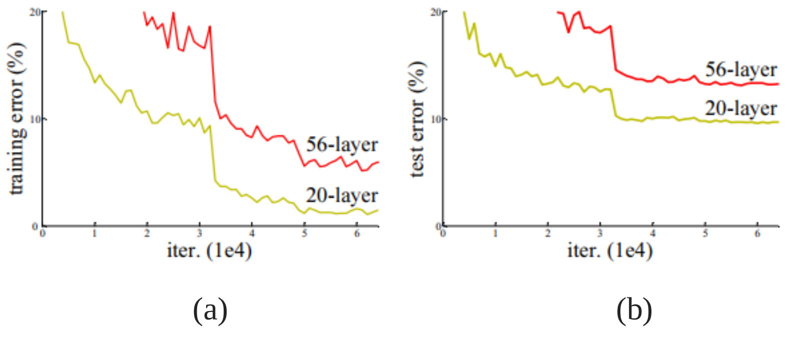


Fig. 4.1 Training error (a) and test error (b) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error [28]

4.1.2 Skip Connections

Use of "Skip Connections" or "Residual Connections" is the remedy for the aforementioned issue. The equation illustrated in figure 4.2 can be used to define the building block of a residual connection, where H(x) and x are the block's output and input, respectively, and F is the residual mapping function that needs to be trained. The residual mapping function that needs to be trained is called F. This block learns the residual mapping F(x) = H(x) - x, which may be simpler to learn because learning the mapping H(x) = x may be quite challenging [28]. An identity mapping is generated by adding the bypass connection/skip connection, enabling the deep neural network to learn straightforward operations that would have been needed a shallower network to learn.

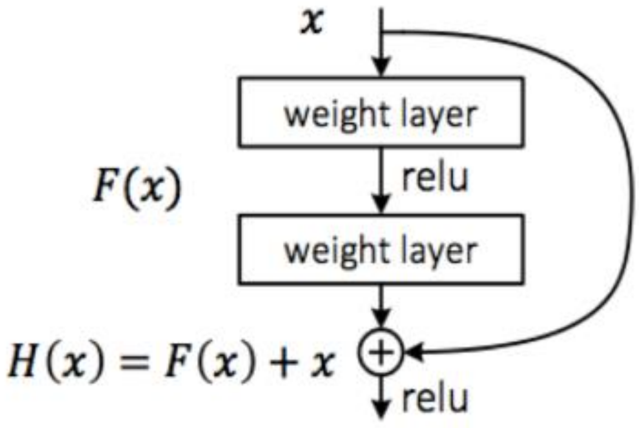


Fig. 4.2 Building block of Residual Network (Resnet)

4.1.3 Summary

With a top-5 error rate of 3.57%, Residual Neural Network (ResNet) took first place in the ILSVRC 2015 classification competition. It also took first place in the categories of ImageNet detection, ImageNet localization, Coco detection, and Coco segmentation at the 2015 ILSVRC and COCO competition.

It is not practicable to use very deep neural networks or simple networks since they are challenging to train because of the vanishing gradient problem. The disappearing gradient issue is helped by the Skip-Connections or the Residual Connections. They also facilitate the learning of an identity function by a ResNet block. The identity block and the convolutional block are the two primary kinds of ResNets blocks. These building elements are stacked to create Deep Residual Networks.

4.2 ResNet based Automatic Modulation Classification

4.2.1 Model Specifications

The model specifications are shown in Table 4.1. This architecture is inspired by the model used in [26] where the dropout rate has been taken to be 0.6.

Table 4.1 Model specifications of ResNet based AMC

Layer (type)	**Output Shape**	**Parameter**	**Connected to**
input_1 (InputLayer)	[(None, 2, 128, 1)]	0	[]
conv2d (Conv2D)	(None, 2, 128, 256)	1024	[‘input_1[0][0]’]
dropout (Dropout)	(None, 2, 128, 256)	0	[‘conv2d[0][0]’]
conv2d_1 (Conv2D)	(None, 2, 128, 256)	393472	[‘dropout[0][0]’]
add (Add)	(None, 2, 128, 256)	0	[‘input_1[0][0]’, ‘conv2d_1[0][0]’]
conv2d_2 (Conv2D)	(None, 2, 128, 80)	61520	[‘ add[0][0]’]
dropout_1 (Dropout)	(None, 2, 128, 80)	0	[‘ conv2d_2[0][0]’]
conv2d_3 (Conv2D)	(None, 2, 128, 80)	19280	[‘ dropout_1[0][0]’]
dropout_2 (Dropout)	(None, 2, 128, 80)	0	[‘ conv2d_3[0][0]’]
flatten (Flatten)	(None, 20480)	0	[‘ dropout_2[0][0]’]
dense (Dense)	(None, 128)	2621568	[‘ flatten [0] [0’]
dropout_3 (Dense)	(None, 128)	0	[‘ dense[0][0]’]
dense_1 (Dense)	(None, 11)	1419	[‘ dropout_3[0][0]’]

Total number of parameters are 3,098,283. Total number of trainable parameters are 3,098,283. The number of non-trainable parameters are 0.

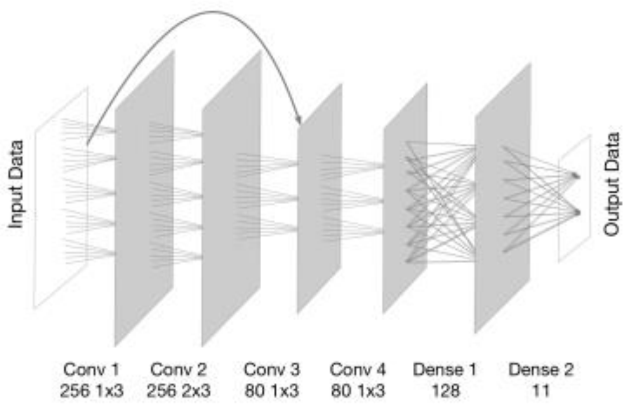


Fig. 4.3 Architecture of seven-layer ResNet [26]

4.2.2 Pseudocode

The code for the model function or definition is as shown:

```
def resnet(x_shape):
    x = Input(x_shape)
    #print(x.shape)
    y = Conv2D(256, (1, 3), activation='relu', padding='same')(x)
    y = Dropout(0.6)(y)
    y = Conv2D(256, (2, 3), activation='relu', padding='same')(y)
    #print(y.shape)
    z = Add()([x, y])
    #print(z.shape)
    z = Conv2D(80, (1, 3), activation='relu', padding='same')(z)
    z = Dropout(0.6)(z)
    z = Conv2D(80, (1, 3), activation='relu', padding='same')(z)
    z = Dropout(0.6)(z)
    #print(z.shape)
    zz = Flatten()(z)
    #print(z.shape)
    z = Dense(128, activation='relu',)(zz)
    z = Dropout(0.6)(z)
    print(z.shape)
    z = Dense(11, activation='softmax',)(z)
    print(z.shape)
    model = Model(x,z)
    model.summary()
    return model
```

4.2.3 Results and Analysis

We have run our model for two different epoch values: 1 and 10. The performance of the model in 1 epoch value is poor for obvious reasons. We have considered the model performance with epoch value 10 as it gives better results.

Epoch = 1 :

The model performance for 1 epoch is as shown in figure 4.4 to 4.7. The training duration is 1 hour 30 minutes in this case.

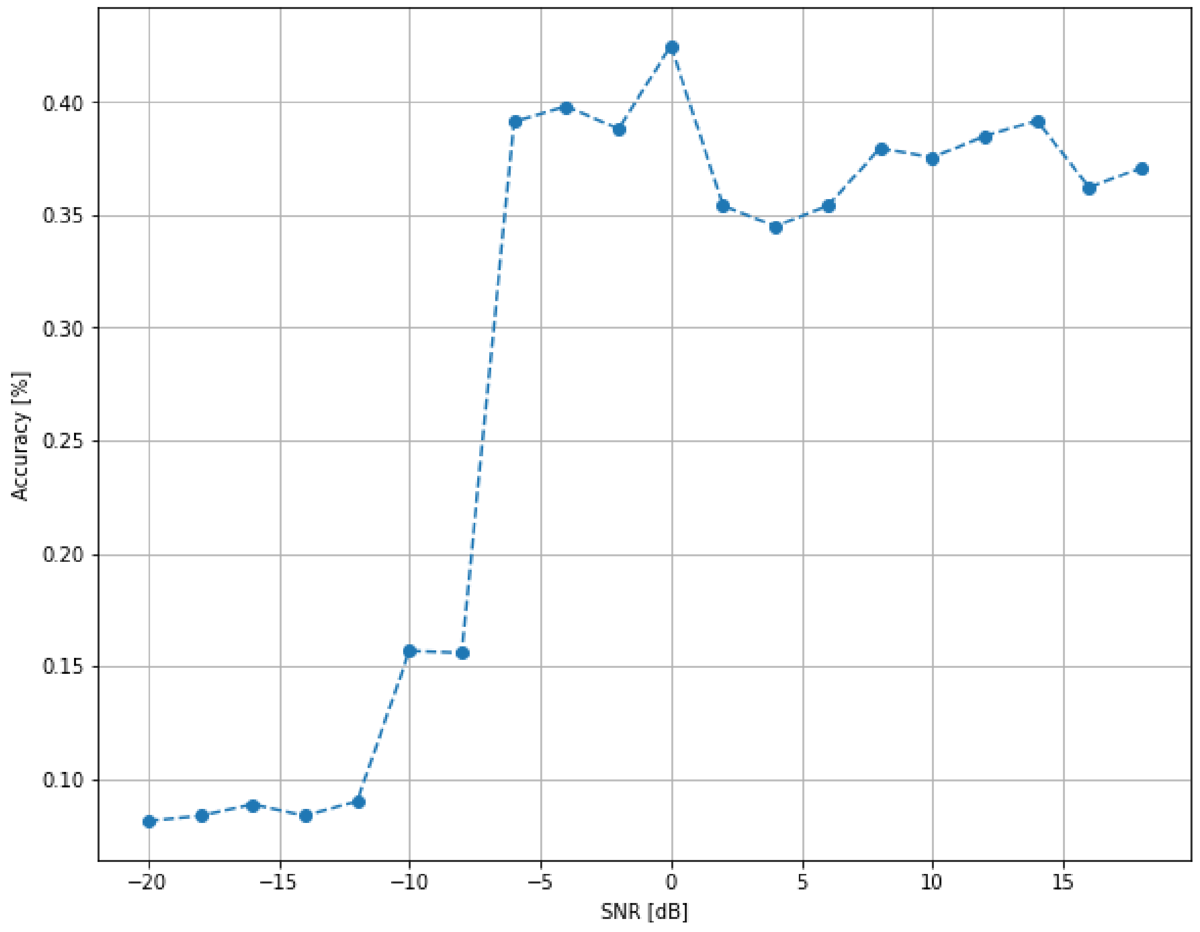


Fig. 4.4 Accuracy vs. SNR plot for ResNet based AMC in 1 epoch

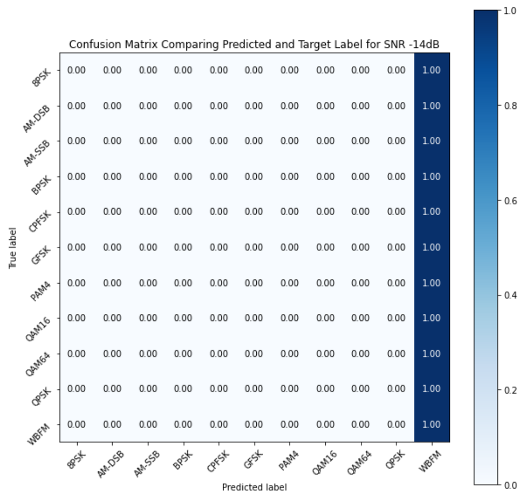


Fig. 4.5 Confusion matrix of ResNet based AMC for -14 dB SNR in 1 epoch

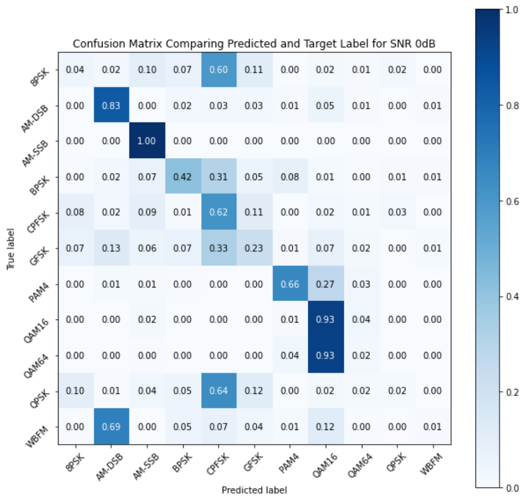


Fig. 4.6 Confusion matrix of ResNet based AMC for 0 dB SNR in 1 epoch

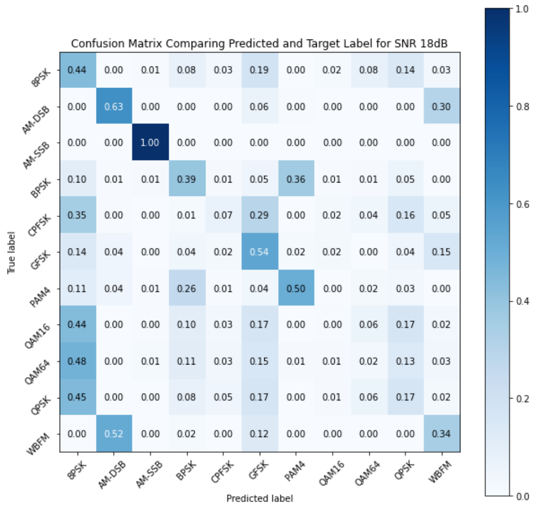


Fig. 4.7 Confusion matrix of ResNet based AMC for 18 dB SNR in 1 epoch

Epoch = 10 :

The model performance for 10 epoch is as shown in figure 4.8 to 4.20. The model shows best results in SNR 2 dB with an accuracy of 70.545%. The confusion matrix corresponding to 2 dB SNR, shown in figure 4.14, highlights a distinct diagonal. The training duration is 6 hours 30 minutes.

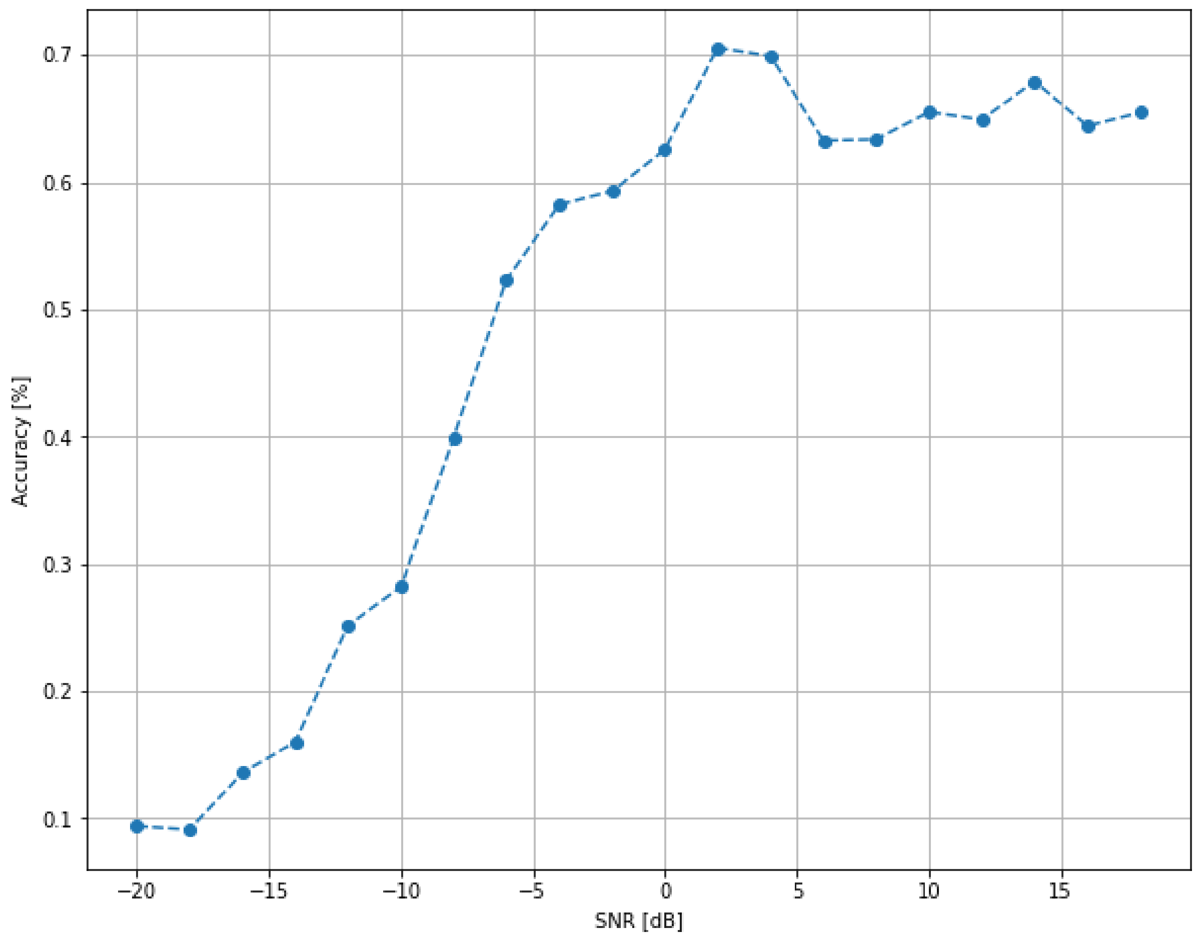


Fig. 4.8 Accuracy vs. SNR plot of ResNet based AMC in 10 epochs

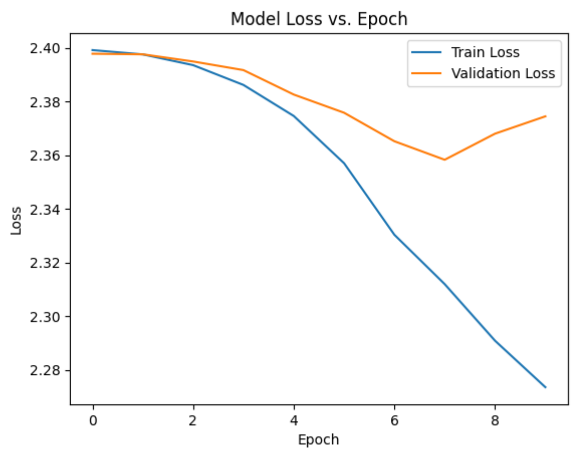


Fig. 4.9 Loss vs. epoch plot of ResNet based AMC for -16 dB SNR in 10 epochs

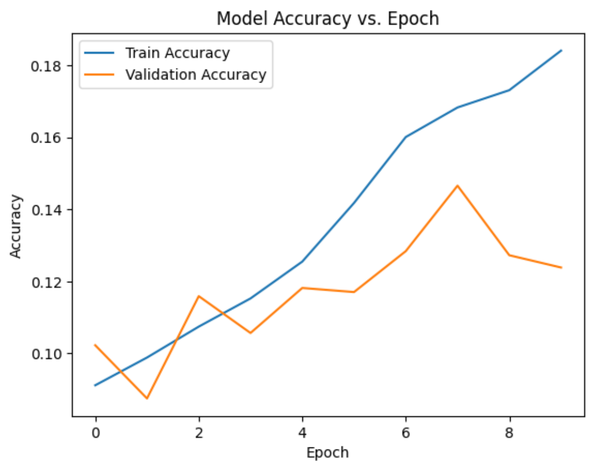


Fig. 4.10 Accuracy vs. epoch plot of ResNet based AMC for -16 dB SNR in 10 epochs

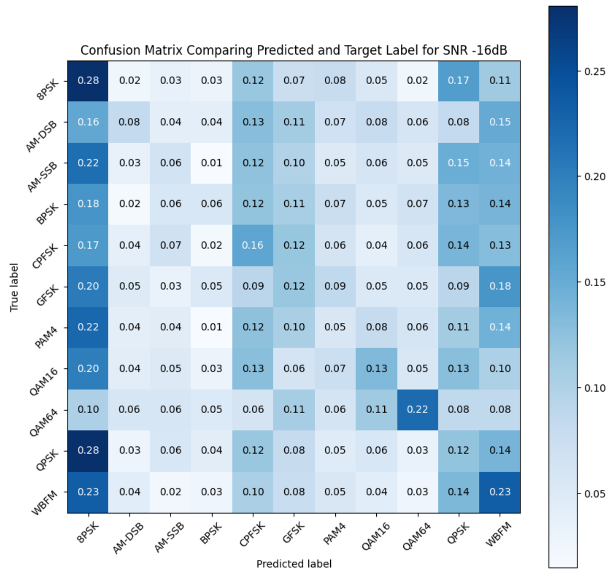


Fig. 4.11 Confusion matrix of ResNet based AMC for -16 dB SNR in 10 epochs

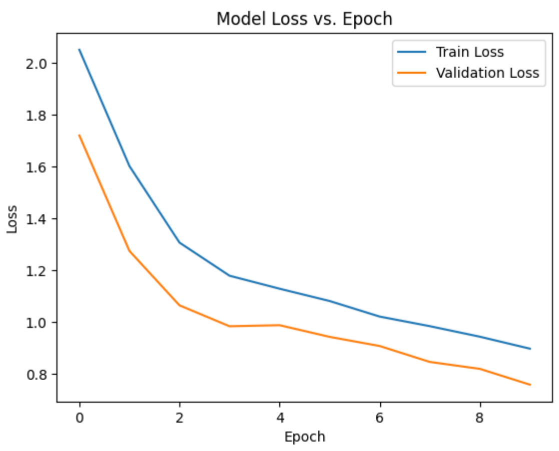


Fig. 4.12 Loss vs. epoch plot of ResNet based AMC for 2 dB SNR in 10 epochs

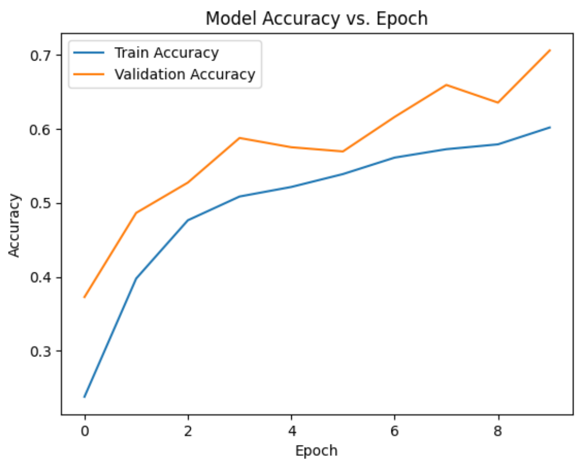
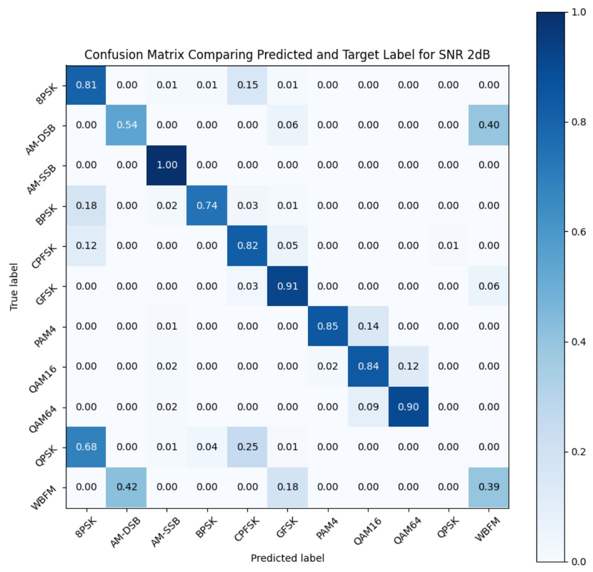


