An Ensemble Automatic Modulation Classification Model With Weight Pruning and Data Preprocessing

by

Xueting Yang

B. Eng, Xidian University, 2017

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Electrical and Computer Engineering)

The University of British Columbia
(Vancouver)

February 2020

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

An Ensemble Automatic Modulation Classification Model With Weight Pruning and Data Preprocessing

submitted by **Xueting Yang** in partial fulfillment of the requirements for the degree of **MASTER OF APPLIED SCIENCE** in **Electrical and Computer Engineering**.

Examining Committee:

Victor C.M. Leung, Electrical and Computer Engineering, UBC, Vancouver
Supervisor

Julian Cheng, School of Engineering, UBC, Okanagan
Co-supervisor, Supervisory Committee Member

Jane Z. Wang, Electrical and Computer Engineering, UBC, Vancouver
Supervisory Committee Member

Abstract

Automatic Modulation Classification (AMC) detects the modulation type and order of the received signal using limited prior knowledge within a short observation interval. In this thesis, we aim to provide a computation-efficient and high-performance AMC model for resource-constrained mobile devices.

We use a public RadioML dataset and introduce three data pre-processing methods including noise reduction, normalization, and label smoothing before training the raw signals. Besides four common signal representations, we propose a new signal representation called a three-dimensional constellation image.

For each signal representation, we carefully design a Deep Learning (DL) model. In addition to the traditional Convolutional Neural Network (CNN), two new AMC model structures are proposed. The attention module is integrated into the AMC model structure based on conventional Long Short-term Memory (LSTM) networks. Another proposed AMC model structure connects CNN, LSTM, and densely connected neural networks with two additional connections.

After training the AMC models, we analyze the overall and per-class performance. We also study the computational complexity of trained AMC models in terms of memory consumption and detection efficiency. Overall, the results indicate that the proposed data pre-processing methods and the new AMC model structures can significantly improve the classification performance. To reduce the complexity of proposed AMC models, we introduce weight pruning to remove unnecessary connections in DL models. After weight pruning, the proposed AMC models have negligible performance degradation.

To further improve the performance of AMC models, we also propose ensemble learning to train a second-level model based on multiple first-level AMC models. With three-fold cross-validation, the second-level model can train on the whole dataset and have an F1-score improvement of at least 10%. We also conduct weight pruning to reduce the unnecessary parameters of the ensemble learned model. Overall, after weight pruning, the ensemble learned AMC model receives an F1-score of 0.965 when the signal-to-noise ratio is greater than 6 dB.

Lay Summary

Automatic Modulation Classification (AMC) is an important technology that finds applications in both civilian and military communication systems. AMC classifies the modulation types and orders of the received signals in short observation time and typically requires less prior statistical information. This thesis aims to provide a high-performance and memory-efficient AMC model for mobile devices.

We propose two improved Deep Learning (DL) models based on the existing AMC models and three data pre-processing methods before training AMC models. To further improve the classification accuracy of the proposed AMC models, we combine multiple AMC models into one model. To improve computational efficiency, we further remove the unnecessary connections in the model structure. Both performance analysis and experimental results validate the efficiency of the proposed three data pre-processing methods, two new DL model structures, and the combination of single models.

Preface

I hereby declare that I am the author of this thesis. The original and unpublished work outlined in this thesis was conducted in the Department of Electrical and Computer Engineering at The University of British Columbia, under the supervision of Professor Julian Cheng and Professor Victor C.M. Leung.

Table of Contents

List of Tables

List of Figures

Symbols

argmax	Points of the domain of the function where the function values are maximized.
argmin	Points of the domain of the function where the function values are minimized.
$cum(\cdot)$	Cumulant function in probability theory.
$\mathfrak{I}$	Imaginary part of a complex variable.
$\mathfrak{R}$	Real part of a complex variable
$\mathbb{R}$	A set of all real numbers
M	Number of modulation schemes
N	Number of examples in the dataset
L	Sample length of received signals

Glossary

1D One-dimensional

2D Two-dimensional

3D-CI Three-dimensional Constellation Image

AMC Automatic Modulation Classification

AM Amplitude Modulation

AP Amplitude and Phase

AWGN Additive White Gaussian Noise

BER Bit Error Rate

BPSK Binary Phase Shift Keying

CD Constellation Diagram

CNN Convolutional Neural Network

CPFSK Continuous-phase Frequency-shift Keying

CR Cognitive Radio

D/A Digital-to-analog

DFT Discrete Fourier Transform

DLM Deep Learning-based Method

DL Deep Learning

DNN Densely-connected Neural Network

DSB Double Side-band

EM Expectation Maximization

FBM Feature-based Method

FFT Fast Fourier Transform

GFSK Gaussian Frequency-shift Keying

HOC High-order Cumulants

HOM High-order Moments

HOS High-order Statics

I.I.D Independent and Identically Distributed

IQ In-phase and Quadrature

KDE Kernel Density Estimation

KNN K-Nearest Neighborhood

LBM Likelihood-based Method

LSTM Long Short-Term Memory

MLE Maximum Likelihood Estimation

MR Modulation Recognition

NN Neural Network

PAM Pulse Amplitude Modulation

PAR Peak to Average Ratio

PDF Probability Density Function

PRC Precision-Recall Curve

PRR Peak to Root-mean-square Ratio

PSK Phase Shift Keying

QAM Quadrature Amplitude Modulation

QPSK Quadrature Phase Shift Keying

RELU Rectified Linear Unit

RNN Recurrent Neural Network

ROC Receiver Operating Characteristics

SGD Stochastic Gradient Descent

SNR Signal-to-Noise Ratio

SSB Single Side-band Modulation

WBFM Wide-band Frequency Modulation

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisors Professor Victor C.M. Leung and Professor Julian Cheng, for their great support and persistent encouragement during my master's program. It is a great pleasure to work under their supervision. Also, I greatly thank my committee members for the time and effort in reading my thesis and providing feedback.

I have been truly fortunate to have all the brilliant colleagues and friends in Vancouver Lab Kaiser 4090 and Okanagan Lab EME 2255. I am also thankful to my roommates Yubo Sun and Siqi An for always being a huge support in my life.

Thank you to my sweet, reliable and caring boyfriend Zhen Wang, for all the precious understanding, encouragement, and always keeping me company. Thanks for being my soulmate, and I love you with all my heart.

Finally, I own my heartfelt gratitude to my beloved parents for giving me endless love and support. I am lucky and proud to be your daughter.

Chapter 1

Introduction

In this chapter, we first introduce the purpose of Automatic Modulation Classification (AMC). We then introduce the classification of AMC methods with motivations, and state our contributions. The structure of the thesis is outlined at the end of this chapter.

1.1 Background

Radio spectrum is scarce. To use the radio spectrum efficiently, we need to monitor and understand the spectrum resource usage. Effective use of radio spectrum calls for developing advanced algorithms to share the available spectrum dynamically. To achieve this goal, we are required to extract useful information from complex radio signals over a wide frequency range [1].

Cognitive Radio (CR) can mitigate the long-standing spectrum scarcity problem [2]. CR automatically detects available channels in the wireless spectrum to utilize unused spectrum. In CR, as an important element in determining the throughput, robustness, and overall implementation overhead, the modulation type is automatically determined according to the external environments.

As an intermediate step between signal detection and demodulation, Modulation Recognition (MR) is the task of detecting the modulation type and modula-

tion order of a received radio signal [3]. AMC is a task to complete the MR task in an autonomous way using limited prior knowledge within a short observation interval.

Adaptive modulation uses different modulation schemes, such as different orders of Quadrature Amplitude Modulation (QAM) and Phase Shift Keying (PSK) according to changing channel conditions. In the adaptive modulation, pilot symbols are commonly used to help demodulate in fading environments [4]. Under a limited power budget, applying AMC to adaptive modulation without pilot tone can increase spectral efficiency without sacrificing the Bit Error Rate (BER) performance [5].

AMC has numerous civilian and military applications [1, 3, 6–9]. For civilian applications, AMC is essential for signal sensing for cooperative communication and spectrum interference monitoring. For military applications, AMC provides added advantages for signal interception, jamming and localization of a hostile signal in electronic warfare and surveillance.

1.2 AMC Methods

In general, MR can be formulated as a classification problem. Various MR approaches in traditional wireless networks can be divided into two broad categories: Likelihood-based Method (LBM) and Feature-based Method (FBM) [1]. Recently, many researchers have also applied Deep Learning-based Method (DLM) to AMC, since DLM is simple to design, robust to model mismatch and less dependent on prior features [6, 7, 10].

1.2.1 LBM

In LBM, the MR is presented as multiple composite hypothesis-testing problems. LBM builds a probabilistic model for the received signal with a proper decision criterion, where the modulation type having the largest likelihood value is the

[1]The LBM and FBM are also known as the decision-theoretic method and statistical pattern recognition method, respectively.

identified output. LBM assumes that the Probability Density Function (PDF) of the transmitted signal contains all information for classification. Even though LBM can achieve the highest recognition rate in the Bayesian sense for a given model, LBM has the following four disadvantages [6].

First, obtaining accurate prior PDF information of the transmitted signal is typically infeasible in most practical scenarios, e.g., non-cooperative communication.

Second, for LBM, it is challenging to obtain an exact closed-form solution for the decision function of this hypothesis-testing problem. Even though such a closed-form solution exists, for example in the PSK classification problem with unknown carrier phase [8], the high computation complexity makes such a classifier impractical [6].

Third, the classification performance of the LBM models is significantly deteriorated in the presence of a model mismatch, which indicates that there is a discrepancy, e.g., unknown channel conditions, and other receiver discrepancies, between the ideal system model of LBM and the true model.

Fourth, the practical implementation of LBM suffers from high computational complexity due to the computation of PDF over unknown channel conditions and the buffering requirement for a large number of samples. To ease the computational complexity, three sub-optimal methods have been proposed based on their assumptions for the unknown parameters [9, 11, 12].

1.2.2 FBM

As a mapping function between features and multi-hypothesis, FBMs normally focus on three major stages: 1) pre-processing; 2) feature extraction; 3) classification.

Pre-processing estimates channel state information, eliminates noise, frequency and phase offset, or transforms inputs into proper forms for better equalization and easier feature extraction.

In the feature extraction stage, five statistical features of the instantaneous

amplitude, phase, and frequency are subsequently extracted [13–21].

- Signal spectral based features [14, 16, 17].
- Wavelet, short-time Fourier transform-based features [17].
- High-order Statics (HOS) based features [14–17, 19, 21]. HOS-based features include cumulants-based and moments-based features, which often have better anti-noise and anti-interference properties.
- Cyclo-stationary analysis-based features [14, 16, 17, 20].
- Graph-based cyclic-spectrum analysis features, such as the constellation diagram [18].

For the classification process, the existing classifiers include various techniques from the decision tree [16, 19], lightweight support vector machine [21], K-Nearest Neighborhood (KNN) [15] to artificial Neural Network (NN) [14, 20].

Compared with LBM, FBM can achieve reasonable probability of correct classification (P_c) with favorable computational complexity. Moreover, since FBM exploits training data to extract features, it is more robust to variations of systems, such as fading, path-loss and time shift.

However, it has been shown that FBM performance is stable only at relatively high Signal-to-Noise Ratio (SNR) in an Additive White Gaussian Noise (AWGN) channel, and is dependent on the number of fine-designed features [14–17]. The manual selection of features [19–21] is also tedious and makes it impossible to model all changes in time, location, velocity, and propagation conditions of the transmitter or the receiver.

Overall, all these aforementioned LBM and FBM exploit knowledge about the structure of different modulation schemes to formulate the classification rules for AMC. Both methods require intensive processing power to deploy on low-cost distributed sensors. Also, both methods are inflexible to adjust for various environments where we need to extract different features for MR.

1.2.3 DLM

The past decade has witnessed the rapid development of high-performance graphics-card processing power, improved Stochastic Gradient Descent (SGD) methods, and strong regularization techniques in Deep Learning (DL) area. With hardware and software breakthroughs, DL has achieved a series of exciting achievements in natural language processing, computer vision and other pattern recognition tasks. This revolution has also sparked interests in extending DL to other domains, including optimization algorithms for wireless communication to achieve better end-to-end performance [22].

Many research works have been conducted to include DL in AMC [1, 3, 10, 18, 23, 24].

Meng et al. [23] specified an idea of end-to-end Convolutional Neural Network (CNN)-based AMC, which shows a performance superior to FBM and close to ideal LBM. Furthermore, Wang et al. [24] combined a CNN model based on In-phase and Quadrature (IQ) data and another CNN model based on Constellation Diagram (CD) to improve the poor classification of QAM16 and QAM64 in [23]. However, both works are limited to a relatively simple and small dataset, and as a result, their models are not suitable in practical applications and not comparable to other AMC models.

In [3], a public dataset with raw IQ time series radio signals for AMC is generated using GNU Radio[2], and this dataset is named as RadioML dataset. Experiments in [3] show that the proposed DLM based on CNNs are robust to noise and corruption.

Based on the public dataset [3], Kulin et al. [1] exploited three wireless signal representations with CNNs for signal classification problems. The results demonstrated the feasibility of DLM using correct signal representation. However, this work only considered CNN and Densely-connected Neural Network (DNN) for the model structure, and the computational complexity was not discussed [1]. To achieve real-time AMC for varying environmental conditions, the quantized Long

[2]https://www.gnuradio.org/about/

Short-Term Memory (LSTM)-based model for Amplitude and Phase (AP) signals was proposed by Rajendran et al. [10], where the robustness of LSTM is established for variable sample lengths. Besides regular raw IQ signals or AP signals, Peng et al. [18] also demonstrated the feasibility of constellation diagrams in different DLMs.

To date, a comprehensive performance and complexity comparison of AMC models based on all possible signal representations is still missing. Whilst some research has been carried out on different model structures, e.g., CNN and LSTM, little attention has been paid to ensemble learning. It is also surprising that model pruning [3] has not been investigated in all the aforementioned papers.

Therefore, in this thesis, we investigate five different signal representations of RadioML dataset [3] and propose a corresponding DL-based model for each signal form. Three data pre-processing methods are introduced to improve AMC performance. To validate the performance of the proposed AMC models, we also conduct extensive numerical experiments to compute and depict various performance metrics including computational complexity. Pruning is further introduced to achieve real-time AMC on edge devices having constrained computational resources. Besides pruning, we illustrate the remarkably improved performance of stacking learning. At last, pruned stacking models are proposed for improved performance and realistic complexity in practice.

1.3 Contributions

This thesis proposes real-time pruned stacking AMC models based on RadioML dataset. We summarize the main contributions of this thesis as follows:

- We consider five commonly-used signal representations for AMC: time-domain IQ signal, time-domain AP signal, frequency-domain Fast Fourier Transform (FFT) signal, image-domain Three-dimensional Constellation Image (3D-CI), and the statistical-domain manually extracted 23 features

[3] Model pruning removes the less salient connections in NNs to reduce the number of non-zero parameters with little loss in final model quality.

(denoted as "Features"). Noise reduction, normalization, and label smoothing are conducted for signals in various representations before feeding into the AMC model.

- For AP signal, we combine attention module with LSTMs to add weights for lower features extracted by the LSTM module. For IQ signal, we add two additional multi-scale connections to capture information at different resolutions produced by CNN, LSTM and DNN modules. The well-known MobileNet version 2 [25] is chosen for 3D-CI and simple CNNs are exploited for FFT and Features to control the computation complexity.

- Detailed per-class and overall performance metrics are computed for each AMC model. Memory consumption and detection efficiency are also considered.

- Pruning AMC models with minimal performance degradation is introduced to save the memory resource in practical deployment. The performance comparison of the pruned model with the non-pruned model is also included.

- Stacking with three-fold cross-validation is proposed to combine the strengths of previous single AMC models. We also prune the stacking learned models to ensure efficient storage memory. Results show that the pruned stacking models can achieve superior performance while having a relatively low computation complexity, when compared with single AMC models and conventional AMC models.

1.4 Outline of the Thesis

The rest of the thesis is organized as follows. Chapter 2 presents the signal model, problem formulation, and fundamental information of DL in AMC. Chapter 3 introduces five signal representations and data pre-processing methods. In Chapter

4, we illustrate the AMC model structures and experiment setup. Chapter 5 summarizes the performance metrics and evaluates the performance of AMC models. In Chapter 6, pruning and stacking are conducted for improved end-to-end performance. Conclusions and future work are presented in Chapter 7.

Chapter 2

System Model and Machine Learning Basics for AMC

2.1 Signal Model

This section describes the system model and formulates the AMC problem. The processing pipeline for wireless communication system with AMC model is illustrated in Figure 2.1. A wireless signal model consists of a transmitter, a receiver and a channel model at the system level. The AMC module is an intermediate process that occurs between signal detection and demodulation at the receiver.

2.1.1 Transmitter

The transmitter transforms a stream of source information bits $b_k \in \{0, 1\}$ into transmission signal $s(t)$. After coding and modulation, the message is mapped to a discrete waveform or signal denoted by s_k via a pulse-shaping filter. The Digital-to-analog (D/A) converter module transforms s_k into the continuous baseband signal $s_b(t)$.

The real-valued bandpass signal $s(t)$ having carrier frequency f_c can be written

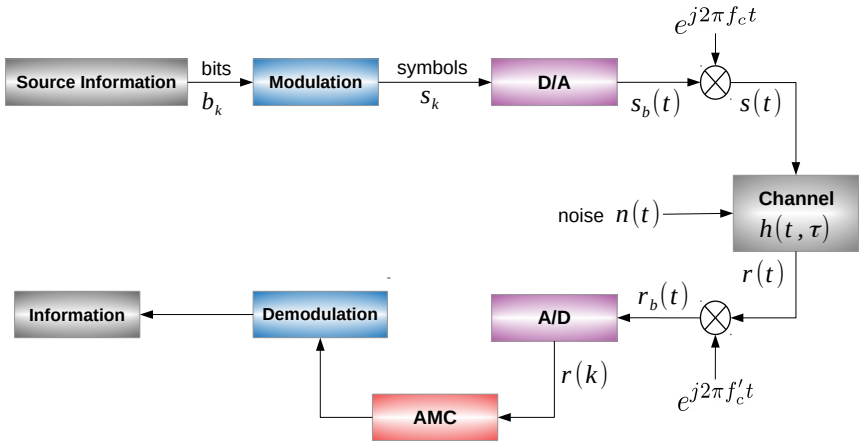


Figure 2.1: Simplified block diagram of signal model processing chain

as [1]

$$\begin{aligned} s(t) &= \Re\{s_b(t)e^{j2\pi f_c t}\} \\ &= \Re\{s_b(t)\}\cos(2\pi f_c t) - \Im\{s_b(t)\}\sin(2\pi f_c t) \end{aligned} \tag{2.1}$$

where $s_b(t) = \Re\{s_b(t)\} + j\Im\{s_b(t)\}$ is the baseband complex envelope of $s(t)$.

