Pattern Recognition
Modulation Classification LaTeX report

Nouran Hisham 6532

June 2022

Index

Problem statement

DeepSig Dataset: RadioML 2016.04C
A synthetic dataset, generated with GNU Radio, consisting of 11 modulations. This is a variable-SNR dataset with moderate LO drift, light fading, and numerous different labeled SNR increments for use in measuring performance across different signal and noise power scenarios.

The code can be viewed from here

1 Download the dataset

1.1 Dataset format

Every sample is presented using two vectors each of them has 128 elements. It contains 10 different modulation techniques.

```
[ ]  from google.colab import drive
     drive.mount('/content/drive')
```

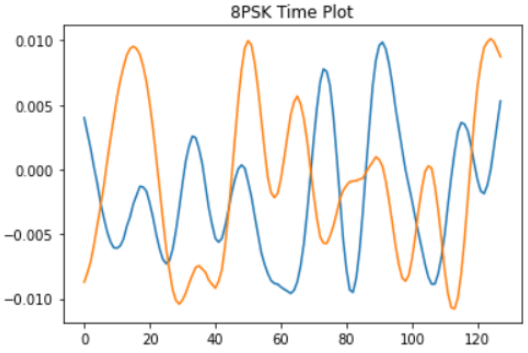
```
Mounted at /content/drive
```

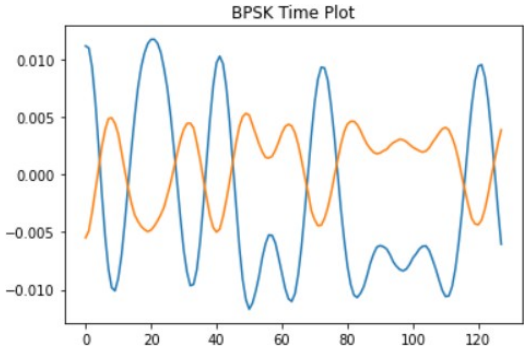
```
[ ]  !tar -xvf '/content/drive/MyDrive/RML2016/RML2016.10b.tar.bz2'
```

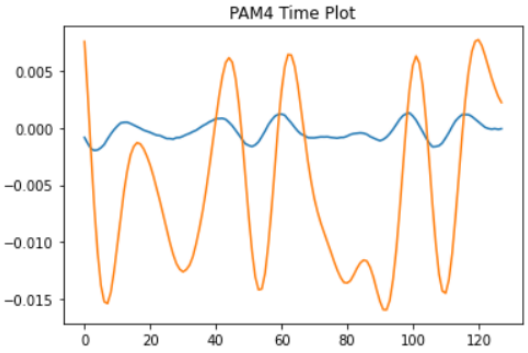
```
RML2016.10b.dat
LICENSE.TXT
```

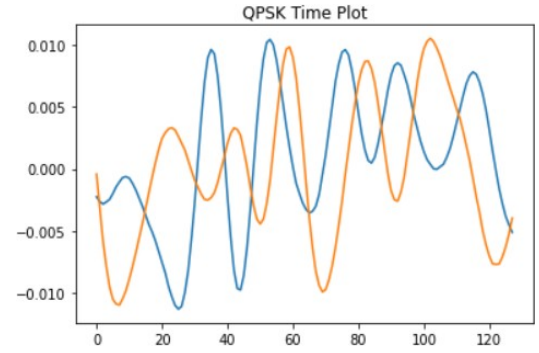
```
[2]  file = open("RML2016.10b.dat",'rb')
     Xd = pickle.load(file, encoding = 'bytes')
     snrs, mods = map(lambda j: sorted(list(set(map(lambda x: x[j], Xd.keys())))), [1,0])
     X = []
     lbl = []
     for mod in mods:
         for snr in snrs:
             X.append(Xd[(mod,snr)])
             for i in range(Xd[(mod,snr)].shape[0]):  lbl.append((mod,snr))
     X = np.vstack(X)
     file.close()
```