Fig. 4.13 Accuracy vs. epoch plot of ResNet based AMC for 2 dB SNR in 10 epochs



	8PSK	AM-DSB	AM-SSB	BPSK	CPFSK	GFSK	PAM4	QAM16	QAM64	QPSK	WBFM
8PSK	0.81	0.00	0.01	0.01	0.15	0.01	0.00	0.00	0.00	0.00	0.00
AM-DSB	0.00	0.54	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.40
AM-SSB	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
BPSK	0.18	0.00	0.02	0.74	0.03	0.01	0.00	0.00	0.00	0.00	0.00
CPFSK	0.12	0.00	0.00	0.00	0.82	0.05	0.00	0.00	0.00	0.01	0.00
GFSK	0.00	0.00	0.00	0.00	0.03	0.91	0.00	0.00	0.00	0.00	0.06
PAM4	0.00	0.00	0.01	0.00	0.00	0.00	0.85	0.14	0.00	0.00	0.00
QAM16	0.00	0.00	0.02	0.00	0.00	0.00	0.02	0.84	0.12	0.00	0.00
QAM64	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.09	0.90	0.00	0.00
QPSK	0.68	0.00	0.01	0.04	0.25	0.01	0.00	0.00	0.00	0.00	0.00
WBFM	0.00	0.42	0.00	0.00	0.00	0.18	0.00	0.00	0.00	0.00	0.39

Fig. 4.14 Confusion matrix of ResNet based AMC for 2 dB SNR in 10 epochs

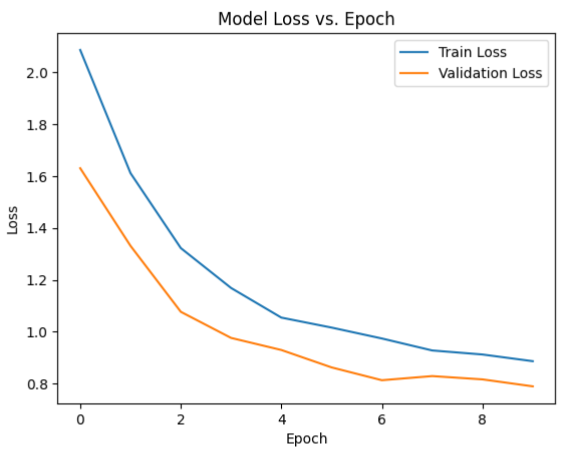


Fig. 4.15 Loss vs. epoch plot of ResNet based AMC for 14 dB SNR in 10 epochs

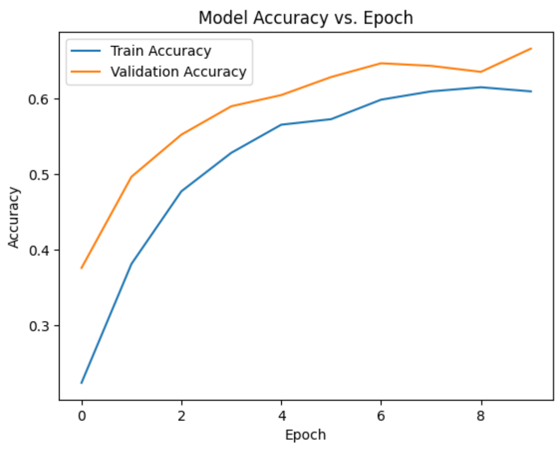


Fig. 4.16 Accuracy vs. epoch plot of ResNet based AMC for 14 dB SNR in 10 epochs

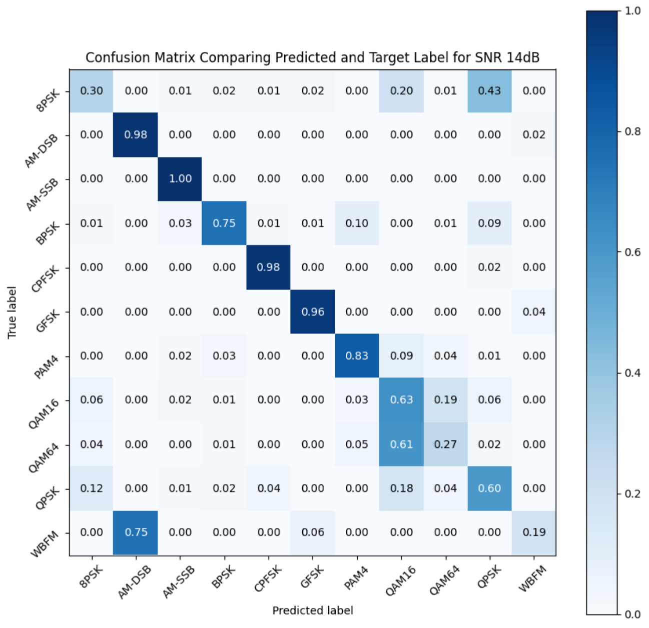


Fig. 4.17 Confusion matrix of ResNet based AMC for 14 dB SNR in 10 epochs

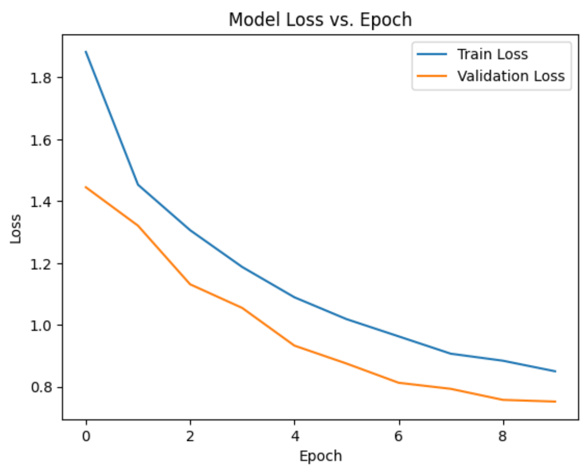


Fig. 4.18 Loss vs. epoch plot of ResNet based AMC for 18 dB SNR in 10 epochs

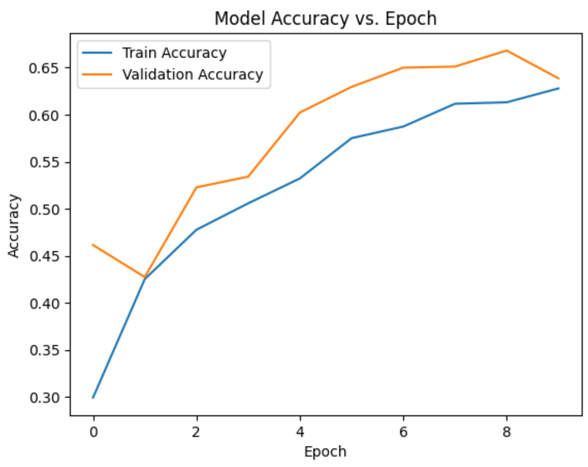
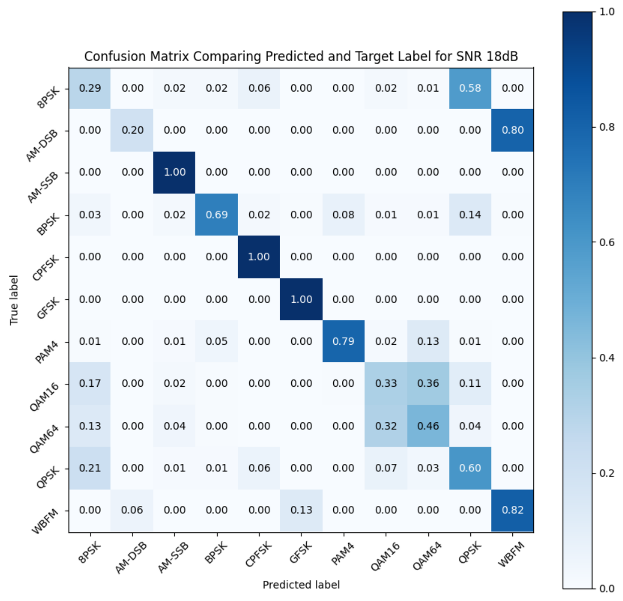


Fig. 4.19 Accuracy vs. epoch plot of ResNet based AMC for 18 dB SNR in 10 epochs



True label	8PSK	AM-DSB	AM-SSB	BPSK	CPFSK	GFSK	PAM4	QAM16	QAM64	QPSK	WBFM
8PSK	0.29	0.00	0.02	0.02	0.06	0.00	0.00	0.02	0.01	0.58	0.00
AM-DSB	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80
AM-SSB	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
BPSK	0.03	0.00	0.02	0.69	0.02	0.00	0.08	0.01	0.01	0.14	0.00
CPFSK	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
GFSK	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
PAM4	0.01	0.00	0.01	0.05	0.00	0.00	0.79	0.02	0.13	0.01	0.00
QAM16	0.17	0.00	0.02	0.00	0.00	0.00	0.00	0.33	0.36	0.11	0.00
QAM64	0.13	0.00	0.04	0.00	0.00	0.00	0.00	0.32	0.46	0.04	0.00
QPSK	0.21	0.00	0.01	0.01	0.06	0.00	0.00	0.07	0.03	0.60	0.00
WBFM	0.00	0.06	0.00	0.00	0.00	0.13	0.00	0.00	0.00	0.00	0.82

Fig. 4.20 Confusion matrix of ResNet based AMC for 18 dB SNR in 10 epochs

4.3 Modification in ResNet based Automatic Modulation Classification

4.3.1 Model Specifications

The model specifications are shown in Table 4.1. We have tuned the dropout rate to 0.3 and observed significant improvement in model accuracy without much increase in overfitting.

Table 4.2 Model specifications of ResNet based AMC with dropout rate 0.3

Layer (type)	Output Shape	Parameter	Connected to
input_1 (InputLayer)	[(None, 2, 128, 1)]	0	[]
conv2d (Conv2D)	(None, 2, 128, 256)	1024	['input_1[0][0]']
dropout (Dropout)	(None, 2, 128, 256)	0	['conv2d[0][0]']
conv2d_1 (Conv2D)	(None, 2, 128, 256)	393472	['dropout[0][0]']
add (Add)	(None, 2, 128, 256)	0	['input_1[0][0]', 'conv2d_1[0][0]']
conv2d_2 (Conv2D)	(None, 2, 128, 80)	61520	[' add[0][0]']
dropout_1 (Dropout)	(None, 2, 128, 80)	0	[' conv2d_2[0][0]']
conv2d_3 (Conv2D)	(None, 2, 128, 80)	19280	[' dropout_1[0][0]']
dropout_2 (Dropout)	(None, 2, 128, 80)	0	[' conv2d_3[0][0]']
flatten (Flatten)	(None, 20480)	0	[' dropout_2[0][0]']
dense (Dense)	(None, 128)	2621568	[' flatten [0] [0']
dropout_3 (Dense)	(None, 128)	0	[' dense[0][0]']
dense_1 (Dense)	(None, 11)	1419	[' dropout_3[0][0]']

Total number of parameters are 3,098,283. Total number of trainable parameters are 3,098,283. The number of non-trainable parameters are 0.

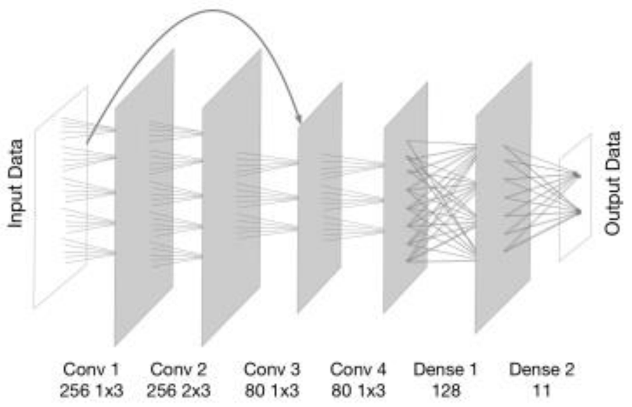


Fig. 4.21 Architecture of seven-layer ResNet with dropout rate 0.3

4.3.2 Pseudocode

The code for the model function or definition is as shown:

```
def resnet(x_shape):
    x = Input(x_shape)
    #print(x.shape)
    y = Conv2D(256, (1, 3), activation='relu', padding='same')(x)
    y = Dropout(0.3)(y)
    y = Conv2D(256, (2, 3), activation='relu', padding='same')(y)
    #print(y.shape)
    z = Add()([x, y])
    #print(z.shape)
    z = Conv2D(80, (1, 3), activation='relu', padding='same')(z)
    z = Dropout(0.3)(z)
    z = Conv2D(80, (1, 3), activation='relu', padding='same')(z)
    z = Dropout(0.3)(z)
    #print(z.shape)
    zz = Flatten()(z)
    #print(z.shape)
    z = Dense(128, activation='relu',)(zz)
    z = Dropout(0.3)(z)
    print(z.shape)
    z = Dense(11, activation='softmax',)(z)
    print(z.shape)
    model = Model(x,z)
    model.summary()
    return model
```

4.3.3 Results and Analysis

We have run our model for two different epoch values: 1 and 10. The performance of the model in 1 epoch value is poor for obvious reasons. We have considered the model performance with epoch value 10 as it gives better results.

Epoch = 1 :

The model performance for 1 epoch is as shown in figure 4.22 to 4.25. The training duration is 1 hour 13 minutes in this case.

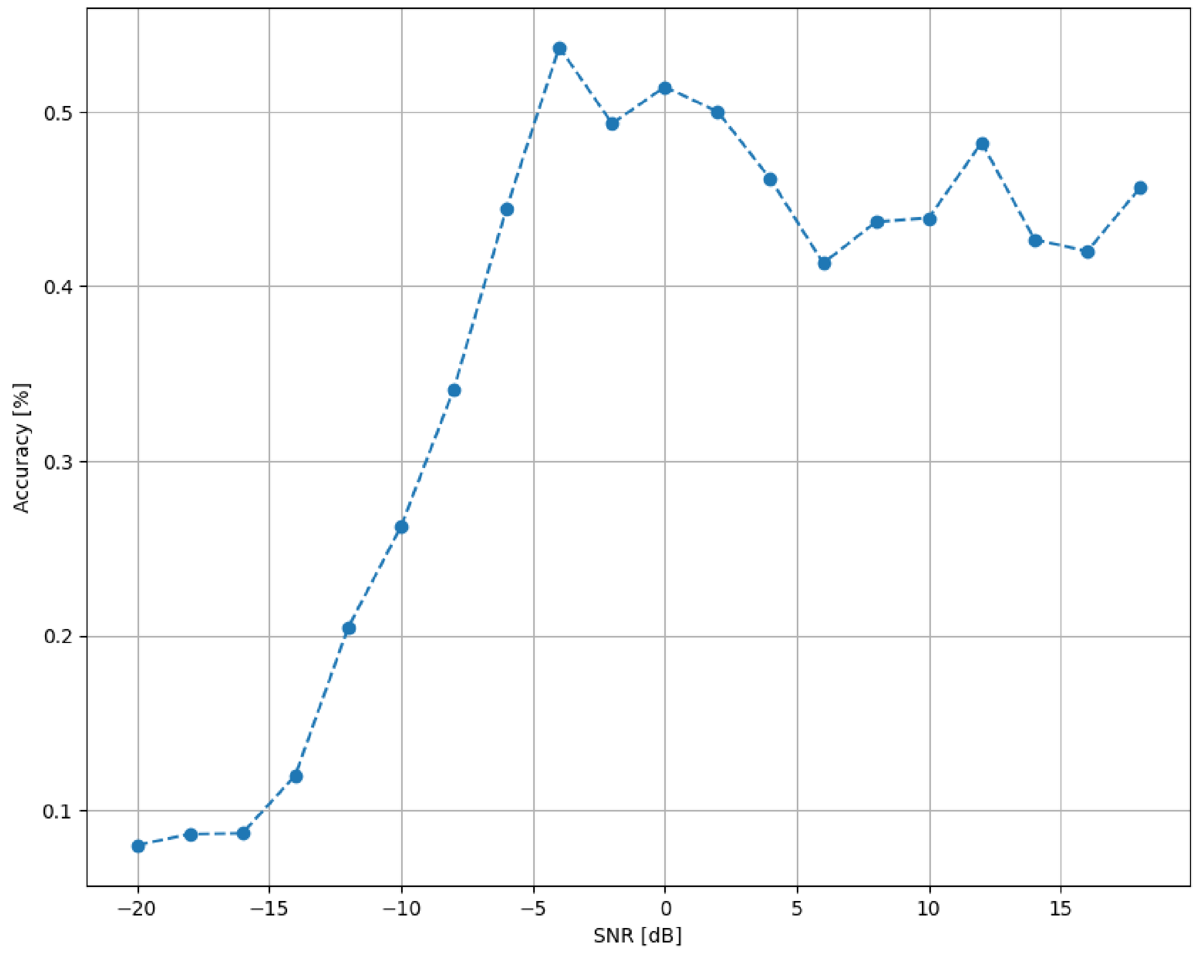


Fig. 4.22 Accuracy vs. SNR plot of ResNet based AMC with dropout rate 0.3 in 1 epoch

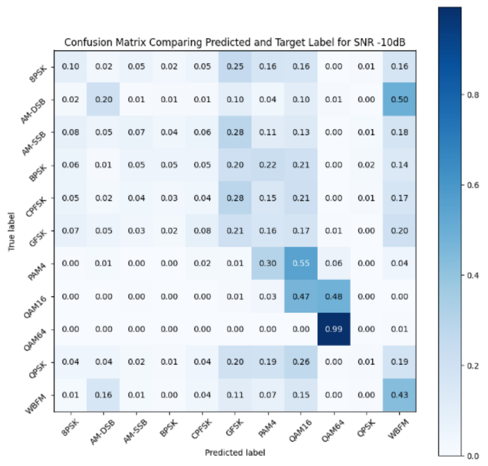


Fig. 4.23 Confusion matrix of ResNet based AMC with dropout rate 0.3 for -10 dB SNR in 1 epoch

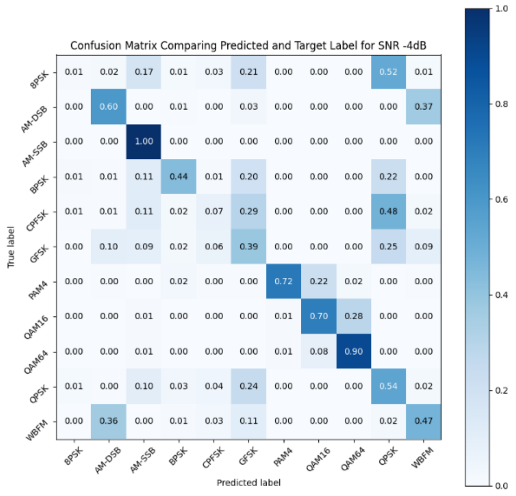


Fig. 4.24 Confusion matrix of ResNet based AMC with dropout rate 0.3 for -4 dB SNR in 1 epoch

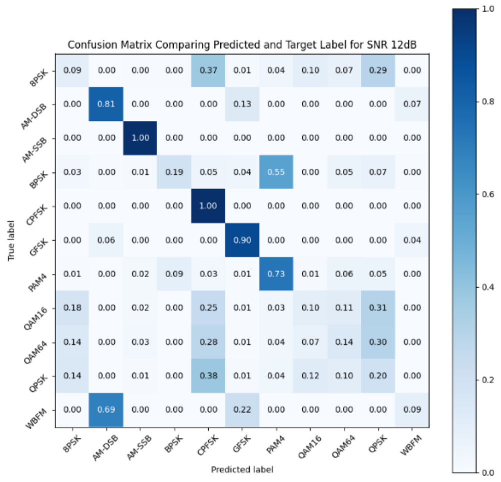


Fig. 4.25 Confusion matrix of ResNet based AMC with dropout rate 0.3 for 12 dB SNR in 1 epoch

Epoch = 10 :

The model performance for 10 epoch is as shown in figure 4.26 to 4.38. The model shows best results in SNR 6 dB with an accuracy of 73.773%. The confusion matrix corresponding to 6 dB SNR, shown in figure 4.35, highlights a distinct diagonal. The training duration is 8 hours 32 minutes.