2.1.2 Wireless Channel

The channel effects are modelled as a linear time-varying bandpass channel impulse response $h(t,\tau)$. A general channel output $r(t)$ under $h(t,\tau)$ is

$$r(t) = s(t) * h(t,\tau) + n(t) \tag{2.2}$$

where $n(t)$ is AWGN having mean zero and variance σ_n^2, and $*$ denotes the convolutional operation.

Equation 2.2 is widely used in traditional expert features computation. However, the practical input/output relation is more complicated with a frequency-

selective fading channel and imperfect receiver hardware. For more details, please refer to Section B.1.

2.1.3 Receiver

The relationship of $r(t)$ and its baseband complex envelope $r_b(t)$ is given by

$$r(t) = \Re\left\{r_b(t)e^{j2\pi f_c t}\right\} \tag{2.3}$$

$$r_b(t) = (s_b(t) * h_b(t,\tau))\frac{1}{2}e^{j(2\pi f_0 t+\phi(t))} + n(t) \tag{2.4}$$

where f_0 is frequency offset of transmitter local oscillator frequency f_c and receiver local oscillator frequency f_c', $\phi(t)$ is the phase offset, and $h_b(t,\tau)$ is the baseband channel.

Let $r(k)$ denote the discrete-time observed signal at sampling index k after the received signal is amplified, mixed, low-pass filtered, and passed through the D/A converter module. After sampling $r_b(t)$ at time $\frac{k}{f_s}$ where f_s is the sampling rate, $r(k)$ is given by

$$r(k) = r_b(t)|_{t=k/f_s}, \quad -\infty < k < +\infty. \tag{2.5}$$

The input $r(k)$ to AMC is presented in a set of IQ complex form $I + jQ$ due to its flexibility and simplicity for mathematical operations and algorithm design. The expression of $I + jQ$ form is given by

$$r(k) = r_I(k) + jr_Q(k), \quad k = 0,\cdots,L-1 \tag{2.6}$$

where L is the sample length that specifies the number of received signal samples.

2.2 Problem Formulation

In general, AMC can be treated as a multi-class classification problem. The objective of supervised AMC is to produce the probabilities $P(s(k) \in \Theta_m|r(k))$, where

Θ_m represents the mth class in modulation schemes pool $\boldsymbol{\Theta}$, which is defined as

$$\boldsymbol{\Theta} = \{\Theta_m\}_{m=1}^{M} \tag{2.7}$$

where M is the number of possible modulation schemes.

Let $P_r(f_{AMC}(r(k)) = m'|\mathcal{H}_m)$ denote the probability of detecting the mth modulation format of the transmitted signal as the m'th modulation format. $f_{AMC}(\cdot)$ is a classification function of the AMC classifier. $f_{AMC}(r(k))$ denotes the modulation format estimated by the classifier according to the received signal $r(k)$. $\mathcal{H}_m$ is the hypothesis that the transmitted sequence $s(t)$ is generated from Θ_m.

We suppose each modulation type Θ_m has equiprobability. The typical AMC approaches maximize the average probability of correct classification P_c in a short observation interval for a wide range of SNR values, and P_c is defined as [6]

$$P_c = \frac{1}{M}\sum_{m=1}^{M} P_r\left(f_{AMC}(r(k)) = m|\mathcal{H}_m\right) \tag{2.8}$$

2.3 Machine Learning in AMC

Machine learning trains a parametric data-driven model from historical data without using explicit instructions, but relying on patterns and inference instead. For a training dataset $D = \{(\mathbf{x}_1, y_1),(\mathbf{x}_2, y_2),\ldots,(\mathbf{x}_N, y_N)\}$, $\boldsymbol{x}_i$ is the i-th received sequence and y_i is the corresponding modulation scheme index. We assume examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ are Independent and Identically Distributed (I.I.D). Each y_i was generated by an unknown function $y_i = h(\boldsymbol{x}_i)$. The goal of machine learning algorithm is to discover a function f that approximates the true function h.

The input signal matrix $\mathbf{X}$ and the output label vector $\mathbf{Y}$ are denoted by

$$\mathbf{X} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_N^T]^T \in \mathbb{R}^{N\times L} \tag{2.9}$$

$$\mathbf{Y} = [y_1, y_2, \ldots, y_N]^T \in \mathbb{R}^N \tag{2.10}$$

where $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,L}]^T \in \mathbb{R}^L$ for $i = 1, \ldots, N$ is the received signal or the feature vector at the ith observation, L is the signal sample length, and N is the number of samples.

For continuous output variable $y_i \in \mathbb{R}$, f is called a regressor; for categorical output variable $y_i \in \{1, \ldots, M\}$, f is described as a classifier. Therefore, our proposed model is called a modulation classifier.

For a new signal $\boldsymbol{x}_{new}$ in the testset, the modulation predictor is given by $\hat{y} = f(\boldsymbol{x}_{new}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents the AMC model parameters. The estimation of $\boldsymbol{\theta}$ is an optimization problem regarding the training loss $J(\boldsymbol{\theta})$ averaged over all training examples. Hence, $\boldsymbol{\theta}$ is computed as

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} J(\boldsymbol{\theta}) = \frac{\sum_{i=1}^{N} \mathcal{L}(\boldsymbol{x}_i, y_i, \boldsymbol{\theta})}{N} \tag{2.11}$$

where $\mathcal{L}(\boldsymbol{x}_i, y_i, \boldsymbol{\theta})$ is the point-wise loss function of the true modulation scheme y_i and the predicted modulation label $f(\boldsymbol{x}_i; \boldsymbol{\theta})$.

2.4 DL in AMC

2.4.1 Classic NN

Based on the model of human brain neurons, classic NN projects the input signal sequence space into a linearly separable space by feeding weighted sum of inputs into a non-linear activation function $f_{act}(\cdot)$. Let the input and output vector of the classic NN layer l be x^{l-1} and x^l, then the mathematical representation of x^l can be described as

$$\boldsymbol{x}^l = f_{act}(\boldsymbol{W}^l \boldsymbol{x}^{l-1} + \boldsymbol{b}^l) \tag{2.12}$$

where weight matrix $\boldsymbol{W}$ has the dimension $N^l \times N^{l-1}$, the bias $\boldsymbol{b}$ has the dimension $N^l \times 1$, the activation function f_{act} is applied element-wise N^l, and the number of trainable parameters is $N^l \times N^{l-1} + N^l$.

2.4.2 CNN

CNN uses a cascade of multiple hidden layers with non-linear logistic functions to transform high-level pertinent information directly from the original data into a manageable reduced dimension representation.

Derived from feed-forward NNs, CNN replaces general matrix multiplication with convolution. CNN is more memory-efficient and invariant to various data transformations with the characteristics of parameters sharing and local connection.

- Parameters sharing: Different from the Equation 2.12, CNN uses the convolutions to compute the output neuron. Each hidden neuron in CNN has the same shared weight matrix and bias connected to its local receptive field. Compared with the fully-connected NN, CNN shares parameters between different neurons to reduce parameter storage and enforce translation invariance [7].

- Local connection: Classic NN connects every input neuron to every hidden unit, while CNN achieves sparse connectivity by connecting local receptive fields. The local receptive field slides across the entire input matrix with certain movement step size known as stride length.

CNN relies on back-propagation with SGD to extract low-level features from raw inputs and higher-level features from previous layers.

In general, there are three typical layers in CNN architectures: convolutional layer, activation layer, and fully-connected or dense layer. Please refer to Appendix A for details.

2.4.3 Prevent Overfitting in DL

There are many modern techniques (such as pooling and dropout) that can be applied to prevent overfitting.

Pooling layer or down-sampling layer reduces the input dimensionality by computing the average value or the maximum value of a windowed input vector (average-pooling or max-pooling), which is defined as

$$\boldsymbol{x}^{l+1} = f_{pool}(\boldsymbol{x}^{l}) \tag{2.13}$$

where $f_{pool}(\cdot)$ is the pooling function.

Based on the fact that the exact locations of found features is not as important as the rough location relative to other features, a pooling layer produces a condensed feature map by throwing away the exact position information. After pooling, $\boldsymbol{x}^{l+1}$ is more computational efficient and robust to the small translations of $\boldsymbol{x}^{l}$.

Apart from pooling, dropout is also commonly applied to prevent overfitting. Dropout neglects the updating of part nodes' weights and results in more independent nodes in DL models.

2.4.4 Loss function in DL

Back-propagation is the optimization process to update the parameters effectively and iteratively, such as Equation 2.14 in traditional machine learning. In AMC experiments, the Adam optimizer [26] is utilized. Under Adam optimizer, the parameter θ_{n+1} is updated as

$$\theta_{n+1} = \theta_n - \eta \frac{\partial \mathcal{L}(\boldsymbol{Y}, f_{AMC}(\boldsymbol{X}, \theta_n))}{\partial \theta_n} \tag{2.14}$$

where η is the learning rate for Adam optimizer and $\mathcal{L}(\cdot)$ is the chosen loss function.

In the multi-class classification problem, the cross-entropy loss function is

commonly introduced to measure the deviation between the desired and actual output across an entire layer, and this loss function can be expressed as:

$$\mathcal{L}(y, \hat{y}) = -\frac{1}{M}\sum_{i=0}^{M} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \tag{2.15}$$

where y_i represents the true label, $\hat{y}_i = f_{AMC}(\boldsymbol{x}_i)$ is the predicted probability of the ith class by the model f_{AMC}, and M is the number of modulation categories.

2.5 Summary

In this chapter, we firstly describe the wireless communication system model and formulate the AMC problem. In general, AMC is a multi-class classification problem. The fundamentals of ML and DL for AMC problem are discussed in Section 2.3 and Section 2.4. For this specific multi-class classification problem, the AMC problem, we choose the cross-entropy function as the loss function.

Chapter 3

Signal Representations and AMC Models

Since the RadioML 2016.10a has been widely used to evaluate the AMC models, we choose the RadioML 2016.10a dataset to compare fairly with the benchmark models without generating new datasets. The simulation setup to obtain the data samples is presented in Appendix B.

In this chapter, the five forms of signal representation and three data preprocessing methods are reviewed for AMC. In Section 3.3, we correspondingly tailor the different types of DL-based AMC models for the five forms of signal representation. At last, the dataset split and implementation details are introduced in Section 3.4.

The notations in this chapter are as follows:

- $\boldsymbol{y}_i$: The ith one-hot encoding label vector[1].
- $\boldsymbol{x}_i$: The ith complex signal with L data points.

[1]In AMC, each signal is associated with categorical data, i.e., the corresponding modulation scheme. One-hot encoding quantifies the categorical data into numerical data. One-hot encoding produces a vector with length equal to the number of categories in the dataset. If an element belongs to the ith category, then the elements are assigned with a value of 0 except for the ith element, which is assigned a value of 1 [27].

- $\boldsymbol{x}_i^I$: The ith in-phase signal with L data points. Each element of $\boldsymbol{x}_i^I$ is represented by $x_{i,j}^I$, where $j \in \{1, 2, \cdots L\}$.
- $\boldsymbol{x}_i^Q$: The ith quadrature signal with L data points. Each element of $\boldsymbol{x}_i^Q$ is represented by $x_{i,j}^Q$, where $j \in \{1, 2, \cdots L\}$.
- $\boldsymbol{x}_i^A$: The ith amplitude vector of complex signal $\boldsymbol{x}_i$. Each element of $\boldsymbol{x}_i^A$ is represented by $x_{i,j}^A$, where $j \in \{1, 2, \cdots L\}$.
- $\boldsymbol{x}_i^P$: The ith phase vector of complex signal $\boldsymbol{x}_i$. Each element of $\boldsymbol{x}_i^P$ is represented by $x_{i,j}^P$, where $j \in \{1, 2, \cdots L\}$.
- $\boldsymbol{x}_i^F$: The ith frequency domain signal of $\boldsymbol{x}_i$. Each element of $\boldsymbol{x}_i^F$ is represented by $x_{i,k}^F$, where $k \in \{1, 2, \cdots L\}$.

3.1 Signal Representations

3.1.1 Time-Domain IQ Signal

When the RadioML dataset is used, an IQ data sample consists of N time-domain complex IQ signals. Denote the ith L-dimensional in-phase term by $\boldsymbol{x}_i^I$, and the ith L-dimensional quadrature term by $\boldsymbol{x}_i^Q$, an IQ sample is given by

$$D^{IQ} = \left\{(\boldsymbol{x}_i^I, \boldsymbol{x}_i^Q), \boldsymbol{y}_i\right\}_{i=1}^{N}. \tag{3.1}$$

3.1.2 Time-Domain AP Signal

When the jth term of the ith in-phase vector is $x_{i,j}^I$, and the jth term of the ith quadrature vector is $x_{i,j}^Q$, we respectively define the terms $x_{i,j}^A$ and $x_{i,j}^P$ as

$$x_{i,j}^A = Amplitude(x_{i,j}) = \sqrt{\left(x_{i,j}^I\right)^2 + \left(x_{i,j}^Q\right)^2} \tag{3.2}$$

and

$$x_{i,j}^{P} = Phase(x_{i,j}) = \arctan\left(x_{i,j}^{Q}/x_{i,j}^{I}\right). \tag{3.3}$$

Therefore, each IQ sample can be transformed to an AP sample in the polar coordinate as

$$D^{AP} = \left\{\left(\boldsymbol{x}_i^A, \boldsymbol{x}_i^P\right), \boldsymbol{y}_i\right\}_{i=1}^{N}. \tag{3.4}$$

3.1.3 Frequency Spectrum Signal

We perform the one-dimensional (1D) L-point Discrete Fourier Transform (DFT) with the efficient FFT algorithm over the IQ sample. Since computing the L-point DFT requires $O(L^2)$ arithmetic operations, which are time-consuming. Therefore, the FFT algorithm is proposed to exploit symmetries in signals to reduce the complexity to $O(L \log L)$.

Performing the FFT operation over the ith complex IQ signal, we obtain a complex vector $\boldsymbol{x}_i^F$. More specifically, the kth element of $\boldsymbol{x}_i^F$ is obtained as

$$x_{i,k}^{F} = \sum_{p=1}^{L} x_{i,p} \cdot e^{-\frac{j2\pi}{L}kp} \qquad k = 1, \ldots, L. \tag{3.5}$$

Therefore, a frequency spectrum sample is given by

$$D^{FFT} = \left\{(\Re\{\boldsymbol{x}_i^F\}, \Im\{\boldsymbol{x}_i^F\}), \boldsymbol{y}_i\right\}_{i=1}^{N}. \tag{3.6}$$

3.1.4 Image-Domain 3D-CI

In the constellation diagram, the horizontal x-axis is the in-phase term of complex IQ signals, and the vertical imaginary y-axis is the quadrature term of complex IQ signals. An L-dimensional complex IQ signal corresponds to the L points in the constellation diagram. More specifically, the jth point is represented as $(x_{i,j}^I, x_{i,j}^Q)$ where $x_{i,j}^I$ is the jth in-phase term of the ith complex IQ signal, and $x_{i,j}^Q$ is the ith quadrature term of the ith complex IQ signal. Unless otherwise specified, we

select the CD having a size of 0.015×0.015 for the RadioML dataset to avoid the overlapping of constellation points and include as more signal points as possible.

The carrier phase shift from a reference phase is equal to the counterclockwise angle of the constellation point from the horizontal x-axis. The distance of a constellation point to the origin represents the signal amplitude. The distance between different points indicates the ability of a receiver to differentiate modulation schemes under additive noise. In practice, the CDs are usually a "cloud" of points surrounding each symbol position. Since the different "cloud" areas in CD have different densities of sample points, we use the differences in density to make the disturbing signals more discernible. Therefore, we convert the traditional 2D-CD into the 3D-CI where the third dimension is the signal density.

Density estimation techniques consist of mixture models and neighbor-based approaches e.g., the non-parametric Kernel Density Estimation (KDE). We use KDE to reconstruct the probability density function for the 3D-CI.

Assuming the observed signal $\{x_{i,j}\}_{j=1}^{L}$ is univariate i.i.d, we want to estimate the underlying unknown PDF. Mathematically, the formal definition of kernel density estimator at a point z within $\{x_{i,j}\}_{j=1}^{L}$ is given by

$$\widehat{\rho}_h(z) = \frac{1}{Lh} \sum_{j=1}^{L} K\Big(\frac{z - x_{i,j}}{h}\Big) \tag{3.7}$$

where $K(\cdot)$ is the non-negative, smooth and symmetric kernel function controlled by the bandwidth parameter h.

We choose the common Gaussian kernel function, which is defined by

$$K(x; h) \propto e^{-\frac{x^2}{2h^2}} \tag{3.8}$$

where the smoothing bandwidth h controls the trade-off between the bias and variance of the estimator. More specifically, the smoothness of $\widehat{\rho}_h(y)$ increases with h. With the estimated PDF $\widehat{\rho}_h(y)$, we can convert CD into colorful 3D-CI. Figure 3.1 illustrates the CD, 3D-CI, and 3D-CI with noise reduction of 30,000

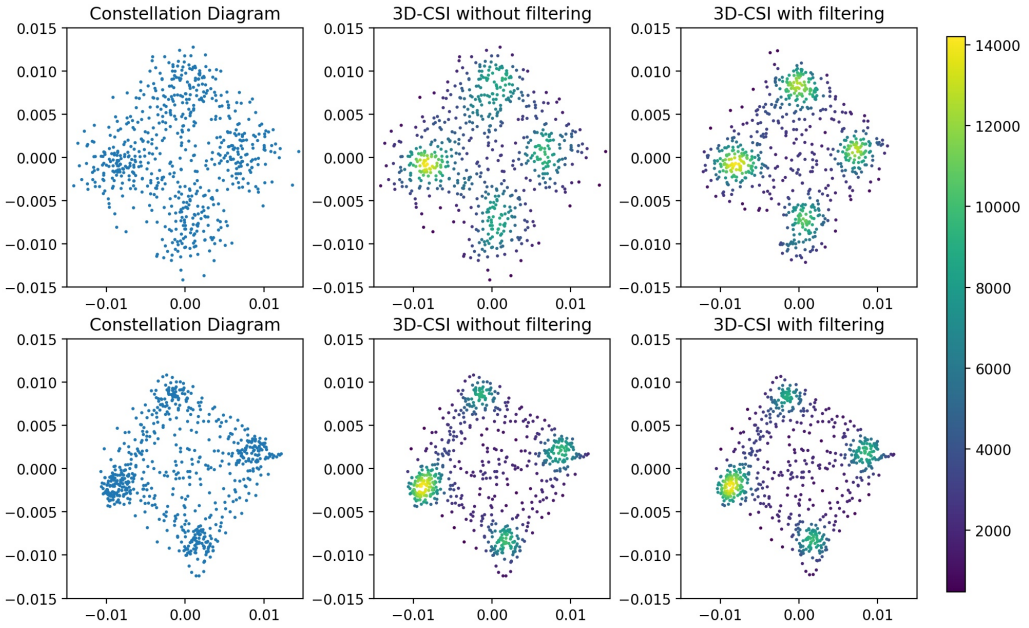


Figure 3.1: 30,000 QPSK constellation points at SNR = 2dB (upper) and 14 dB (lower)

QPSK samples when the values of SNR are 2 dB and 14 dB.

