University of Alexandria

Computer and Communication Engineering Department

Modulation Classification

CC484 - Pattern Recognition

Made By:

Eyad Salama[1] and Nour Hesham[2]

[1]Computer and Communication Department, Alexandria University

[2]Computer and Communication Department, Alexandria University

January 27, 2023

Contents

1 Abstract **6**

2 Motivation and Background **7**

3 Introduction **8**

4 Modulations and Neural Networks **9**

4.1 About Our DataSet 9

4.2 OurCNN 10

4.3 OurRNN 12

4.4 OurLSTM 13

4.5 RNN vs LSTM 15

4.6 RNNs have recurrent connections and/or layers 15

4.7 LSTM can refer to a unit, layer or neural network 16

4.8 When should you use RNNs and LSTMs? 16

5 Results and Discussion **17**

5.1 Train all models on different feature spaces, no hyperparameter tweaking, find best combination, and tweak hyperparameters on diff models . 17

5.2 Train best feature space "raw" with diff learning rates on diff models . . 18

5.3 Testing best models on each SNR alone 19

5.3.1 MODEL: OurCNN LR=1e-04 19

5.3.2 MODEL: OurCNN LR=1e-04 20

5.3.3 MODEL: OurRNN LR=1e-03 21

5.3.4 MODEL: OurRNN$_LR = 5e-044$ 22

5.3.5 MODEL: OurLSTM$_LR = 1e-03$ 23

5.3.6 MODEL: |MODEL: OurLSTM$_L R = 5e - 04$ 24

5.4 Using Raw time series 25

5.5 Using First Derivative in time 26

5.6 Confusion Matrices 27

5.6.1 At SNR -20 27

5.6.2 At SNR -18 28

5.6.3 At SNR -16 29

5.6.4 At SNR -14 30

5.6.5 At SNR -12 31

5.6.6 At SNR -10 32

5.6.7 At SNR -8 33

5.6.8 At SNR -6 34

5.6.9 At SNR -4 35

5.6.10 At SNR -2 36

5.6.11 At SNR 0 37

5.6.12 At SNR 2 38

5.6.13 At SNR 4 39

5.6.14 At SNR 6 40

5.6.15 At SNR 8 41

5.6.16 At SNR 10 42

5.6.17 At SNR 12 43

5.6.18 At SNR 14 44

5.6.19 At SNR 16 45

5.6.20 At SNR 18 46

5.6.21 Plots for Some NN Training 46

6 Conclusion **48**

1 Abstract

Modulation identification is an essential component of signal interception analysis, which has long been a study focus in radio communication. Interference in signal transmission gets increasingly problematic as the electromagnetic spectrum environment grows more complicated. This work offers a modulation recognition strategy based on multimodal feature fusion to increase modulation identification performance across various channels.

In the signal preprocessing stage, various time- and frequency-domain properties are extracted as the network input. The residual shrinkage building unit with channel-wise thresholds (RSBU-CW) was used to create deep convolutional neural networks to extract spatial features that interact in pairs with temporal information collected by LSTM to boost feature diversity. Finally, the PNN model was modified to cross-fuse the network's extracted features in order to improve feature complementarity.

The simulation results show that the proposed technique outperforms existing feature fusion systems in terms of recognition performance, and it can also achieve high recognition performance in multipath fading channels.

2 Motivation and Background

There are numerous important reasons why machine learning has grown so fast in the last ten years. Algorithms have advanced considerably in a variety of ways, including momentum techniques of gradient descent and advances in regularisation and dropout, among others. Computa-tional power and concurrent programming models to fully use it have also increased substantially, thanks in large part to high-level languages that compile quickly to contemporary GPU implementations.
Another significant enabler has been access to big and well-curated datasets, which has allowed individuals to quickly compare ideas, test new techniques, and train models with enormous parameter spaces.
Many of the most recent improvements in machine learning are being driven by researchers in computer vision, speech recognition, natural language processing, medical images, and finance. Unfortunately, radio signal processing has been conspicuously absent from most of this recent work;
yet, the potential for applying recent improvements in machine learning to the radio domain is huge right now. Researchers are increasingly studying these approaches, however there are no clear benchmarks or available datasets for evaluating advancements, as seen in other application fields.

3 Introduction

Because of the high demand for wireless data researchers have recently focused on developing smart radios that can coexist.
The concept of Cognitive Radios was initially proposed by Mitola.
His suggested model has two sorts of users: licenced users or primary users (PUs) who have priority in using the spectrum bands in question, and unlicensed users or secondary users (SUs) who are permitted to broadcast in the specified frequency band as long as no interference occurs on the PUs side.
A CR's capacity to scan the spectrum to determine the type of users in the system is one of the listed features.
Following that, it will adjust its broadcast to avoid interference with neighbouring users.
To correctly categorise these users, the CR must begin a learning phase in which he studies the transmission properties of every incoming signal. The modulation technique utilised is one of the most important aspects of any signal.
To properly recover the signal, the transmitter and receiver must agree on the modulation scheme as part of the physical layer of any wireless radio.
A number of neural network types were also employed. Among the most plausible methods were Long Term Short Memory and Residual Networks (ResNet).
Another approach employed in was making a vanilla RNN instead of using the LSTM but the first approach had better results.
We also tried implementing an CNN neural network and most of the times it performed better than the vanilla RNN neural network
TODO:
The last approach that we tried was using a CONV-LSTM neural network

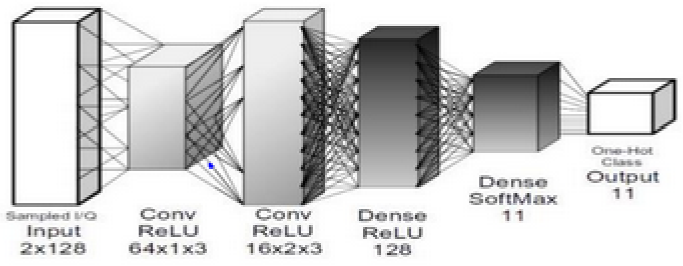
4 Modulations and Neural Networks

As we discussed in the previous section we used Three approaches but before discussing each architecture we need an brief introduction about our DataSet that we used.

4.1 About Our DataSet

A synthetic dataset of 10 modulations created with GNU Radio. This is a variable-SNR dataset with modest LO drift, light fading, and a variety of labelled SNR increments for assessing performance in various signal and noise power circumstances.
We have 20 SNR values [-18, -16,... 18] and 10 modulation types (8 digital and 2 analog modulation types).
So in total we have 200 combinations and each combination has 6000 samples so, in total we have 1.2M samples as our dataset. In this experiment we only care about the modulation type and not the corresponding SNR value so our neural networks is going to have 10 heads (each head for a specific modulation type)

4.2 OurCNN

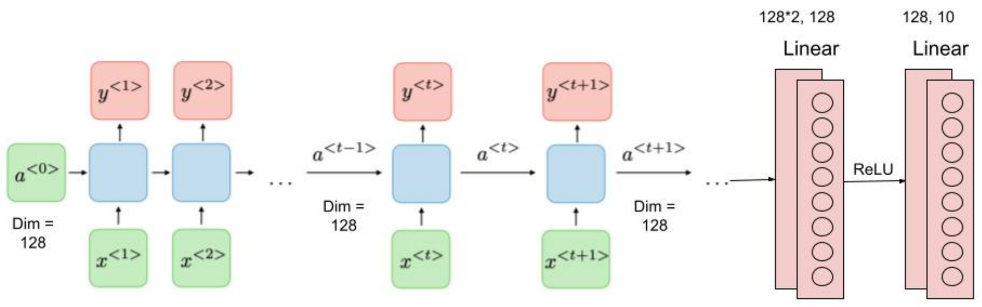


```python
class OurCNN(nn.Module):
    def __init__(self, num_classes, input_shape=(2, 128)):
        super().__init__()
        self.num_classes = num_classes
        self.input_shape = input_shape
        self.batchnorm = nn.BatchNorm2d(2)
        self.conv1 = nn.Sequential(
            nn.Conv2d(
                in_channels=1,
                out_channels=64,
                kernel_size=(1, 3),
                stride=1,
                padding='same'
            ),
            nn.BatchNorm2d(64),
            nn.ReLU(),
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(
                in_channels=64,
                out_channels=16,
                kernel_size=(2, 3),
                stride=1,
                padding='same'
            ),
            nn.BatchNorm2d(16),
            nn.ReLU()
        )
        self.relu = nn.ReLU()
        self.flatten = nn.Flatten()
        self.softmax = nn.Softmax(dim=1)
        self.dropout = nn.Dropout(p=0.5)
        n_size = self._get_conv_output(self.input_shape)
        self.linear_1 = nn.Linear(n_size, 128)
        self.linear_2 = nn.Linear(128, 10)

    def forward(self, input):
        x = torch.unsqueeze(input, dim=1)
        x = self.conv1(x)
        x = self.dropout(x)
        x = self.conv2(x)
        x = self.dropout(x)
        x = self.flatten(x)
        x = self.linear_1(x)
        x = self.relu(x)
        x = self.linear_2(x)
        #x = self.softmax(x)
        return x

    def _get_conv_output(self, shape):
        bs = 1
        input_data = Variable(torch.rand(bs, *shape))
        input_data = torch.unsqueeze(input_data, dim=1)
        x = self.conv1(input_data)
        x = self.conv2(x)
        x = self.flatten(x)
        n_size = x.data.view(bs, -1).size(1)
        return n_size
```