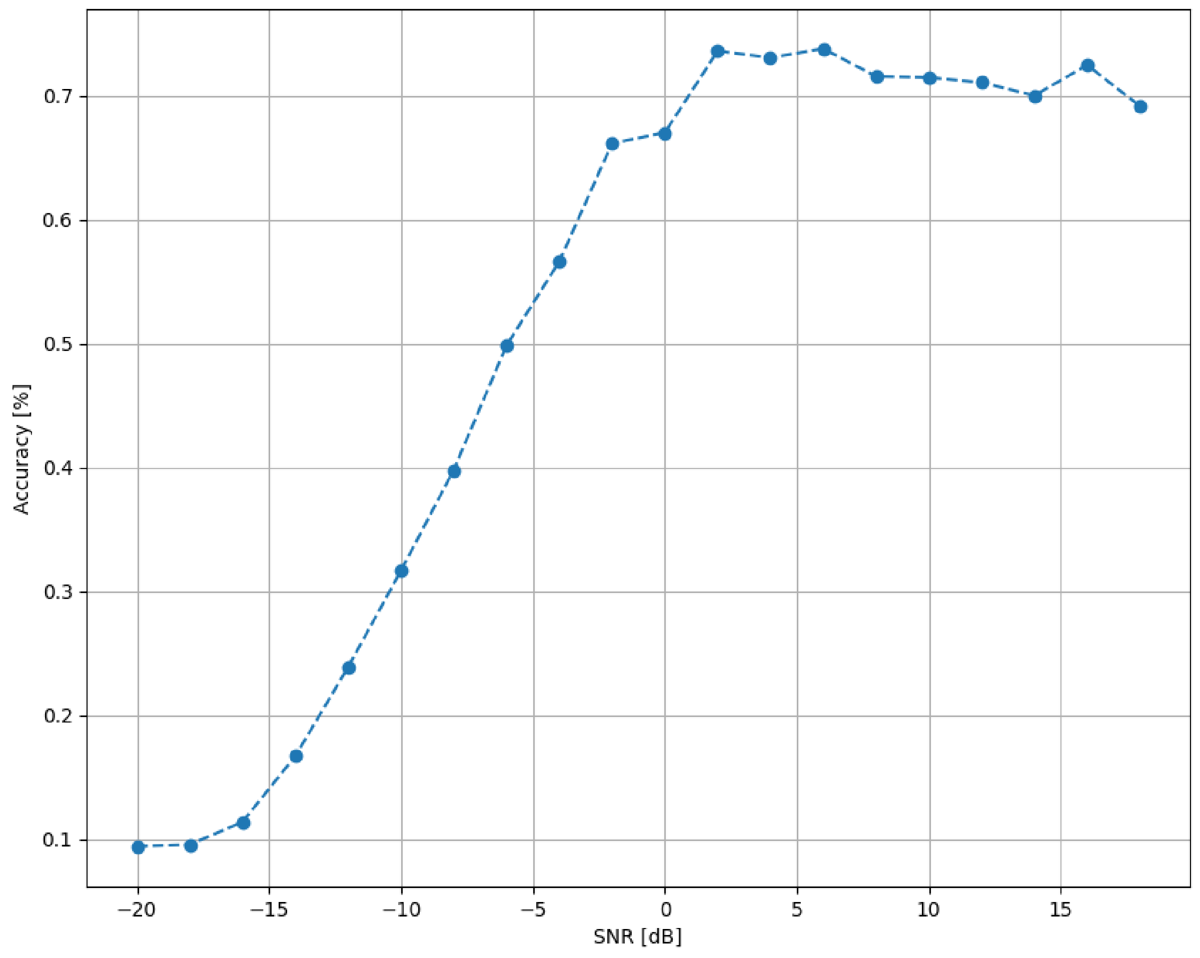


Fig. 4.26 Accuracy vs. SNR plot of ResNet based AMC with dropout rate 0.3 in 10 epochs

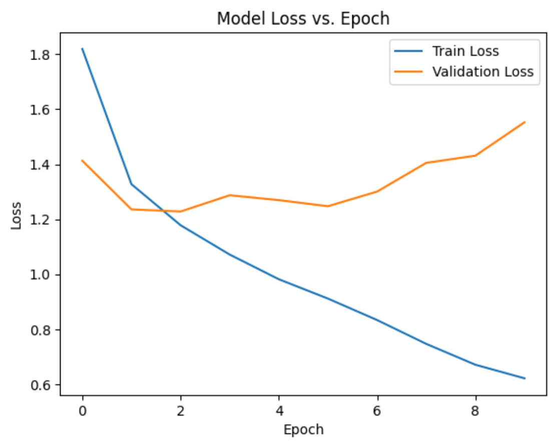


Fig. 4.27 Loss vs. epoch plot of ResNet based AMC with dropout rate 0.3 for -6 dB SNR in 10 epochs

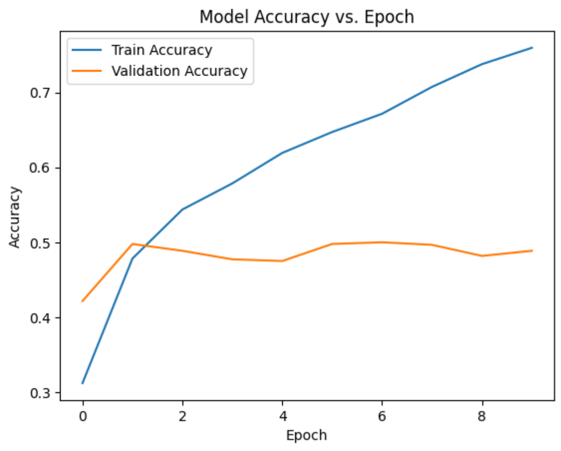
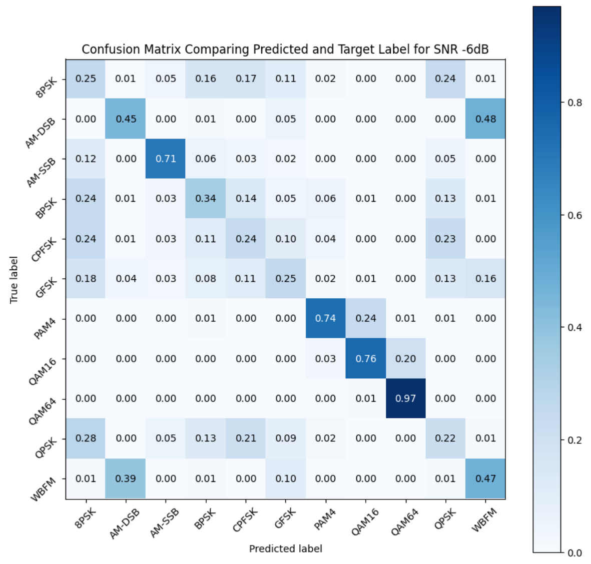


Fig. 4.28 Accuracy vs. epoch plot of ResNet based AMC with dropout rate 0.3 for -6 dB SNR in 10 epochs



	8PSK	AM-DSB	AM-SSB	BPSK	CPFSK	GFSK	PAM4	QAM16	QAM64	QPSK	WBFM
8PSK	0.25	0.01	0.05	0.16	0.17	0.11	0.02	0.00	0.00	0.24	0.01
AM-DSB	0.00	0.45	0.00	0.01	0.00	0.05	0.00	0.00	0.00	0.00	0.48
AM-SSB	0.12	0.00	0.71	0.06	0.03	0.02	0.00	0.00	0.00	0.05	0.00
BPSK	0.24	0.01	0.03	0.34	0.14	0.05	0.06	0.01	0.00	0.13	0.01
CPFSK	0.24	0.01	0.03	0.11	0.24	0.10	0.04	0.00	0.00	0.23	0.00
GFSK	0.18	0.04	0.03	0.08	0.11	0.25	0.02	0.01	0.00	0.13	0.16
PAM4	0.00	0.00	0.00	0.01	0.00	0.00	0.74	0.24	0.01	0.01	0.00
QAM16	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.76	0.20	0.00	0.00
QAM64	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.97	0.00	0.00
QPSK	0.28	0.00	0.05	0.13	0.21	0.09	0.02	0.00	0.00	0.22	0.01
WBFM	0.01	0.39	0.00	0.01	0.00	0.10	0.00	0.00	0.00	0.01	0.47

Fig. 4.29 Confusion matrix of ResNet based AMC with dropout rate 0.3 for -6 dB SNR in 10 epochs

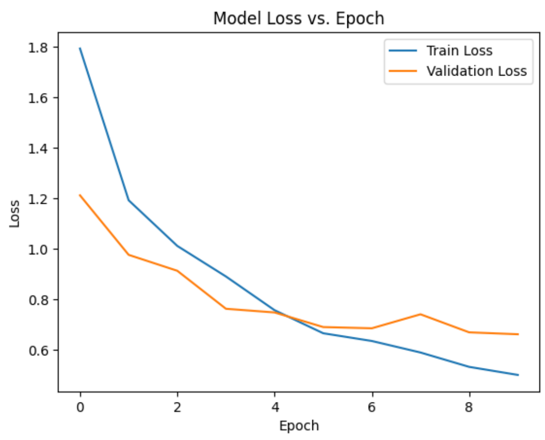


Fig. 4.30 Loss vs. epoch plot of ResNet based AMC with dropout rate 0.3 for 2 dB SNR in 10 epochs

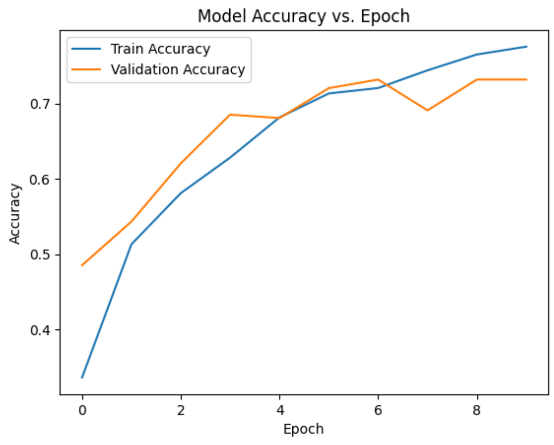


Fig. 4.31 Accuracy vs. epoch plot of ResNet based AMC with dropout rate 0.3 for 2 dB SNR in 10 epochs

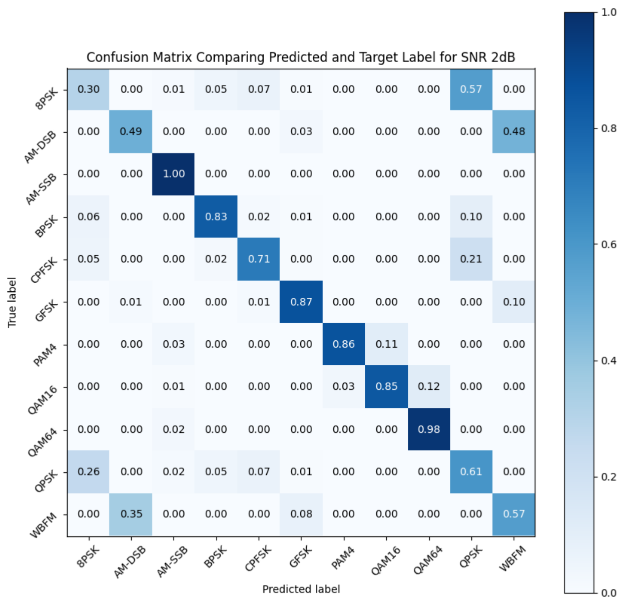


Fig. 4.32 Confusion matrix of ResNet based AMC with dropout rate 0.3 for 2 dB SNR in 10 epochs

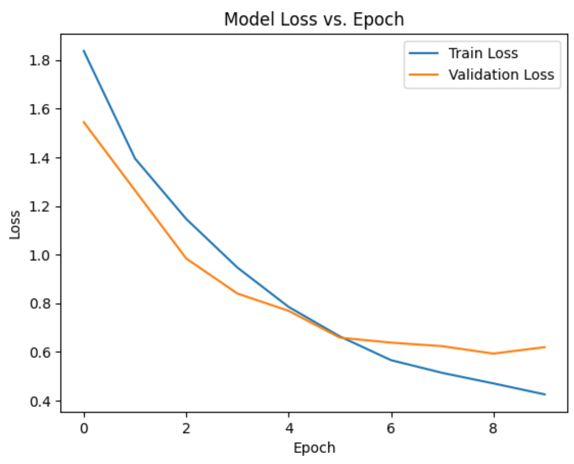


Fig. 4.33 Loss vs. epoch plot of ResNet based AMC with dropout rate 0.3 for 6 dB SNR in 10 epochs

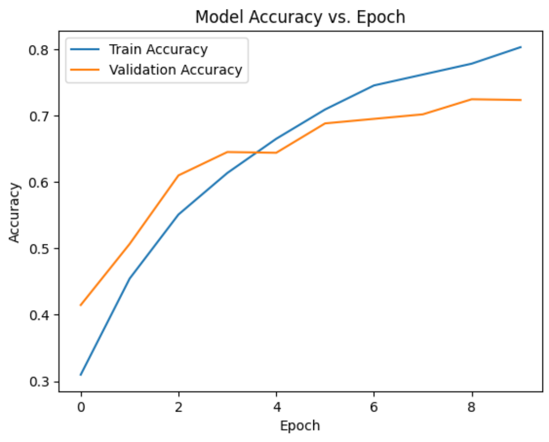


Fig. 4.34 Accuracy vs. epoch plot of ResNet based AMC with dropout rate 0.3 for 6 dB SNR in 10 epochs

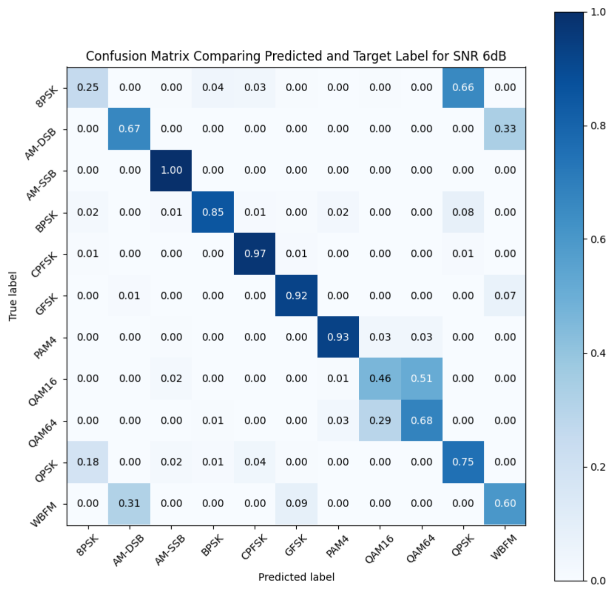


Fig. 4.35 Confusion matrix of ResNet based AMC with dropout rate 0.3 for 6 dB SNR in 10 epochs

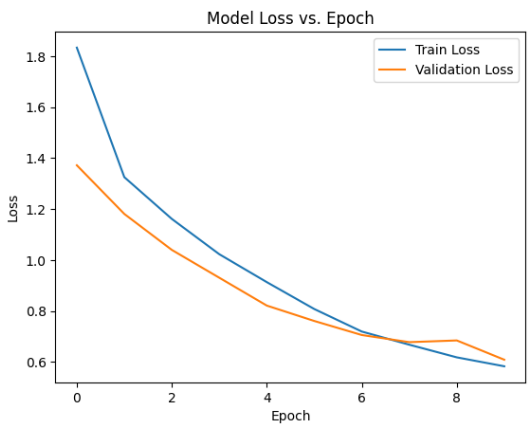


Fig. 4.36 Loss vs. epoch plot of ResNet based AMC with dropout rate 0.3 for 16 dB SNR in 10 epochs

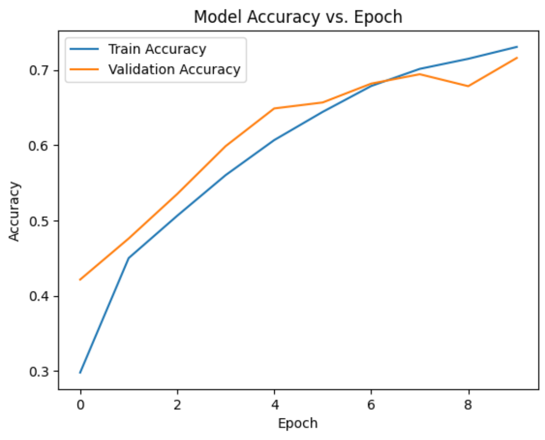


Fig. 4.37 Accuracy vs. epoch plot of ResNet based AMC with dropout rate 0.3 for 16 dB SNR in 10 epochs

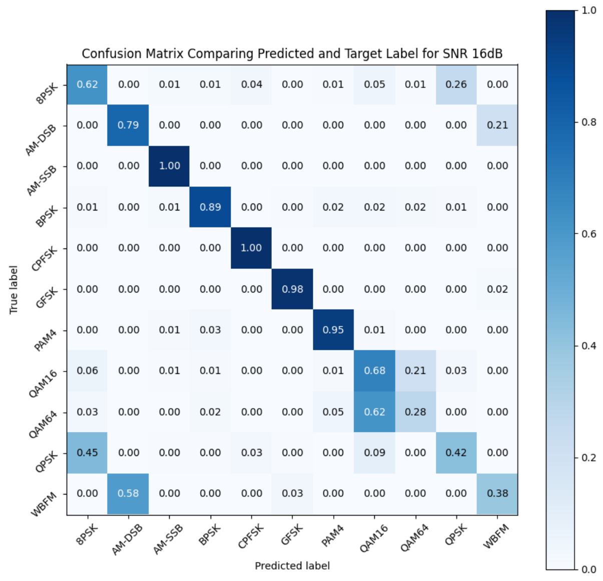


Fig. 4.38 Confusion matrix of ResNet based AMC with dropout rate 0.3 for 16 dB SNR in 10 epochs

4.3.4 Comparison between ResNet based AMC with dropout rate 0.6 & 0.3

The graph shown in figure 4.39 compares the performance of ResNet based AMC with dropout rate 0.6 (paper model [26]) and with dropout rate 0.3 (proposed model). The model with dropout rate 0.3 (proposed model) shows **4.575% better** performance than the model with dropout rate 0.6 (paper model).

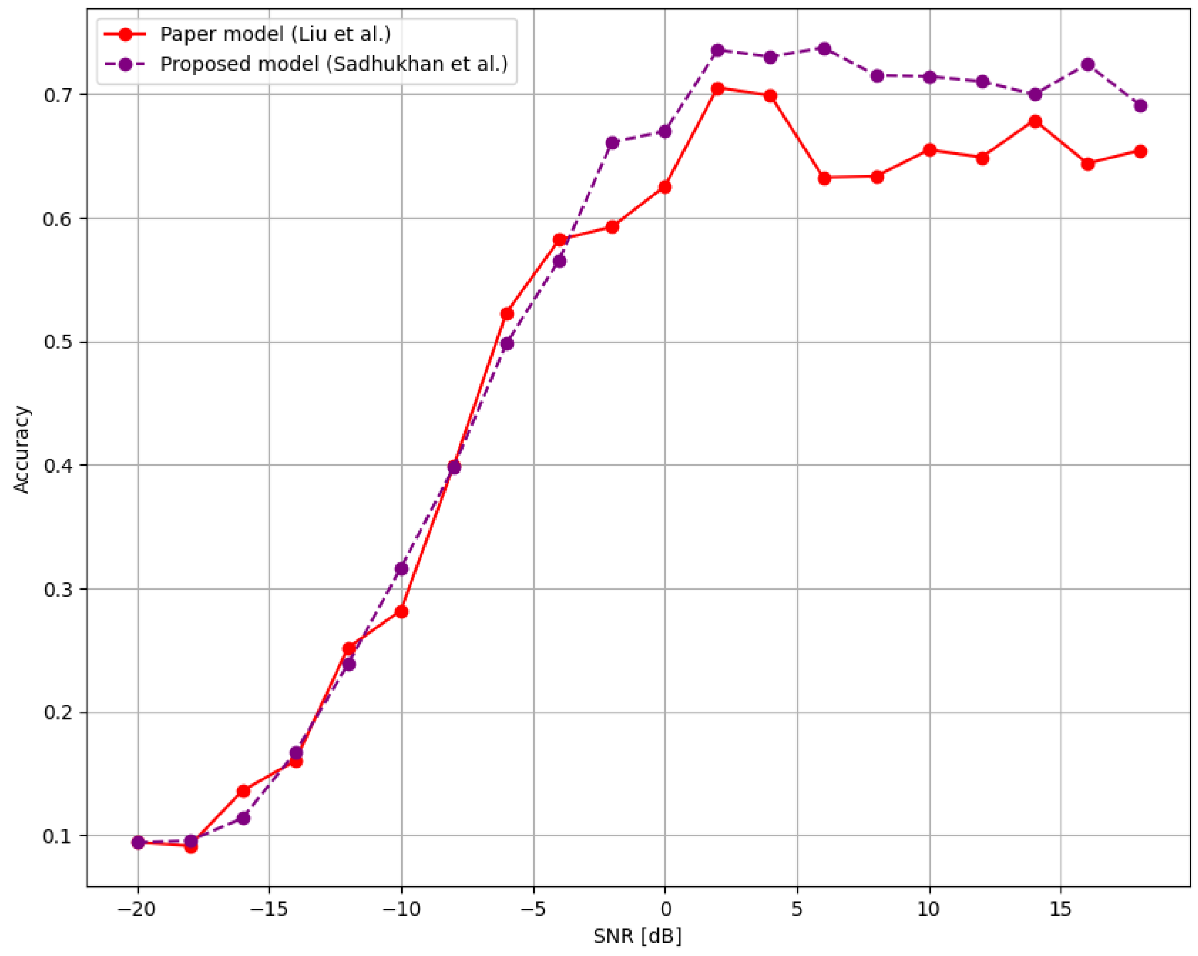


Fig. 4.39 Accuracy vs. SNR plot comparing ResNet based AMC with dropout rate 0.6 & 0.3 in 10 epochs

4.4 Summary

Vanishing or exploding gradients and accuracy loss after a given network depth have been issues for CNN. When deep neural networks are trained, a phenomenon known as the ‘vanishing gradient problem’ takes place in which the gradients that are used to update the network shrink dramatically or ‘vanish’ as they are backpropagated from the output layers to the previous layers. Significant error gradients that build and cause very significant modifications to the weights of neural network models during training are known as ‘exploding gradients’. The model becomes unstable and unable to gain knowledge from the training set in such cases. To overcome these limitations of CNN, we have introduced a new type of DL model for our problem namely, Residual Network (ResNet) model. ResNet improves feature propagation in the neural network by establishing shortcuts between its various layers. We have added a shortcut path between the input layer and third convolution layer as can be seen from figure 4.3 and figure 4.21. The paper model in [26] used 0.6 as dropout rate. We have tuned this hyperparameter to 0.3 and observed almost 4.6% better performance. Moreover, the proposed ResNet model shows an accuracy of 73.773% at SNR 6 dB which is 7.635% better compared to the CNN model using 4 convolution layers with batch normalization. However, the increase in accuracy has led to an increase in training time as well. It takes almost 8 hours 32 minutes to train a ResNet model with dropout rate 0.3 and 6.5 hours for a ResNet model with dropout rate 0.6.

CHAPTER 5

Convolutional Long Short-Term Deep Neural Network based Automatic Modulation Classification

5.1 Convolutional Long Short-Term Deep Neural Network

5.1.1 Overview

In [27], a Convolutional Long Short-Term Deep Neural Network (CLDNN) is introduced. Using the complementary properties of CNNs, LSTMs, and DNNs, it combines the LSTM and CNN designs into a Deep Neural Network. In a wide variety of speech recognition applications, Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) have both demonstrated appreciable advancements over Deep Neural Networks (DNNs). In terms of modelling abilities, CNNs, LSTMs, and DNNs are complementing since CNNs are good at reducing frequency variations, LSTMs are good at temporal modelling, and DNNs are suitable for mapping features to a more separable space.[30]

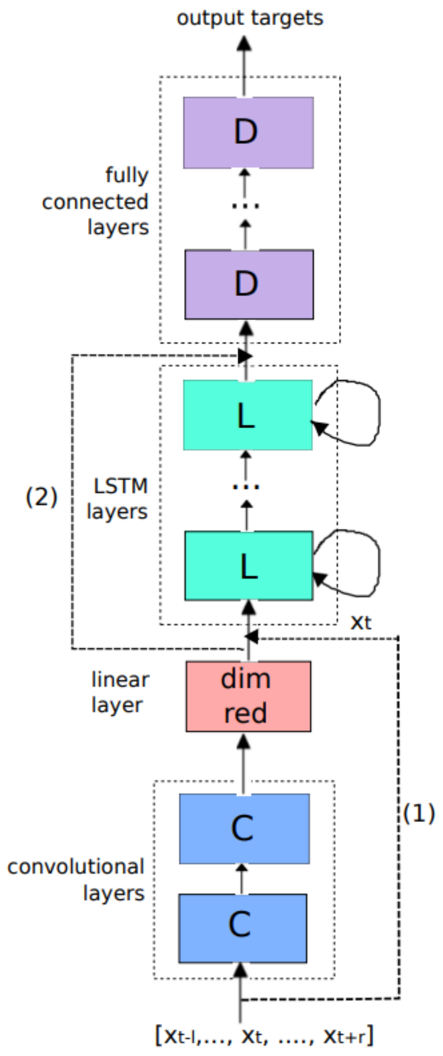


Fig 5.1 CLDNN architecture [30]

5.1.2 Long Short-Term Memory (LSTM)

Recurrent neural network (RNN) architectures of the Long Short-Term Memory (LSTM) type are frequently employed in deep learning. It excels at capturing long-term dependencies, which makes it the best choice for tasks requiring sequence prediction.