At high SNR, CD and 3D-CI have already revealed enough statistics information for AMC, and the noise reduction has little improvement. Overall, both CD and 3D-CI can reveal statistical modulation information at high SNR. However, at low SNR, CD polluted by noise is disguised, and 3D-CI with color density information can provide richer amplitude and phase information. In addition, noise reduction can concentrate the constellation points to build a clear "cloud". The details of noise reduction will be discussed in Section 3.2.1.

3.1.5 Statistical-Domain Manually Extracted Features

We convert a complex IQ signal into K-dimensional feature vector $\boldsymbol{f}_i$ based on the extensive feature set, which contains the statistical and instantaneous features [28]. Using the time-average features, the negative effects of background noise

and phase rotation can be mitigated [29]. The High-order Moments (HOM) and High-order Cumulants (HOC) vary for different modulated signals, and have good anti-noise performance. Therefore, we include the HOM and HOC as two features. Besides, kurtosis (K), skewness (S), Peak to Average Ratio (PAR) and Peak to Root-mean-square Ratio (PRR) are also included. Table C.1 in Appendix C lists the selected 23 statistical and instantaneous features [30–32].

3.2 Data Preprocessing

Before feeding the data samples to the AMC models, several pre-processing operations are performed, namely noise reduction, signal normalization and label smoothing.

3.2.1 Noise Reduction

Since the AWGN term $n(t)$ can compromise the performance of AMC models, we use the Gaussian low-pass filter $f_G(\cdot)$ to reduce the noise before any other data pre-processing method. The filter $f_G(\cdot)$ attenuates high-frequency outliers and keeps the low-frequency details of the received signal. The impulse response of the one-dimensional Gaussian filter $f_G(\cdot)$ with standard deviation σ is given by

$$f_G(x) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot e^{-\frac{x^2}{2\sigma^2}}. \tag{3.9}$$

Mathematically, the input signal is convolved with the Gaussian function $f_G(x)$. In theory, $f_G(x)$ is non-zero everywhere, which would require an infinitely large convolution kernel. In practice, $f_G(x)$ is effectively zero more than about four standard deviations from the mean, and therefore we truncate the convolution kernel at this point.

The effectiveness of Gaussian low-pass filtered signals can be found in Figure 3.1. Experiments have demonstrated that setting $\sigma = 0.8$ can remove most outliers and keep more constellation points in 3D-CI signal representation.

3.2.2 Normalization

After passing the Gaussian filter, we normalize the training samples to robustify the automatic feature extraction. The normalization is to perform a linear operation to the original data samples such that the normalized data samples have zero mean and unit variance. For example, the normalization operation to the jth point of the ith signal is given by

$$x'_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}, \quad j \in \{1,2,\cdots L\} \tag{3.10}$$

where the jth element of the mean vector, denoted by μ_j, is obtained as

$$\mu_j = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} x_{i,j}, \quad j \in \{1,2,\cdots L\} \tag{3.11}$$

and the jth element of the standard deviation vector, denoted by σ_j, is obtained as

$$\sigma_j = \sqrt{\frac{1}{N_{train}} \sum_{i=1}^{N_{train}} (x_{i,j} - \mu_j)^2}, \quad j \in \{1,2,\cdots L\}. \tag{3.12}$$

Here, N_{train} is the number of training signals, and the values of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are computed over the training set and fixed for the test set[2].

3.2.3 Label Smoothing

In our experiments, the modulation scheme is represented by one-hot encoding and later modified by label smoothing. With label smoothing, the difference of true label value and wrong label value becomes a constant relying on a smoothing parameter α. Moreover, the activation values of the penultimate layer of the model are close to the true class and equidistant to the wrong classes [33]. Hence, the model with label smoothing is automatically calibrated and less overfitting. For

[2]This trick can accelerate the training process and prevent overfitting.

the mth modulation, we smooth the traditional label vector y_m by

$$\boldsymbol{y}'_m = \boldsymbol{y}_m(1-\alpha) + \frac{\alpha}{M} \tag{3.13}$$

where the smoothing parameter α equals 0.1.

3.3 Corresponding AMC Model for Each Signal Representation

Based on the five forms of signal representation in Section 3.1, we correspondingly introduce five models by tailoring different types of neural networks as

- Attention-based model of AP signal (AP-Attention).
- Combined CNN, LSTM and DNN model with multi-scale additions of IQ signals (IQ-CLDNN).
- MobileNet version 2 model of 3D-CI images (Img-MobileNetV2).
- CNN-based model of statistical features (Features-CNN).
- CNN-based model of frequency spectrum signals (FFT-CNN).

Among the five models, we propose AP-Attention and IQ-CLDNN models for the first time. Though the MobileNet model has achieved great success in the discipline of computer vision, the performance is unknown in the AMC problems. Therefore, we tailor the MobileNet model for the proposed 3D-CI signals. We consider the Features-CNN and FFT-CNN as benchmarks [1, 17].

3.3.1 AP-Attention Model

The proposed AP-Attention model consists of an LSTM-based module to extract low-level features, an attention-based module to include importance weights, and a classification module to obtain the probabilities for different modulations. The

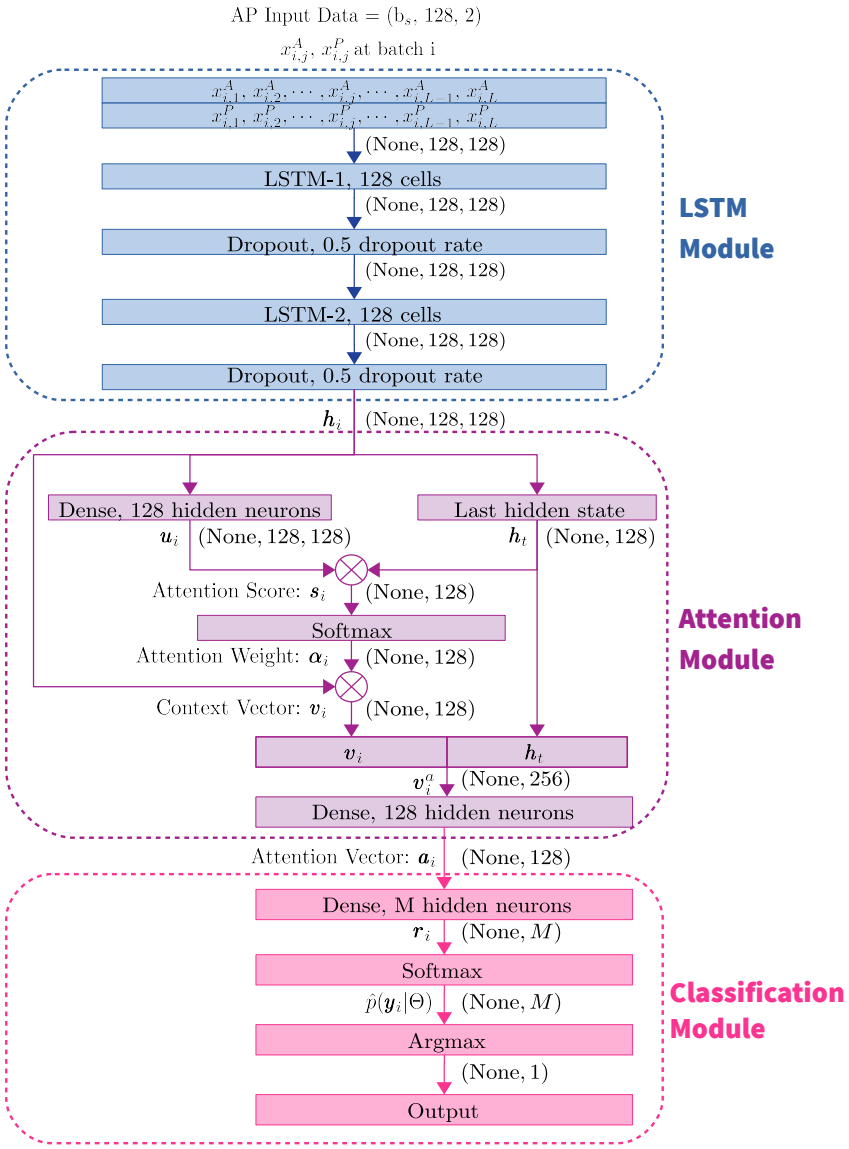


Figure 3.2: AP-Attention model schematic diagram for batch *i*

Table 3.1: LSTM Parameters

Variables	Definition
h	The number of hidden units
d	The number of input features
σ_g	The sigmoid activation function
σ_h	The tanh activation function
$\boldsymbol{x}_t \in \mathbb{R}^d$	The input vector to the LSTM unit
$\boldsymbol{f}_t \in \mathbb{R}^h$	The activation vector of forget gate
$\boldsymbol{i}_t \in \mathbb{R}^h$	The activation vector of input gate
$\boldsymbol{o}_t \in \mathbb{R}^h$	The activation vector of output gate
$\boldsymbol{h}_t \in \mathbb{R}^h$	The hidden state vector of the LSTM unit
$\boldsymbol{c}_t \in \mathbb{R}^h$	The current cell state vector
$\boldsymbol{W} \in \mathbb{R}^{h \times d}$	The weight matrix of input vector $\boldsymbol{x}_t$
$\boldsymbol{U} \in \mathbb{R}^{h \times h}$	The weight matrix of previous hidden state vector $\boldsymbol{h}_{t-1}$
$\boldsymbol{b} \in \mathbb{R}^h$	The trained bias vector

input of the AP-Attention model has a unified size of 128×2 for the AP signals. Figure 3.2 provides the schematic diagram of AP-Attention model. The term "None" in the diagram represents the first dimension of a variable. This dimension is usually ignored when building a model and is defined during the prediction period.

The LSTM module can alleviate the gradient vanishing problem by exploiting a gating mechanism with explicit memory releasing and updating. The variables of the LSTM module are listed in Table 3.1.

Typically, an LSTM block is composed of a cell state vector $\boldsymbol{c}_t$ to record values over arbitrary time intervals and three gates to regulate the information flow. The functionalities of the three gates are [34]:

- Input gate with weight matrices $\boldsymbol{W}_i$, $\boldsymbol{U}_i$, and $\boldsymbol{b}_i$ to control the extent where a new value flows into the cell state.

- Forget gate with weight matrices $\boldsymbol{W}_f$, $\boldsymbol{U}_f$, and $\boldsymbol{b}_f$ to control the extent where the value is kept or discarded from the cell state.

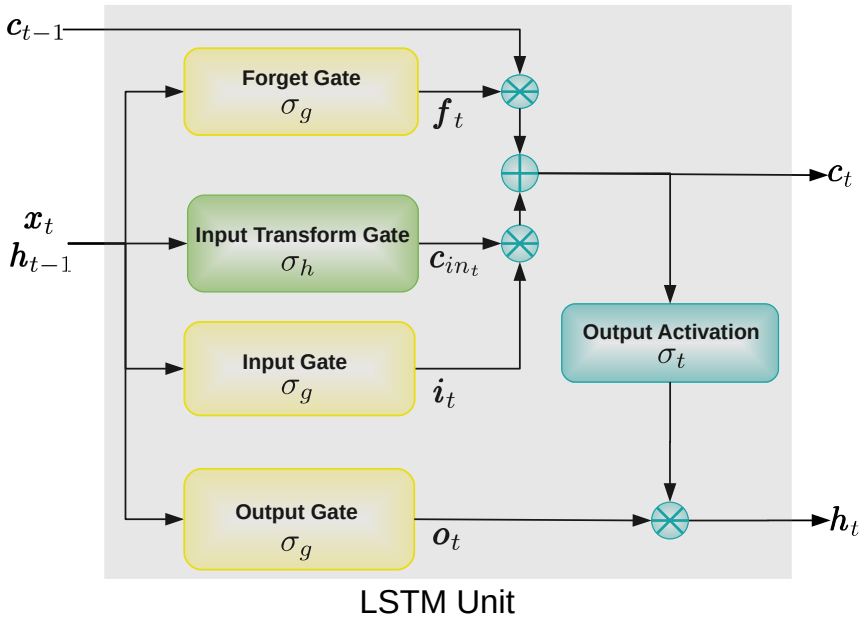


Figure 3.3: LSTM block diagram

- Output gate with weight matrices $\boldsymbol{W}_o$, $\boldsymbol{U}_o$, and $\boldsymbol{b}_o$ to control the extent where the cell value computes the output activation of the LSTM unit.

The mechanism of the LSTM module is described by the following equations using current input $\boldsymbol{x}_t$ and the previous state $\boldsymbol{h}_{t-1}$ [35]:

- Gates

$$\boldsymbol{i}_t = \sigma_g(\boldsymbol{W}_i\boldsymbol{x}_t + \boldsymbol{U}_i\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{3.14}$$

$$\boldsymbol{f}_t = \sigma_g(\boldsymbol{W}_f\boldsymbol{x}_t + \boldsymbol{U}_f\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \tag{3.15}$$

$$\boldsymbol{o}_t = \sigma_g(\boldsymbol{W}_o\boldsymbol{x}_t + \boldsymbol{U}_o\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \tag{3.16}$$

- Input transform

$$\boldsymbol{c}_{in_t} = \sigma_h(\boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{U}_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \tag{3.17}$$

- State update

$$\boldsymbol{c}_t = \boldsymbol{f}_t \circledast \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circledast \boldsymbol{c}_{in_t} \tag{3.18}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \circledast \sigma_t(\boldsymbol{c}_t) \tag{3.19}$$

where the initial values c_0 and h_0 are zero, the subscript t represents the time step, and the operator $\circledast$ denotes the element-wise product. Figure 3.3 presents the LSTM block diagram and illustrates the above equations. The input, output, and forget gate functions are the sigmoid function. The input transform and output activation functions are the tanh function.

In the LSTM module, we exploit two layers of LSTM with a hidden size l_s of 128 along with a dropout layer with a ratio of 0.5 to encode the AP sequences. In AP-Attention model, the input $\boldsymbol{x}_t$ of LSTM module is denoted by $\boldsymbol{x}_i$, where $i \in \{1, \cdots, b_s\}$. Thus, the output of LSTM $\boldsymbol{h}_i \in \mathbb{R}^{L \times l_s}$ is described as

$$\boldsymbol{h}_i = f_{LSTM}(\boldsymbol{x}_i; \theta_{LSTM}) \tag{3.20}$$

where L is the signal length and θ_{LSTM} is the parameters of LSTM module.

The second module in AP-Attention model is the attention module with architecture illustrated in Figure 3.4. We explain Figure 3.4 in the following equations from Equation 3.21 to Equation 3.25.

We first generate the attention score [35]. Let $\boldsymbol{H}$ be the annotation matrix with the LSTM extracted vector $[\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{b_s}]^T$, where b_s is the batch size and $\boldsymbol{h}_i \in \mathbb{R}^{L \times l_s}$ is the LSTM feature vector with sample length L and hidden size l_s. We feed $\boldsymbol{h}_i$ into an additional feed-forward densely-connected network such that the attention score vector $\boldsymbol{u}_i \in \mathbb{R}^{L \times l_s}$ is obtained.

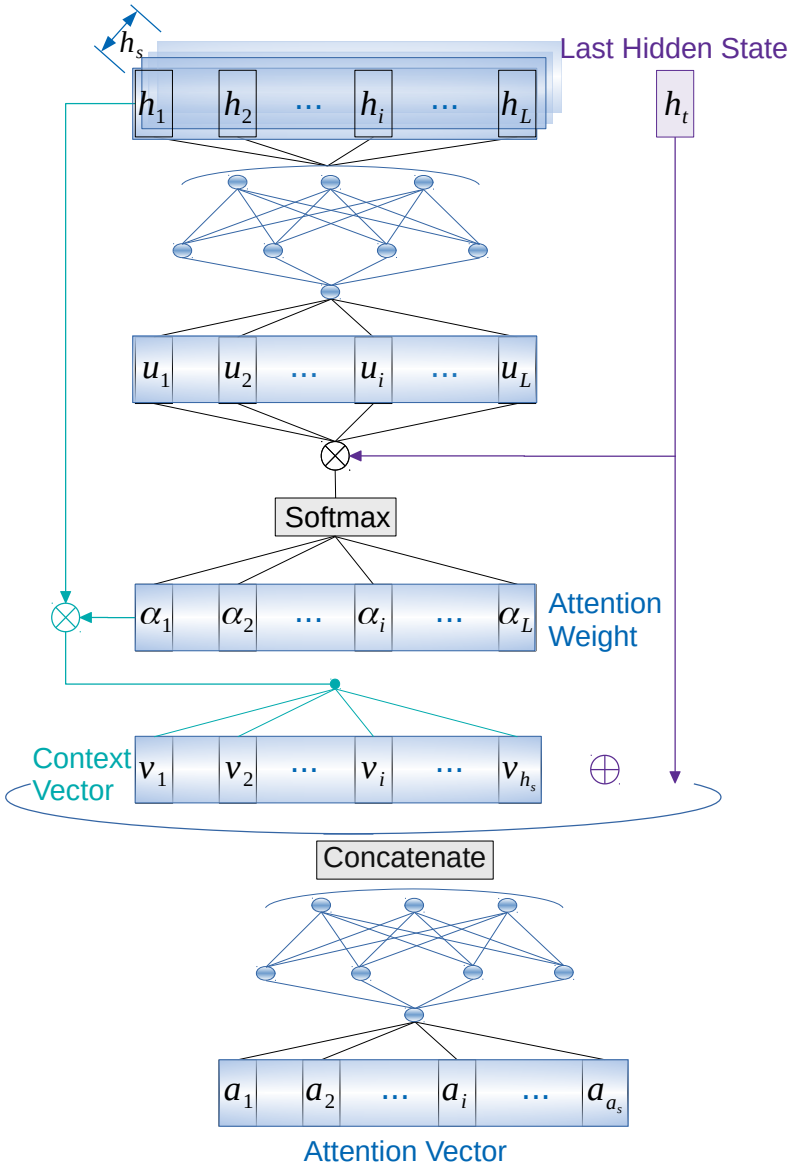


Figure 3.4: Schematic diagram of the attention module

As a hidden representation of $\boldsymbol{h}_i$, the vector $\boldsymbol{u}_i$ is defined as

$$\boldsymbol{u}_i = f_c(\boldsymbol{W}_c \cdot \boldsymbol{h}_i) \tag{3.21}$$

where f_c represents the fully-connected network and $\boldsymbol{W}_c$ is the weight matrix randomly initialized and jointly trained.