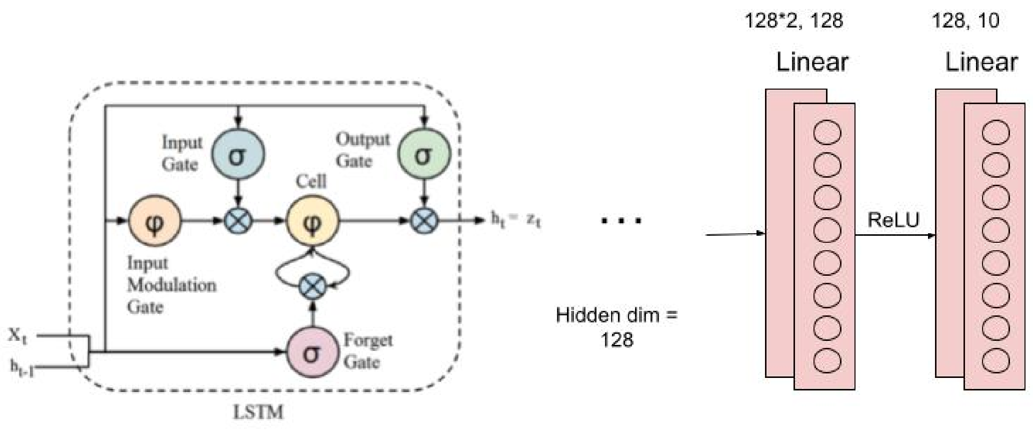
4.3 OurRNN



```
    class OurRNN(nn.Module):

    def __init__(self, num_classes, input_dim, hidden_dim):
        super().__init__()
        self.device = torch.device("cuda:0"if torch.cuda.
                                        is_available()else"cpu")
        self.num_classes = num_classes
        self.input_dim = input_dim
        self.hidden_dim = hidden_dim
        self.batchnorm = nn.BatchNorm1d(2)
        self.rnn = nn.RNN(
            input_size=input_dim,
            hidden_size=hidden_dim,
            num_layers = 1,
            batch_first=True,
            nonlinearity='tanh')
        self.fc1 = nn.Linear(hidden_dim*2, 128)
        self.fc2 = nn.Linear(128, num_classes)
        self.relu = nn.ReLU()
    def forward(self, x):
        x = self.batchnorm(x)
        h0 = Variable(torch.zeros(1, x.size(0), self.hidden_dim)).
                                        to(self.device)
        out, hn = self.rnn(x, h0)
        out = out.contiguous().view(x.size(0),-1)
        out = self.fc1(out)
        out = self.relu(out)
        out = self.fc2(out)
        return out
```

4.4 OurLSTM

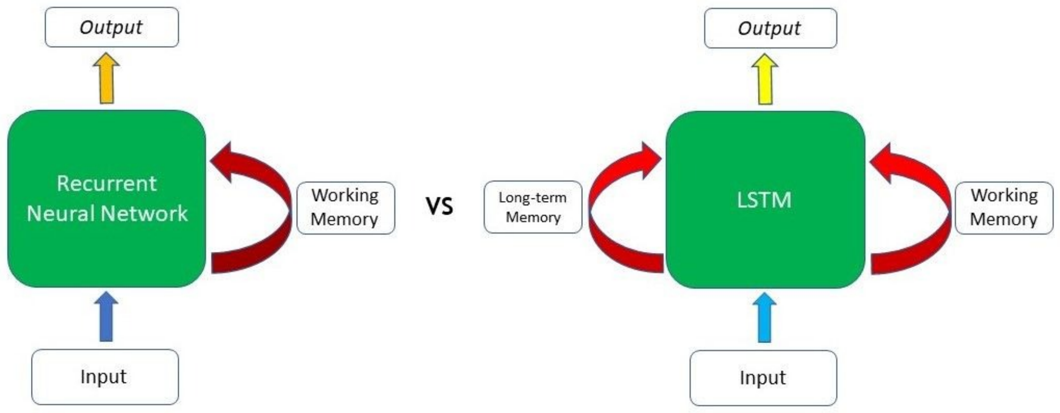


```
class OurLSTM(nn.Module):

def __init__(self, num_classes, input_dim, hidden_dim):
    super().__init__()
    self.device = torch.device("cuda:0" if torch.cuda.
                                        is_available() else "cpu"
                                        )
    self.num_classes = num_classes
    self.input_dim = input_dim
    self.hidden_dim = hidden_dim
    self.batchnorm = nn.BatchNorm1d(2)
    self.lstm = nn.LSTM(
        input_size=input_dim,
        hidden_size=hidden_dim,
        num_layers = 1,
        batch_first=True)
    self.fc1 = nn.Linear(hidden_dim*2, 128)
    self.fc2 = nn.Linear(128, num_classes)
    self.relu = nn.ReLU()

def forward(self, x):
    x = self.batchnorm(x)
    h0 = Variable(torch.zeros(1, x.size(0), self.hidden_dim)).
                                        to(self.device)
    c0 = Variable(torch.zeros(1, x.size(0), self.hidden_dim)).
                                        to(self.device)
    out, (hn, cn) = self.lstm(x, (h0,c0))
    out = out.contiguous().view(x.size(0),-1)
    out = self.fc1(out)
    out = self.relu(out)
    out = self.fc2(out)
    return out
```

4.5 RNN vs LSTM



4.6 RNNs have recurrent connections and/or layers

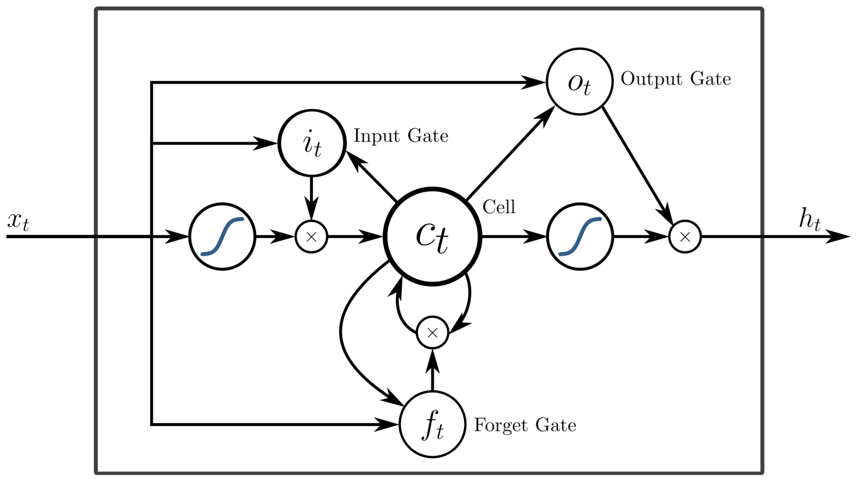
Depending on the context, you may define a recurrent neural network (RNN) or a long short-term memory (LSTM) at different degrees of abstraction. An RNN, for example, might be defined as any neural network with one or more recurrent (or cyclic) connections. Layer l of neural network N is a recurrent layer since it includes units (or neurons) with recurrent connections, whereas N is not.

may not only have recurring layers (for example, it may also be composed of feedforward layers, i.e. layers with units that contain only feedforward connections).

In any event, a recurrent neural network (RNN) is generally always referred to as a neural network (NN) rather than a layer (this should also be obvious from the name).

4.7 LSTM can refer to a unit, layer or neural network

However, depending on the context, the word "LSTM" might refer to an LSTM unit (or neuron), an LSTM layer (a collection of LSTM units), or an LSTM neural network (a neural network with LSTM units or layers).
LSTMs are another name for neural networks incorporating LSTM units (plural version of LSTM).



we can see from this diagram that an LSTM unit (or layer) is composed of gates

1. i -> input gate.
2. o -> output gate.
3. f -> forget gate

4.8 When should you use RNNs and LSTMs?

RNNs are particularly well-suited for jobs involving sequences (thanks to the recurrent connections). They are frequently employed in machine translation, for example, when the sequences represent phrases or words.
In practise, an LSTM is frequently used instead of a vanilla (or standard) RNN since it is more computationally efficient.
In reality, the LSTM was developed to address a difficulty that regular RNNs face, namely the vanishing gradient problem.

5 Results and Discussion

5.1 Train all models on different feature spaces, no hyperparameter tweaking, find best combination, and tweak hyperparameters on diff models

Arch	CNN	RNN, hid = 128	LSTM, hid = 128	CNN	RNN
LR	1.00E-03	1.00E-03	1.00E-03	1.00E-03	1.00E-03
Batch	256	256	256	256	256
Epochs	10	10	10	10	10
Features	Raw	Raw	Raw	Derivative	Derivative
Acc (train, val) E	0.545, 0.512	0.487, 0.481	0.537, 0.5159	0.3705, 0.354	0.317, 0.195
F1 (train, val) E	0.548, 0.518	0.499, 0.495	0.55, 0.54	0.3953, 0.401	0.3327, 0.243
Acc (train, val) S	0.652, 0.625	0.585, 0.57	0.666, 0.6875	0.503, 0.5	0.414, 0.375
F1 (train, val) S	0.658, 0.642	0.587, 0.658	0.6837, 0.684	0.559, 0.494	0.476, 0.405

Arch	LSTM
LR	1.00E-03
Batch	256
Epochs	10
Features	Derivative
Acc (train, val) E	0.334, 0.328
F1 (train, val) E	0.3605, 0.35
Acc (train, val) S	0.4414, 0.4257
F1 (train, val) S	0.4698, 0.443