Since LSTM includes feedback connections as opposed to standard neural networks, it can handle complete data sequences as opposed to just single data points. Because of this, it excels at identifying and forecasting patterns in sequential data, including time series, text, and voice. By extracting important insights from sequential data, LSTM has developed into a powerful tool in artificial intelligence and deep learning, leading to advancements in numerous domains.

5.1.3 Architecture of Long Short-Term Memory (LSTM)

According to figure 5.2, the LSTM network architecture is composed of three sections, each of which has a distinct role.

The first component decides if the data from the previous timestamp should be remembered or if it is unnecessary and can be forgotten. The cell attempts to learn new information from the input that is provided to it in the second half. The cell then transmits the revised data from the present timestamp to the following timestamp in the third section. One LSTM cycle is regarded as one single-time step.

These three LSTM unit components are referred to as gates. They control how data enters and exits the memory cell (LSTM cell). Forget gate, Input gate, and Output gate are the names of the first, second, and third gates, respectively. A layer of neurons in a typical feedforward neural network can be compared to an LSTM unit, which consists of these three gates and a memory cell (LSTM cell), with each neuron having a hidden layer and a current state.

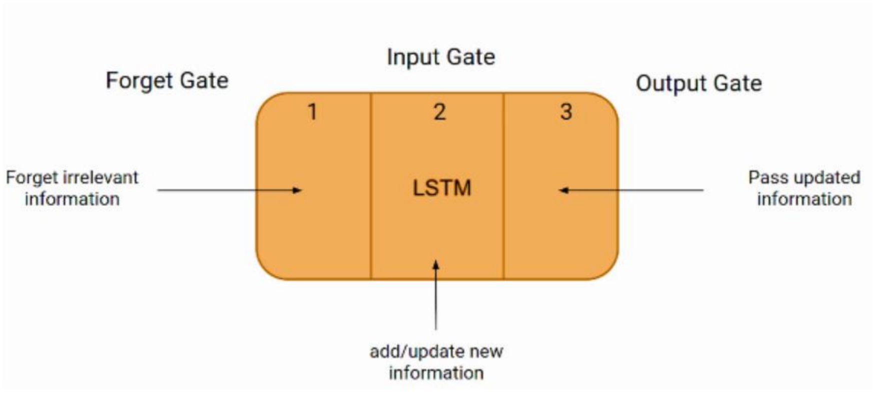


Fig. 5.2 Architecture of LSTM

An LSTM has a hidden state, just like a straightforward RNN, with Ht-1 standing for the hidden state of the prior timestamp and Ht for the hidden state of the present timestamp. Additionally, the LSTM has a cell state that is represented by the timestamps Ct-1 for the past and Ct for the present.

In this case, the cell state is referred to as Long term memory, while the hidden state is referred to as Short term memory, as shown in figure 5.3.

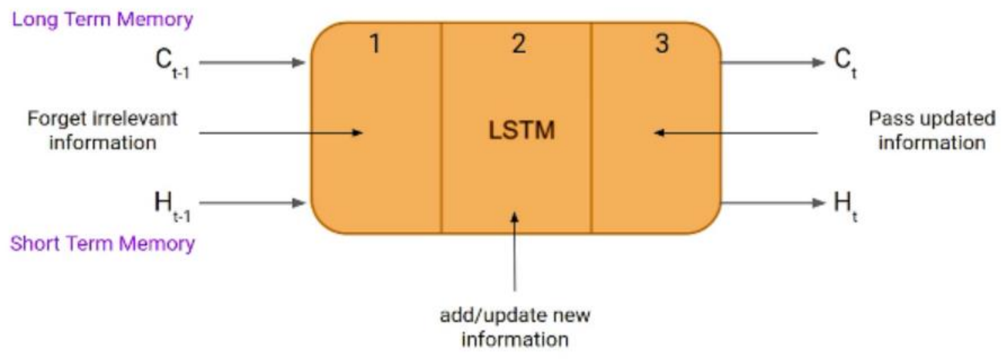


Fig. 5.3 Long-Term and Short-Term Memory in LSTM

5.1.4 Summary

The Long Short-Term Memory (LSTM) architecture employs CNNs to reduce the input feature's spectral variation before passing it to LSTM layers for temporal modelling. Finally, this output is generated to Deep Neural Network (DNN) layers, which creates a feature representation that is simple to separate.

5.2 CLDNN based Automatic Modulation Classification

5.2.1 Model Specifications

The model specifications are shown in table 5.1. This model is inspired by the work done in [26]. The dropout rate has been taken to be 0.6 and Adam optimizer has been used and the batch size was taken to be 128 in the original work.

Table 5.1 Model specifications of CLDNN based AMC

Layer (type)	**Output Shape**	**Parameter**
conv2d (Conv2D)	(None, 2, 128, 256)	1024
max_pooling2D (Maxpooling2D)	(None, 2, 64, 256)	0
dropout (Dropout)	(None, 2, 64, 256)	0
conv2d_1 (Conv2D)	(None, 2, 64, 256)	393472
max_pooling2d_1 (Maxpooling2D)	(None, 2, 32, 256)	0
dropout_1 (Dropout)	(None, 2, 32, 256)	0
conv2d_2 (Conv2D)	(None, 2, 32, 80)	61520
max_pooling2d_2 (Maxpooling2D)	(None, 2, 16, 80)	0
dropout_2 (Dropout)	(None, 2, 16, 80)	0
conv2d_3 (Conv2D)	(None, 2, 16, 80)	19280
max_pooling2d_3 (Maxpooling2D)	(None, 2, 8, 80)	0
dropout_3 (Dropout)	(None, 2, 8, 80)	0
reshape (Reshape)	(None, 2, 640)	0
lstm (LSTM)	(None, 50)	138200
dropout_4 (Dropout)	(None, 50)	0
dense (Dense)	(None, 128)	6528
dropout_5 (Dense)	(None, 128)	0
dense_1 (Dense)	(None, 11)	1419

Total number of parameters are 621,443. Total number of trainable parameters are 621,443. The number of non-trainable parameters are 0.

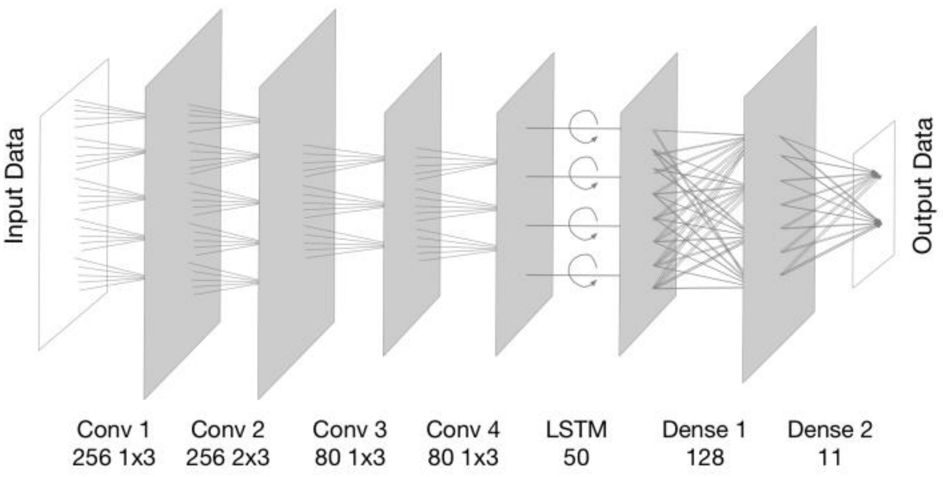


Fig. 5.4 Architecture of eight-layer CLDNN [26]

5.2.2 Pseudocode

The code for the model function or definition is as shown:

```
def CLDNN():

   model = Sequential()

   model.add(Conv2D(256, (1, 3), activation='relu', padding='same', input_shape=(2, 128,
1)))
   model.add(MaxPooling2D(pool_size=(1, 2), padding='valid',  data_format=None))
   model.add(layers.Dropout(0.6))

   model.add(Conv2D(256, (2, 3), activation='relu', padding='same'))
   model.add(MaxPooling2D(pool_size=(1, 2), padding='valid',  data_format=None))
   model.add(layers.Dropout(0.6))

   model.add(Conv2D(80, (1, 3), activation='relu', padding='same'))
   model.add(MaxPooling2D(pool_size=(1, 2), padding='valid',  data_format=None))
   model.add(layers.Dropout(0.6))

   model.add(Conv2D(80, (1, 3), activation='relu', padding='same'))
   model.add(MaxPooling2D(pool_size=(1, 2), padding='valid',  data_format=None))
   model.add(layers.Dropout(0.6))

   model.add(Reshape((2, 640)))

   model.add(LSTM(50, activation='relu'))
   model.add(layers.Dropout(0.6))

   model.add(Dense(128, activation='relu'))
   model.add(layers.Dropout(0.6))

   model.add(Dense(11, activation='softmax'))
   model.summary()

   return model
```

5.2.3 Results and Analysis

We have run our model for four different epoch values: 1, 10, 15 and 20. The performance of the model in 1 & 10 epoch value is poor for obvious reasons. We have considered the model performance with epoch value 15 as it gives comparatively better results.

Epoch = 1 :

The model performance for 1 epoch is as shown in figure 5.5 to 5.8. The training duration is 30 minutes in this case.

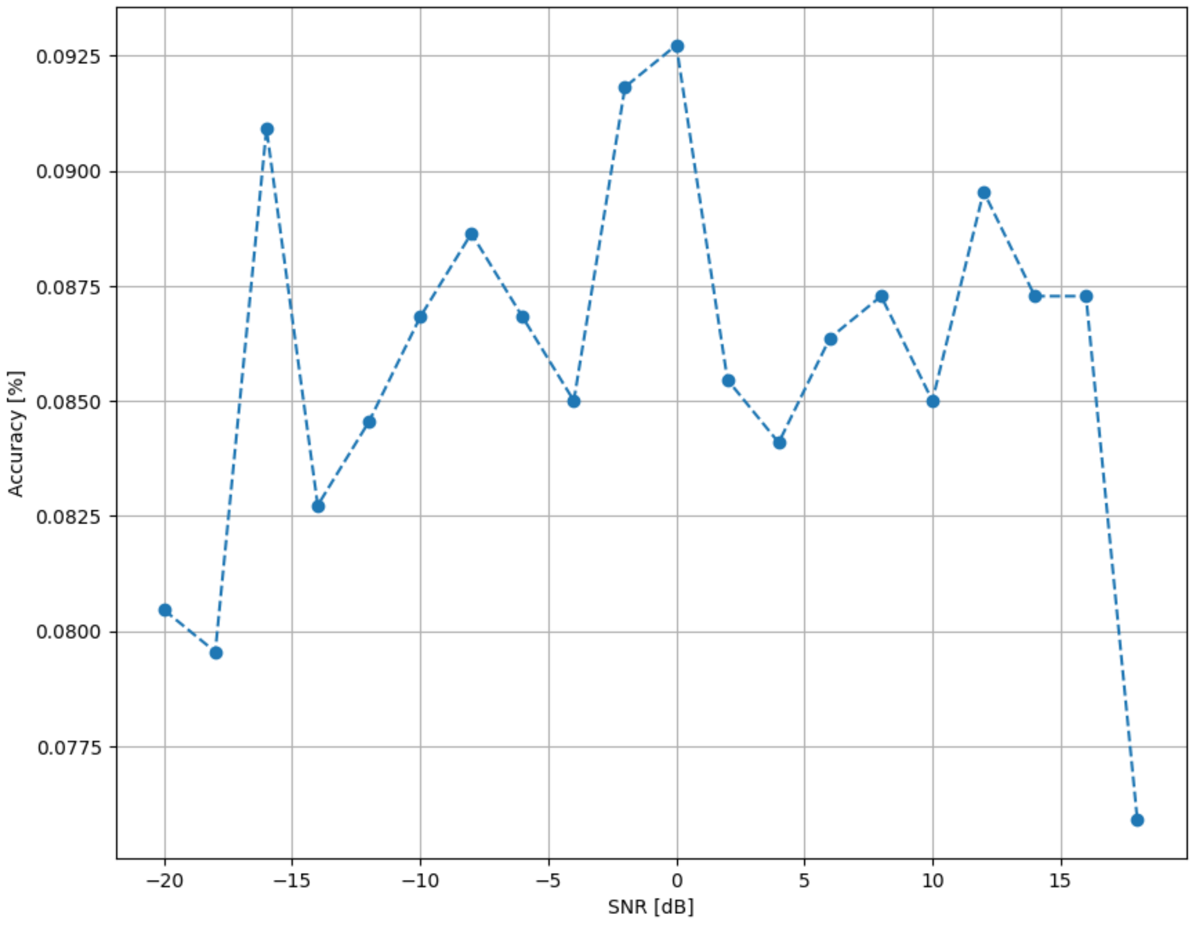


Fig. 5.5 Accuracy vs. SNR plot of CLDNN based AMC in 1 epoch

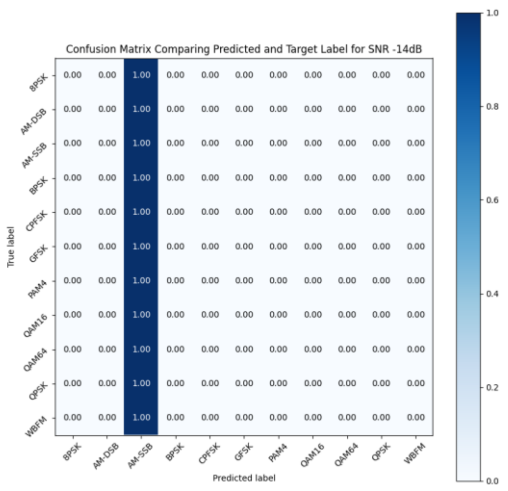


Fig. 5.6 Confusion matrix of CLDNN based AMC for -14 dB SNR in 1 epoch

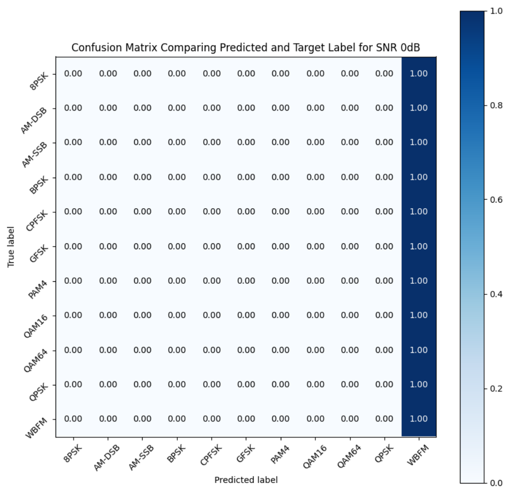


Fig. 5.7 Confusion matrix of CLDNN based AMC for 0 dB SNR in 1 epoch

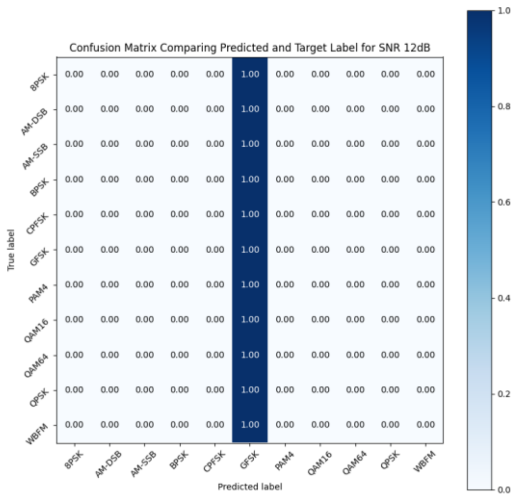


Fig. 5.8 Confusion matrix of CLDNN based AMC for 12 dB SNR in 1 epoch

Epoch = 10 :

The model performance for 10 epoch is as shown in figure 5.9 to 5.12. The training duration is 3 hours 30 minutes in this case.

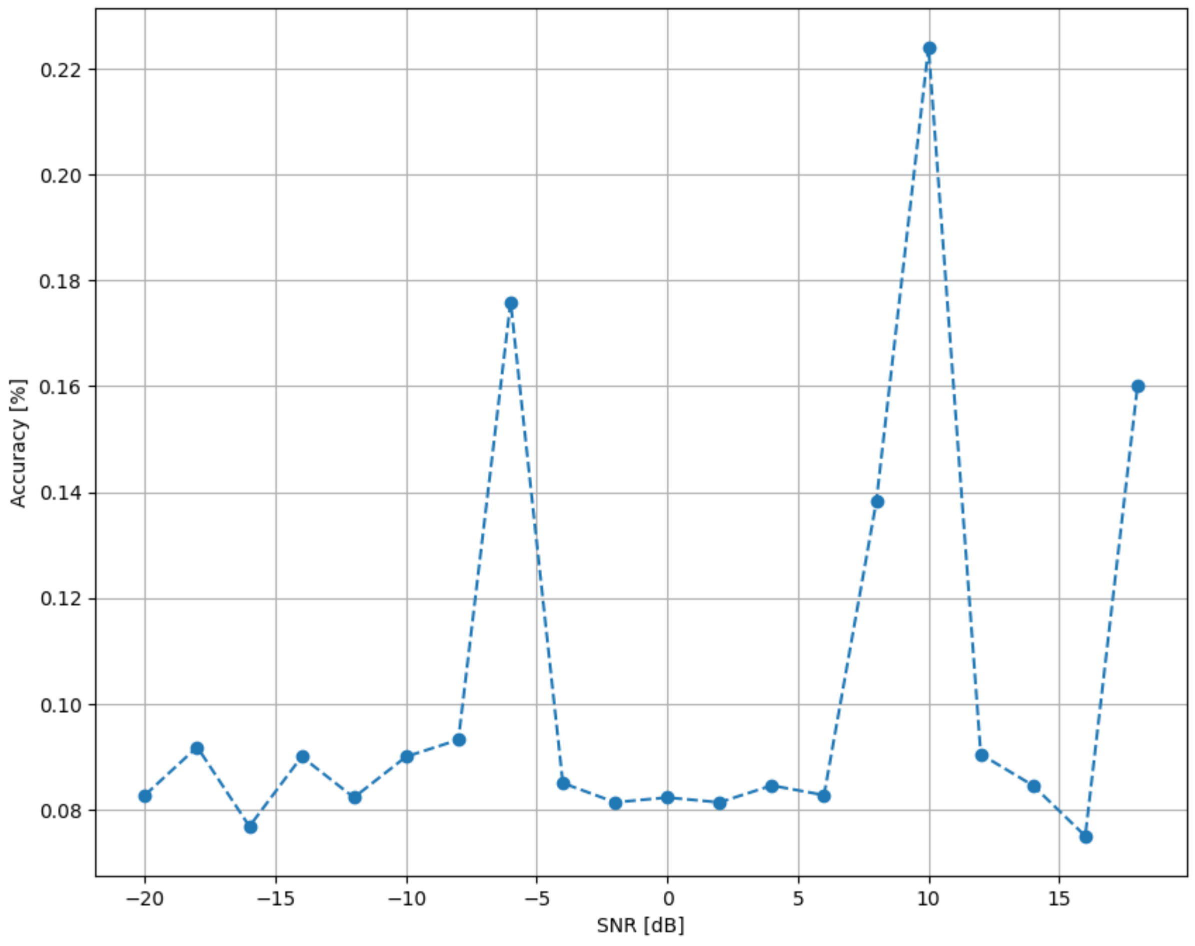


Fig. 5.9 Accuracy vs. SNR plot of CLDNN based AMC in 10 epochs

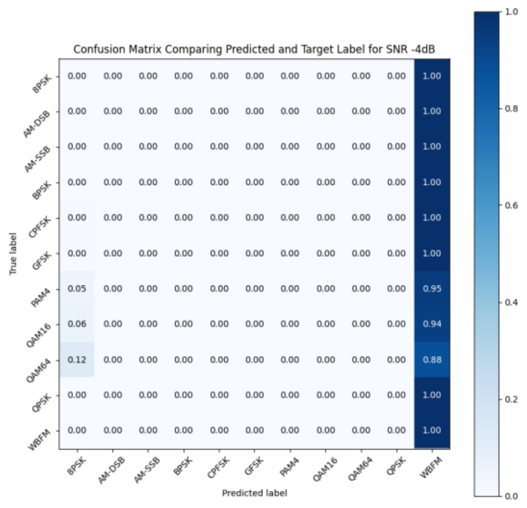


Fig. 5.10 Confusion matrix of CLDNN based AMC for -4 dB SNR in 10 epochs

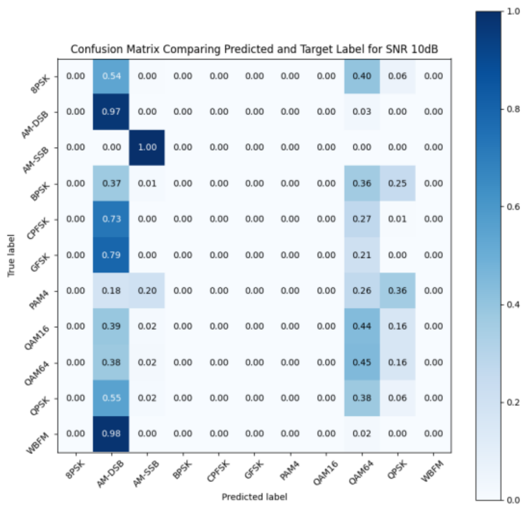


Fig. 5.11 Confusion matrix of CLDNN based AMC for 10 dB SNR in 10 epochs

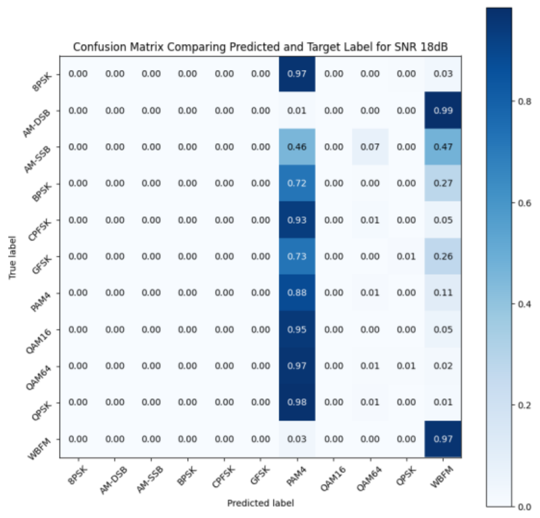


Fig. 5.12 Confusion matrix of CLDNN based AMC for 18 dB SNR in 10 epochs

Epoch = 15 :

The model performance for 15 epoch is as shown in figure 5.13 to 5.25. The model shows best results in SNR 10 dB with an accuracy of 27.182%. The confusion matrix corresponding to 10 dB SNR is shown in figure 5.22. The training duration is 6 hours.