The attention score $\boldsymbol{s}_i \in \mathbb{R}^L$ is computed by

$$\boldsymbol{s}_i = \boldsymbol{u}_i \cdot \boldsymbol{h}_t \tag{3.22}$$

where $\boldsymbol{h}_t \in \mathbb{R}^{l_s}$ is the last LSTM hidden state vector and is extracted from the matrix $\boldsymbol{H}$.

With attention scores $\boldsymbol{s}_i$ and softmax activation function $f_{softmax}(\cdot)$, the attention weights $\boldsymbol{\alpha}_i$ is given by $\boldsymbol{\alpha}_i = f_{softmax}(\boldsymbol{s}_i) \in \mathbb{R}^L$. The large values in $\boldsymbol{\alpha}_i$ force the network to focus on the corresponding part in input $\boldsymbol{x}_i$.

After that, we measure the context vector $\boldsymbol{v}_i \in \mathbb{R}^{l_s}$. $\boldsymbol{v}_i$ is a weighted combination of attention weight $\boldsymbol{\alpha}_i$ and $\boldsymbol{h}_i$, which is given by

$$\boldsymbol{v}_i = \boldsymbol{\alpha}_i \cdot \boldsymbol{h}_i. \tag{3.23}$$

We then concatenate the context vector $\boldsymbol{v}_i$ and the last hidden states $\boldsymbol{h}_t$ into an attention output vector $\boldsymbol{v}_i^a \in \mathbb{R}^{l_s}$. At last, the attention vector $\boldsymbol{a}_i \in \mathbb{R}^{l_a}$ with the attention output size l_a is computed by feeding $\boldsymbol{v}_i^a$ into a DNN having 128 neurons,

$$\boldsymbol{v}_i^a = [\boldsymbol{v}_i, \boldsymbol{h}_t] \tag{3.24}$$

$$\boldsymbol{a}_i = f_a(\boldsymbol{W}_a \cdot \boldsymbol{v}_i^a) \tag{3.25}$$

where f_a is the DNN layer with attention weight matrix $\boldsymbol{W}_a \in \mathbb{R}^{l_a \times l_s}$.

In summary, the attention module has full access to the input sequences. Although the weighted combination increases the computational burden, attention

module produces a more targeted and better-performing model.

The last module is the classification module. The classification module takes the attention layer output $\boldsymbol{a}_i$ as the input. We feed $\boldsymbol{a}_i$ into a DNN with M units, where M is the number of modulation schemes. Then a softmax classifier is used to predict the modulation scheme $\hat{y}_i$. The process in the classification module is as follows:

$$\boldsymbol{r}_i = \tanh(\boldsymbol{W}_c \boldsymbol{a}_i + \boldsymbol{b}_c) \tag{3.26}$$

$$\hat{p}(\boldsymbol{y}_i|\Theta) = \text{softmax}\,(\boldsymbol{W}_s \boldsymbol{r}_i + \boldsymbol{b}_s) \tag{3.27}$$

$$\hat{y}_i = \underset{\boldsymbol{y}_i}{\text{argmax}}\, \hat{p}(\boldsymbol{y}_i|\Theta) \tag{3.28}$$

where $\boldsymbol{W}_c$ and $\boldsymbol{b}_c$ are DNN layer parameters, $\boldsymbol{W}_s$ and $\boldsymbol{b}_s$ are softmax layer parameters, and the $\hat{p}(\boldsymbol{y}_i|\Theta)$ represents the probability vector for each modulation scheme.

3.3.2 IQ-CLDNN Model

Speech recognition performance can be improved by combining CNNs, LSTMs, and DNNs in a unified framework [36]. Thus, we propose a CLDNN model to combine CNN, LSTM, and DNN. We also introduce the multi-scale additional connections to include more features. Figure 3.5 shows the structure of the IQ-CLDNN model.

The first module is the CNN module. Passing the input signal to CNNs before LSTMs can help reduce the input variance. Thus the temporal modeling in LSTM is processed on higher-level features extracted by CNNs. Specifically, we choose two 1D-CNNs with 256 and 128 units. The convolutional kernel with size of nine is chosen for each CNN.

After the CNN module, we pass the output to the LSTM module to model temporally the sequence input. We use two layers of LSTM with returned se-

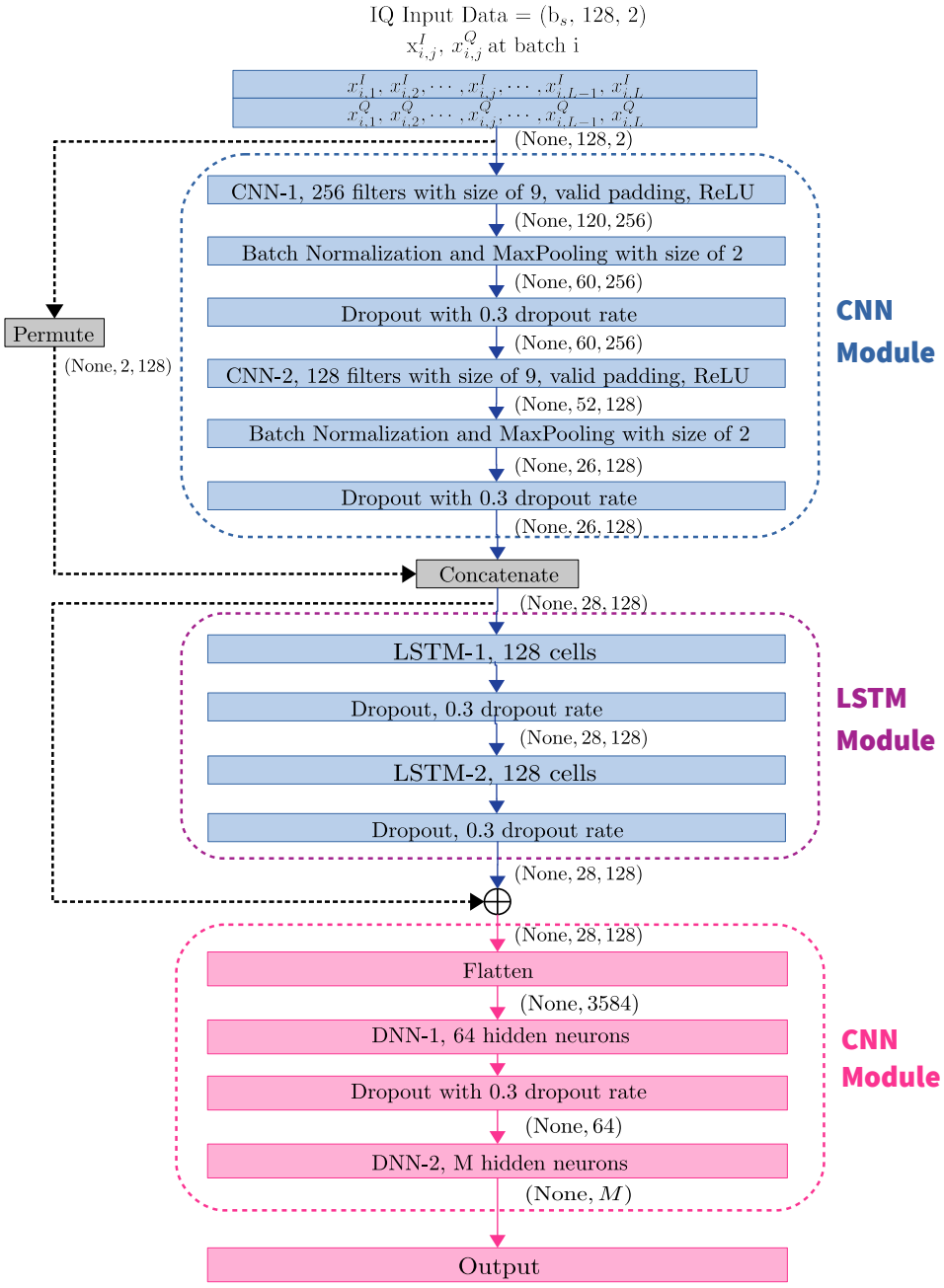


Figure 3.5: IQ-CLDNN model schematic diagram for batch *i*

quences and 128 cells.

A DNN module provides a mapping between hidden units and outputs, which help the AMC model separate different modulation schemes. Therefore, we pass the output of the LSTM module to two DNNs with 64 and 11 hidden units. For activation functions, the first DNN layer uses ReLU and the second DNN layer exploits softmax to output probability scores for each modulation scheme.

Compared with the CLDNN model in [37], we add two additional multi-scale connections to capture information at different resolutions. The additional connections are shown in Figure 3.5 by dashed streams. The "Permute" block changes the input dimensions, and the "Concatenate" block concatenates the two inputs together.

The first additional connection forward the long-term features from the CNN module and the original short-term input features to the LSTM module. The second connection directly passes the outputs from the LSTM module and CNN module into the DNN module without extra layers. The two direct connections increase only a negligible amount of the number of network parameters.

3.3.3 Img-MobileNetV2 Model

In Section 3.1.4, our inputs are batches of Gaussian filtered 3D-CIs with a size of $(224, 224, 3)$. We convert the AMC problem into a multi-class classification problem. The popular MobileNetV2 [25] model is proposed for running deep networks efficiently on personal mobile devices. Hence, we choose the MobileNetV2 model for the real-time AMC problem under resource-constrained environments. The training parameters are provided in Section 3.4.

3.3.4 Features-CNN Model

Statistical features have been proven to be a robust input format for the AMC problem [17]. Based on Section 3.1.5, our input is in the form of $(b_s, 23, 2)$ with 23 manually selected features. To prevent overfitting, we use a simple model with four 1D-CNNs and two DNNs. A dropout layer with a dropout rate of 0.6 is

exploited after each CNN and DNN layer.

The numbers of CNN filters for four 1D-CNN layers are 256, 128, 64, and 64. The classification module has a DNN of 128 neurons with Rectified Linear Unit (RELU) activation function and another DNN of 11 neurons with softmax activation function.

3.3.5 FFT-CNN Model Architecture

Based on the definition in Section 3.1.3, the frequency spectrum data D^{FFT} is in the form of $(b_s, 128, 2)$ with batch size b_s. the FFT-CNN model has three layers of one-dimensional (1D)-CNN and two layers of DNN.

Specifically, the three 1D-CNNs have 256, 128, and 64 filters of the same size of nine. Each 1D-CNN is followed by a batch normalization layer and a dropout layer with a dropout rate of 0.4. After CNNs, the input feature maps are flattened to 1D. The DNN classification module has two DNN layers. The first DNN has 128 neurons and the second DNN has 11 neurons.

3.4 Experiment Setup

3.4.1 Dataset Split

In the RadioML dataset, there are total $N_{SNR} \times M \times L = 20 \times 11 \times 1000 = 220{,}000$ samples. We split the dataset into three parts: training set, validation set, and test set. We randomly selected 50% samples for training with a batch size b_s of 1024, 25% samples for validation. After training, the performance of the AMC model is tested using the remaining 25% samples. All sets are uniformly distributed in different SNR values.

3.4.2 Implementation Details

Benefit from the user-friendliness, modularity and easy extensibility features, we choose the open-source NN library "Keras" to develop our models[3]. With additional support for CNN, Recurrent Neural Network (RNN), and other commonly used layers, Keras allows non-ML researchers to focus on their model construction and training setting.

The AMC models are trained and validated with the central processing unit Intel(R) Core i7-8700 @ 3.20GHz with 12 threads and 16GB RAM.

As mentioned in Section 2.4.4, the categorical cross-entropy measures the probability error, and the Adam optimizer estimates the model parameters with a learning rate of 0.001. The training of AMC models will stop when the validation loss dose not improve for the last 20 epochs. The model having the lowest validation loss is selected for evaluation. All AMC models use the same training parameters unless specified explicitly.

In the Img-MobileNetV2 model, the SGD optimizer with an initial learning rate of 0.1, a momentum of 0.9, and a weight decay of 10^{-4} is used for training. The total number of training epochs is 120. The learning rate is divided by 10 every 40 epochs.

3.5 Summary

In this chapter, we introduce five different signal representations for AMC problem. Before we feed raw signals of different signal representations into AMC models, three data pre-processing methods are conducted in Section 3.2. In Section 3.3, we build a DL-based AMC model for each signal representation. At last, the experiment setting including dataset split and implementation details are introduced in Section 3.4.

[3]https://keras.io/

Chapter 4

Performance of AMC Models

In this chapter, we perform extensive experiments to evaluate the performance of AMC models. Accuracy rate is introduced in Section 4.1, and confusion matrix is introduced in Section 4.2. In Section 4.3, we regard AMC models as soft classifiers or hard classifiers. We also discuss the micro-averaging and macro-averaging algorithms when combing the per-class metrics in Section 4.3. We evaluate the computational complexity in Section 4.4.

4.1 Accuracy Rate

The accuracy rate P_{acc}, also known as the correct probability, assesses the ratio of correct AMC predictions. We evaluate the overall per-class accuracy rate using different SNR values to gain insights on the effective SNR range of each AMC model. The overall accuracy rate is defined as the ratio of correct predictions over the total samples

$$P_{acc} = \frac{N_c}{N_{test}} \tag{4.1}$$

where N_c is the number of correct predictions, and N_{test} is the number of samples in test set.

For each modulation scheme, the per-class correct accuracy rate is defined as

$$P_{acc}^{m} = \frac{N_c^m}{N_{test}^m} \tag{4.2}$$

where N_c^m is the number of correct classifications for modulation scheme Θ_m, and N_{test}^m is the number of samples with modulation scheme Θ_m in test set.

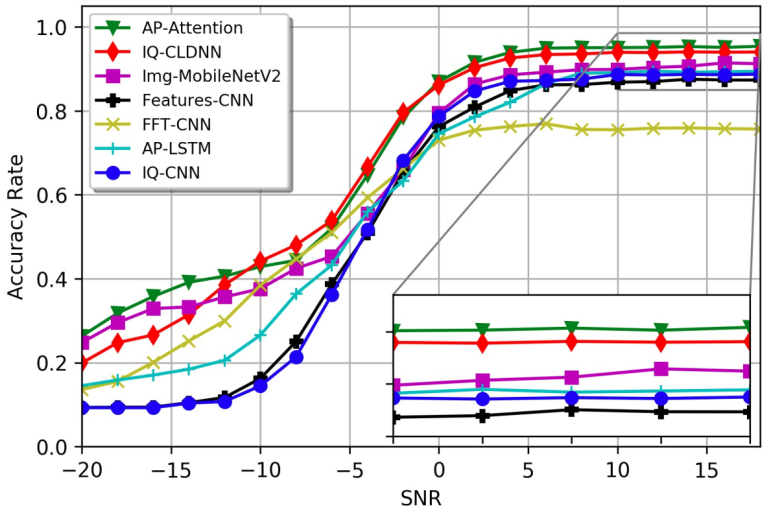


Figure 4.1: Overall accuracy rate of AMC classifiers under various SNR values.

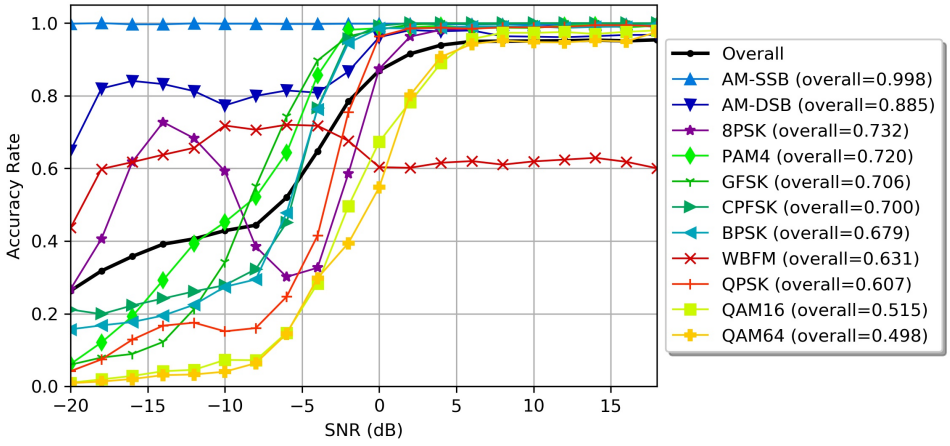


Figure 4.2: Accuracy rate of AP-Attention model

4.1.1 Overall Accuracy Rate

Figure 4.1 depicts the overall accuracy rate P_{acc} over various SNR values for five AMC models in Section 3.3 (AP-Attention, IQ-CLDNN, Img-MobilenetV2, features-CNN, and FFT-CNN) and previous AMC models (AP-LSTM [10] and IQ-CNN [3]). We use the same signal representation and model structure in [3, 10] for comparison. Seven AMC models are trained, validated, and tested on the same dataset. Compared with the results in [1, 3, 10, 17], noise reduction and label smoothing help the model discriminate modulation schemes better.

At high SNR, AP-Attention achieves a higher P_{acc} than the other models. Moreover, AP-Attention, IQ-CLDNN, and Img-MobilenetV2 achieve a stable P_{acc} over 86% for most modulation schemes in Figure 4.2. On the contrary, the AMC models based on statistical features and frequency-domain features require at least 6 dB to converge and have low converged P_{acc} values, especially for FFT-CNN model with a final P_{acc} less than 80%. Hence, we conclude that the AP signal representation has the best modulation discrimination ability, which verifies that the signal representation is a key factor for the AMC problem.

Compared with an 87.4% accuracy rate for IQ-CNN model [3], our updated IQ-CNN with noise reduction and label smoothing has a roughly 1.3% improvement under the same model structure. Similarly, for $SNR > 10$ dB, AP-LSTM achieves a nearly 1.2% accuracy gain over the model based on partial high-SNR training data in [10]. This further verifies the efficiency of proposed data preprocessing methods.

4.1.2 Per-class Accuracy Rate

We further investigate the per-class accuracy rate of AP-Attention in Figure 4.2 due to its highest overall P_{acc}. The curves are obtained by averaging various modulation schemes. The value in the brackets represents the P_{acc}^{m} averaged over all possible SNR values from -20 dB to $+18$ dB with step size of 2 dB.