5.2 Train best feature space "raw" with diff learning rates on diff models

Arch	CNN	RNN	LSTM	CNN	RNN
LR	1.00E-04	1.00E-04	1.00E-04	5.00E-04	5.00E-04
Batch	256	256	256	256	256
Epochs	10	10	10	10	10
Features	Raw	Raw	Raw	Raw	Raw
Acc (train, val) E	0.51, 0.5527	0.371, 0.373	0.415, 0.415	0.514, 0.545	0.395, 0.3902
F1 (train, val) E	0.4738, 0.5244	0.313, 0.329	0.349, 0.366	0.4907, 0.511	0.3506, 0.416
Acc (train, val) S	0.617, 0.75	0.4609, 0.625	0.5833, 0.625	0.644, 0.8125	0.496, 0.75
F1 (train, val) S	0.626, 0.7455	0.492, 0.659	0.5816, 0.6406	0.647, 0.7927	0.5416, 0.7946

Arch	LSTM	CNN	RNN	LSTM
LR	5.00E-04	5.00E-03	5.00E-03	5.00E-03
Batch	256	256	256	256
Epochs	10	10	10	10
Features	Raw	Raw	Raw	Raw
Acc (train, val) E	0.49, 0.479	0.474, 0.38	0.403, 0.414	0.498, 0.518
F1 (train, val) E	0.427, 0.435	0.459, 0.269	0.3688, 0.386	0.458, 0.481
Acc (train, val) S	0.597, 0.6875	0.574, 0.6875	0.519, 0.625	0.605, 0.8125
F1 (train, val) S	0.6222, 0.709	0.588, 0.78125	0.557, 0.717	0.616, 0.803

5.3 Testing best models on each SNR alone

5.3.1 MODEL: OurCNN LR=1e-04

SNR	Accuracy
-20	0.1171875
-18	0.1328125
-16	0.140625
-14	0.1875
-12	0.2109375
-10	0.29296875
-8	0.375
-6	0.51171875
-4	0.609375
-2	0.7421875
0	0.7734375
2	0.75
4	0.78515625
6	0.8203125
8	0.80859375
10	0.80078125
12	0.7734375
14	0.82421875
16	0.81640625
18	0.76171875

5.3.2 MODEL: OurCNN LR=1e-04

SNR	Accuracy
-20	0.109375
-18	0.11328125
-16	0.109375
-14	0.1328125
-12	0.16015625
-10	0.20703125
-8	0.27734375
-6	0.42578125
-4	0.54296875
-2	0.703125
0	0.66015625
2	0.73046875
4	0.7578125
6	0.78125
8	0.765625
10	0.80859375
12	0.72265625
14	0.75
16	0.7734375
18	0.75390625

5.3.3 MODEL: OurRNN LR=1e-03

SNR	Accuracy
-20	0.09375
-18	0.10546875
-16	0.1171875
-14	0.16015625
-12	0.1953125
-10	0.33203125
-8	0.40625
-6	0.4296875
-4	0.609375
-2	0.71484375
0	0.76953125
2	0.73828125
4	0.77734375
6	0.8125
8	0.79296875
10	0.81640625
12	0.78515625
14	0.80078125
16	0.8125
18	0.76953125

5.3.4 MODEL: OurRNN$_L R = 5e - 044$

SNR	Accuracy
-20	0.14453125
-18	0.1484375
-16	0.1328125
-14	0.16796875
-12	0.203125
-10	0.23828125
-8	0.390625
-6	0.41015625
-4	0.49609375
-2	0.50390625
0	0.61328125
2	0.44140625
4	0.5546875
6	0.5546875
8	0.55078125
10	0.515625
12	0.50390625
14	0.5546875
16	0.5234375
18	0.49609375

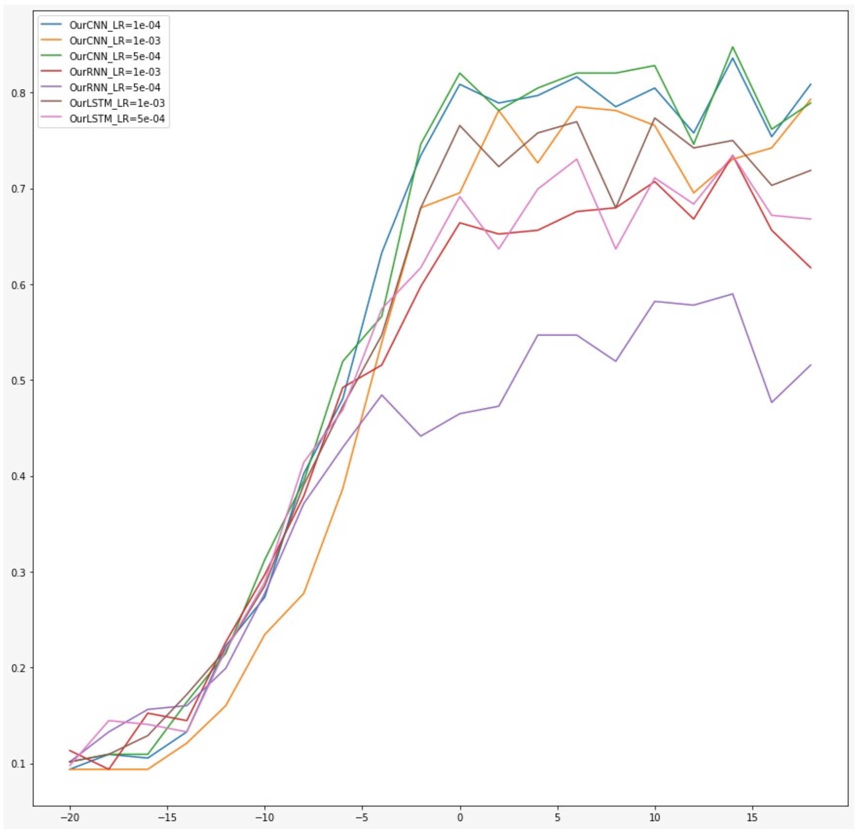
5.3.5 MODEL: OurLSTM$_L R = 1e-03$

SNR	Accuracy
-20	0.10546875
-18	0.1328125
-16	0.1640625
-14	0.1875
-12	0.2265625
-10	0.34765625
-8	0.39453125
-6	0.5
-4	0.60546875
-2	0.6484375
0	0.76953125
2	0.66796875
4	0.734375
6	0.77734375
8	0.76171875
10	0.734375
12	0.69140625
14	0.75
16	0.70703125
18	0.6796875

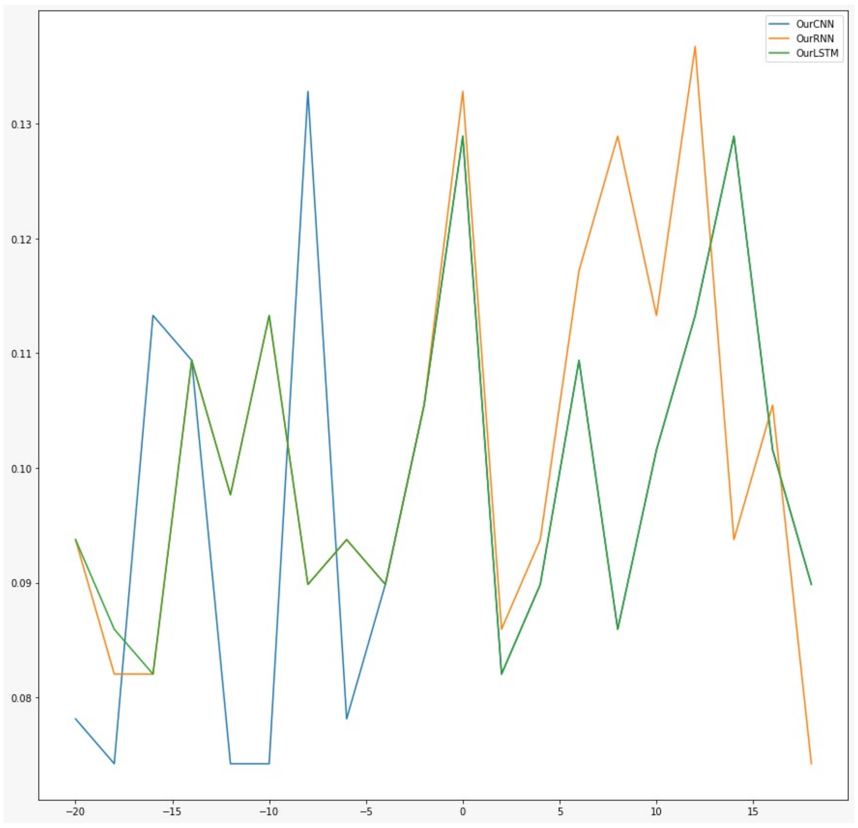
5.3.6 MODEL: |MODEL: OurLSTM$_LR = 5e - 04$

SNR	Accuracy
-20	0.15625
-18	0.14453125
-16	0.14453125
-14	0.140625
-12	0.203125
-10	0.3203125
-8	0.421875
-6	0.48046875
-4	0.55078125
-2	0.63671875
0	0.703125
2	0.64453125
4	0.69140625
6	0.75
8	0.73828125
10	0.6875
12	0.67578125
14	0.69140625
16	0.68359375
18	0.66015625