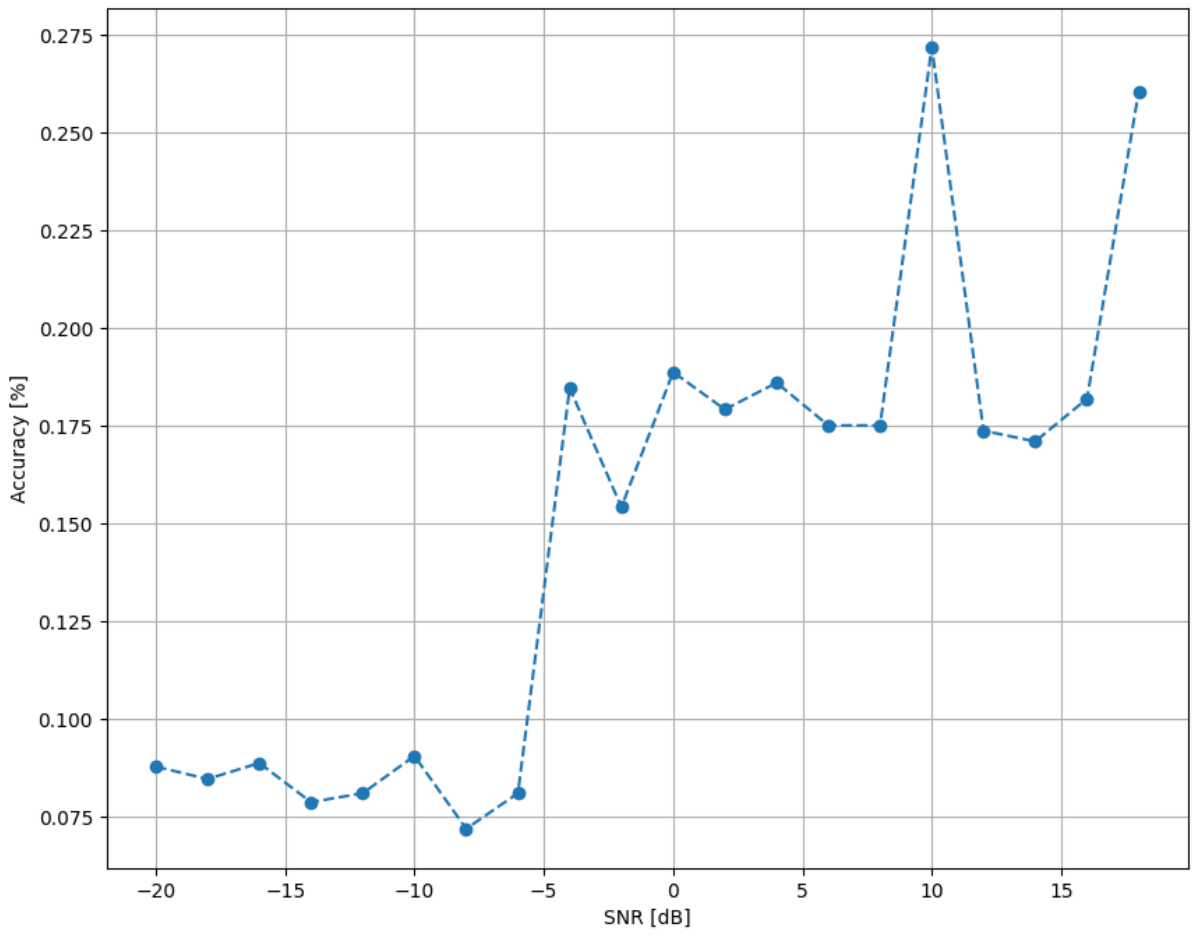


Fig. 5.13 Accuracy vs. SNR plot for CLDNN based AMC in 15 epochs

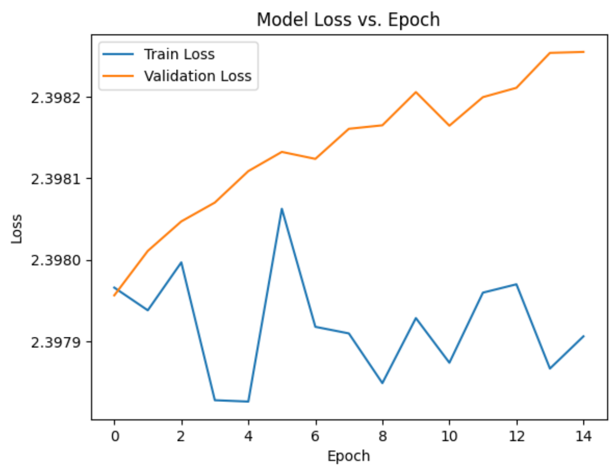


Fig. 5.14 Loss vs. epoch plot of CLDNN based AMC for -10 dB SNR in 15 epochs

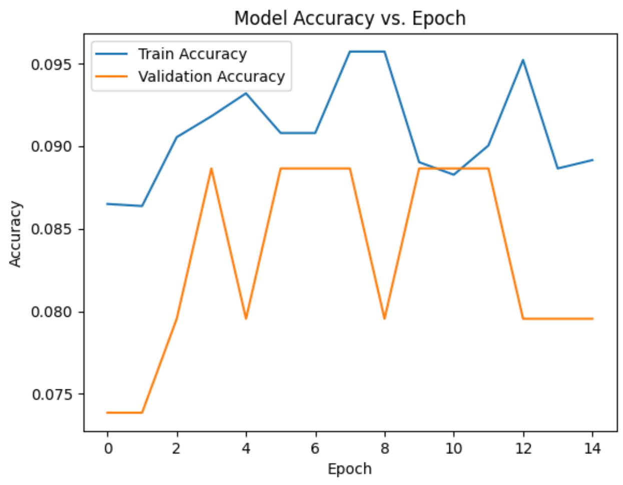


Fig. 5.15 Accuracy vs. epoch plot of CLDNN based AMC for -10 dB SNR in 15 epochs

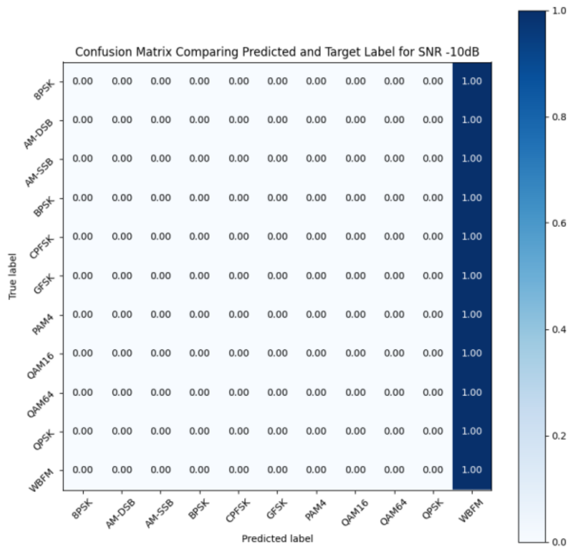


Fig. 5.16 Confusion matrix of CLDNN based AMC for -10 dB SNR in 15 epochs

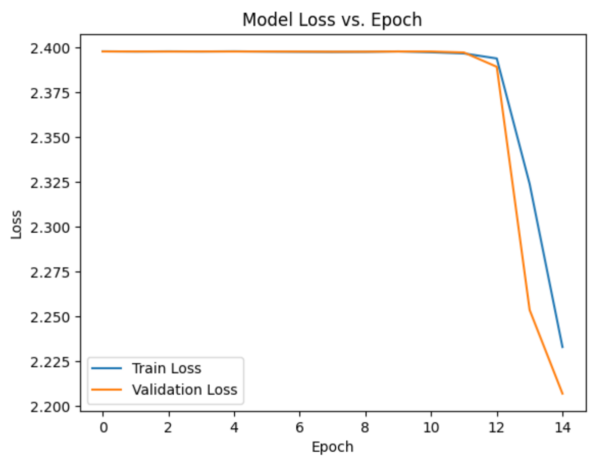


Fig. 5.17 Loss vs. epoch plot of CLDNN based AMC for 0 dB SNR in 15 epochs

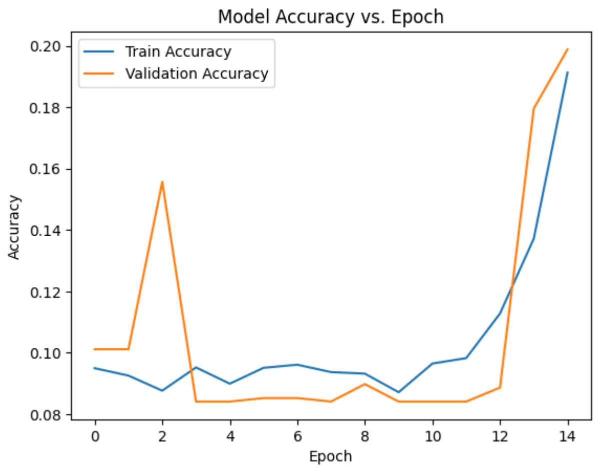


Fig. 5.18 Accuracy vs. epoch plot of CLDNN based AMC for 0 dB SNR in 15 epochs

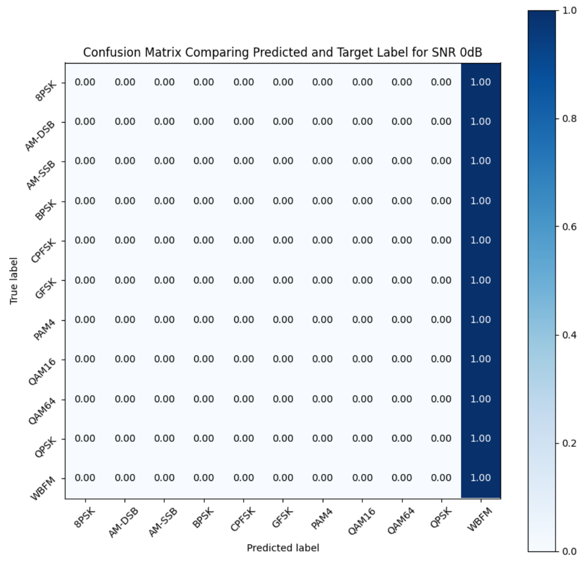


Fig. 5.19 Confusion matrix of CLDNN based AMC for 0 dB SNR in 15 epochs

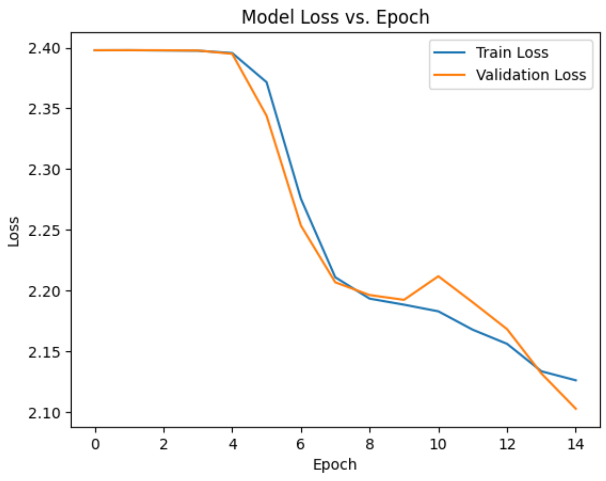


Fig. 5.20 Loss vs. epoch plot of CLDNN based AMC for 10 dB SNR in 15 epochs

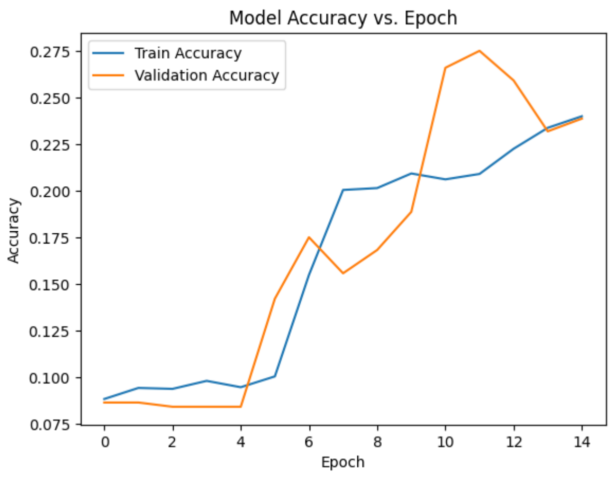


Fig. 5.21 Accuracy vs. epoch plot of CLDNN based AMC for 10 dB SNR in 15 epochs

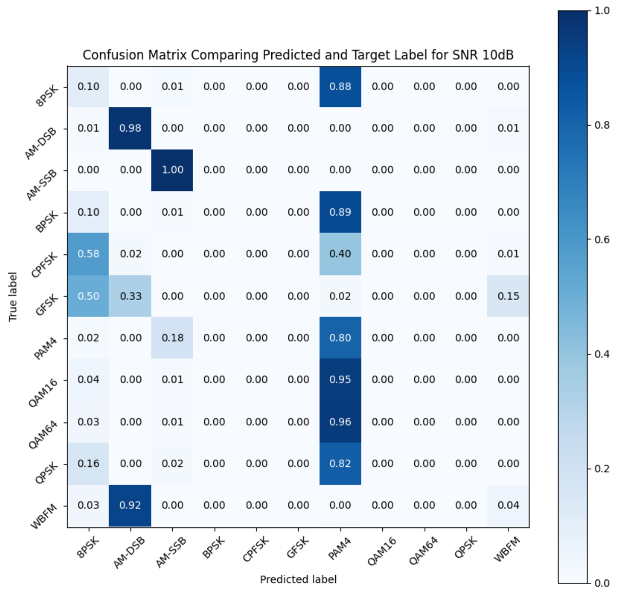


Fig. 5.22 Confusion matrix of CLDNN based AMC for 10 dB SNR in 15 epochs

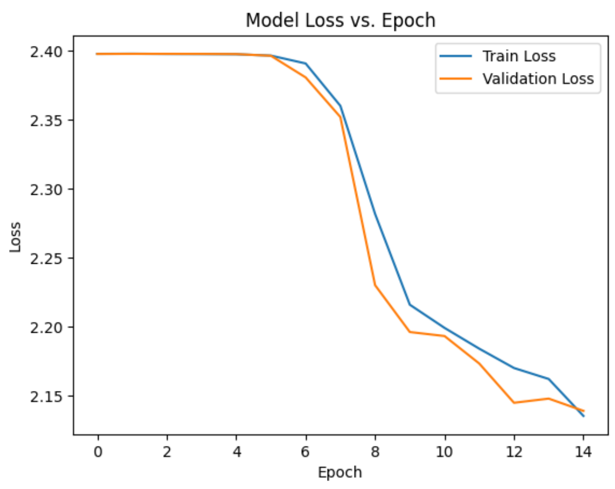


Fig. 5.23 Loss vs. epoch plot of CLDNN based AMC for 18 dB SNR in 15 epochs

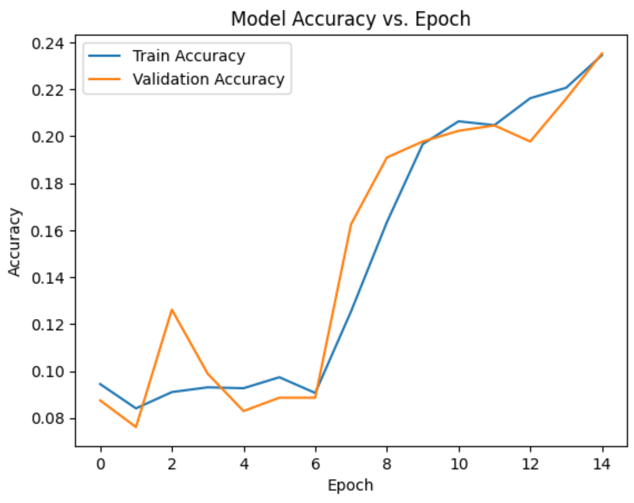


Fig. 5.24 Accuracy vs. epoch plot of CLDNN based AMC for 18 dB SNR in 15 epochs

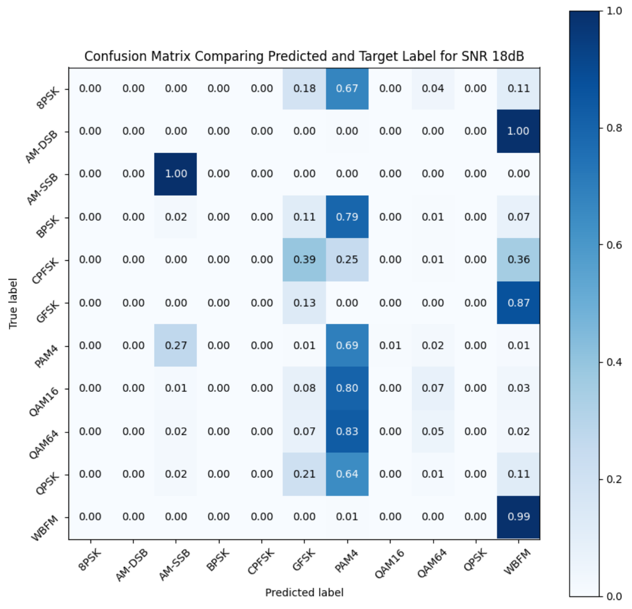


Fig. 5.25 Confusion matrix of CLDNN based AMC for 18 dB SNR in 15 epochs

Epoch = 20 :

The model performance for 20 epoch is as shown in figure 5.26 to 5.38. The model shows best results in SNR 0 dB with an accuracy of 28.409%. The training duration is 6 hours 30 minutes.