Figure 4.2 shows that AM-SSB is the most easily recognized modulation type with an almost 100% accuracy rate for all considered SNR values. For high SNR

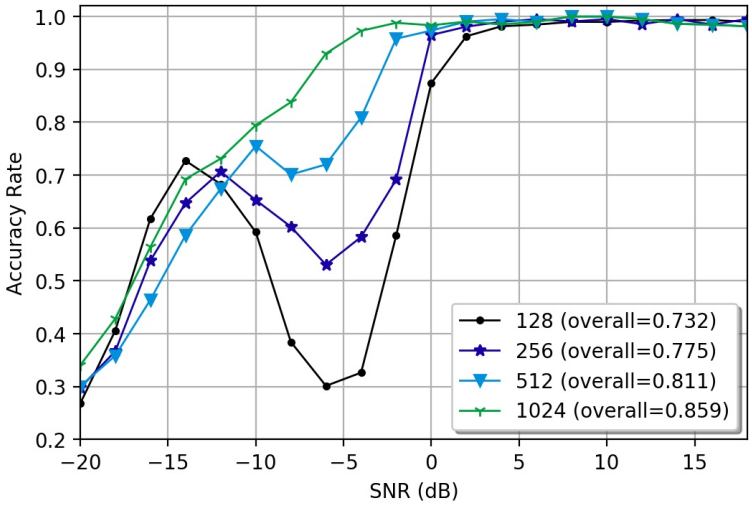


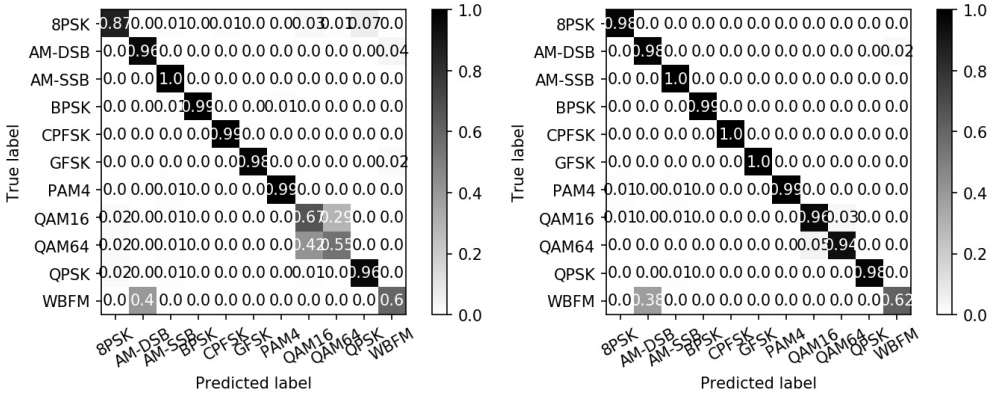
Figure 4.3: Accuracy rate of 8PSK for different signal sample lengths

values (> 5 dB), almost all modulation schemes have P_{acc} over 95%, except for WBFM. It is also found that the accuracy rate of 8PSK has a sudden decrease at low SNR from around −14 dB to −4 dB, which is because that the noise limits the representation learning capability of attention module and results in unstable performance.

Figure 4.3 illustrates that the performance degradation of 8PSK can be alleviated by increasing the dimension of signals. Based on the original RadioML dataset, three new datasets are generated with sample lengths of 256, 512, and 1024, respectively. We train the same AP-Attention model for new datasets and plot their accuracy rates of 8PSK. The results show that the accuracy rate curve grows more steadily and converges faster when we increase the sample dimension L of training samples.

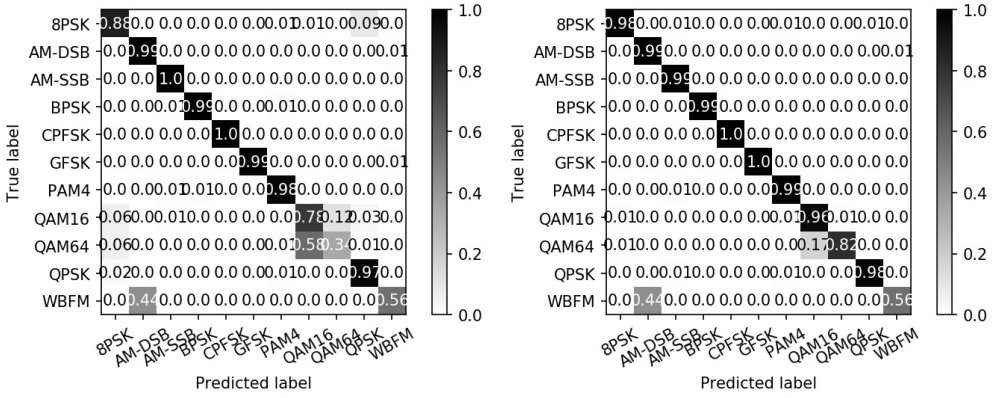
4.2 Confusion Matrix

Since the accuracy rate is unreliable for imbalanced datasets, we evaluate the confusion matrix of AMC models. A detailed overview of the per-class performance



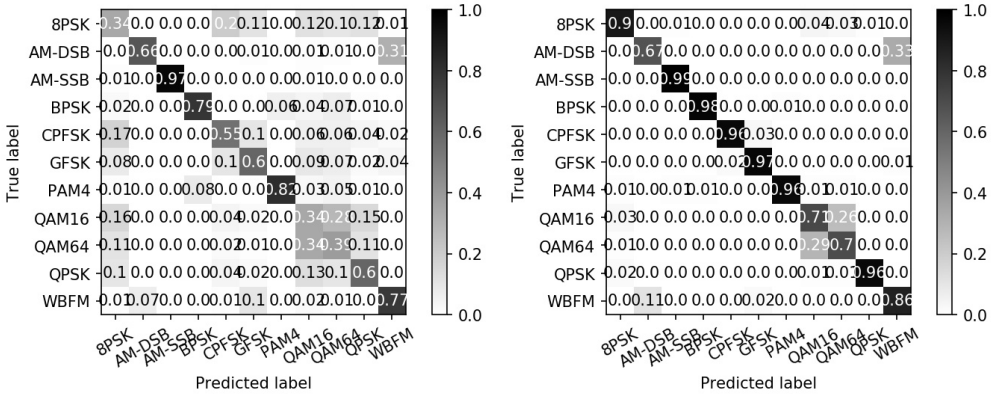
(a) AP-Attention at 0 dB

(b) AP-Attention at 6 dB



(c) IQ-CLDNN at 0 dB

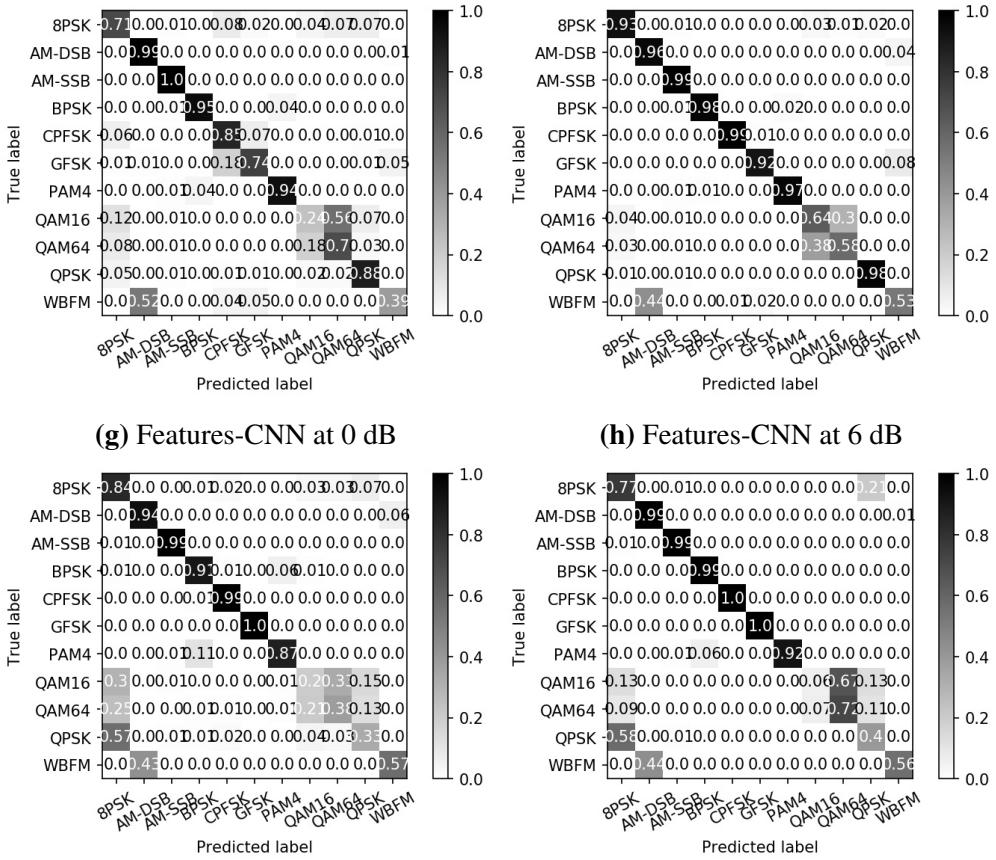
(d) IQ-CLDNN at 6 dB



(e) Img-MobileNetV2 at 0 dB

(f) Img-MobileNetV2 at 6 dB

Figure 4.4: Confusion matrices when SNR is 0 dB or 6 dB



(g) Features-CNN at 0 dB

(h) Features-CNN at 6 dB

(i) FFT-CNN at 0 dB

(j) FFT-CNN at 6 dB

Figure 4.4: Confusion matrices when SNR is 0 dB or 6 dB

is also evaluated and visualized by a confusion matrix for the AMC problem. In an $M \times M$ confusion matrix, the ith row represents the signals with actual Θ_i, while the jth column represents the signals predicted as Θ_j. The (i, j)th entry represents the percentage of signals with Θ_i but predicted as Θ_j. The summation over the ith row is equal to 1. The diagonal element is the correct classification probability for the mth modulation. Therefore, a good classifier will have a clear diagonal in its confusion matrix.

From Figure 4.4a and Figure 4.4b, the main source of error for AP-Attention is the misclassification between QAM16/QAM64 and AM-DSB/WBFM. IQ-CLDNN

tends to misclassify more QAM64 as QAM16 when $SNR = 0$ dB, while AP-Attention discriminates QAM16/QAM64 better with a clear diagonal. This is due to the fact that QAM16 is a subset of QAM64, and it is hard to differentiate them. Separating AM-DSB and WBFM is another challenge for both AP-Attention and IQ-CLDNN.

Compared with other models, Img-MobileNetV2 cannot classify most modulation schemes when SNR is 0 dB, while the performance for SNR at 6 dB is better and comparable to AP-Attention and IQ-CLDNN. The performance of Features-CNN also degrades for low SNR values. Thus, in low SNR regime, 3D-CI and statistical features are not a good choice. From Figure 4.4i and Figure 4.4j, compared with the other AMC algorithms, FFT-CNN has no benefits for QAM16, QAM64, QPSK, and WBFM.

4.3 Performance Metrics for Hard and Soft Classifiers

Accuracy rate and confusion matrix are relatively simple metrics. To obtain a balanced and overall performance description, we further evaluate AMC models from two scopes: hard classifier with a single cutoff and soft classifier with multiple cutoffs. The per-class metrics are combined by two averaging methods: micro-averaging and macro-averaging.

4.3.1 Basic Statistics

We introduce some fundamental statics before defining the performance metrics for hard and soft classifiers. For each sample in the test set, true positive (TP), true negative (TN), false positive (FP) and false-negative (FN) are computed at first. We use them to derive per-class performance metrics: sensitivity, specificity, false positive rate (FPR), false negative rate (FNR), and precision. For simplification, we denote precision by P and recall by R.

- TP: Number of signals with Θ_m and predicted as positive.

- TN: Number of signals not with Θ_m and predicted as negative.
- FP: Number of signals not with Θ_m but predicted as positive.
- FN: Number of signals with Θ_m but predicted as negative.
- Sensitivity, recall, or true positive rate (TPR): Number of signals correctly predicted as positive over total true items

$$Sensitivity = R = TPR = \frac{TP}{TP + FN}. \tag{4.3}$$

- Specificity or true negative rate (TNR): Number of signals correctly predicted as negative over total false items

$$Specificity = TNR = \frac{TN}{TN + FP}. \tag{4.4}$$

- FPR: Number of signals wrongly predicted as positive over total false items

$$FPR = 1 - Specificity = \frac{FP}{FP + TN}. \tag{4.5}$$

- FNR: Number of signals wrongly predicted as negative over total true items

$$FNR = 1 - Sensitivity = \frac{FN}{FN + TP}. \tag{4.6}$$

- Precision: Number of signals correctly predicted as positive over total positive predicted items

$$P = \frac{TP}{TP + FP}. \tag{4.7}$$

4.3.2 Hard Classifiers: Balanced Accuracy and F1-score

Balanced accuracy is a combination of sensitivity and specificity. Based on the definition of sensitivity and specificity in Equation 4.3 and Equation 4.4, balanced

accuracy is a holistic measure considering all entries in the confusion matrix, and it is calculated by

$$Balanced\ Accuracy = \frac{Sensitivity + Specificity}{2}. \tag{4.8}$$

F1-score is the harmonic mean of precision and recall, and it is defined by

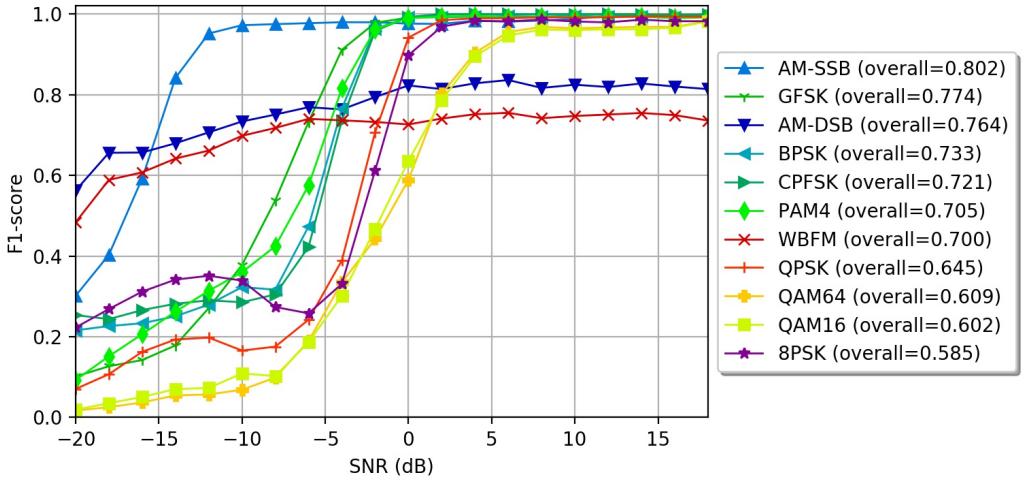
$$F1 = 2 \cdot \frac{R \cdot P}{R + P}. \tag{4.9}$$

However, from Equation 4.3 and Equation 4.7, F1-score should only be used when TN does not play a role, as TNs are not taken into account in F1-score.

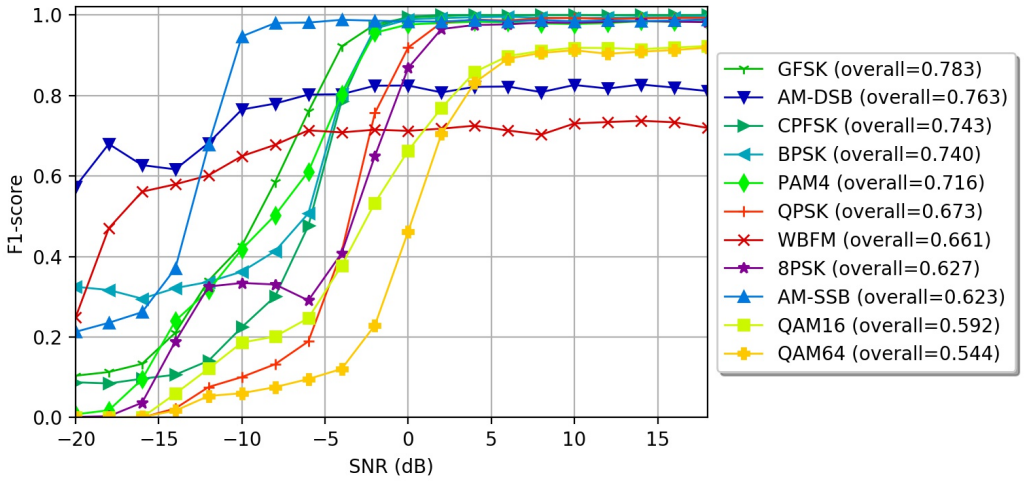
We depict the per-class F1-scores for AP-Attention and IQ-CLDNN in Figure 4.5. The number in parentheses represents the F1-score for a certain modulation class average over all possible SNR values. We will discuss the reason why balanced accuracy is unreliable later.

Figure 4.5a indicates that at high SNR, AP-Attention has excellent performance on most modulations except for AM-DSB and WBFM. The F1-score performance of AM-SSB is far superior to other modulations with the highest overall F1-score of 0.802 and the fastest convergence speed. AM-DSB and WBFM can be easily classified at low SNR, but they have little improvement when SNR is increased. QAM16 and QAM64 have similar performance of per-class F1-scores.

Figure 4.5b shows the per-class F1-scores for IQ-CLDNN. Similarly, AM-SSB is still the most easily recognized modulation. Different from AP-Attention, QAM16 and QAM64 both have a slight performance degradation for around 8% and a larger performance gap between them in high SNR values. In the low SNR regime, all modulation schemes have lower F1-scores in IQ-CLDNN than in AP-Attention. This indicates that AP-Attention can extract robust and distinct features at lower SNR.



(a) F1-scores for AP-Attention.



(b) F1-scores for IQ-CLDNN.

Figure 4.5: Per-class F1-scores for AP-Attention and IQ-CLDNN

4.3.3 Micro and Macro Averaging

To obtain one metric that quantifies the overall performance of the AMC classifier, we combine the aforementioned per-class performance measures using micro or macro averaging algorithms. Since the AMC problem belongs to the category of multi-class classification problems, each AMC model has a larger TN than the binary classification problems[1]. Since the specificity becomes inflated, the balanced accuracy does not provide a good performance measurement for the multi-class classification problem. Therefore, the F1-score is used for the micro and macro averaging algorithms.

Micro-averaging algorithm considers each modulation type as a binary classification problem and assigns equal weight to the individual decision. The micro-averaging precision and recall are defined as

$$P_{micro} = \frac{\sum_{m=1}^{M} TP_m}{\sum_{m=1}^{M} (TP_m + FP_m)} \tag{4.10}$$

and

$$R_{micro} = TPR_{micro} = \frac{\sum_{m=1}^{M} TP_m}{\sum_{m=1}^{M} (TP_m + FN_m)} \tag{4.11}$$

where m is the index of modulation scheme, and M represents the number of modulation schemes.