5.4 Using Raw time series

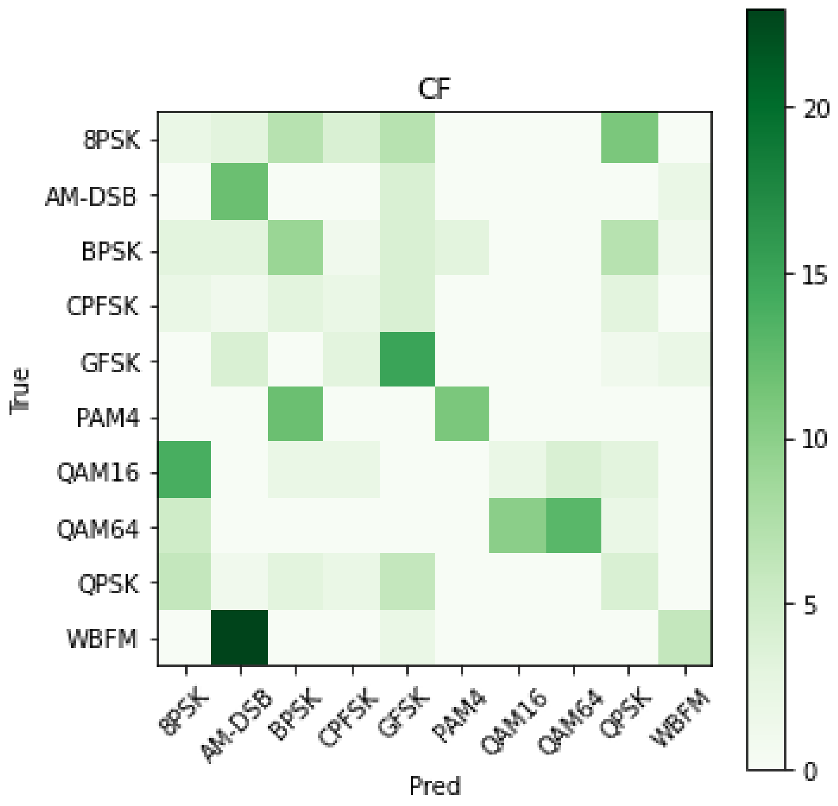


5.5 Using First Derivative in time

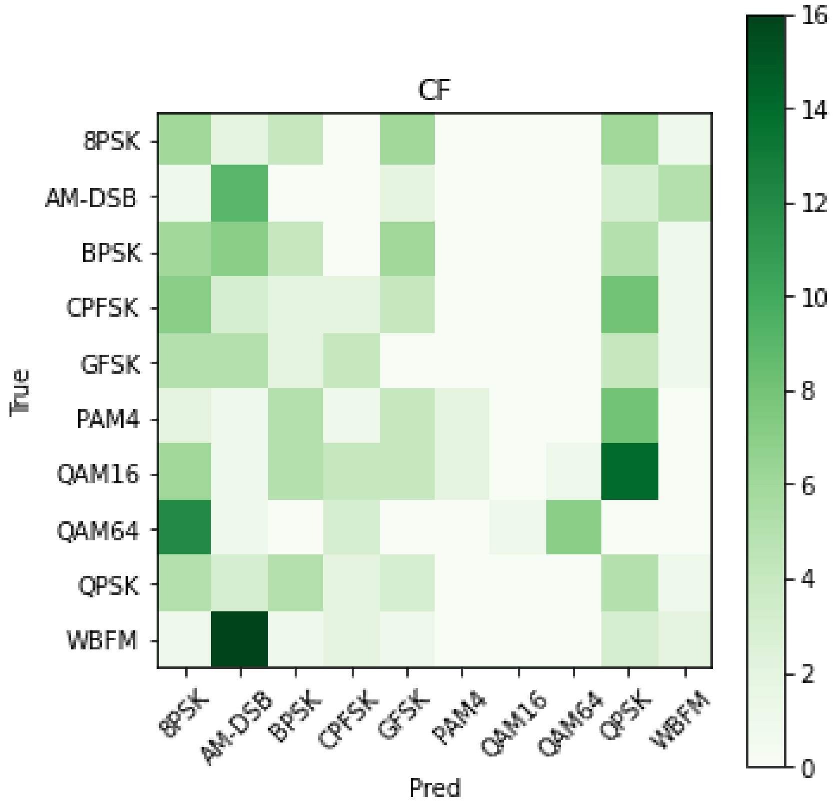


5.6 Confusion Matrices

5.6.1 At SNR -20

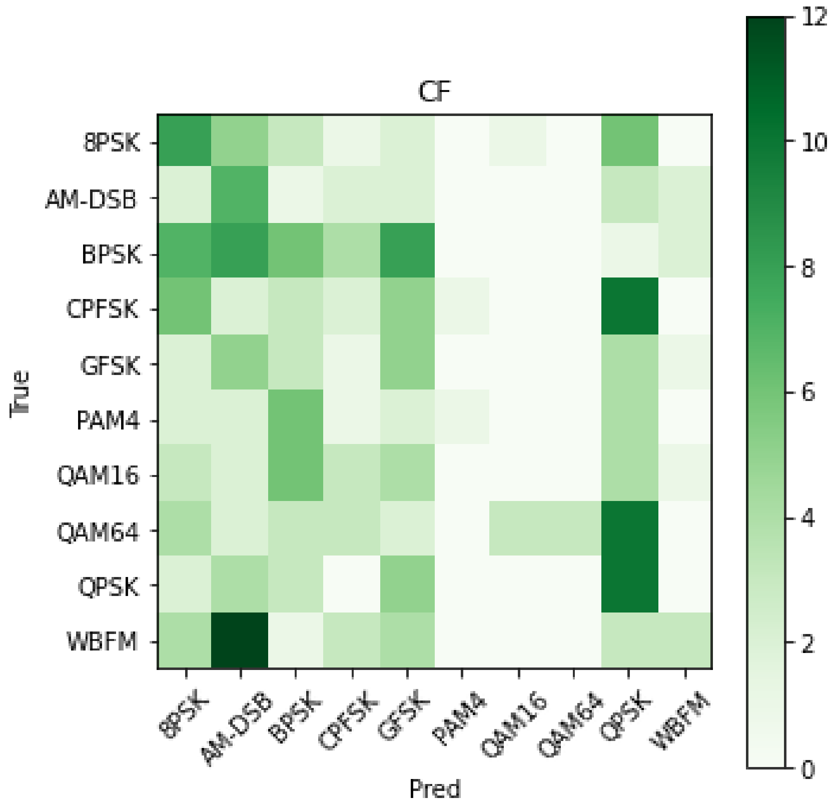


5.6.2 At SNR -18

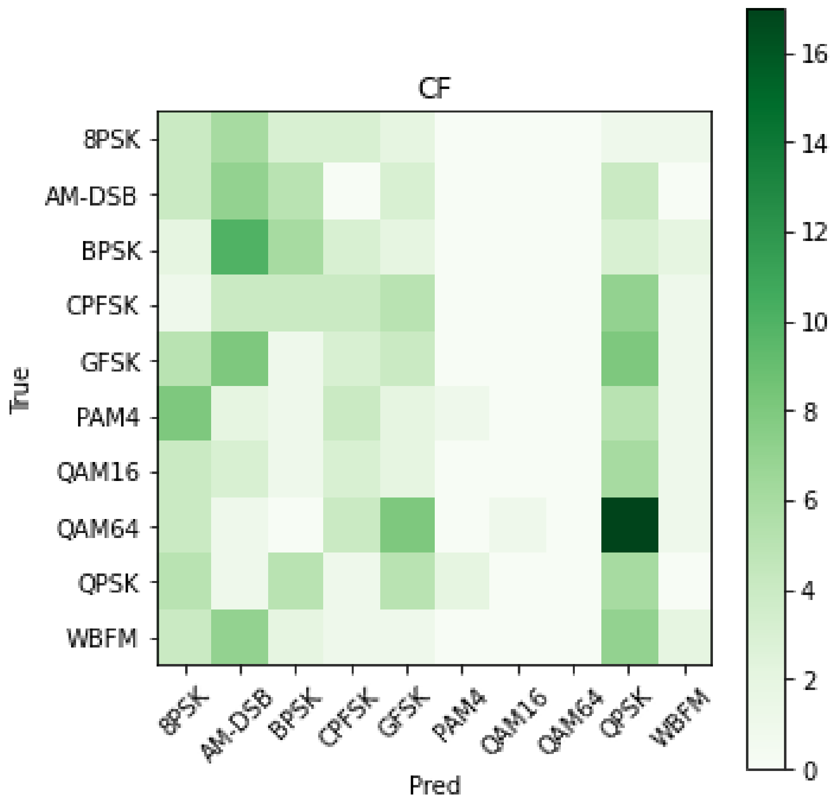


CF
True
Pred
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0
2
4
6
8
10
12
14
16

5.6.3 At SNR -16

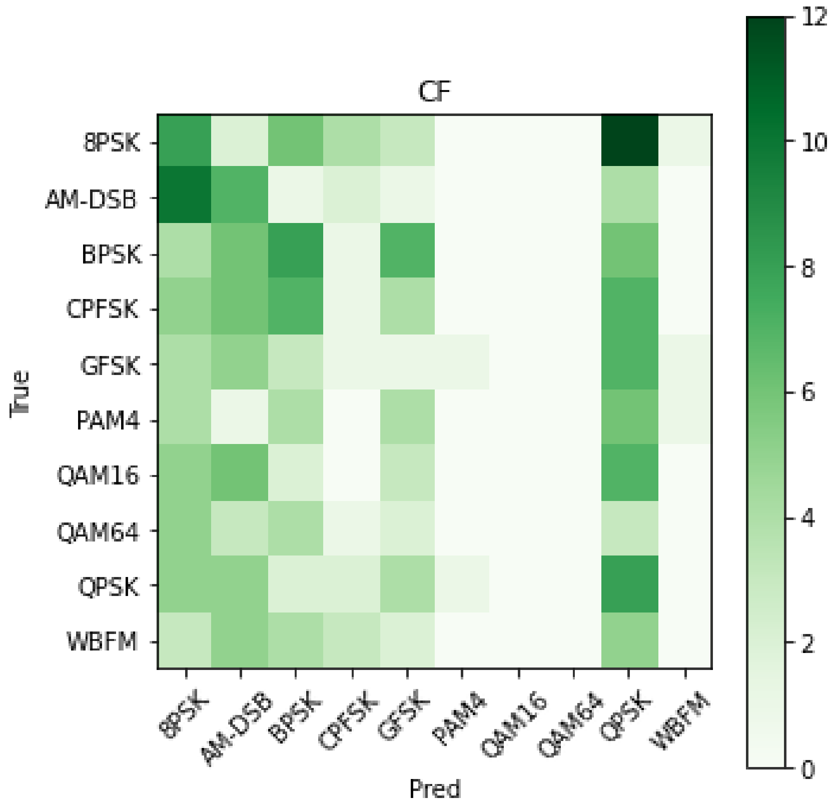

CF
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
True
Pred
0
2
4
6
8
10
12