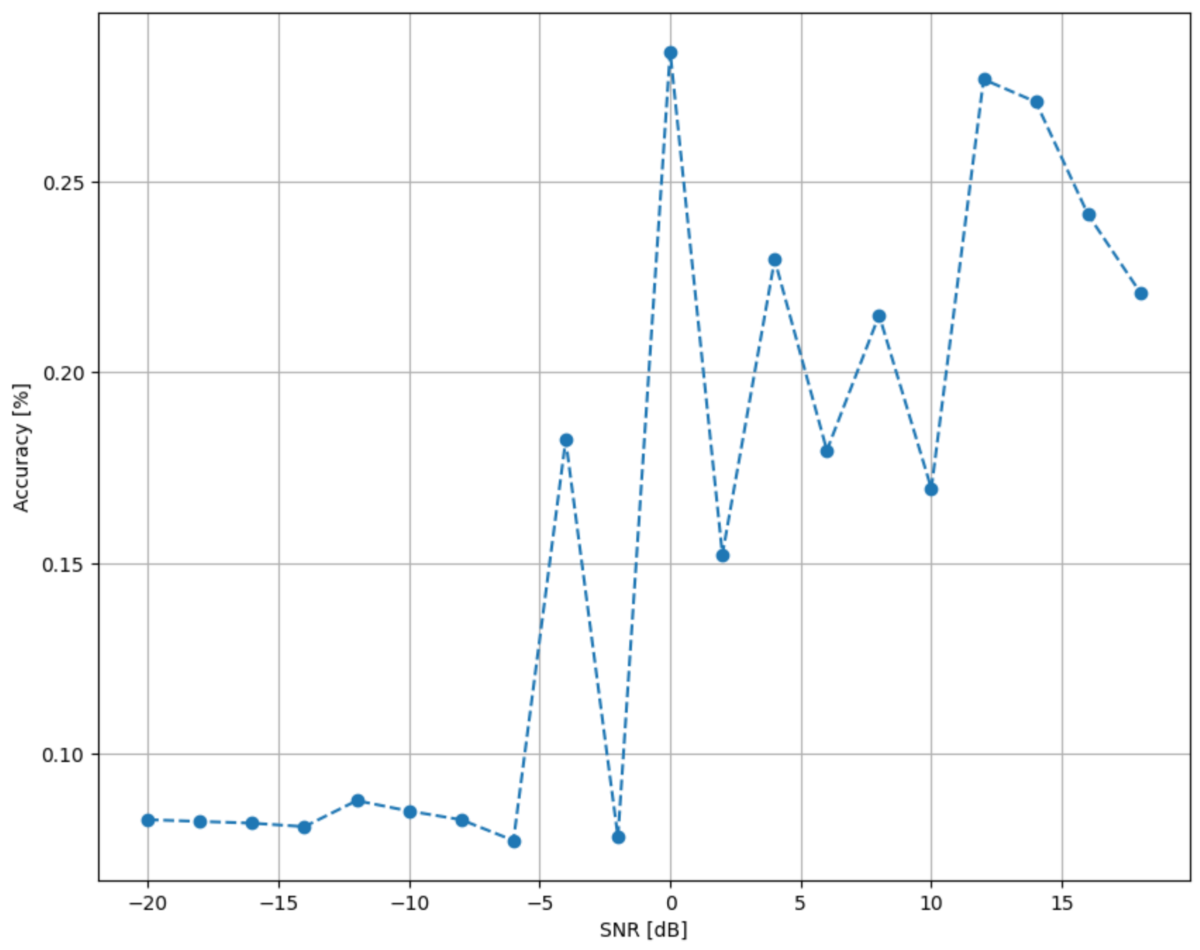


Fig. 5.26 Accuracy vs. SNR plot of CLDNN based AMC in 20 epochs

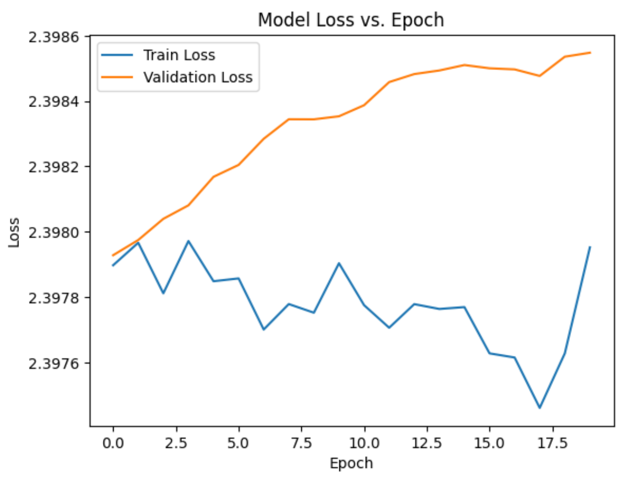


Fig. 5.27 Loss vs. epoch plot of CLDNN based AMC for -10 dB SNR in 20 epochs

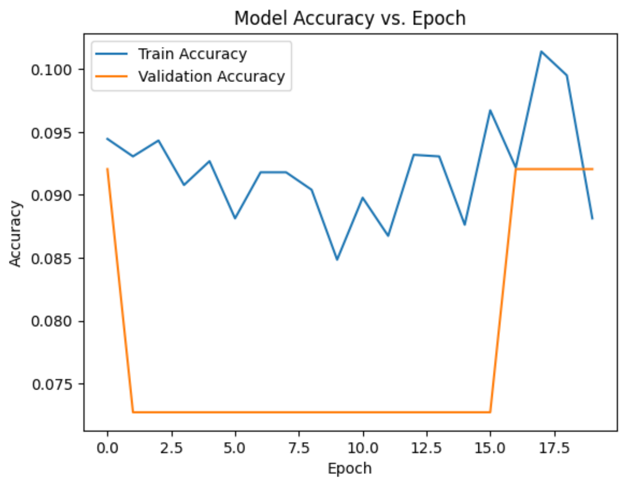


Fig. 5.28 Accuracy vs. epoch plot of CLDNN based AMC for -10 dB SNR in 20 epochs

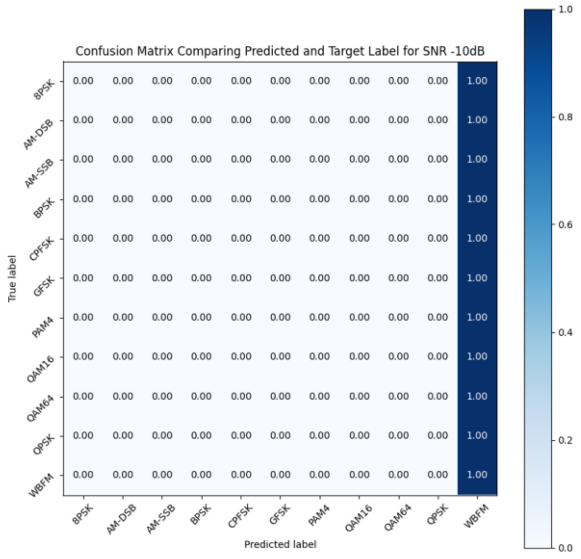


Fig. 5.29 Confusion matrix of CLDNN based AMC for -10 dB SNR in 20 epochs

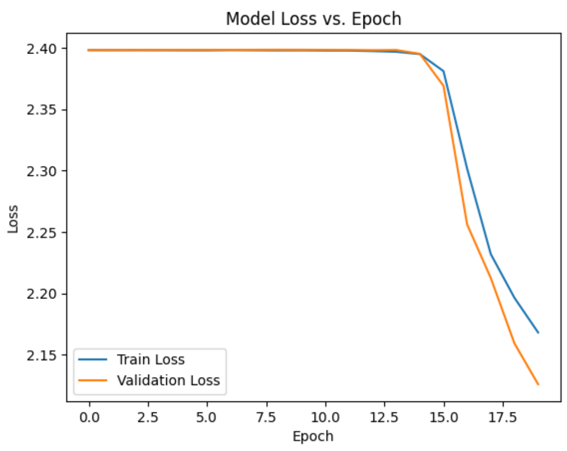


Fig. 5.30 Loss vs. epoch plot of CLDNN based AMC for -4 dB SNR in 20 epochs

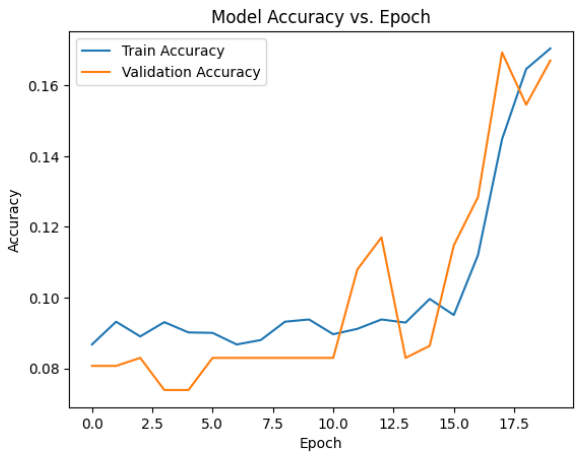


Fig. 5.31 Accuracy vs. epoch plot of CLDNN based AMC for -4 dB SNR in 20 epochs

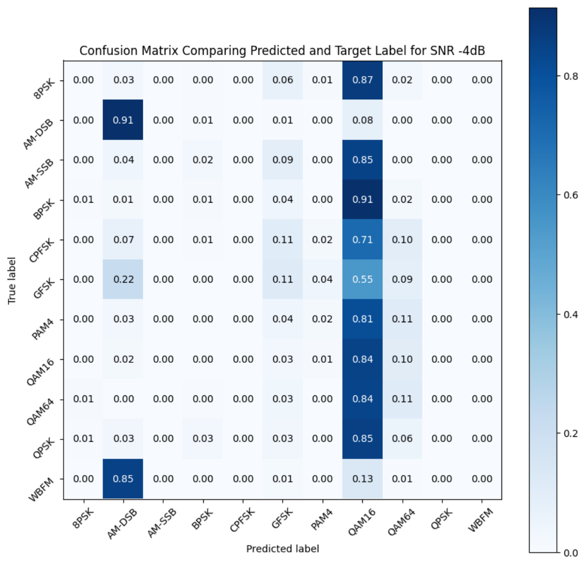


Fig. 5.32 Confusion matrix of CLDNN based AMC for -4 dB SNR in 20 epochs

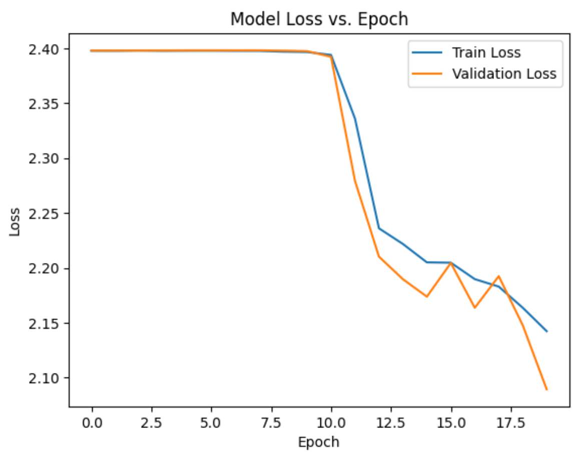


Fig. 5.33 Loss vs. epoch plot of CLDNN based AMC for 0 dB SNR in 20 epochs

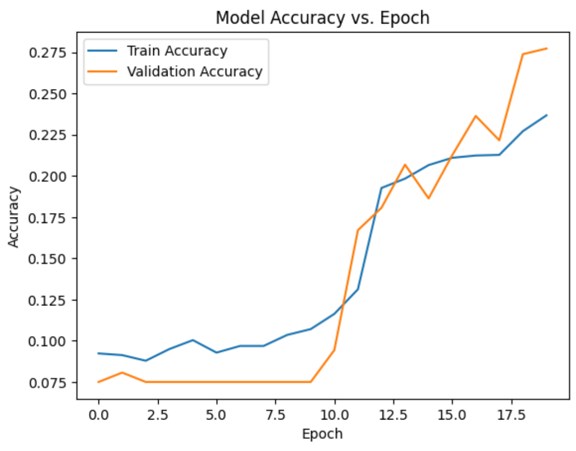


Fig. 5.34 Accuracy vs. epoch plot of CLDNN based AMC for 0 dB SNR in 20 epochs

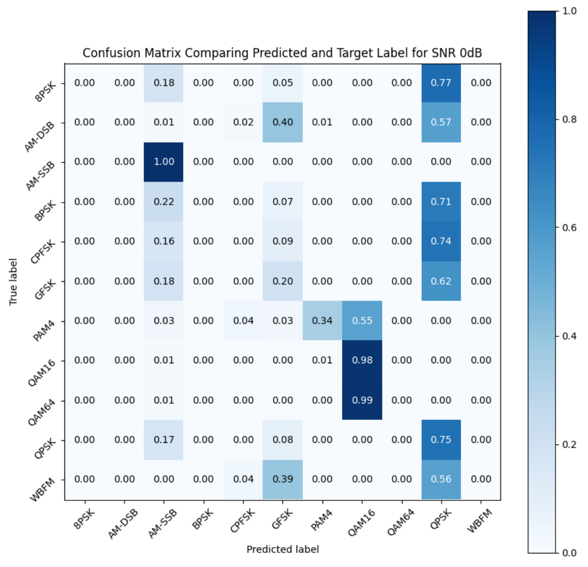


Fig. 5.35 Confusion matrix of CLDNN based AMC for 0 dB SNR in 20 epochs

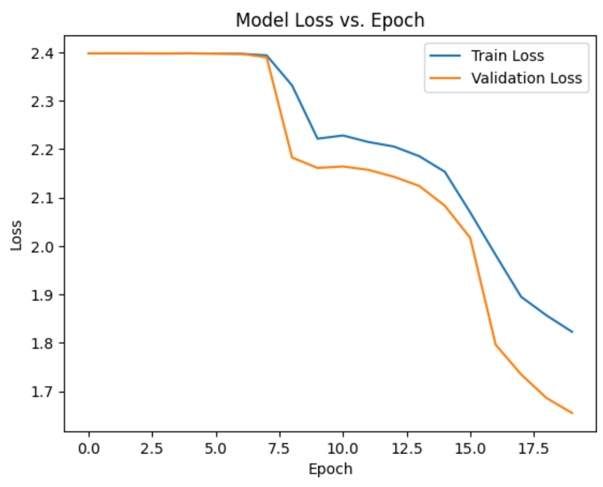


Fig. 5.36 Loss vs. epoch plot of CLDNN based AMC for 12 dB SNR in 20 epochs

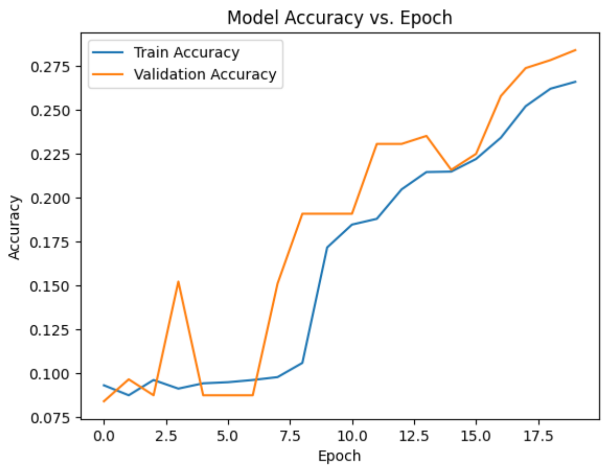


Fig. 5.37 Accuracy vs. epoch plot of CLDNN based AMC for 12 dB SNR in 20 epochs

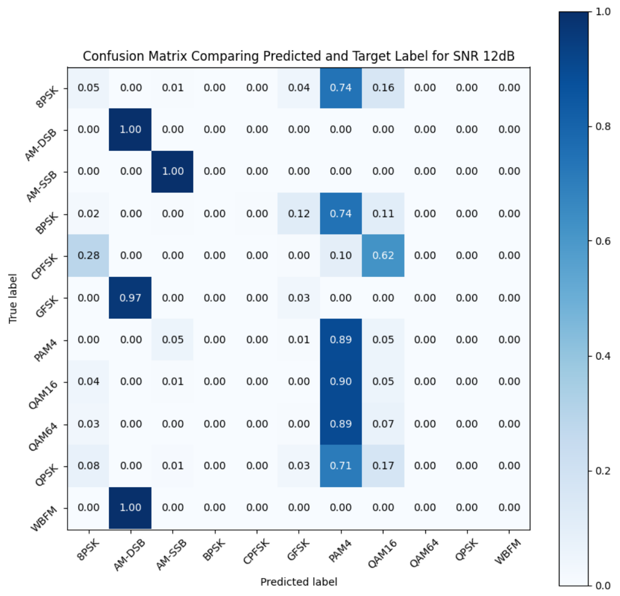


Fig. 5.38 Confusion matrix of CLDNN based AMC for 12 dB SNR in 20 epochs

5.3 Modification in CLDNN based Automatic Modulation Classification

5.3.1 Model Specifications

The batch size determines how many samples will be sent via the network at once. A mini-batch is another name for a batch.

Simply explained, an epoch is one single run through the whole training set to the network, whereas a batch size is the number of samples that are delivered to the network at once.

One of the hyperparameters of a deep neural network model is batch size. We significantly increased the model accuracy after tuning this hyperparameter to the value 32.

The model specifications of the proposed model are shown in table 5.2.

Table 5.2 Model specifications of CLDNN based AMC with batch size 32

Layer (type)	**Output Shape**	**Parameter**
conv2d (Conv2D)	(None, 2, 128, 256)	1024
max_pooling2D (Maxpooling2D)	(None, 2, 64, 256)	0
dropout (Dropout)	(None, 2, 64, 256)	0
conv2d_1 (Conv2D)	(None, 2, 64, 256)	393472
max_pooling2d_1 (Maxpooling2D)	(None, 2, 32, 256)	0
dropout_1 (Dropout)	(None, 2, 32, 256)	0
conv2d_2 (Conv2D)	(None, 2, 32, 80)	61520
max_pooling2d_2 (Maxpooling2D)	(None, 2, 16, 80)	0
dropout_2 (Dropout)	(None, 2, 16, 80)	0
conv2d_3 (Conv2D)	(None, 2, 16, 80)	19280
max_pooling2d_3 (Maxpooling2D)	(None, 2, 8, 80)	0
dropout_3 (Dropout)	(None, 2, 8, 80)	0
reshape (Reshape)	(None, 2, 640)	0
lstm (LSTM)	(None, 50)	138200
dropout_4 (Dropout)	(None, 50)	0
dense (Dense)	(None, 128)	6528

dropout_5 (Dense)	(None, 128)	0
dense_1 (Dense)	(None, 11)	1419

Total number of parameters are 621,443. Total number of trainable parameters are 621,443. The number of non-trainable parameters are 0.

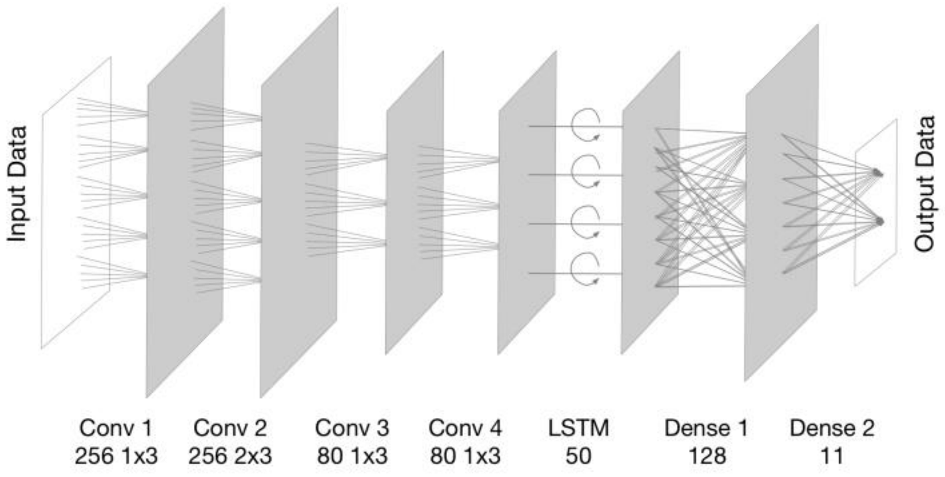


Fig. 5.39 Architecture of eight-layer CLDNN with batch size 32

5.3.2 Pseudocode

The code for the model function or definition is as shown:

```
def CLDNN():

    model = Sequential()

    model.add(Conv2D(256, (1, 3), activation='relu', padding='same', input_shape=(2, 128,
1)))
    model.add(MaxPooling2D(pool_size=(1, 2), padding='valid',  data_format=None))
    model.add(layers.Dropout(0.6))

    model.add(Conv2D(256, (2, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(1, 2), padding='valid',  data_format=None))
    model.add(layers.Dropout(0.6))

    model.add(Conv2D(80, (1, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(1, 2), padding='valid',  data_format=None))
    model.add(layers.Dropout(0.6))

    model.add(Conv2D(80, (1, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(1, 2), padding='valid',  data_format=None))
    model.add(layers.Dropout(0.6))

    model.add(Reshape((2, 640)))

    model.add(LSTM(50, activation='relu'))
    model.add(layers.Dropout(0.6))

    model.add(Dense(128, activation='relu'))
    model.add(layers.Dropout(0.6))

    model.add(Dense(11, activation='softmax'))
    model.summary()

    return model
```

5.3.3 Results and Analysis

We have run our proposed model in two different epoch values: 1 and 15. The performance of the model in 1 epoch value is poor for obvious reasons. We have considered the model performance with epoch value 15 as it gives comparatively better results.

Epoch = 1 :

The model performance for 1 epoch is as shown in figure 5.40 to 5.43. The training duration is 30 minutes in this case.

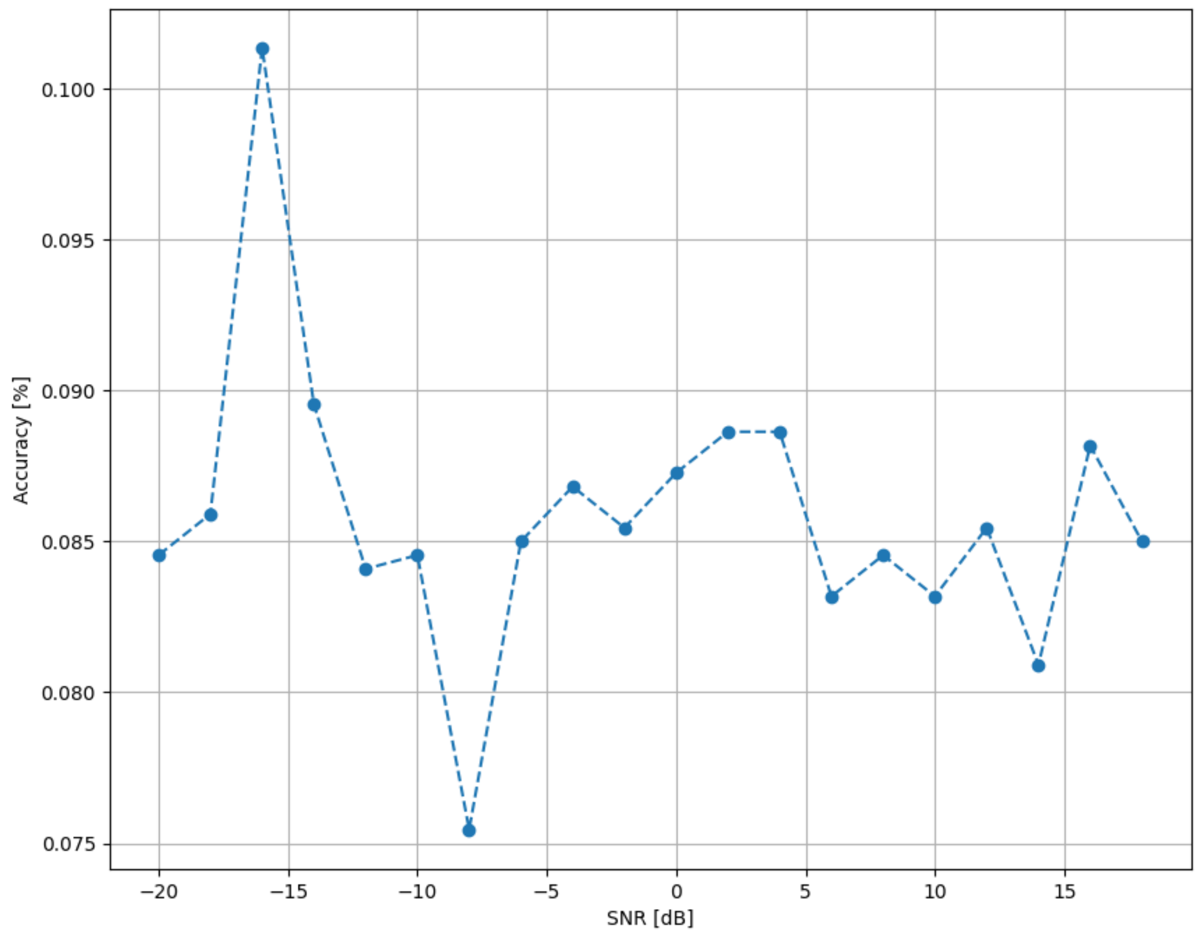


Fig. 5.40 Accuracy vs. SNR plot of CLDNN based AMC with batch size 32 in 1 epoch

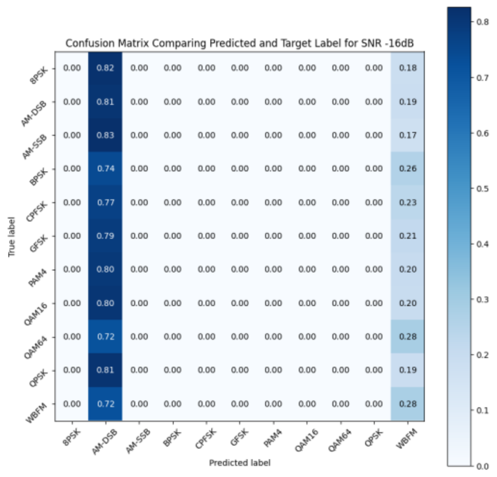


Fig. 5.41 Confusion matrix of CLDNN based AMC with batch size 32 for -16 dB SNR in 1 epoch

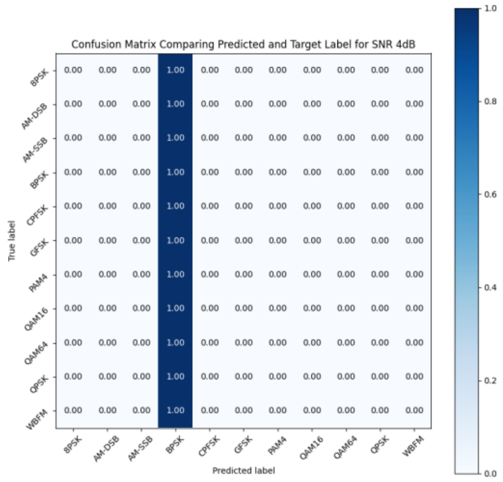


Fig. 5.42 Confusion matrix of CLDNN based AMC with batch size 32 for 4 dB SNR in 1 epoch

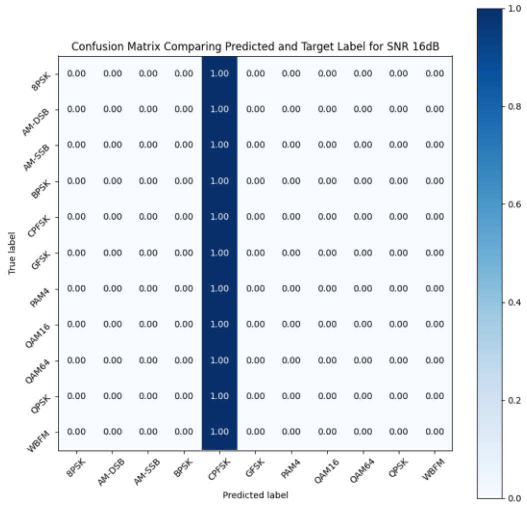


Fig. 5.43 Confusion matrix of CLDNN based AMC with batch size 32 for 16 dB SNR in 1 epoch

Epoch = 15 :

The model performance for 15 epoch is as shown in figure 5.44 to 5.56. The model shows best results in SNR 14 dB with an accuracy of 30.682%. The confusion matrix corresponding to 14 dB SNR is shown in figure 5.56. The training duration is 7 hours.