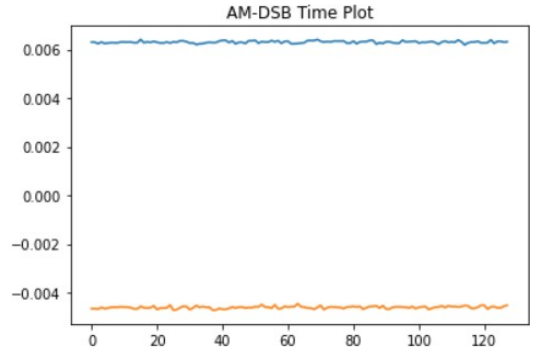
1.2 Dataset visualisation

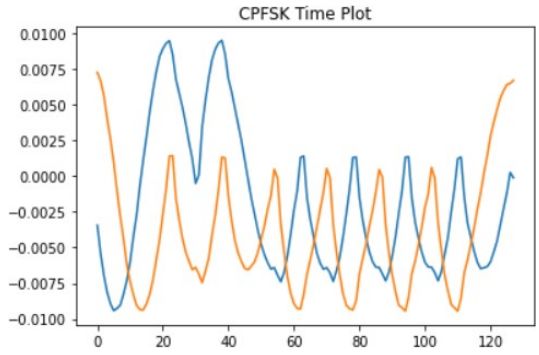


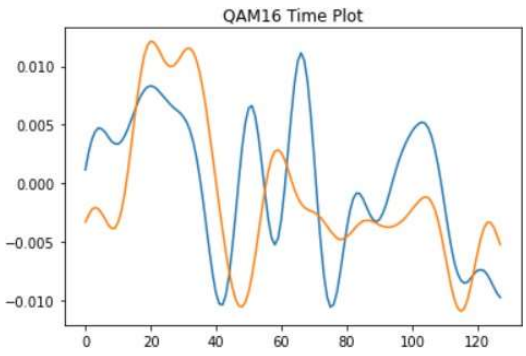


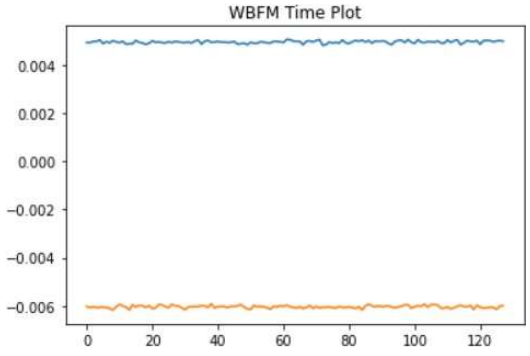


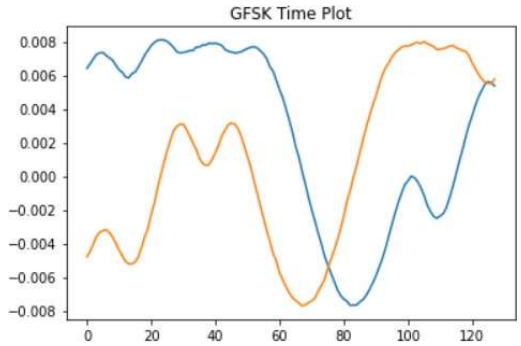


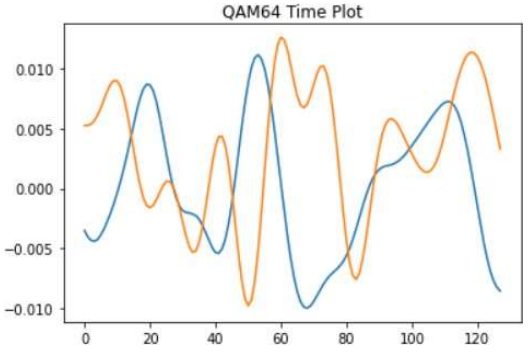





QAM16 Time Plot
0.010
0.005
0.000
−0.005
−0.010
0
20
40
60
80
100
120


WBFM Time Plot
0.004
0.002
0.000
−0.002
−0.004
−0.006
0
20
40
60
80
100
120


GFSK Time Plot
0.008
0.006
0.004
0.002
0.000
−0.002
−0.004
−0.006
−0.008
0
20
40
60
80
100
120


QAM64 Time Plot
0.010
0.005
0.000
−0.005
−0.010
0
20
40
60
80
100
120

2 Create feature space

Function for encoding labels

```
[3]  def to_onehot(yy):
         yy1 = np.zeros([len(yy), max(yy)+1])
         yy1[np.arange(len(yy)),yy] = 1
         return yy1
```

Function for plotting confusion matrices

```
[4]  def plot_confusion_matrix(cm, title='Confusion matrix', labels=[]):
         plt.imshow(cm, interpolation='nearest')
         plt.title(title)
         plt.colorbar()
         tick_marks = np.arange(len(labels))
         plt.xticks(tick_marks, labels, rotation=45)
         plt.yticks(tick_marks, labels)
         plt.tight_layout()
         plt.ylabel('True label')
         plt.xlabel('Predicted label')
```