5.6.4 At SNR -14

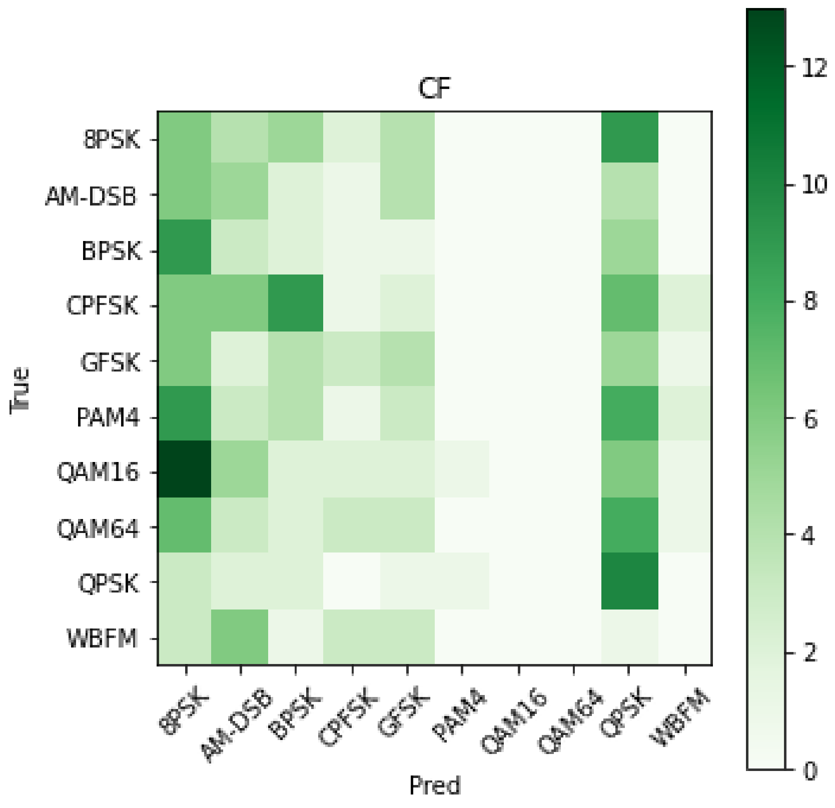


CF
True
Pred
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0
2
4
6
8
10
12
14
16

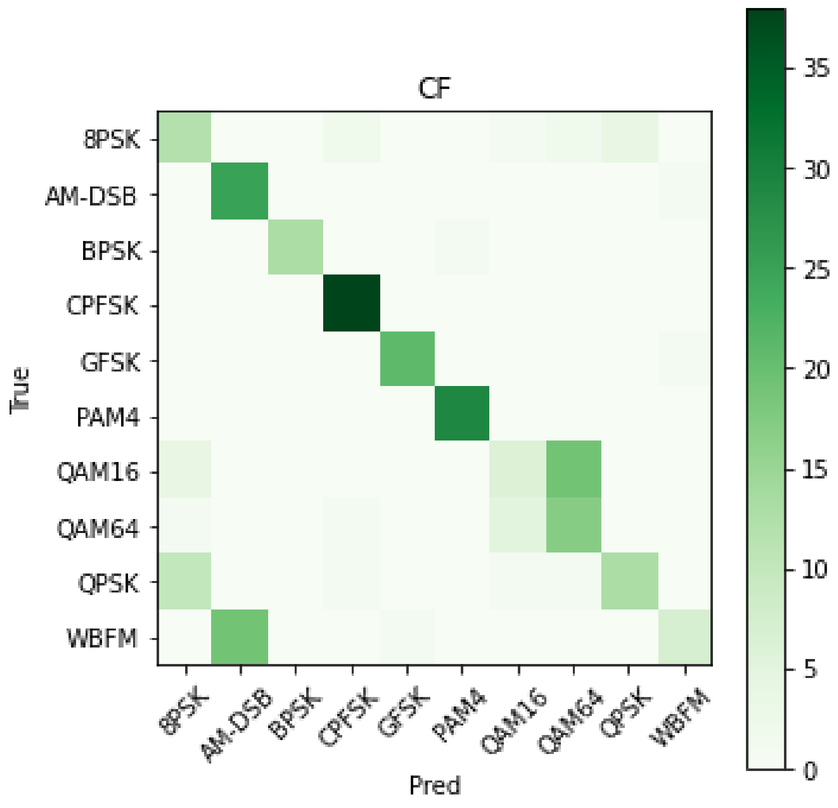
5.6.5 At SNR -12


CF
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
True
Pred
0
2
4
6
8
10
12

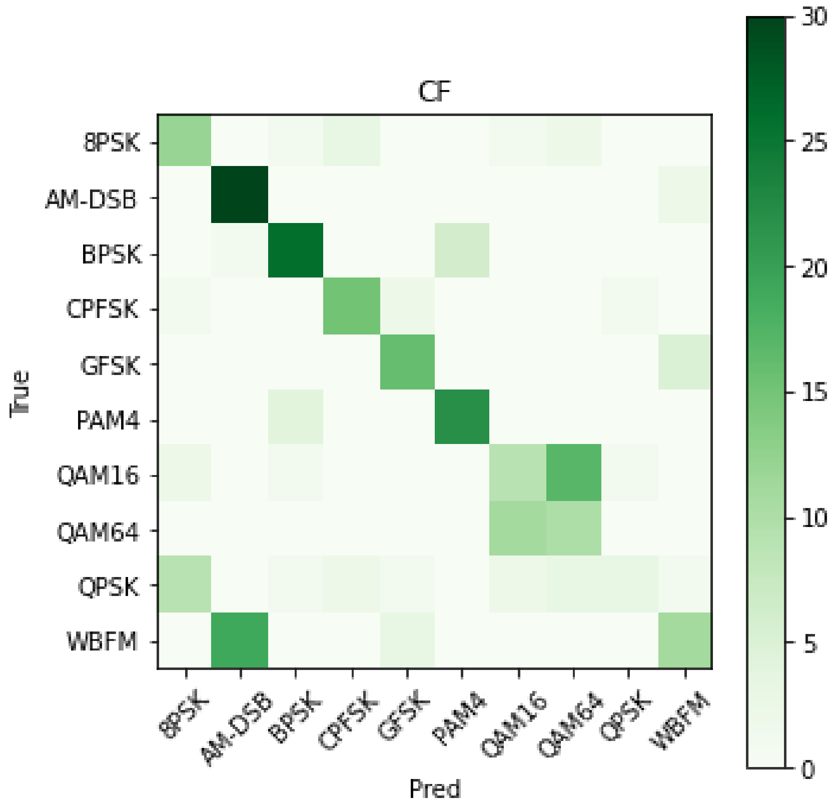
5.6.6 At SNR -10


CF
True
Pred
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0
2
4
6
8
10
12

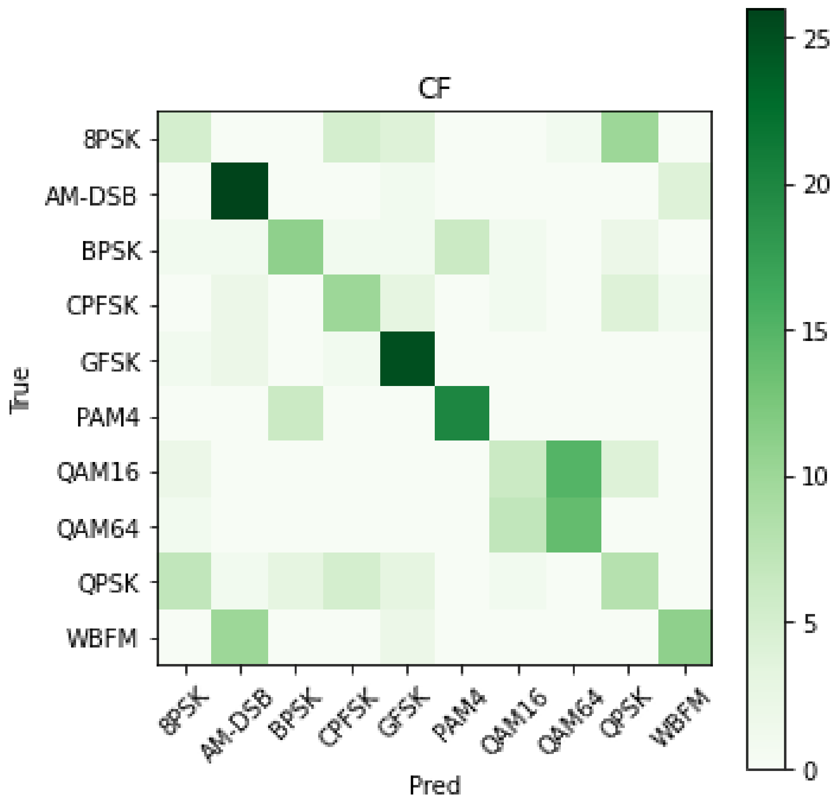
5.6.7 At SNR -8


CF
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
True
Pred
35
30
25
20
15
10
5
0