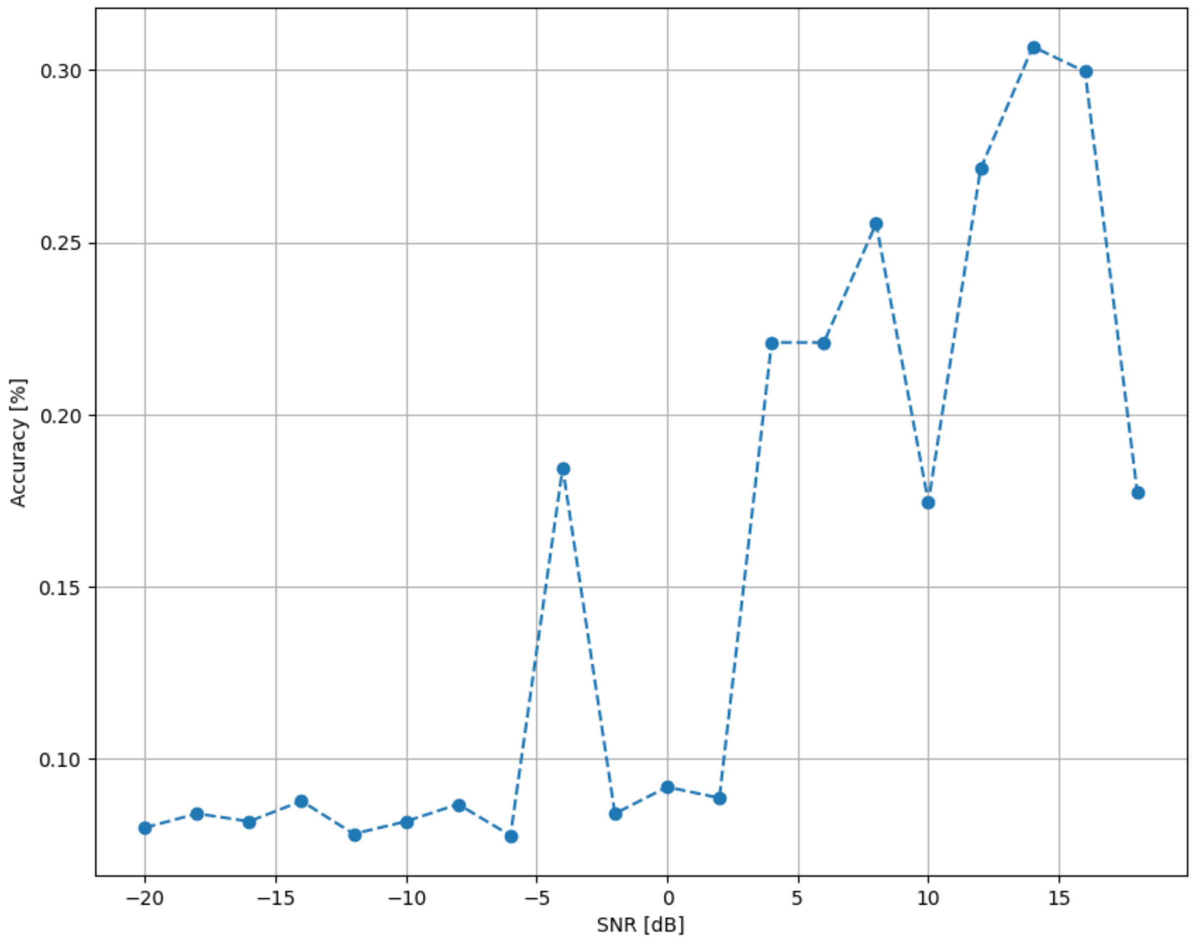


Fig. 5.44 Accuracy vs. SNR plot of CLDNN based AMC with batch size 32 in 15 epochs

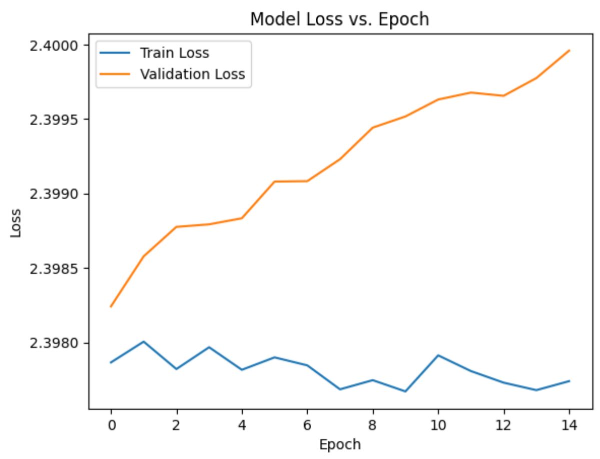


Fig. 5.45 Loss vs. epoch plot for CLDNN based AMC with batch size 32 for -10 dB SNR in 15 epochs

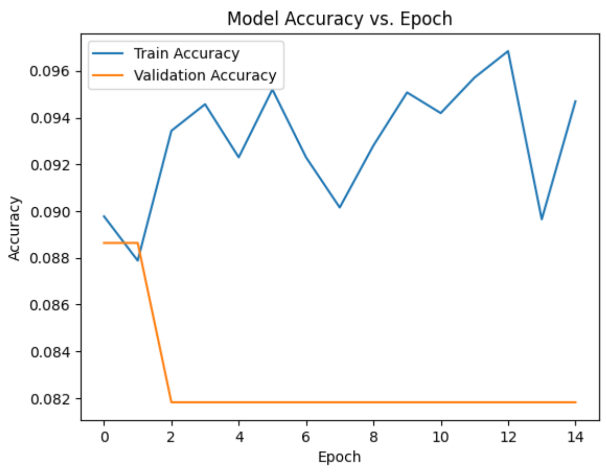
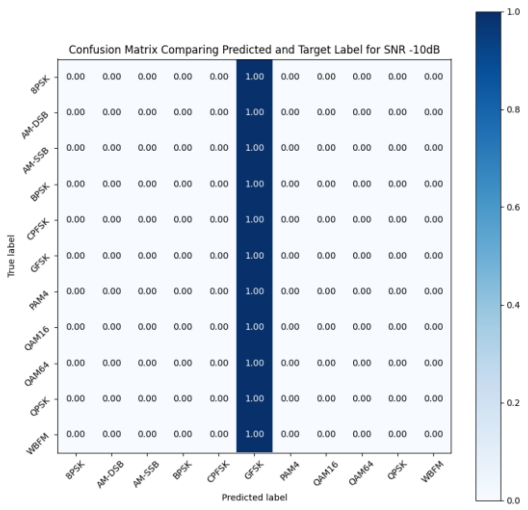


Fig. 5.46 Accuracy vs. epoch plot of CLDNN based AMC with batch size 32 for -10 dB SNR in 15 epochs



	8PSK	AM-DSB	AM-SSB	BPSK	CPFSK	GFSK	PAM4	QAM16	QAM64	QPSK	WBFM
8PSK	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
AM-DSB	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
AM-SSB	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
BPSK	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
CPFSK	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
GFSK	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
PAM4	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
QAM16	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
QAM64	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
QPSK	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
WBFM	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00

Fig. 5.47 Confusion matrix of CLDNN based AMC with batch size 32 for -10 dB SNR in 15 epochs

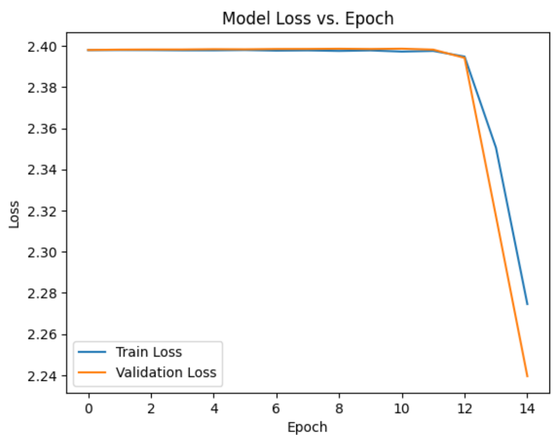


Fig. 5.48 Loss vs. epoch plot for CLDNN based AMC with batch size 32 for -4 dB SNR in 15 epochs

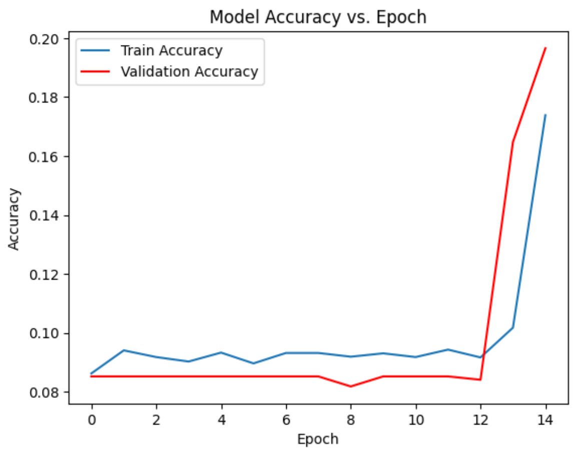


Fig. 5.49 Accuracy vs. epoch plot of CLDNN based AMC with batch size 32 for -4 dB SNR in 15 epochs

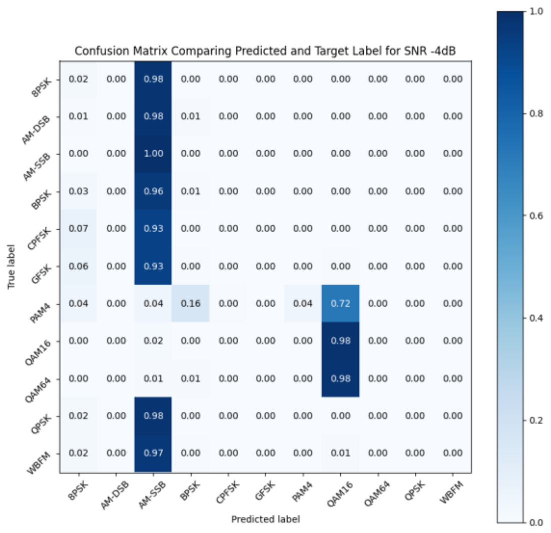


Fig. 5.50 Confusion matrix of CLDNN based AMC with batch size 32 for -4 dB SNR in 15 epochs

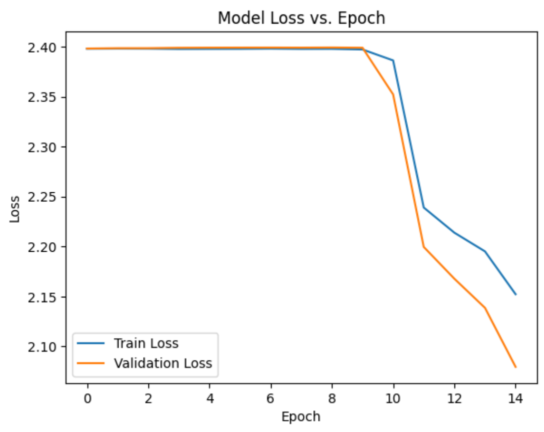


Fig. 5.51 Loss vs. epoch plot for CLDNN based AMC with batch size 32 for 8 dB SNR in 15 epochs

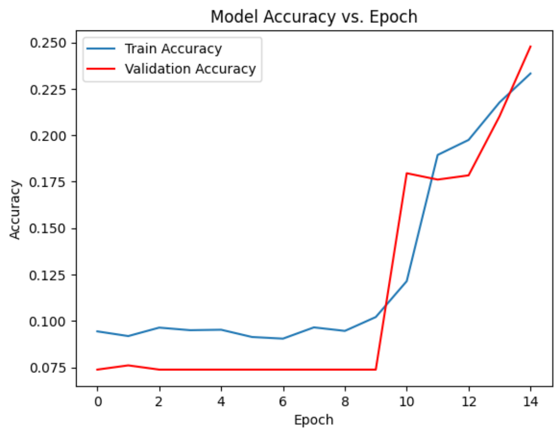


Fig. 5.52 Accuracy vs. epoch plot of CLDNN based AMC with batch size 32 for 8 dB SNR in 15 epochs

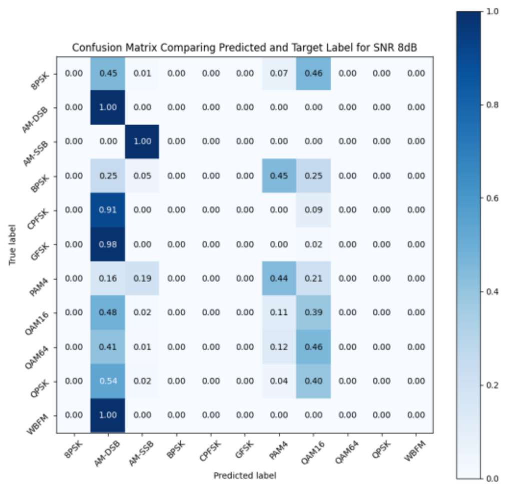


Fig. 5.53 Confusion matrix of CLDNN based AMC with batch size 32 for 8 dB SNR in 15 epochs

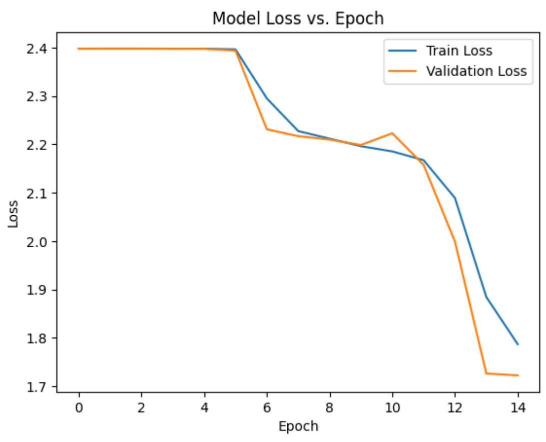


Fig. 5.54 Loss vs. epoch plot for CLDNN based AMC with batch size 32 for 14 dB SNR in 15 epochs

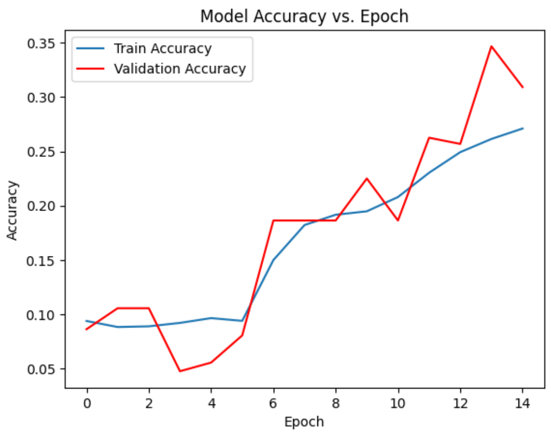
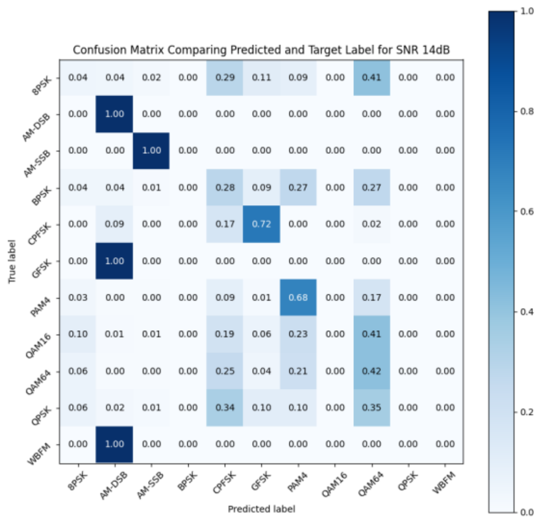


Fig. 5.55 Accuracy vs. epoch plot of CLDNN based AMC with batch size 32 for 14 dB SNR in 15 epochs



	8PSK	AM-DSB	AM-SSB	BPSK	CPFSK	GFSK	PAM4	QAM16	QAM64	QPSK	WBFM
8PSK	0.04	0.04	0.02	0.00	0.29	0.11	0.09	0.00	0.41	0.00	0.00
AM-DSB	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AM-SSB	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
BPSK	0.04	0.04	0.01	0.00	0.28	0.09	0.27	0.00	0.27	0.00	0.00
CPFSK	0.00	0.09	0.00	0.00	0.17	0.72	0.00	0.00	0.02	0.00	0.00
GFSK	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PAM4	0.03	0.00	0.00	0.00	0.09	0.01	0.68	0.00	0.17	0.00	0.00
QAM16	0.10	0.01	0.01	0.00	0.19	0.06	0.23	0.00	0.41	0.00	0.00
QAM64	0.06	0.00	0.00	0.00	0.25	0.04	0.21	0.00	0.42	0.00	0.00
QPSK	0.06	0.02	0.01	0.00	0.34	0.10	0.10	0.00	0.35	0.00	0.00
WBFM	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Fig. 5.56 Confusion matrix of CLDNN based AMC with batch size 32 for 14 dB SNR in 15 epochs

5.3.4 Comparison between CLDNN based AMC with batch size 128 and with batch size 32

The graph shown in figure 5.57 compares the performance of CLDNN based AMC with batch size 128 (paper model [26]) and with batch size 32 (proposed model). The model with batch size 32 (proposed model) shows **12.876% better** performance than the model with batch size 128 (paper model).

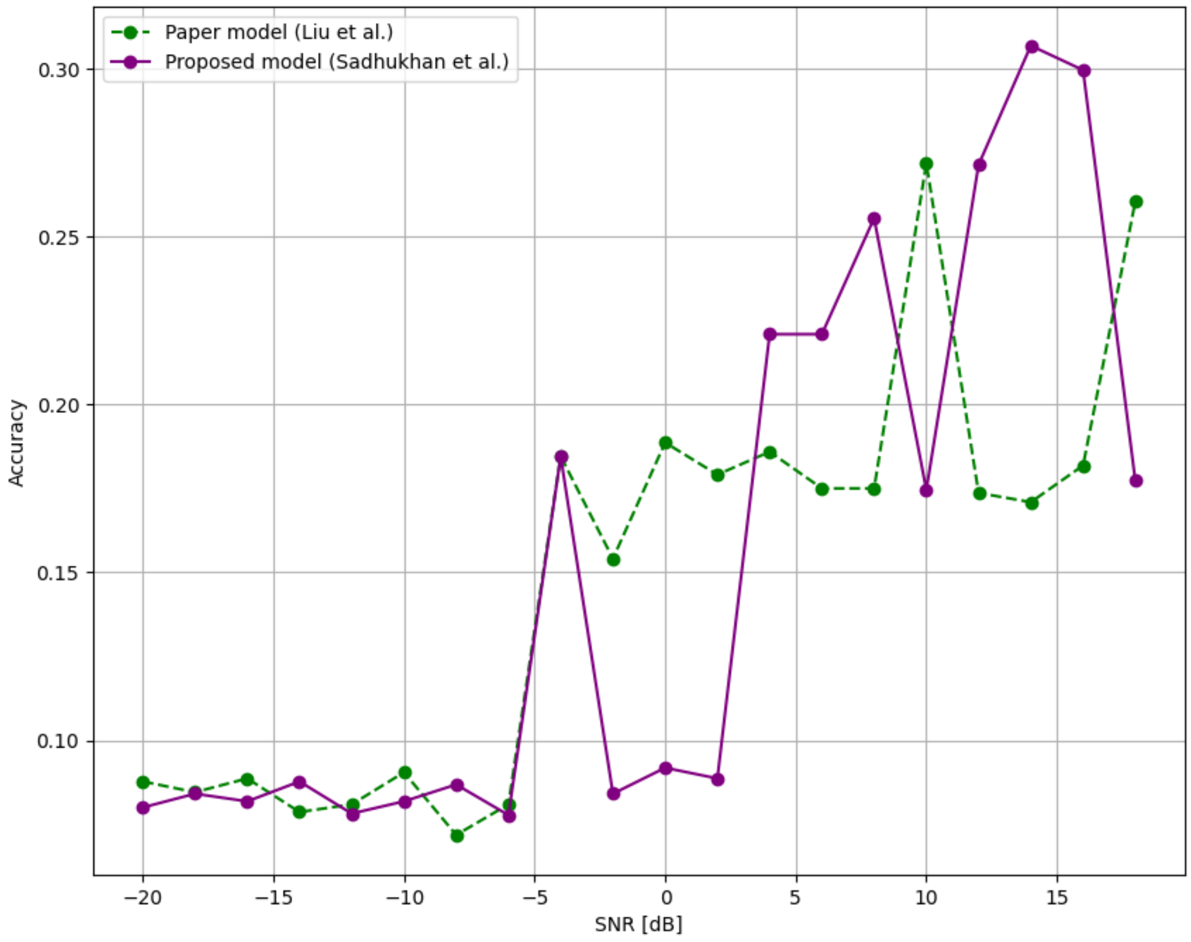


Fig. 5.57 Accuracy vs. SNR plot comparing CLDNN based AMC with batch size 128 (paper model) and 32 (proposed model) in 15 epochs

5.4 Summary

By utilizing the complementarity of CNNs, LSTMs, and DNNs, Convolutional Long Short-Term Deep Neural Network (CLDNN) integrates the architectures of CNN and Long Short-Term Memory (LSTM) into a deep neural network. The memory unit of a recurrent neural network (RNN) is the LSTM unit. RNNs are neural networks containing memories that function well for tasks requiring learning sequences, such as speech and handwriting recognition. By utilizing a forget gate in its memory cell, which permits the learning of long-term dependencies, LSTM optimizes the gradient vanishing problem in RNNs. We have used an LSTM with 50 cells along with 4 convolution layers in our model. We investigated various CLDNN designs with various LSTM layer memory cell counts. Our tests demonstrate that, when compared to other layer configurations, an LSTM layer with 50 cells provides the best accuracy results. But CLDNN model fails to learn well using small number of epochs. It learns well for at least a 100-epoch value. But due to resource constraint we could only run our model for 15 and 20 epoch values. The model shows best results in SNR 0 dB with an accuracy of 28.409% in 20 epochs and in 15 epochs it shows best results in SNR 10 dB with an accuracy of 27.182%. In case of 15 epochs the training duration is 6 hours and in case of 20 epochs it is 6.5 hours. We have tuned the hyperparameter, batch size to 32 from 128 and observed major improvement in accuracy. The model with batch size 32 shows best results in SNR 14 dB with an accuracy of 30.682%. The model with batch size 32 which is the proposed model shows 12.876% better performance than the model with batch size 128 which has been used in the paper model.

5.5 Comparison between CNN, ResNet and CLDNN based AMC

Figure 5.58 compares three different DNN based AMC models – CNN with 4 convolutional layer and batch normalization in 10 epochs, Resnet with dropout rate 0.3 in 10 epochs and CLDNN with batch size 32 in 15 epochs. Among the three different models ResNet based AMC with dropout rate 0.3 shows the best results as far as the accuracy is concerned. It shows 7.635% better accuracy than the CNN based model with 4 convolution layer and batch normalization. The model shows best result in SNR 6 dB with an accuracy of 73.773% which is highest among all the three models in this epoch setting. Table 5.3 summarizes the relative merits and demerits of the three models.