(2) Create feature space

Getting derivatives for dataset

```
[ ]  X_driv=np.diff(np.array(X))
     X_driv=np.concatenate((np.zeros((1200000,2,1)), X_driv), axis=2)
```

Getting integrals for dataset

```
[ ]  import numpy as np
     from scipy import integrate

     S=integrate.cumtrapz(np.array(X), initial=0)
```

3 Supervised learning step

3.1 Train/test split

Raw data (train/test split)

```
[5]  np.random.seed(2016)
     n_examples = X.shape[0]
     n_train = n_examples * 0.7
     train_idx = np.random.choice(range(0,n_examples), size= int(n_train), replace=False)
     test_idx = list(set(range(0,n_examples))-set(train_idx))
     X_train = X[train_idx]
     X_test =  X[test_idx]

     Y_train = to_onehot(list(map(lambda x: mods.index(lbl[x][0]), train_idx)))
     Y_test = to_onehot(list(map(lambda x: mods.index(lbl[x][0]), test_idx)))
```

First derivative of data (train/test split)

```
[ ]  np.random.seed(2016)
     n_train_driv = X_driv.shape[0] * 0.7
     train_idx_driv = np.random.choice(range(0,X_driv.shape[0]), size= int(n_train_driv), replace=False)
     test_idx_driv = list(set(range(0,X_driv.shape[0]))-set(train_idx_driv))
     X_train_driv = X_driv[train_idx_driv]
     X_test_driv =  X_driv[test_idx_driv]

     Y_train_driv = to_onehot(list(map(lambda x: mods.index(lbl[x][0]), train_idx_driv)))
     Y_test_driv = to_onehot(list(map(lambda x: mods.index(lbl[x][0]), test_idx_driv)))
```

Integrals of data (train/test split)

```
[ ]  np.random.seed(2016)
     n_train_int = S.shape[0] * 0.7
     train_idx_int = np.random.choice(range(0,S.shape[0]), size= int(n_train_int), replace=False)
     test_idx_int = list(set(range(0,S.shape[0]))-set(train_idx_int))
     X_train_int = S[train_idx_int]
     X_test_int =  S[test_idx_int]

     Y_train_int = to_onehot(list(map(lambda x: mods.index(lbl[x][0]), train_idx_int)))
     Y_test_int = to_onehot(list(map(lambda x: mods.index(lbl[x][0]), test_idx_int)))
```

Combination for raw, derivatives and integrals of data (train/test split)

```
[ ]  combined_Data=np.zeros((1200000,6,128))
     for (i) in range(len(combined_Data)):
         combined_Data[i]=np.vstack((X[i],S[i]))

     np.random.seed(2016)
     n_examples_all = combined_Data.shape[0]
     n_train_all = n_examples_all * 0.7
     train_idx_all = np.random.choice(range(0,n_examples_all), size= int(n_train_all), replace=False)
     test_idx_all = list(set(range(0,n_examples_all))-set(train_idx_all))
     X_train_all = combined_Data[train_idx_all]
     X_test_all =  combined_Data[test_idx_all]

     Y_train_all = to_onehot(list(map(lambda x: mods.index(lbl[x][0]), train_idx_all)))
     Y_test_all = to_onehot(list(map(lambda x: mods.index(lbl[x][0]), test_idx_all)))
```

3.2 CNN model

3.2.1 Model design

CNN model for raw data

```
[ ]  import keras.models as models

     dr = 0.1
     model = keras.models.Sequential()
     model.add(Reshape(list(X_train.shape[1:])+[1], input_shape=list(X_train.shape[1:])))
     model.add(ZeroPadding2D((0, 2)))
     model.add(Conv2D(64, (1, 3),padding='valid', activation="relu", name="conv1",kernel_initializer='glorot_uniform',data_format="channels_last"))
     model.add(Dropout(dr))
     model.add(ZeroPadding2D((0, 2)))
     model.add(Conv2D(16,(2, 3), padding='valid', activation="relu", name="conv2", kernel_initializer='glorot_uniform',data_format="channels_last"))
     model.add(Dropout(dr))
     model.add(Flatten())
     model.add(Dense(128, activation='relu', kernel_initializer='he_normal', name="dense1"))
     model.add(Dropout(dr))
     model.add(Dense( 10, kernel_initializer='he_normal', name="dense2" ))
     model.add(Activation('softmax'))
     model.add(Reshape([10]))
     model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
     model.summary()
```