The micro-averaging F1-scores are calculated by

$$F1_{micro} = 2 \cdot \frac{R_{micro} \cdot P_{micro}}{R_{micro} + P_{micro}}. \tag{4.12}$$

Since the micro-averaging is unreliable for imbalanced class distribution, the macro-averaging algorithm is introduced. Macro-averaging algorithm is suitable for imbalanced datasets. Macro-averaging algorithm assigns equal weights to each modulation schemes and averages over M possible modulation schemes. The

[1]The ith type of modulation is TP, and the other types of modulations are TNs when $\forall j \neq i$.

Table 4.1: Micro and Macro-averaging F1-scores for AMC models

AMC Model	$F1_{\boldsymbol{micro}}$	$F1_{\boldsymbol{macro}}$
AP-Attention	0.6974	0.6946
IQ-CLDNN	0.6800	0.6785
Img-MobileNetV2	0.5951	0.5844
Features-CNN	0.5488	0.5721
FFT-CNN	0.5602	0.5457

macro-averaging precision and recall are defined as

$$P_{macro} = \frac{1}{M}\sum_{m=1}^{M}\frac{TP_m}{TP_m + FP_m} = \frac{\sum_{m=1}^{M} P_m}{M} \tag{4.13}$$

and

$$R_{macro} = TPR_{macro} = \frac{1}{M}\sum_{m=1}^{M}\frac{TP_m}{TP_m + FN_m} = \frac{\sum_{m=1}^{M} R_m}{M}. \tag{4.14}$$

Similarly, the macro-averaging $F1_{macro}$ is defined by

$$F1_{macro} = 2 \cdot \frac{R_{macro} \cdot P_{macro}}{R_{macro} + P_{macro}}. \tag{4.15}$$

Table 4.1 provides the $F1_{micro}$ and $F1_{macro}$ computed by the same test set. The AP-Attention obtains the largest $F1_{micro}$ and $F1_{macro}$. Note that, except for Features-CNN, other AMC models have a larger $F1_{micro}$ than $F1_{macro}$. This indicates that Features-CNN is more stable when the modulation scheme distribution is imbalanced, as $F1_{macro}$ is sensitive to the predictive performance for individual classes.

4.3.4 Soft Classifiers: Receiver Operating Characteristics (ROC) and Precision-Recall Curve (PRC)

The AMC models can be regarded as soft classifiers that produce predictions with a decision cutoff applied on scores for each modulation scheme. ROC curve and PRC are plotted to visualize the classification performance for soft classifiers.

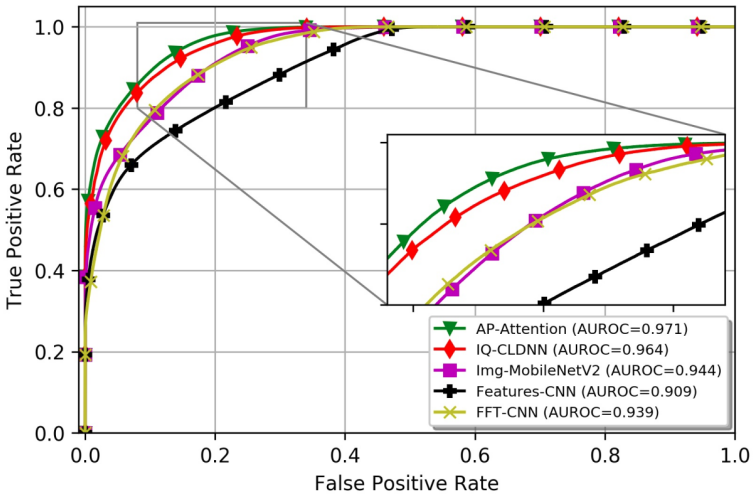


Figure 4.6: Micro-averaging ROC Curves for AMC models

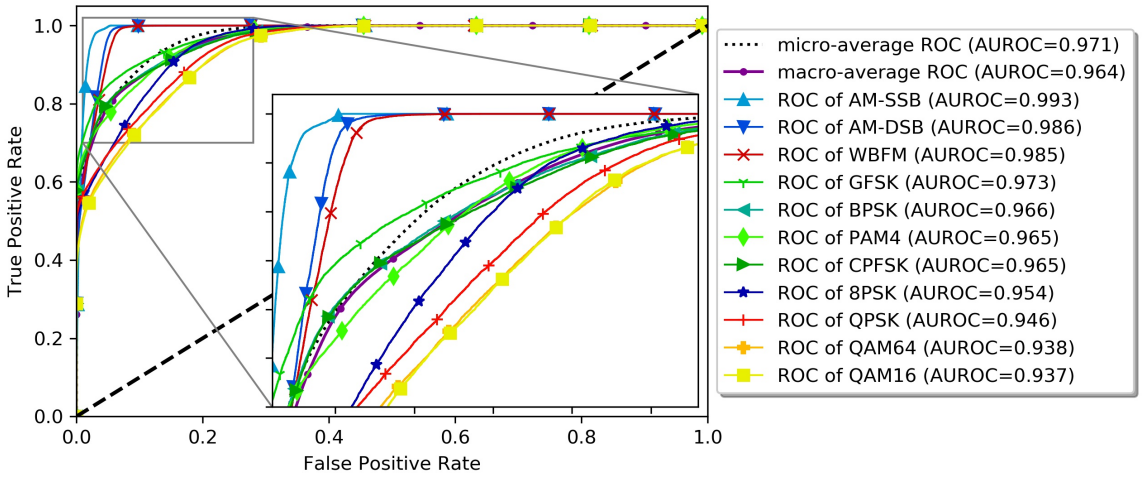


Figure 4.7: Per-class ROC Curves for AP-Attention

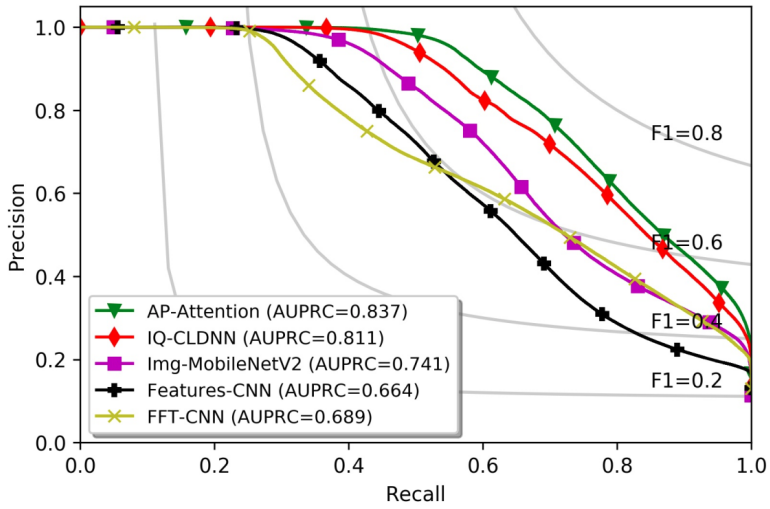


Figure 4.8: Micro-averaging PRCs for AMC models

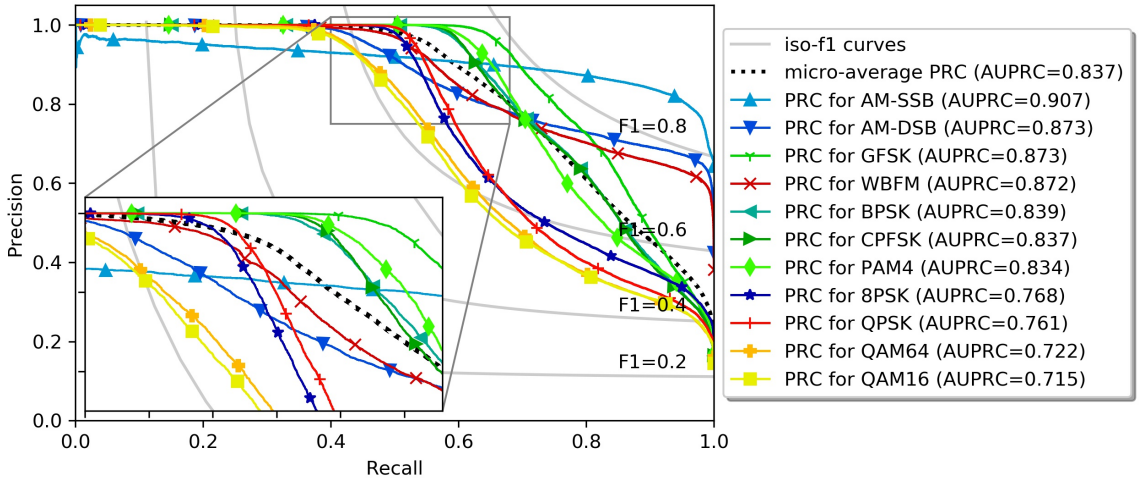


Figure 4.9: Per-class PRC for AP-Attention

ROC curve is a probability curve with FPR as the x-axis and TPR as the y-axis. Each point of the ROC curve represents a TPR/FPR pair at different decision thresholds. The point in the upper left corner $(0,1)$ of the ROC space yields the best prediction result, representing no FNs and FPs.

The area under ROC (AUROC) measures the model distinguishing ability between classes by computing the 2D area under ROC from point $(0,0)$ to $(1,1)$.

In probabilistic interpretation, AUROC is a decision-threshold-invariant measure, while F1-score is a threshold-sensitive measure based on hard (0 or 1) outputs. In our experiment, AUROC is computed by the trapezoidal rule [38] in Python.

As a useful measure for imbalanced datasets, PRC shows the trade-off between precision and recall for different decision thresholds. A larger area under PRC (AUPRC) represents high scores for both precision and recall. However, similar to F1-score, PRC does not consider TNs.

Compared with the macro-averaging ROC curves, micro-averaging ROC curves for AMC models are closer to the ideal point $(0,1)$ and has a slightly larger AUROC for all AMC models. This indicates that AMC models have better overall performance (micro-averaging) than class-specific performance (macro-averaging). Compared with the macro-averaging ROC, micro-averaging ROC has a similar growing tendency and comparable AUROC value. Thus, we only present the micro-averaging ROC curves for AMC models in Figure 4.6, unless specified explicitly. Per-class ROC curves of AP-Attention are also plotted in Figure 4.7.

From Figure 4.6, almost all algorithms converge to $TPR = 1.0$ at the same value of $FPR \approx 0.35$, except for Features-CNN. For per-class ROC curves of AP-Attention in Figure 4.7, AM-SSB, AM-DSB, and WBFM have the lowest FPR when reaching $TPR = 1$. In other words, classifying AM-SSB, AM-DSB, and WBFM is easier than other modulations. QAM16 and QAM64 have the nearly coincident ROC curves and the smallest AUROCs because of their similar features under low SNR values.

Similarly, micro-averaging and per-class PRCs are illustrated in Figure 4.8 and Figure 4.9. Iso-F1 curve consisting of points with the same F1-score is also included in these two figures to show how close the PRCs are to different F1 scores. Good PRC are close to point $(1,1)$ and good AUROC is close to 1.

Overall, micro-averaging AUPRCs are lower than micro-averaging AUROCs. From Figure 4.8, AP-Attention and IQ-CLDNN still show an AUPRC advantage of up to 0.16 than other AMC models. AP-Attention and IQ-CLDNN reach a recall of roughly 46% without any FP predictions. Features-CNN and FFT-CNN

start to predict FP at a low recall rate of 29%.

For the per-class PRCs for AP-Attention model in Figure 4.9, AM-SSB has the highest AUPRC. It is also noticed that at first plotted PRC points, only the precision of AM-SSB is slightly lower than 100%. This indicates that at the first threshold, the AP-Attention model already makes FPs for AM-SSB. However, to reach a sensitivity of 100%, the precision of AM-SSB only reduces to around 70%. Except for QAM16 and QAM64, the modulation classes have an AUPRC above 75% in AP-Attention. From the micro-averaging PRC, AP-Attention reaches a recall rate of roughly 48% without any FPs.

4.4 Computational Complexity

Computational complexity includes memory consumption and detection efficiency. The memory consumption can be measured by the number of trainable parameters (and corresponding required storage size) and the peak memory usage. The detection efficiency is evaluated by the average inference time per input signal. Since the model parameters of AMC models can be trained and stored in memory, the computational complexity of AMC models is mainly determined by the model structure (neural-network architecture and input size) the feature extraction stage (signal representation transformation and data pre-processing). The modulation schemes and channel states have negligible influence on the computational complexity. To make a fair comparison, we analyze the complexity metrics (the number of trainable parameters, storage size of parameters, peak memory usage and average inference time) under the same hardware and software implementations.

As shown in Table 4.2, AP-Attention has the smallest number of trainable parameters and the smallest required memory size. Therefore, AP-Attention presents an attractive choice when the computational complexity and classification performance are preferred. We also observe that the proposed IQ-CLDNN model has the second-highest efficiency in inference time. The Img-MobileNetV2 model requires more memory and inference time than other models because of the larger

Table 4.2: Computational Complexity of AMC Models

AMC Model	Number of Trainable Parameters	Storage Size (MB)	Peak Memory Usage (MiB)	Average Inference Time per Example (ms)
AP-Attention	249,227	3.0	4,243.492	0.329
IQ-CLDNN	794,763	9.6	6,317.871	0.303
Img-MobileNetV2	2,237,963	16.1	7155.203	17.045
Features-CNN	821,259	9.9	3530.207	0.655
FFT-CNN	1,175,883	14.2	6518.602	0.291

input size of $(224, 224, 2)$ and complex model structure. The average inference time for Features-CNN is almost twice than that of AP-Attention since the computation of statistical features consumes more time. Using the pooling layer and a smaller number of CNN filters, FFT-CNN has the lowest inference time among AMC models.

Attributing to the hardware support of parallel processing and the software optimization of data flow, it is envisioned that the inference time of DL-based models can be sharply reduced in implementation. Therefore, the graphics processing unit (GPU) assisted AMC models are envisioned to have lower computation complexities by processing different received sequences simultaneously.

4.5 Summary

This chapter analyzes the results of previous introduced five AMC models. Different performance metrics are introduced to evaluate the AMC models. Experiment results indicate that AP-Attention and IQ-CLDNN have the superior classification performance over other AMC models. Computational complexity including memory consumption and detection efficiency is discussed in Section 4.4. Complexity results show that AP-Attention has less trainable parameters and short inference

time, while Img-MobileNetV2 suffers from high complexity.

Chapter 5

Weight Pruning and Stacking for a Better End-to-end AMC Model

5.1 Weight Pruning

Real-time AMC for mobile devices with constrained computational resources requires more memory-efficient and power-efficient AMC models. Therefore, pruned NNs are introduced at the expense of negligible loss in accuracy. Back to the 1990s, pruned NNs have been proposed based on the fact that many NN parameters are redundant and have less contributions to the final output [39].

We eliminate the unnecessary model parameters to improve the memory efficiency and power efficiency of the proposed AMC models. More specifically, the weight pruning operations set the low-magnitude model parameters to zero such that the required memory resource is reduced. After weight pruning, the models become sparse. Therefore, performing the compression operation[1] can reduce the latency. The detailed procedures of our proposed weight pruning operation are as follows. Based on the obtained models in Chapter 4, the training is performed for 150 epochs among which the pruning operation is conducted in the first 60

[1]The compression operation is defined as recording the non-zero elements and skipping the computations related to those zeros.

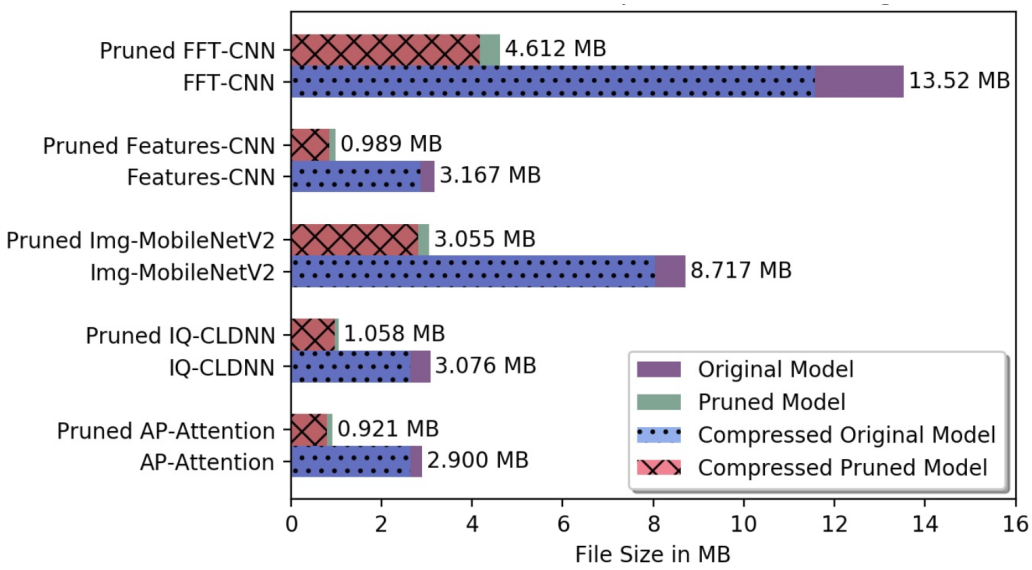


Figure 5.1: Model sizes comparison after weight pruning

epochs. Since pruning too many model parameters in each iteration can severely degrade the performance, the pruning operation is performed iteratively. For example, we prune every 500 steps to give the AMC model more recovery time. We gradually train the model until the sparsity target of 70% is reached. When the validation accuracy does not improve in consecutive 30 epochs, the pruned model is obtained.

We compress the original models and the pruned models by a generic file compression algorithm, namely zip compression. As illustrated in Figure 5.1, bars with texture represent the model after zip compression. The value labeled on each bar is the model size before zip compression. Figure 5.1 shows that pruned models occupy roughly 30% memory of the original models. After zip compression, we can save roughly 10% more storage space. From these results, the sizes of pruned models do not exceed 5 MB. Moreover, pruned AP-Attention, pruned IQ-CLDNN, and pruned Features-CNN only occupy around 1 MB. Therefore, weight pruning greatly lightens the storage burden for end-to-end mobile devices.