5.6.8 At SNR -6

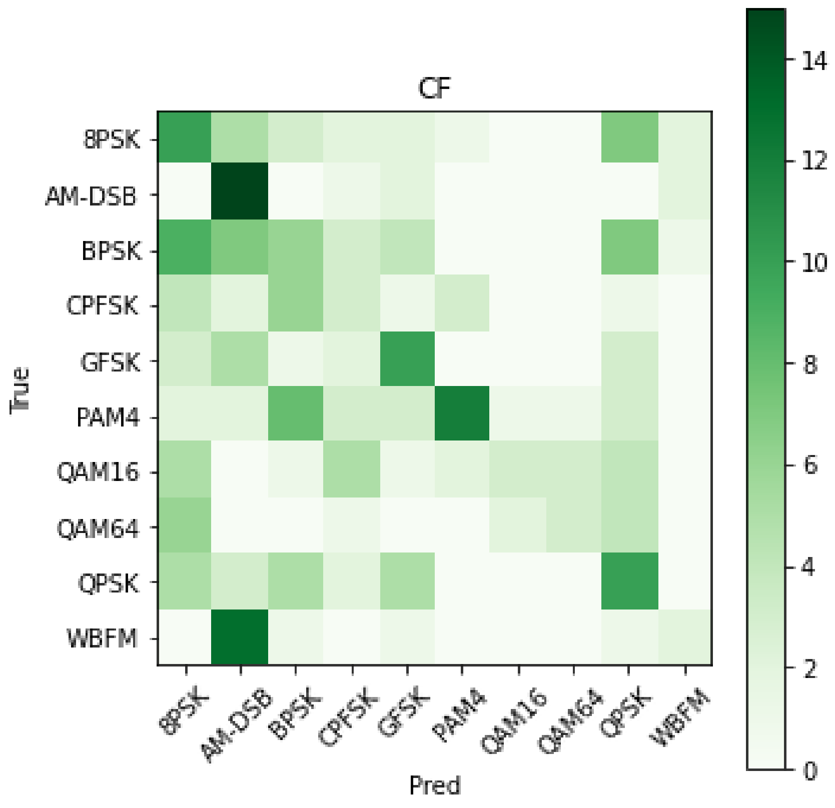

CF
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
True
Pred
30
25
20
15
10
5
0

5.6.9 At SNR -4

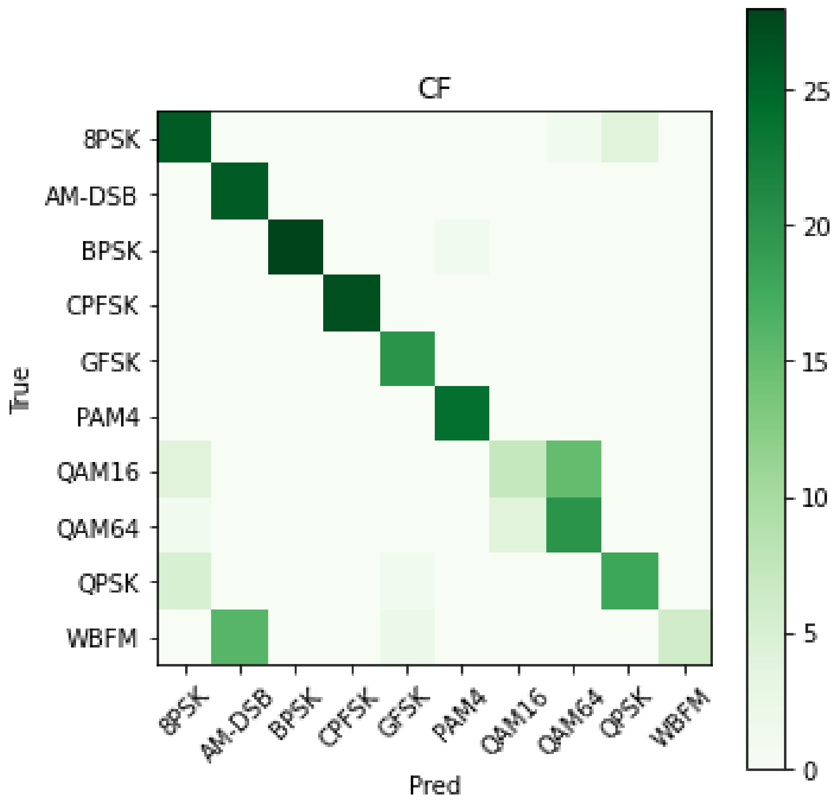


CF
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
True
Pred
0
5
10
15
20
25

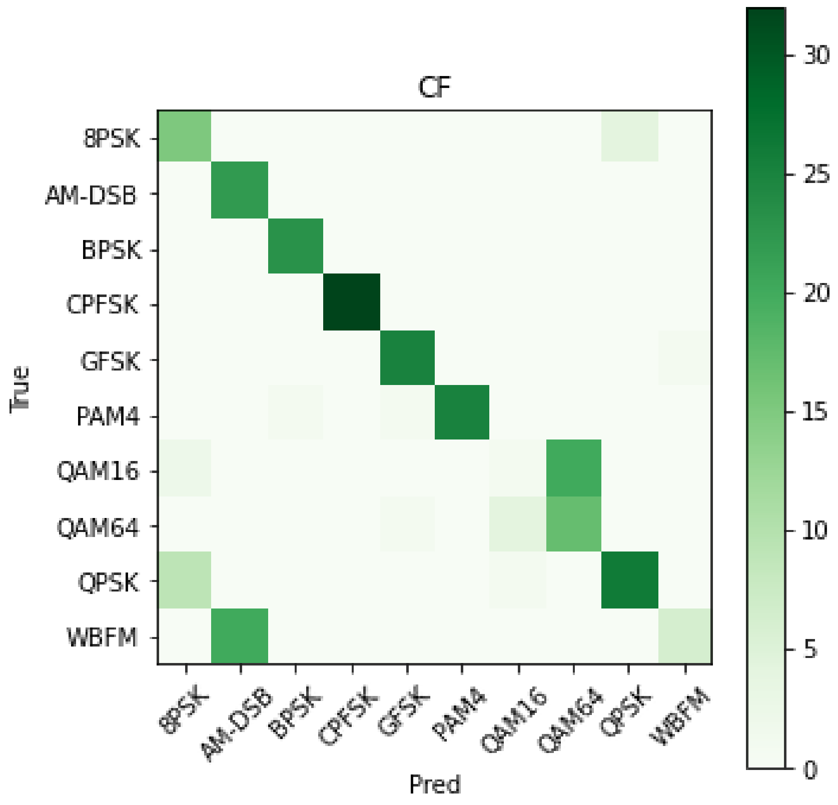
5.6.10 At SNR -2


CF
True
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
Pred
0
2
4
6
8
10
12
14

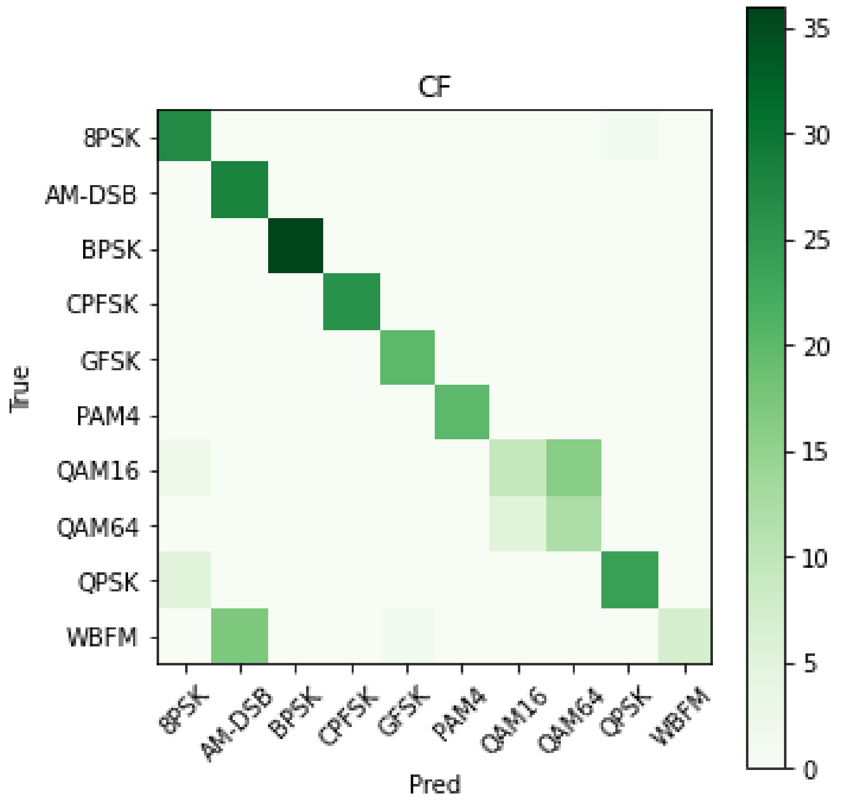
5.6.11 At SNR 0


CF
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
True
Pred
25
20
15
10
5
0