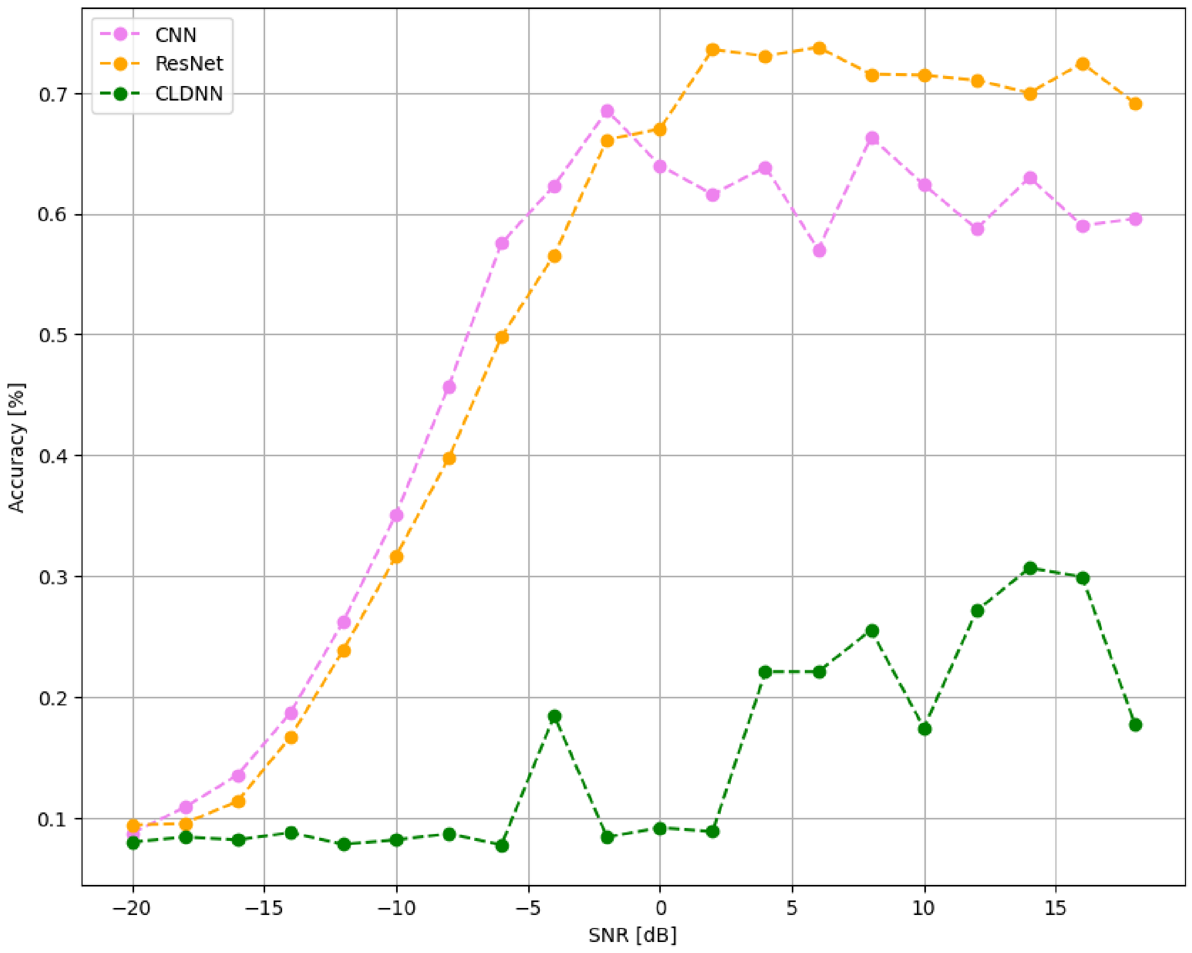


Fig. 5.58 Accuracy vs. SNR plot comparing CNN, ResNet and CLDNN based AMC models

Table 5.3 Relative merits and demerits of CNN, ResNet and CLDNN based AMC models

	CNN based AMC	ResNet based AMC	CLDNN based AMC
Merits	CNN greatly reduces the number of parameters to learn. Hence it takes less amount of time to train the model.	ResNet improves feature propagation in the neural network by establishing shortcut paths between its various layers which in turn improves the accuracy	CLDNN performs very well and shows highest accuracy among the three models for high value of epochs (>100)
Demerits	CNN suffers from vanishing or exploding gradients and accuracy loss after a given network depth	ResNet models take huge amount of time for training	CLDNN learns almost nothing for small number of epoch values

---

CHAPTER 6

Conclusion and Future Scope

6.1 Concluding Remarks

In this thesis work, we have proposed three different Deep Learning (DL) or Deep Neural Network (DNN) models to solve the Automatic Modulation Classification problem. During simulation we have used the RadioML2016.10a dataset which comprises of 11 different types of modulated synthetic signals. The dataset includes the following modulation types: AM-DSB, AM-SSB, WBFM, GFSK, CPFSK, PAM-4, BPSK, QPSK, 8-PSK, 16-QAM, and 64-QAM. All the simulations have been run on top of Google Colaboratory.

At first, two types of CNN have been considered here- one with 3 convolutional layers and another with 4 convolutional layers. Three convolution layers with different number of filters with size 3x3 are used in the first model. Each convolution layer is followed by a maxpooling layer with pool size 1x2 and the maxpooling layers are followed by dropout layer with dropout rate 0.3. The first convolution layer has 256 number of filters, the second convolution layer has 128 number of filters and the last convolution layer has 64 number of filters. The activation function that is used here is ReLU. Zero padding is used here which makes the output vector dimension same as the input vector dimension. Lastly a flatten layer has been used followed by two dense layers. The first dense layer contains 128 neurons and the second dense layer contains 11 neurons which is followed by a SoftMax activation function. The simulation is done with 2 different epoch values: epoch 1 and epoch 10. The model shows best results in SNR 0 dB with an accuracy of 61.5% in 10 epochs. The four convolution layered model has everything same as the three convolution model except it has another convolutional layer with 64 number of filters. The model shows best results in SNR 0 dB with an accuracy of 62.73%. We have added batch normalization Layer after each convolution layer and first dense layer in the CNN based AMC model with 4 convolution layers. The model shows best results in SNR -2 dB with an accuracy of 68.54% which is 9.275% better compared to the model without batch normalization layer.

Secondly, we have considered Residual Network (ResNet). This model also consists of 4 convolution layers, each one followed by a dropout layer with dropout rate 0.6. The first convolution layer has 256 number of filters with size 1x3. The second convolution layer has 256 number of filters with size 2x3. The third and fourth convolution layer has 80 number of filters with size 1x3. This is followed by a flatten layer which is followed by two dense layers with 128 and 11 neurons respectively. Residual connection is made between the input layer and the convolutional layers. The last dense layer is followed by a SoftMax activation function and ReLU is used in the remaining layers. Zero padding is used here which makes the output vector dimension same as the input vector dimension. The simulation results for 1 and 10 have been considered. We have tuned the dropout rate, which is one of the hyperparameters of the model, to 0.3 and noticed major improvement in accuracy without any significant increase in overfitting. . The model shows best results in SNR 6 dB with an accuracy of 73.773% which is 4.575% higher compared to the ResNet model with 0.6 dropout rate.

Lastly, we have considered the Convolutional Long Short-term Deep Neural Network (CLDNN). The structure of the CLDNN model is similar to the 4-convolution layer CNN model except that in CLDNN there is an extra LSTM layer (Long Short-Term Memory) between the last convolution layer and the first dense layer. We have considered the model performance for 4 different epoch values: 1, 10, 15 and 20.

We have tried the following methods to increase the accuracy of the model:

- By decreasing the batch size
- Early Stopping
- Using Stochastic Gradient Descent (SGD) optimizer
- Increasing the depth of the dense layer

But only the first approach has shown fruitful result. We decreased (fine-tuned) the batch size to 32 from 128 and it has increased the accuracy of the model to 30.682% at 14 dB SNR which is 12.876% higher than the model with 128 Batch Size. Having fewer examples in the batch enables the model to learn from each one separately. Hence it improves the accuracy but at the cost of longer training time.

6.2 Future Work

We would like to perform the following tasks in future:

- Implementation of two different models namely Densely Connected Deep Neural Network (DenseNet) and Residual Network 50 (ResNet50).
- Testing our model using two new datasets :
    - RadioML2016.10b which includes 10 different digital and analog modulation types: AM-DSB, WBFM, GFSK, CPFSK, PAM-4, BPSK, QPSK, 8-PSK, 16-QAM and 64-QAM.
    - HisarMod2019.1 which includes 26 modulation classes with 5 different fading types.

Bibliography

[1] O. Dobre, A. Abdi, Y. Bar-Ness, W. Su, Survey of automatic modulation classification techniques: classical approaches and new trends, IET Commun. 1 (2) (2007) 137–156.

[2] F. Hameed, O. Dobre, D. Popescu, On the likelihood-based approach to modulation classification, IEEE Trans. Wireless Commun. 8 (12) (2009) 5884–5892.

[3] K. Assaleh, K. Farrell, R. Mammone, A new method of modulation classification for digitally modulated signals, in: IEEE Military Communications Conference, MILCOM'92, Vol. 2, 1992, pp. 712–716.

[4] F. Xie, C. Li, G. Wan, An efficient and simple method of MPSK modulation classification, in: 4th International Conference on Wireless Communications, Networking and Mobile Computing WiCOM'08, 2008, pp. 1–3.

[5] W. Dan, G. Xuemai, G. Qing, A new scheme of automatic modulation classification using wavelet and wsvm, in: 2nd International Conference on Mobile Technology, Applications and Systems, 2005, p.

[6] L. Liu, J. Xu, A novel modulation classification method based on high order cumulants, in: International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM, 2006, pp. 1–5.

[7] A. Swami, B. Sadler, Hierarchical digital modulation classification using cumulants, IEEE Trans. Commun. 48 (3) (2000) 416–429.

[8] Y. Han, G. Wei, C. Song, L. Lai, Hierarchical digital modulation recognition based on higher-order cumulants, in: 2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control, IMCCC, 2012, pp. 1645–1648. http://dx.doi.org/10.1109/IMCCC.2012.398

[9] O. Dobre, Y. Bar-Ness, W. Su, Higher-order cyclic cumulants for high order modulation classification, in: Military Communications Conference, MILCOM, Vol. 1, 2003, pp. 112–117.

[10] W. Su, Feature space analysis of modulation classification using very high-order statistics, IEEE Commun. Lett. 17 (9) (2013) 1688–1691.

[11] C. Zhendong, J. Weining, X. Changbo, L. Min, Modulation recognition based on constellation diagram for M-QAM signals, in: 11th IEEE International Conference on Electronic Measurement Instruments, ICEMI, Vol. 1, 2013, pp. 70–74.

[12] J. Popoola, R. Van Olst, Automatic classification of combined analog and digital modulation schemes using feedforward neural network, in: AFRICON, 2011, pp. 1–6.

[13] M. Valipour, M. Homayounpour, M. Mehralian, Automatic digital modulation recognition in presence of noise using SVM and PSO, in: Sixth International Symposium on Telecommunications, IST, 2012, pp. 378–382.

[14] A. Abdelmutalab, K. Assaleh, M. El-Tarhuni, Automatic modulation classification using polynomial classifiers, in: 25th IEEE International Symposium on Personal Indoor and Mobile Radio Communications, PIMRC, 2014, pp. 785–789

[15] M. Aslam, Z. Zhu, A. Nandi, Automatic modulation classification using combination of Genetic Programming and KNN, IEEE Trans. Wireless Communication 11 (8) (2012) 2742–2750.

[16] M. Mirarab, M. Sobhani, Robust modulation classification for PSK /QAM/ASK using higher-order cumulants, in: 6th International Conference on Information, Communications Signal Processing, 2007, pp. 1–4.

[17] M. Wong, S.K. Ting, A. Nandi, Naïve Bayes classification of adaptive broadband wireless modulation schemes with higher order cumulants, in: 2nd International Conference on Signal Processing and Communication Systems, ICSPCS, 2008, pp. 1–5

[18] M. Aslam, Z. Zhu, A. Nandi, Automatic digital modulation classification using Genetic Programming with K-Nearest Neighbour, in: Military Communications Conference, MILCOM, 2010, pp. 1731–1736.

[19] N. Ahmadi, R. Berangi, Modulation classification of QAM and PSK from their constellation using genetic algorithm and hierarchical clustering, in: 3rd International Conference on Information and Communication Technologies: From Theory to Applications, ICTTA, 2008, pp. 1–5.

[20] H.-C. Wu, M. Saquib, Z. Yun, Novel automatic modulation classification using cumulant features for communications via multipath channels, IEEE Trans. Wireless Communication 7 (8) (2008) 3098–3105.

[21] T. Huynh-The, Q. Pham, T. Nguyen, T. T. Nguyen, R. Ruby, M. Zeng, D. Kim, "Automatic Modulation Classification: A Deep Architecture Survey", IEEE Access (Volume: 9), 15 October 2021

[22] A. P. Hermawan, R. R. Ginanjar, D. S. Kim, J. M Lee, "CNN-Based Automatic Modulation Classification for Beyond 5G Communications", IEEE Communications Letters, Volume: 24, Issue: 5.

[23] W. Yongshi, G. Jie, L. Hao, L. Li, W. Zhigang, W. Houjun, "CNN-based modulation classification in the complicated communication channel", 2017 IEEE 13th International Conference on Electronic Measurement & Instruments (ICEMI'2017), pp. 512-516.

[24] K. Tekbıyık, A. R. Ekti, A. Gorcin, G. K. Kurt, C. Kececi, "Robust and Fast Automatic Modulation Classification with CNN under Multipath Fading Channels", 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring).

[25] S. Ramjee, S. Ju, D. Yang, X. Liu, A. E. Gamal, and Y. C. Eldar, "Fast deep learning for automatic modulation classification," arXiv preprint arXiv:1901.05850, 2019.

[26] X. Liu, D. Yang, and A. E. Gamal, "Deep neural network architectures for modulation classification," in 2017 51st Asilomar Conference on Signals, Systems, and Computers, Oct 2017, pp. 915–919.

[27] Timothy J. O'Shea, Latha Pemula, Dhruv Batra, and T. Charles Clancy. "Radio transformer networks: Attention models for learning to synchronize in wireless systems." In Signals, Systems and Computers, 2016 50th Asilomar Conference on, pp. 662-666. IEEE, 2016

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.

[29] Huang G, Liu Z, Weinberger K Q, et al. Densely connected convolutional networks[J]. arXiv preprint arXiv:1608.06993, 2016.

[30] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak. Convolutional, long short-term memory, fully connected deep neural networks. In 2015 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 4580-4584, April 2015.

[31] B. Kim, J. Kim, H. Chae, D. Yoon, and J. W. Choi, "Deep neural network-based automatic modulation classification technique," in Information and Communication Technology Convergence (ICTC), 2016 International Conference on, pp. 579–582, IEEE, 2016.

[32] J. Jagannath, N. Polosky, D. O'Connor, L. N. Theagarajan, B. Sheaffer, S. Foulke, and P. K. Varshney, "Artificial neural network based automatic modulation classification over a software defined radio testbed," in 2018 IEEE International Conference on Communications (ICC), May 2018, pp. 1–6.

[33] J. Jagannath, D. O'Connor, N. Polosky, B. Sheaffer, L. N. Theagarajan, S. Foulke, P. K. Varshney, and S. P. Reichhart, "Design and Evaluation of Hierarchical Hybrid Automatic Modulation Classifier using Software Defined Radios," in Proc. of IEEE Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, Jan 2017.

[34] P. Qi, X. Zhou, S. Zheng, Z. Li, "Automatic Modulation Classification Based on Deep Residual Networks with Multimodal Information", IEEE Transactions on Cognitive Communications and Networking (Volume: 7, Issue: 1, March 2021)

[35] https://keras.io/api/losses/probabilistic_losses/#categoricalcrossentropy-class

[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in Proceeding of the International Conference on Machine Learning, Lille, France, Jul. 2015, pp. 448–456.

[37] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," IEEE Journal on Selected Areas in Communications, vol. 35, no. 6, pp. 1201–1221, June 2017.

[38] R. Li, Z. Zhao, X. Zhou, G. Ding, Y. Chen, Z. Wang, and H. Zhang, "Intelligent 5G: When cellular networks meet artificial intelligence," IEEE Wireless Communications, vol. 24, no. 5, pp. 175–183, October 2017.

[39] M. Matinmikko-Blue, S. Yrj ¨ol ¨a, V. Sepp ¨anen, P. Ahokangas, H. H ¨amm ¨ainen, and M. Latva-Aho, "Analysis of spectrum valuation elements for local 5G networks: Case study of 3.5-GHz band," IEEE Transactions on Cognitive Communications and Networking, vol. 5, no. 3, pp. 741–753, Sep. 2019.

[40] W. Zhang, C. Wang, X. Ge, and Y. Chen, "Enhanced 5G cognitive radio networks based on spectrum sharing and spectrum aggregation," IEEE Transactions on Communications, vol. 66, no. 12, pp. 6304–6316, Dec 2018.

[41] Y. Li, W. Zhang, C. Wang, J. Sun, and Y. Liu, “Deep reinforcement learning for dynamic spectrum sensing and aggregation in multi-channel wireless networks,” IEEE Transactions on Cognitive Communications and Networking, vol. 6, no. 2, pp. 464–475, 2020.

[42] D. Wang, B. Song, D. Chen, and X. Du, “Intelligent cognitive radio in 5G: AI-based hierarchical cognitive cellular networks,” IEEE Wireless Communications, vol. 26, no. 3, pp. 54–61, June 2019.

[43] H. Wang, J. Wang, G. Ding, J. Chen, Y. Li, and Z. Han, “Spectrum sharing planning for full-duplex uav relaying systems with underlaid d2d communications,” IEEE Journal on Selected Areas in Communications, vol. 36, no. 9, pp. 1986–1999, Sep. 2018.

[44] E. Pateromichelakis, O. Bulakci, C. Peng, J. Zhang, and Y. Xia, “LAA as a key enabler in slice-aware 5G ran: Challenges and opportunities,” IEEE Communications Standards Magazine, vol. 2, no. 1, pp. 29–35, March 2018.

[45] R. Bajracharya, R. Shrestha, R. Ali, A. Musaddiq, and S. W. Kim, “LWA in 5G: State-of-the-art architecture, opportunities, and research challenges,” IEEE Communications Magazine, vol. 56, no. 10, pp. 134– 141, October 2018.

[46] D. Guiducci, C. Carciofi, V. Petrini, S. Pompei, M. Faccioli, E. Spina, G. De Sipio, D. Massimi, D. Spoto, F. Amerighi, T. Magliocca, P. Chawdhry, J. Chareau, J. Bishop, P. Viaud, T. Pinato, S. Yrj ¨ol ¨a, V. Hartikainen, L. Tudose, J. Llorente, V. Ferrer, J. Costa-Requena, H. Kokkinen, L. Ardito, P. Muller, M. Gianesin, F. Grazioli, and D. Caggiati, “Regulatory pilot on licensed shared access in a live LTE-TDD network in IMT band 40,” IEEE Transactions on Cognitive Communications and Networking, vol. 3, no. 3, pp. 534–549, Sep. 2017.

[47] C. Xin and M. Song, “Analysis of the on-demand spectrum access architecture for CBRS cognitive radio networks,” IEEE Transactions on Wireless Communications, vol. 19, no. 2, pp. 970–978, Feb 2020.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in International Conference on Neural Information Processing Systems, 2012. [13] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.

[49] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” Computer Science, 2014.

[50] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[51] H. Xing, G. Zhang, and M. Shang, “Deep learning,” International Journal of Semantic Computing, vol. 10, no. 03, pp. 417–439, 2016.

[52] O. Simeone, “A very brief introduction to machine learning with applications to communication systems,” IEEE Transactions on Cognitive Communications and Networking, vol. 4, no. 4, pp. 648–664, 2018.

[53] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, “Imagenet large scale visual recognition challenge,” International Journal of Computer Vision, vol. 115, no. 3, pp. 211–252, 2015.

[54] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, C. Yuan, G. Qin, and K. Macherey, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” 2016.

[55] L. Jin, P. Yi, L. Min, Z. Chen, T. Lu, C. Lu, and J. Wang, “Applications of deep learning to MRI images: A survey,” Big Data Mining and Analytics, vol. 1, no. 1, pp. 1–18, 2018.

[56] C. Jiang and X. Zhu, “Reinforcement learning based capacity management in multi-layer satellite networks,” IEEE Transactions on Wireless Communications, vol. PP, no. 99, pp. 1–1, 2020.

[57] T. J. O’Shea, J. Corgan, and T. C. Clancy, “Convolutional radio modulation recognition networks,” in Engineering Applications of Neural Networks, 2016, pp. 213–226.

[58] F. Meng, P. Chen, L. Wu, and X. Wang, “Automatic modulation classification: A deep learning enabled approach,” IEEE Transactions on Vehicular Technology, vol. 67, no. 11, pp. 10 760–10 772, Nov 2018.

[59] J. Nie, Y. Zhang, Z. He, S. Chen, S. Gong, and W. Zhang, “Deep hierarchical network for automatic modulation classification,” IEEE Access, vol. 7, pp. 94 604–94 613, 2019.

[60] Z. Ming, D. Ming, and L. Guo, “Convolutional neural networks for automatic cognitive radio waveform recognition,” IEEE Access, vol. 5, pp. 11 074–11 082, 2017.

[61] J. H. Lee, K. Kim, and Y. Shin, “Feature image-based automatic modulation classification method using CNN algorithm,” in 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Feb 2019, pp. 1–4.

[62] B. Tang, Y. Tu, Z. Zhang, and Y. Lin, “Digital signal modulation classification with data augmentation using generative adversarial nets in cognitive radio networks,” IEEE Access, vol. 6, pp. 15 713–15 722, 2018.

[63] Y. Li, G. Shao, and B. Wang, “Automatic modulation classification based on bispectrum and CNN,” in 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), May 2019, pp. 311–316.

[64] R. Li, L. Li, S. Yang, and S. Li, “Robust automated vhf modulation recognition based on deep convolutional neural networks,” IEEE Communications Letters, vol. 22, no. 5, pp. 946–949, 2018.

[65] Y. Wang, M. Liu, J. Yang, and G. Gui, “Data-driven deep learning for automatic modulation recognition in cognitive radios,” IEEE Transactions on Vehicular Technology, vol. 68, no. 4, pp. 4074–4077, April 2019.

[66] S. Zheng, P. Qi, S. Chen, and X. Yang, “Fusion methods for CNN-based automatic modulation classification,” IEEE Access, vol. 7, pp. 66 496– 66 504, 2019.

[67] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," IEEE Journal of Selected Topics in Signal Processing, vol. 12, no. 1, pp. 168–179, 2018.

[68] J. Ma, S. Lin, H. Gao, and T. Qiu, "Automatic modulation classification under non-Gaussian noise: A deep residual learning approach," in 2019 IEEE International Conference on Communications (ICC), 2019, pp. 1–6.

[69] H. Zhang, L. Chen, J. Liu, and J. Yuan, "Hierarchical multi-feature fusion for multimodal data analysis," in 2014 IEEE International Conference on Image Processing (ICIP), Oct 2014, pp. 5916–5920.

[70] Z. Guo, X. Li, H. Huang, N. Guo, and Q. Li, "Medical image segmentation based on multi-modal convolutional neural network: Study on image fusion schemes," in 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), April 2018, pp. 903–907.

[71] T. Zhou, K. Thung, X. Zhu, and D. Shen, "Effective feature learning and fusion of multimodality data using stage wise deep neural network for dementia diagnosis," Human Brain Mapping, vol. 40, no. 5, 2018.

[72] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," IEEE Transactions on Cognitive Communications and Networking, vol. 3, no. 4, pp. 563–575, 2017.

[73] S. Wu, G. Li, L. Deng, L. Liu, D. Wu, Y. Xie, and L. Shi, "l1-norm batch normalization for efficient training of deep neural networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 7, pp. 2043–2051, 2019.

[74] "Physical channels and modulations (release 15), document 3gpp tr 38.211 v13.0.0, 3rd generation partnership project," Technical Specification Group Radio Access Network, pp. 1–96, 2018.

[75] "IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements–part 11: Wireless LAN medium access control (mac) and physical layer (phy) specifications–amendment 4: Enhancements for very high throughput for operation in bands below 6 ghz." IEEE Std 802.11ac(TM)-2013 (Amendment to IEEE Std 802.11- 2012, as amended by IEEE Std 802.11ae-2012, IEEE Std 802.11aa2012, and IEEE Std 802.11ad-2012), pp. 1–425, 2013.

[76] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi, "A tutorial on IEEE 802.11ax high efficiency WLANs," IEEE Communications Surveys Tutorials, vol. 21, no. 1, pp. 197–216, 2019