We also perform the comparison of F1-scores based on the original models in

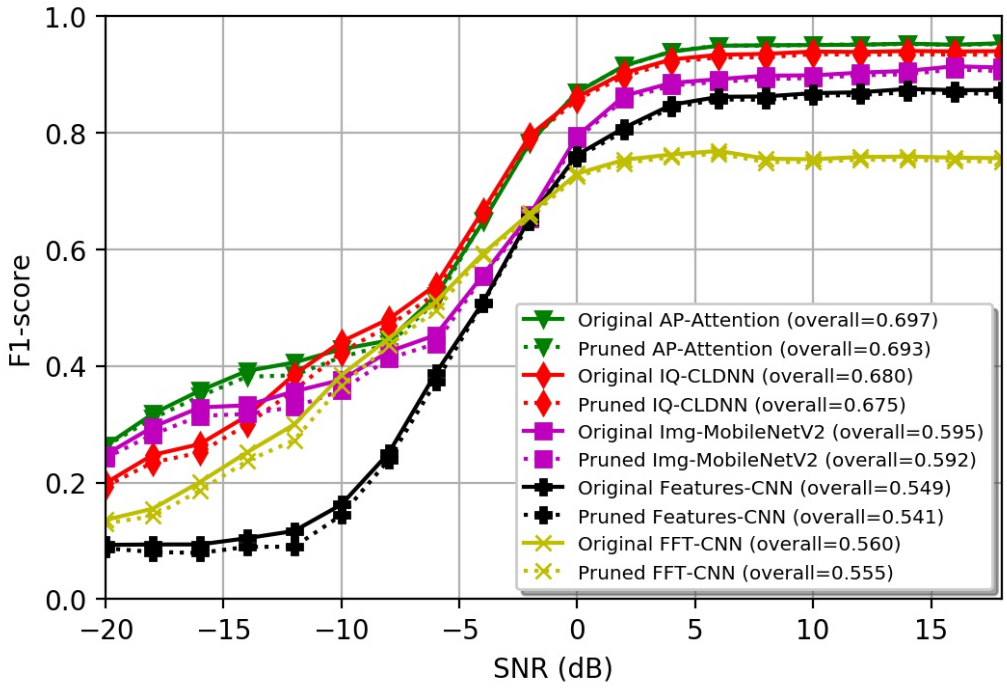


Figure 5.2: Micro-averaged F1-scores comparison after weight pruning

Chapter 4 and the pruned models. The micro-averaged pruned F1-scores results can be found in Figure 5.2. The values in the brackets are the overall averaged F1-scores. We observe that the pruning operation results in minor performance degradation. Although the degradation in F1-scores is more obvious in low SNR regime, the overall decrease does not exceed 0.08. Therefore, the pruning operation can trade the F1-scores for the memory reduction.

5.2 Ensemble Learning

Chapter 4 illustrates the classification accuracy of different models in the descending order as AP-Attention, IQ-CLDNN, Img-MobileNetV2, Features-CNN, and FFT-CNN. Using the parallel processing, we can process multiple signal sequences simultaneously. Therefore, the classification accuracy is more important

than the computational complexity in practice. To improve the classification accuracy, ensemble learning is introduced to integrate the different AMC models in Chapter 4.

5.2.1 Definition

In ensemble learning, multiple first-level models (a.k.a. weak learners) are combined and trained to solve the same problem. Two categories of combination methods can be used for the ensemble learning, namely homogeneous and heterogeneous methods. The homogeneous methods combine the same models trained in different ways, and the heterogeneous methods combine the different models. Different models are based on different learning algorithms.

5.2.2 Stacking

Since the proposed AMC models are all heterogeneous models based on different learning algorithms, our ensemble target is to obtain a more accurate modulation classification. Therefore, a heterogeneous method (i.e., stacking method) is used. Rather than choosing a single model, stacking method integrates the different first-level models into a second-level model (a.k.a meta model) and trains the second-level model based on outputs of first-level models [40].

We denote the original training dataset by D with N individual signal samples $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$ and N_1 first-level models. At the beginning, the general procedure of stacking learns the first-level model $f^1 = \{f_1^1, f_2^1, \ldots, f_{N_1}^1\}$ based on D. Then, stacking trains a second-level model f^2 based on the predictions of first-level models f^1. For sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, the corresponding item in the new dataset is $(\{f_1^1(\boldsymbol{x}_i), f_2^1(\boldsymbol{x}_i), \ldots, f_{N_1}^1(\boldsymbol{x}_i)\}, \boldsymbol{y}_i)$. After training the second-level model f^2, the prediction of unseen sequence $\boldsymbol{x}$ is computed by $f^2(f_1^1(\boldsymbol{x}), f_2^1(\boldsymbol{x}), \ldots, f_{N_1}^1(\boldsymbol{x}))$.

We use the combinations of introduced AMC models in Chapter 4 to generate the first-level models f^1. To maintain low complexity, a three-layer CNN is used as the second-level model f^2. Using the proposed five AMC models, we can obtain $C_5^2 + C_5^3 + C_5^4 + C_5^5 = 26$ first-level models. Based on the performance of five

introduced AMC models discussed in Chapter 4, we consider the five reasonable combinations to include more features and less computation complexity:

- AP-Attention, IQ-CLDNN (AP-IQ)
- AP-Attention, IQ-CLDNN, Img-MobileNetV2 (AP-IQ-Img)
- AP-Attention, IQ-CLDNN, Features-CNN (AP-IQ-Features)
- AP-Attention, IQ-CLDNN, Img-MobileNetV2, Features-CNN (AP-IQ-Img-Features)
- AP-Attention, IQ-CLDNN, Img-MobileNetV2, Features-CNN, FFT-CNN (AP-IQ-Img-Features-FFT).

When the first-level models use the same dataset as the second-level model, the overfitting to the dataset occurs. A heuristic method is to split the dataset into two subsets for the first-level models and the second-level model. However, the data-splitting method has an obvious drawback, i.e., only a fraction of the data samples are used for training the model in each level. Therefore, the K-fold cross-validation is used.

Using the K-fold cross-validation, we can partition D into K disjoint subsets such that the second-level model f^2 can be trained using all samples in dataset D. The procedure of stacking with K-fold cross-validation is illustrated in Figure 5.3. We train f^1 on $K-1$ folds and make predictions on the remaining fold to avoid overfitting. Repeat K times, all predictions from f^1 make up the training dataset for f^2. After training f^2, we re-train f^1 on the whole dataset D. Therefore, the final stacking model F is obtained by applying the second-level model f^2 on re-trained first-level models f^1, which is defined by

$$F(\cdot) = f^2(f_1^1(\cdot), f_2^1(\cdot), \ldots, f_{N_1}^1(\cdot)). \tag{5.1}$$

In our experiments, we choose three-fold cross-validation to train the stacking model. The second-level model f^2 consists of three layers of CNN with 128,

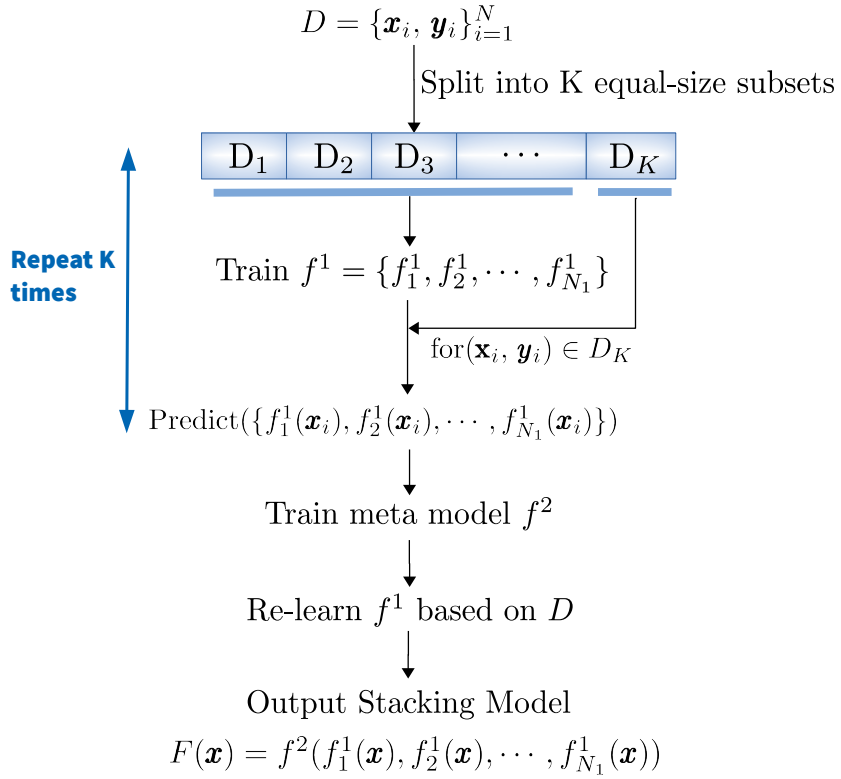


Figure 5.3: Stacking with K-fold Cross-Validation

128, and 64 hidden neurons. The padding and dropout with a rate of 0.4 are utilized. We depict the F1-scores performance of five stacking learned models in Figure 5.4. The values in the brackets are the overall micro-averaged F1-scores.

Compared with the other stacking models and the AMC models in Figure 5.2, we found that the AP-IQ model has the highest overall micro-averaged F1-scores. This observation is reasonable due to the facts that the AP-Attention model and IQ-CLDNN models show the superior performance over the other models. When the highly accurate features are combined, the second-level model f^2 extract more discriminative representations. Besides, AP-IQ model outperforms the single AP-Attention model by roughly 2% in F1-score, which validates the effectiveness of stacking.

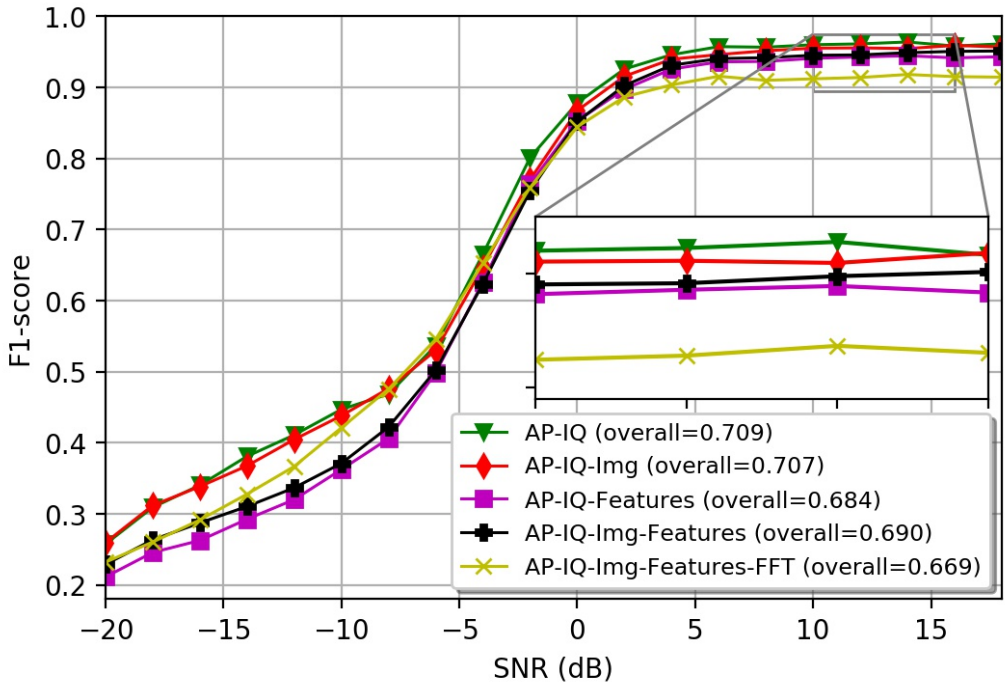


Figure 5.4: Micro-averaged F1-scores for stacking learned models

Predictions of Features-CNN and FFT-CNN limit the representation learning capability of the second-level model f^2. Therefore, the obtained F1-scores is lower than the other stacking models. This observation indicates the possibility of a weighted dataset to give more weights to the first-level models f^1 with better performance and fewer weights to other models.

Following Section 4.4 and Section 5.1, extensive pruning experiments with a target of 70% sparsity are also conducted to control the computational complexity of stacking AMC models. Similar to single AMC models, pruning operation leads to a slight degradation in F1-scores. Although the deterioration is more severe under low SNR regime, overall the effects can be neglected.

Table 5.1 shows the comparison of complexity with the same pruning operations in Section 5.1. After the pruning operations, the stacking classifiers with Img-MobileNetV2 ensembled have remarkably increased inference time. The AP-

Table 5.1: Computational Complexity of Pruned Stacking AMC Models

AMC Model	Number of Trainable Parameters	Storage Size (MB)	Peak Memory Usage (MiB)	Average Inference Time per Example (ms)
AP-IQ	331,656	4.15	6,806.595	0.723
AP-IQ-Img	977,142	7.91	7,382.542	17.325
AP-IQ-Features	633,930	7.52	6,917.368	1.378
AP-IQ-Img-Features	1,177,869	12.19	7,520.321	18.308
AP-IQ-Img-Features-FFT	1,551,467	15.88	7,614.509	18.711

IQ model has a storage size of less than 5 MB and an averaged inference time of less than 1 ms. Therefore, the pruned AP-IQ is the most suitable AMC classifier for end-to-end devices due to its superior F1-scores performance and lower computational complexity.

5.3 Summary

In this chapter, we introduce weight pruning and stacking learning to further improve the classification performance of AMC models. Experiments show that weight pruning can reduce the AMC model complexity with a negligible performance degradation. Stacking learned AMC models have superior performance than single AMC models. Overall, the pruned stacking learned AP-IQ model has the best classification performance and acceptable low computational complexity.

Chapter 6

Conclusions and Future Work

In this chapter, we conclude and highlight the contributions of this thesis in Section 6.1. Then, we also illustrate some ideas for future work in Section 6.2.

6.1 Conclusions

This thesis aims at providing a memory-efficient and high-performance AMC model for resource-constrained devices. Our contributions can be summarized as follows:

- We introduced noise reduction, signal normalization, and label smoothing before training AMC models. To conduct a thorough comparison, we investigated all possible signal representations and designed a DL-based model for each signal representation. For example, we used the KDE algorithm to convert the well-known CD into 3D-CI. We proposed the attention module in AP-Attention and connected CNN, LSTM, and DNN with multi-scale connection IQ-CLDNN.

- We conducted extensive simulations to evaluate the model performance based on the comprehensive metrics, namely F1-score, ROC, and PRC. Numerical results illustrate that the overall performance of AMC models in descending order was: AP-Attention, IQ-CLDNN, Img-MobileNetV2,

Features-CNN, and FFT-CNN. Moreover, the complexity experiments validated the computation-efficiency of AP-Attention and IQ-CLDNN.

Therefore, we conclude that the used data pre-processing methods and proposed AMC models (AP-Attention model and IQ-CLDNN model, Img-MobileNetV2 model, Features-CNN and FFT-CNN) can improve the AMC performance over the benchmark models (AP-LSTM and IQ-CNN). Besides, our work also compared the impacts of the pruning operation and stacking operation on the AMC models. The investigation of pruned models showed that weight pruning greatly reduced the model storage size while only brought negligible performance degradation. Furthermore, the stacking experiments confirmed that pruned AP-IQ achieved the highest F1-scores and kept low computational complexity at the same time.

6.2 Future Work

Several research directions can be considered in the future

- Recently, O'Shea et al. [7] proposed the second version of the RadioML dataset that includes both synthetic simulated channel effects and over-the-air recordings of 24 modulation schemes. We will train and test our models on this dataset in the future to check the performance of our proposed models on this complicated dataset.

- In this thesis, we chose weight pruning to remove unnecessary NN connections. Further efforts to reduce the model size might explore the quantization method. Quantization converts the model weights to eight-bit precision. We should also convert the final pruned model into a suitable format to run on specific back-ends. For example, TensorFlow lite[1] is a format for deploying DL models on mobile devices.

- In Section 5.2, we assign equal weights to the predictions of first-level base models when building the training dataset for the meta-model. However, the

[1] https://www.tensorflow.org/lite

results suggest the possibility for further research about a weighted dataset with more weights to good predictions and fewer weights for other predictions.

- Recent research has shown that DNNs are highly vulnerable to adversarial attacks. Sadeghi and Larsson [41] validated this phenomenon in radio modulation classification tasks. Further studies need to be carried out to determine whether adversarial attacks will affect our proposed AMC models.

- All our proposed models belong to supervised learning. The model performance relies heavily on the quality of training dataset. More research should be undertaken to explore how semi-supervised learning performs in AMC problem.