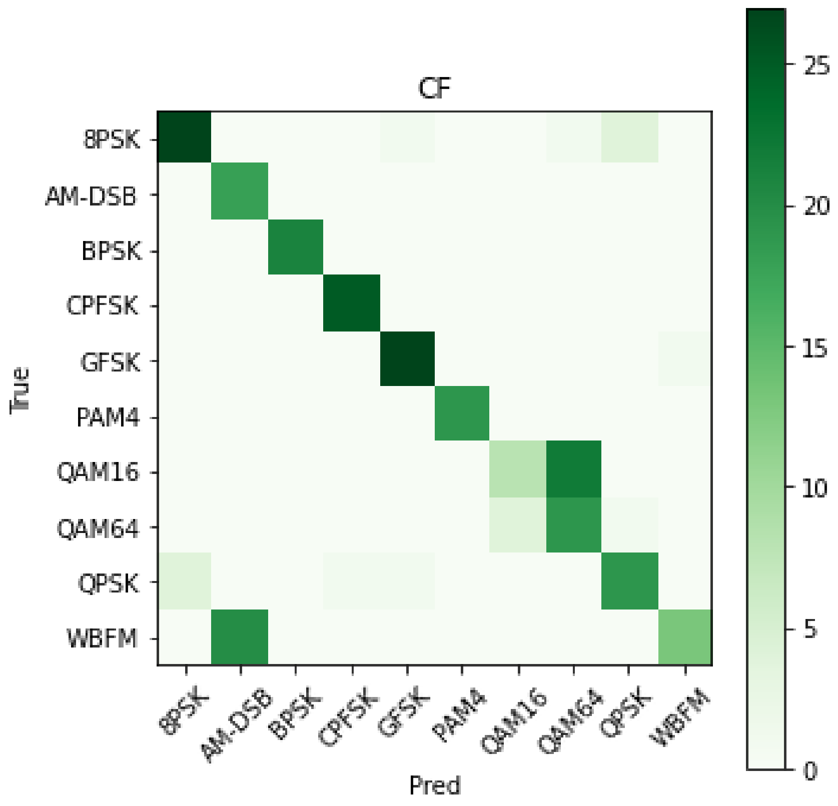
5.6.12 At SNR 2


CF
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
True
Pred
0
5
10
15
20
25
30

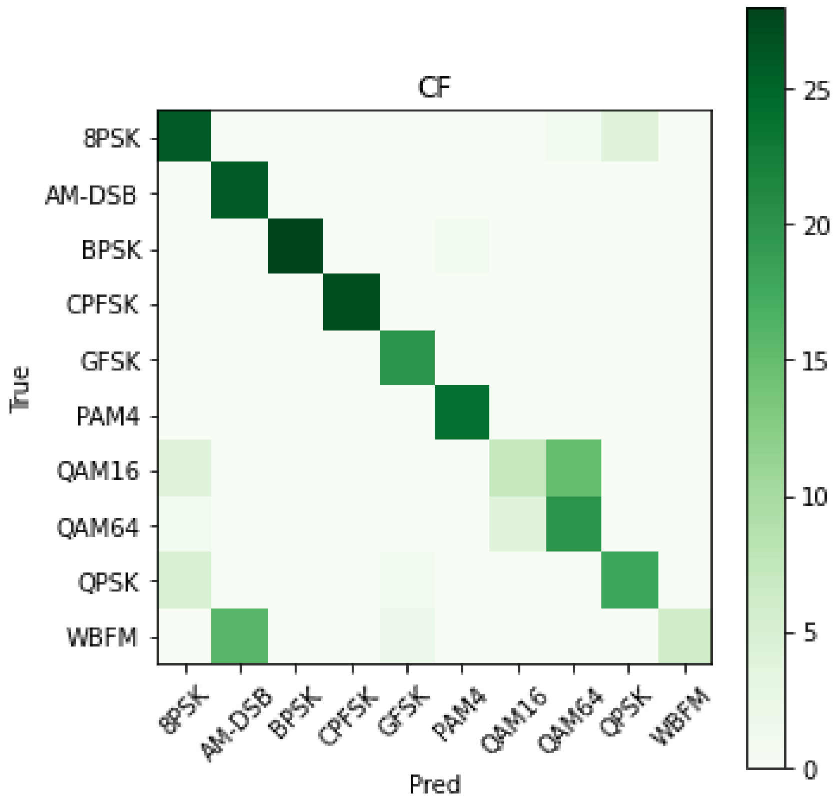
5.6.13 At SNR 4


CF
True
Pred
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
35
30
25
20
15
10
5
0

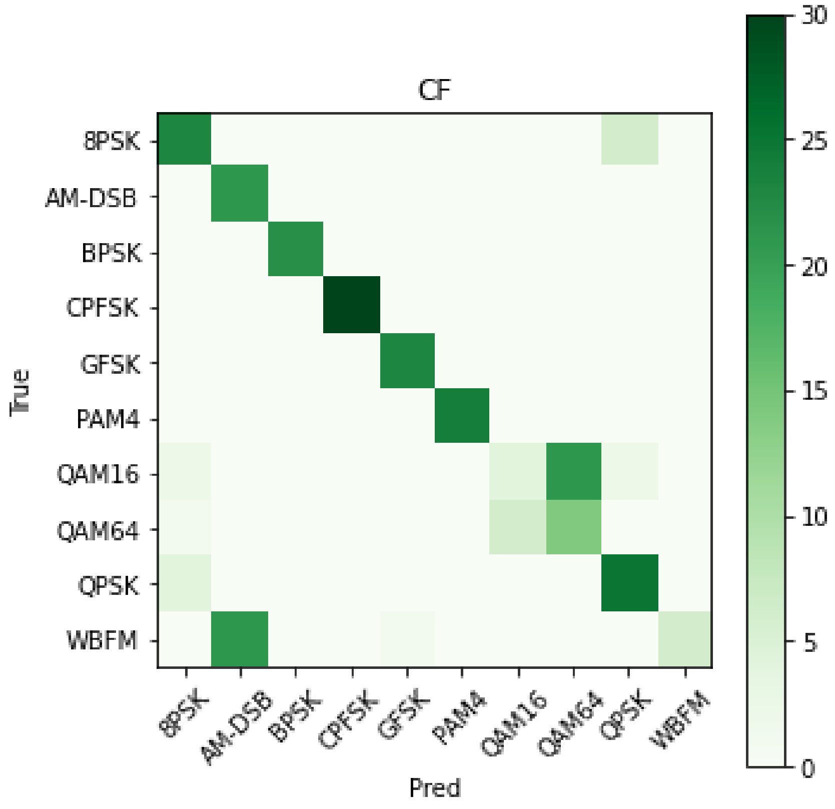
5.6.14 At SNR 6


CF
True
Pred
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0
5
10
15
20
25

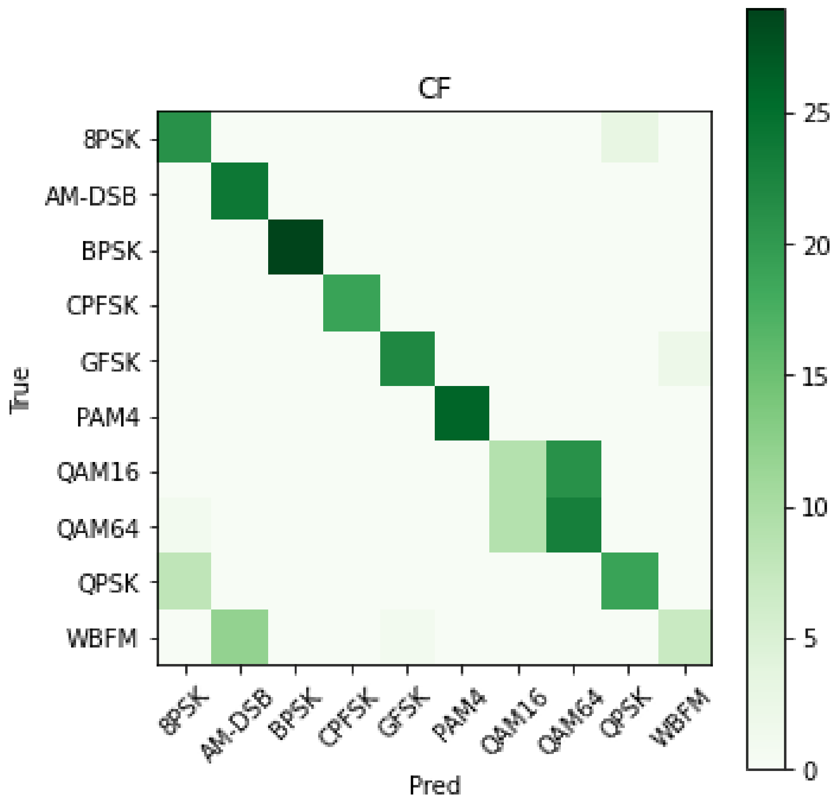
5.6.15 At SNR 8


CF
True
Pred
8PSK
AM-DSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0
5
10
15
20
25