Bibliography

[1] M. Kulin, T. Kazaz, I. Moerman, and E. De Poorter, "End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications," *IEEE Access*, vol. 6, pp. 18 484–18 501, 2018. → pages 1, 2, 5, 10, 24, 38

[2] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 201–220, Feb 2005. → page 1

[3] T. J. O'Shea and J. Corgan, "Convolutional radio modulation recognition networks," *CoRR*, vol. abs/1602.04105, 2016. [Online]. Available: http://arxiv.org/abs/1602.04105 → pages 2, 5, 6, 38, 72

[4] I. Abou-Faycal, M. Medard, and U. Madhow, "Binary adaptive coded pilot symbol assisted modulation over rayleigh fading channels without feedback," *IEEE Transactions on Communications*, vol. 53, no. 6, pp. 1036–1046, June 2005. → page 2

[5] J. Sun, G. Wang, Z. Lin, S. G. Razul, and X. Lai, "Automatic modulation classification of cochannel signals using deep learning," *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pp. 1–5, 2018. → page 2

[6] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "Survey of automatic modulation classification techniques: classical approaches and new trends," *IET Communications*, vol. 1, no. 2, pp. 137–156, April 2007. → pages 2, 3, 12

[7] 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, Feb 2018. → pages 2, 14, 63

[8] C.-Y. Huan and A. Polydoros, "Likelihood methods for mpsk modulation classification," *IEEE Transactions on Communications*, vol. 43, no. 2/3/4, pp. 1493–1504, Feb 1995. → page 3

[9] W. Wei and J. M. Mendel, "Maximum-likelihood classification for digital amplitude-phase modulations," *IEEE Transactions on Communications*, vol. 48, no. 2, pp. 189–193, Feb 2000. → pages 2, 3

[10] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, Sep. 2018. → pages 2, 5, 6, 38

[11] A. E. El-Mahdy and N. M. Namazi, "Classification of multiple m-ary frequency-shift keying signals over a rayleigh fading channel," *IEEE Transactions on Communications*, vol. 50, no. 6, pp. 967–974, June 2002. → page 3

[12] P. Panagiotou, A. Anastasopoulos, and A. Polydoros, "Likelihood ratio tests for modulation classification," in *MILCOM 2000 Proceedings. 21st Century Military Communications. Architectures and Technologies for Information Superiority (Cat. No.00CH37155)*, vol. 2, Oct 2000, pp. 670–674 vol.2. → page 3

[13] Y. Zeng, M. Zhang, F. Han, Y. Gong, and J. Zhang, "Spectrum analysis and convolutional neural network for automatic modulation recognition," *IEEE Wireless Communications Letters*, vol. 8, no. 3, pp. 929–932, June 2019. → page 4

[14] M. L. D. Wong and A. K. Nandi, "Automatic digital modulation recognition using artificial neural network and genetic algorithm," *Signal Process.*, vol. 84, no. 2, pp. 351–365, Feb. 2004. [Online]. Available: http://dx.doi.org/10.1016/j.sigpro.2003.10.019 → page 4

[15] M. W. Aslam, Z. Zhu, and A. K. Nandi, "Automatic modulation classification using combination of genetic programming and knn," *IEEE Transactions on Wireless Communications*, vol. 11, no. 8, pp. 2742–2750, August 2012. → page 4

[16] Huang Fu-qing, Huang Fu-qing, Zhong Zhi-ming, Xu Yi-tao, and Ren Guo-chun, "Modulation recognition of symbol shaped digital signals," in

2008 International Conference on Communications, Circuits and Systems, May 2008, pp. 328–332. → page 4

[17] J. H. Lee, J. Kim, B. Kim, D. Yoon, and J. W. Choi, "Robust automatic modulation classification technique for fading channels via deep neural network," *Entropy*, vol. 19, no. 9, 2017. [Online]. Available: https://www.mdpi.com/1099-4300/19/9/454 → pages 4, 24, 33, 38, 75

[18] S. Peng, H. Jiang, H. Wang, H. Alwageed, Y. Zhou, M. M. Sebdani, and Y. Yao, "Modulation classification based on signal constellation diagrams and deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 3, pp. 718–727, March 2019. → pages 4, 5, 6

[19] A. Swami and B. M. Sadler, "Hierarchical digital modulation classification using cumulants," *IEEE Transactions on Communications*, vol. 48, no. 3, pp. 416–429, March 2000. → page 4

[20] J. J. Popoola and R. van Olst, "Effect of training algorithms on performance of a developed automatic modulation classification using artificial neural network," in *2013 Africon*, Sep. 2013, pp. 1–6. → page 4

[21] Han Gang, Li Jiandong, and Lu Donghua, "Study of modulation recognition based on hocs and svm," in *2004 IEEE 59th Vehicular Technology Conference. VTC 2004-Spring (IEEE Cat. No.04CH37514)*, vol. 2, May 2004, pp. 898–902 Vol.2. → page 4

[22] 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, Dec 2017. → page 5

[23] 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. → page 5

[24] 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. → page 5

[25] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381 → pages 7, 33

[26] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec 2014. → page 15

[27] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. → page 17

[28] E. Azzouz and A. Nandi, "Automatic modulation recognition," *Journal of the Franklin Institute*, vol. 334, no. 2, pp. 241–273, 1997. → pages 21, 76

[29] A. Swami and B. M. Sadler, "Hierarchical digital modulation classification using cumulants," *IEEE Transactions on Communications*, vol. 48, no. 3, pp. 416–429, March 2000. → page 22

[30] A. K. Nandi and E. E. Azzouz, "Algorithms for automatic modulation recognition of communication signals," *IEEE Transactions on Communications*, vol. 46, no. 4, pp. 431–436, April 1998. → pages 22, 76

[31] E. Azzouz and A. Nandi, "Automatic identification of digital modulation types," *Signal Processing*, vol. 47, no. 1, pp. 55 – 69, 1995. → page 76

[32] M. W. Aslam, Z. Zhu, and A. K. Nandi, "Automatic modulation classification using combination of genetic programming and knn," *IEEE Transactions on Wireless Communications*, vol. 11, no. 8, pp. 2742–2750, August 2012. → page 22

[33] R. Müller, S. Kornblith, and G. Hinton, "When does label smoothing help?" 2019. → page 23

[34] M. Zhang, Y. Zeng, Z. Han, and Y. Gong, "Automatic modulation recognition using deep learning architectures," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, June 2018, pp. 1–5. → page 26

[35] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. Xu, "Attention-based bidirectional long short-term memory networks for relation classification," in *ACL*, 2016. → pages 27, 28

[36] T. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," 04 2015, pp. 4580–4584. → page 31

[37] 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. → page 33

[38] T. Fawcett, "Introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, pp. 861–874, 06 2006. → page 50

[39] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed. Morgan-Kaufmann, 1990, pp. 598–605. [Online]. Available: http://papers.nips.cc/paper/250-optimal-brain-damage.pdf → page 54

[40] C. C. Aggarwal, *Data Classification: Algorithms and Applications*, 1st ed. Chapman & Hall/CRC, 2014. → page 57

[41] M. Sadeghi and E. G. Larsson, "Adversarial attacks on deep-learning based radio signal classification," *CoRR*, vol. abs/1808.07713, 2018. [Online]. Available: http://arxiv.org/abs/1808.07713 → page 64

[42] T. O'Shea and N. West, "Radio machine learning dataset generation with gnu radio," *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, 2016. → page 73

Appendix A

Typical Layers in CNN Architecture

A.1 Convolutional Layer

Each convolutional layer convolves input feature map with fixed length filters and then cascades these new feature maps together to form the output layer. We begin with the standard One-dimensional (1D) convolutional layer, which convolves in a single direction for vectors.

For example, if the input feature map of the first 1D convolutional layer is $x^{l-1} \in \mathbb{R}^{L \times N^{l-1}}$, where L is the sample length of 128 points and $N^{l-1} = 2$ is the number of input feature maps, or dimensions, then the 1D convolutional layer l with N^l output feature maps would have $N^{l-1} \times N^l$ kernels with size k_l and N^l biases.

For a certain kernel $i \in \{1,\ldots,N^l\}$ in next layer l, the input vector of 1D convolution layer is the output from previous layer denoted as x_j^{l-1} with $j \in \{1,\ldots,N^{l-1}\}$, then the ith mathematical output vector is represented as

$$x_i^l = f_{act}\left(\sum_{j=1}^{N_{l-1}} x_j^{l-1} * k_{i,j}^l + b_i^l\right) \tag{A.1}$$

where $k_{i,j}^l$ is the kernel for ith output vector and jth input vector with size k^l, b_i^l

represents the bias term, and $*$ is the convolution operation. For kernel with size k_l, the total number of trainable parameters in convolution layer l is $N_l \times N_{l-1} \times k_l + N_l$.

A.2 Activation Layer

After classic NN or convolutional layer, a non-linear activation function, like RELU, is applied to mitigate the effects of gradient vanishing problem. RELU can be described as

$$f_{ReLU}(x_i) = max(0, x_i). \tag{A.2}$$

Activation functions are typically in sigmoidal shape, resulting in a nonlinear system to extract more complex features than a high-order linear system.

Another common activation function, softmax is often used to produce the probability score associated with the ith class. In AMC, the final modulation type is decided by the modulation class with the highest probability score. The ith element in output of softmax is

$$f_{softmax}(x_i) = \frac{e^{x_i}}{\sum_{i=1}^{M} e^{x_i}} \tag{A.3}$$

where x_i is ith pre-activation output, and M is the number of modulation classes.

A.3 Fully-connected or Dense Layer

After stacked convolutional layers and a flatten layer to transfer a 2D vector to a 1D vector, a fully-connected layer is used to extract higher-level information from the previous flatten layer.

Fully-connected layer or dense layer has the same architecture as classic neural networks in Equation 2.12 where neurons have full connections to all activations from the previous layer. The dense layer is usually used as the last layer in the classification problem to output the normalized likelihood vector with activation function being softmax.

Appendix B

RadioML Dataset Generation Setup

B.1 Dataset Simulation Model

Modulated with real voice and text data, the samples in this dataset are generated with 11 different modulation schemes and 20 different SNR levels from −20 dB to +18 dB with a step of 2 dB.

For digital modulations, Binary Phase Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK), 8PSK, QAM16, QAM64, and Pulse Amplitude Modulation (PAM)4, the entire Gutenberg works of Shakespeare in ASCII is used, with whitening block randomizer to ensure equiprobable symbols and bits. Moreover, the gr-mapper OOT module and an interpolating finite impulse response root-raised cosine pulse shaping filter with an excess bandwidth of 0.35 is used to achieve the desired samples per symbol rate [3].

For analog modulations Wide-band Frequency Modulation (WBFM), Amplitude Modulation (AM)-Single Side-band Modulation (SSB), AM-Double Side-band (DSB), Gaussian Frequency-shift Keying (GFSK), and Continuous-phase Frequency-shift Keying (CPFSK), a continuous acoustic voice speech with some interludes and off times is implemented with GNU Radio hierarchical blocks.

The generated signals then pass through a number of realistic channel imperfections and intersymbol interference. Primary amplitude, phase, Doppler, and

delay impairments introduced in the wireless channel consist of [42]:

- Thermal noise: Due to the resistive components in the physical device such as the receiver antenna, this thermal noise maybe modelled as AWGN, $n \sim \mathcal{N}(0, \sigma^2)$, which forms a specific noise power level corresponding to the desired SNR.

- Frequency offset: The frequency offset is caused by the slightly different local oscillator signal frequencies at the transmitter f_c and receiver f_c', and the motion of emitters, reflectors, and/or receivers.

- Phase noise: Oscillator drift and unknown phase-delay of various propagation medium result in the angle of the signal to drift around its intended instantaneous phase $2\pi f_c t$.

- Sample rate offset: Different sample rates at the receiver and transmitter and time dilation are simulated by a fractional interpolator stepping along at a rate of $1 + \epsilon$ input samples per output sample, where ϵ is close to zero and follows a clipped random walk process.

- Multipath fading or frequency selective fading: Implemented by the sum of sinusoids with random phase offset for Rician and Rayleigh fading in GNU Radio, multipath fading usually occurs when signals reflect off any form of reflectors like buildings and vehicles.

- Delay spread: Non-impulsive delay spread is caused by the propagation of delayed multi-path reflection, diffraction, and diffusion.

B.2 Dataset Parameters

The total dataset is split by a short-time rectangular windowing process which is similar for speech recognition to slice continuous acoustic voice signals [42]. After segmentation, each signal example is normalized to average transmit power

Table B.1: RadioML Dataset Parameters

Parameters	Value
Samples per symbol	4
Sample length	128
Sampling frequency	200 kHz
Sampling rate offset standard deviation	0.01 Hz
excess bandwidth for root-raised cosine pulse shaping filter	0.35
Maximum sampling rate offset	50 Hz
Carrier frequency offset standard deviation	0.01 Hz
Maximum carrier frequency offset	500 Hz
Number of sinusoids	8
Maximum Doppler frequency	1
Fading model	Rician
Rician K-factor	4
Fractional sample delays for the power delay profile	$[0, 0.9, 1.7]$
Number of samples per modulation scheme at a specific SNR	1000
Magnitude corresponding to each delay time	$[1, 0.8, 0.3]$
Filter length to interpolate the power delay profile	8
Standard deviation of the AWGN process	$10^{-\frac{SNR}{10}}$
Number of training samples	82500
Number of validation samples	41250
Number of test samples	41250

of 0 dB in a 128×2 vector with IQ components. Each example is approximately 128 μ sec and contains between 8 and 16 symbols.

For dataset storage, numpy and cPickle Python packages are exploited to store it as a pickle file with complex 32-bit floating point samples. The detailed specifications and generation parameters are listed in Table B.1. As a modulation characteristic, the samples per symbol parameter used in the Table B.1 specify the number of samples representing each modulated symbol.

Appendix C

Statistical Features

C.1 HOC

The HOC C_{pq} is defined by

$$C_{pq} = cum(x,\ldots,x,x^*,\ldots,x^*) \tag{C.1}$$

where $cum(\cdot)$ denotes the cumulant function, x is repeated $p-q$ times and the conjugated version x^* is repeated q times. To remove the effect of the signal scale on cumulants, C_{pq} is typically powered by $\frac{2}{p}$.

By adopting Equation C.1, second-order cumulant C_{21} is given by $cum(x,x^*)$, fourth-order cumulant C_{42} is $cum(x,x,x^*,x^*)$, and the sixth-order cumulant C_{62} is computed by $cum(x,x,x,x,x^*,x^*)$.

The joint cumulant function is defined as [17]:

$$cum(x_1,\ldots,x_N) = \sum_A (|A|-1)!(-1)^{|A|-1}\Pi_{B\in A}E(\Pi_{i\in A}x_i) \tag{C.2}$$

where A is the whole partitions of set $[1,\ldots,N]$, and B runs through the list of all blocks of the partition A. A simple example is $cum(\alpha,\beta,\gamma,\nu) = E[\alpha\beta\gamma\nu] - E[\alpha\beta]E[\gamma\nu] - E[\alpha\gamma]E[\beta\nu] - E[\alpha\nu]E[\beta\gamma]$.

C.2 HOM

Given N samples $x(i)$, the function of HOC C_{pq} can be obtained from HOM M_{pq}, where the empirical estimated moment M_{pq} associated with the stationary process $x(i)$ is computed as

$$M_{pq} = E[x(i)^{p-q}(x(i)^*)^q], \quad for\ 0 \leq q \leq p \tag{C.3}$$

where p and q are integers, and the superscript $*$ represents the complex conjugate.

C.3 Other features

Another effective feature type extracted for AMC here is instantaneous features [28, 30, 31]. We introduce three instantaneous features: γ_{max}, kurtosis K and skewness S. γ_{max} is the maximum value of the power spectral density of the normalized signal samples.

K measures whether the PDF of $x(i)$ are heavy-tailed or light-tailed relative to a normal distribution. In other words, K checks the presence of outliers in the data distribution. High K indicates the presence of heavy tails or outliers in data, while low K is an indicator of light tails or lack of outliers.

S is a measure of lacking symmetry in the data distribution. The S of symmetrical distribution is zero. Negative S indicates $x(i)$ is skewed left, which means the left tail is longer than the right tail of the distribution. While positive S suggests the $x(i)$ is skewed right; the mean and median are less than the mode.

Besides, we also include four variances. σ^2_{aa} is the variance of the absolute value of normalized instantaneous amplitude $|x_{cn}(i)|$. σ^2_v represents the variance of the absolute value of normalized signal phase.

In the the variance of the direct instantaneous phase $\varphi_{NL}(i)$, σ^2_{dp}, and the variance of the non-linear component of $\varphi_{NL}(i)$, σ^2_{ap}, there is a threshold $x_t = 1$ below which the estimation of instantaneous phase $\varphi_{NL}(i)$ is sensitive to noise.

Table C.1: List of Features Used in Proposed AMC Method

Features	Definition
$f_1 : C_{20}$	M_{20}
$f_2 : C_{21}$	M_{21}
$f_3 : C_{40}^{1/2}$	$M_{40} - 3M_{20}^2$
$f_4 : C_{41}^{1/2}$	$M_{40} - 3M_{20}M_{21}$
$f_5 : C_{42}^{1/2}$	$M_{42} - \lvert M_{20}\rvert^2 - 2M_{21}^2$
$f_6 : C_{44}^{1/2}$	$M_{44} - M_{40}^2 - 18M_{22}^2 - 54M_{20}^4 - 144M_{11}^4 - 432M_{20}^2M_{11}^2 + 12M_{40}M_{20}^2 + 192M_{31}M_{11}M_{20} + 144M_{22}M_{11}^2 + 72M_{22}M_{20}^2$
$f_7 : C_{60}^{1/3}$	$M_{60} - 15M_{20}M_{40} + 30M_{20}^3$
$f_8 : C_{61}^{1/3}$	$M_{61} - 5M_{21}M_{40} - 10M_{20}M_{41} + 30M_{20}^3M_{21}$
$f_9 : C_{62}^{1/3}$	$M_{62} - 6M_{20}M_{42} - 8M_{21}M_{41} - M_{22}M_{40} + 6M_{20}^2M_{22} + 24M_{21}^2M_{20}$
$f_{10} : C_{63}^{1/3}$	$M_{63} - 9M_{21}M_{42} + 12M_{21}^3 - 3M_{20}M_{43} - 3M_{22}M_{41} + 18M_{20}M_{21}M_{22}$
$f_{11} : C_{80}^{1/4}$	$M_{80} - 28M_{60}M_{20} - 35M_{40}^2 + 420M_{40}M_{20}^2 - 630M_{20}^4$
$f_{12} : C_{84}^{1/4}$	$M_{84} - 16C_{63}C_{21} - \lvert C_{40}\rvert^2 - 18C_{42}^2 - 72C_{42}C_{21}^2 - 24C_{21}^4$
$f_{13} : \gamma_{max}$	$max\lvert\mathrm{DFT}(x(\cdot))\rvert^2/N$
$f_{14} : \sigma_{aa}^2$	$E[x_{cn}^2(i)] - E[\lvert x_{cn}(i)\rvert]^2$
$f_{15} : \sigma_v^2$	$E[x_v^2(i)] - E[\lvert x_v(i)\rvert]^2$
$f_{16} : \sigma_{dp}^2$	$E_{x(i)>x_t}[\varphi_{NL}^2(i)] - E_{x(i)>x_t}[\varphi_{NL}(i)]^2$
$f_{17} : \sigma_{ap}^2$	$E_{x(i)>x_t}[\varphi_{NL}^2(i)] - E_{x(i)>x_t}[\lvert\varphi_{NL}(i)\rvert]^2$
$f_{18} : v_{20}$	M_{42}/M_{21}^2
$f_{19} : \beta$	$\sum_{i=1}^{N} x_I^2(i)/\sum_{i=1}^{N} x_Q^2(i)$
$f_{20} : K$	$\left\lvert E[x_{cn}^4(i)]/E^2[x_{cn}^2(i)]\right\rvert$
$f_{21} : S$	$\left\lvert E[x_{cn}^3(i)]/E^{\frac{3}{2}}[x_{cn}^2(i)]\right\rvert$
$f_{22} : PAR$	$\max(\lvert x(\cdot)\rvert)/E[\lvert x(i)\rvert]$
$f_{23} : PRR$	$\max(\lvert x(\cdot)\rvert^2)/E[\lvert x(i)\rvert^2]$

C.4 Summary of Selected Statistical Features

Partial variables in Table C.1 are given by

$$x(i) = x_I(i) + j x_Q(i) \tag{C.4}$$

$$x_{cn}(i) = \frac{|x(i)|}{E[x(i)]} - 1 \tag{C.5}$$

$$x_v(i) = \sqrt{\frac{|x(i)|}{E^2[|x(i)| - E[x(i)]]}} - 1 \tag{C.6}$$

$$\varphi_{NL}(i) = phase(x(i)) - E[phase(x(i))] \tag{C.7}$$

In the Table C.1, the $E[\cdot]$ operation denotes the mathematical expectation; the $DFT(\cdot)$ denotes the discrete Fourier transform operation; N is the number of the received symbols; and $x(\cdot)$ represents all samples of the received signal x.