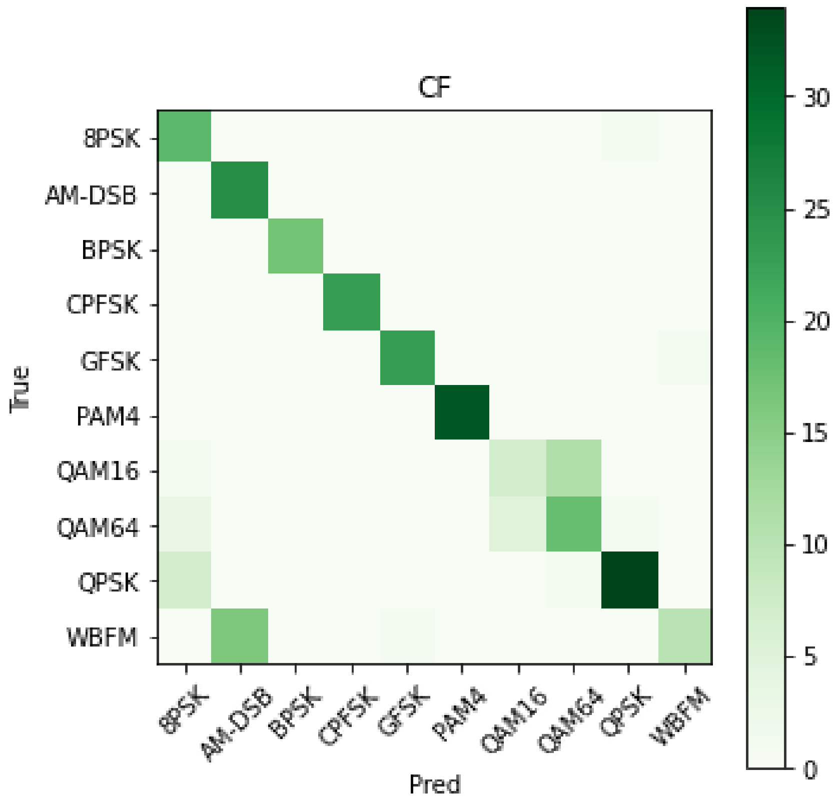
5.6.16 At SNR 10



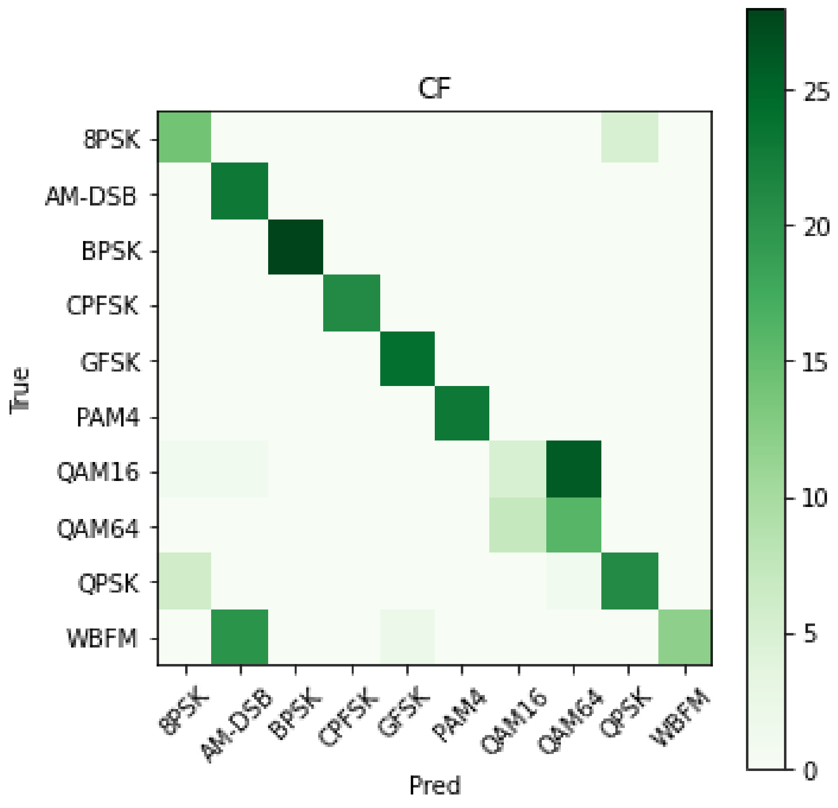
5.6.17 At SNR 12



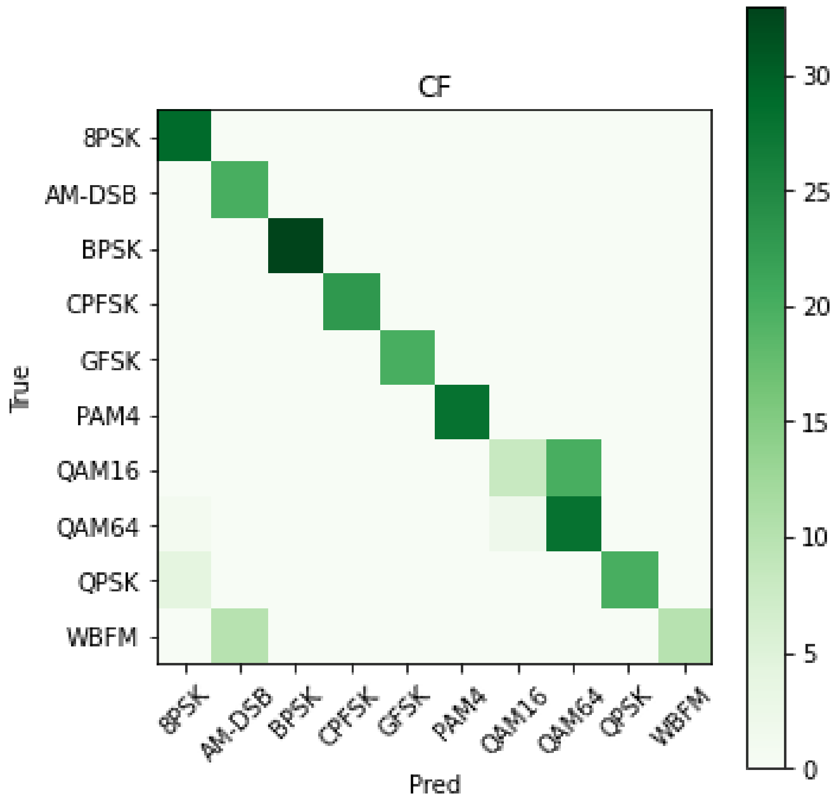
5.6.18 At SNR 14



5.6.19 At SNR 16

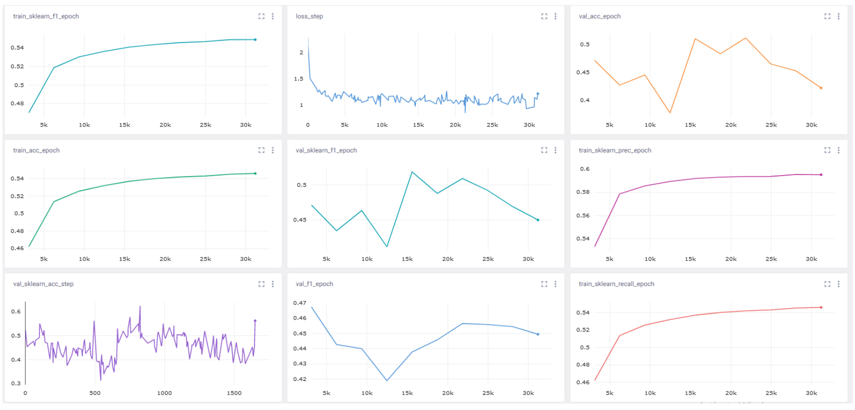


5.6.20 At SNR 18

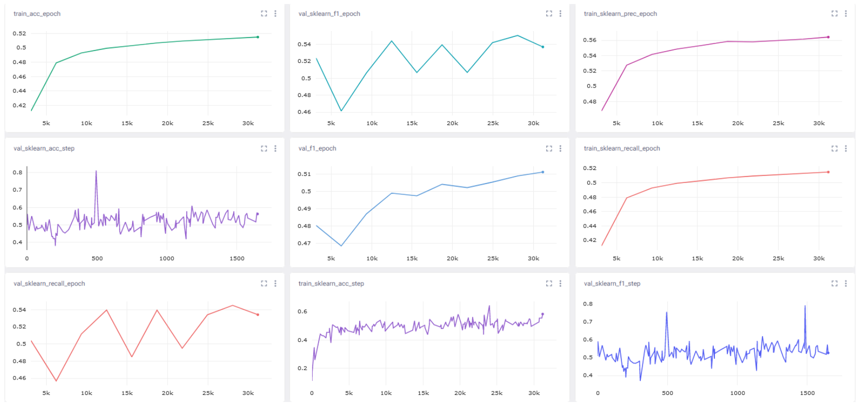


5.6.21 Plots for Some NN Training

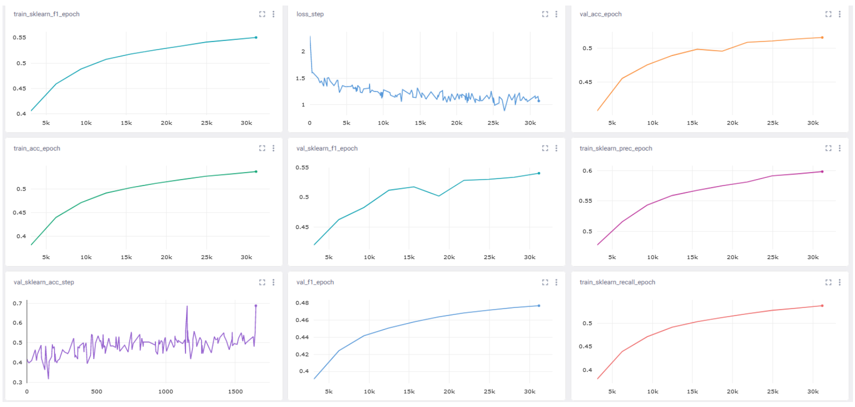
OurCNN-raw-10-epochs



OurCNN-raw-10epochs-b256-lr0.0005-weighted



OurLSTM-raw-10epochs-b256-lr0.001-weighted



6 Conclusion

- Raw is better than derivative.
- CNN and LSTM are usually better than Vanillar RNN.