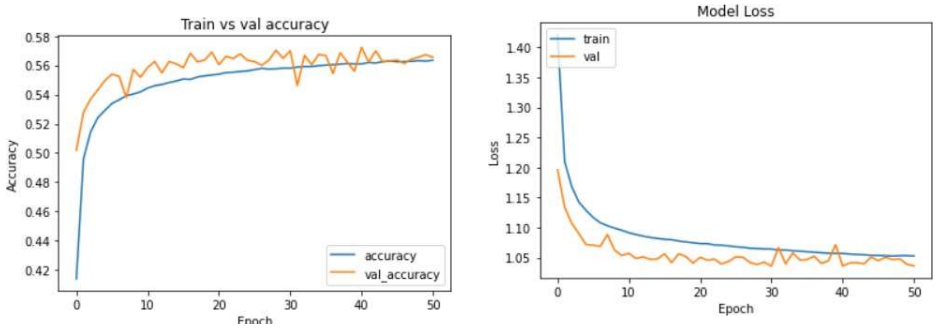
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 reshape (Reshape)           (None, 2, 128, 1)         0

 zero_padding2d (ZeroPadding  (None, 2, 132, 1)        0
 2D)

 conv1 (Conv2D)              (None, 2, 130, 64)        256

 dropout (Dropout)           (None, 2, 130, 64)        0

 zero_padding2d_1 (ZeroPaddi  (None, 2, 134, 64)       0
 ng2D)

 conv2 (Conv2D)              (None, 1, 132, 16)        6160

 dropout_1 (Dropout)         (None, 1, 132, 16)        0

 flatten (Flatten)           (None, 2112)              0

 dense1 (Dense)              (None, 128)               270464

 dropout_2 (Dropout)         (None, 128)               0

 dense2 (Dense)              (None, 10)                1290

 activation (Activation)     (None, 10)                0

 reshape_1 (Reshape)         (None, 10)                0

=================================================================
Total params: 278,170
Trainable params: 278,170
Non-trainable params: 0
```

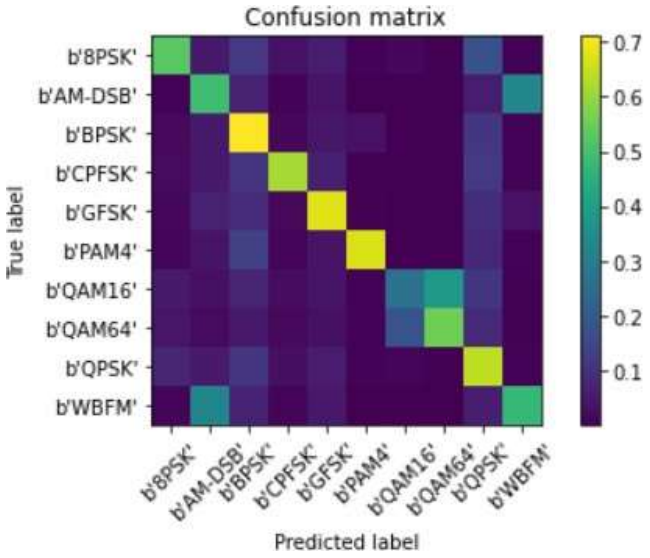
3.2.2 Training/testing results

```
[ ]  score = model.evaluate(X_test, Y_test)
     print(model.metrics_names)
     print (score)
```

```
11250/11250 [==============================] - 45s 4ms/step - loss: 1.0399 - accuracy: 0.5635
['loss', 'accuracy']
[1.039883017539978, 0.5635361075401306]
```

3.2.3 Training/testing visuals





Model design

CNN model for derivatives of data

```
import keras.models as models

dr = 0.1
model1 = keras.models.Sequential()
model1.add(Reshape(list(X_train_driv.shape[1:])+[1], input_shape=list(X_train_driv.shape[1:])))
model1.add(ZeroPadding2D((0, 2)))
model1.add(Conv2D(64, (1, 3),padding='valid', activation="relu", name="conv1",kernel_initializer='glorot_uniform',data_format="channels_last"))
model1.add(Dropout(dr))
model1.add(ZeroPadding2D((0, 2)))
model1.add(Conv2D(16,(2, 3), padding='valid', activation="relu", name="conv2", kernel_initializer='glorot_uniform',data_format="channels_last"))
model1.add(Dropout(dr))
model1.add(Flatten())
model1.add(Dense(128, activation='relu', kernel_initializer='he_normal', name="dense1"))
model1.add(Dropout(dr))
model1.add(Dense( 10, kernel_initializer='he_normal', name="dense2" ))
model1.add(Activation('softmax'))
model1.add(Reshape([10]))
model1.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model1.summary()
```

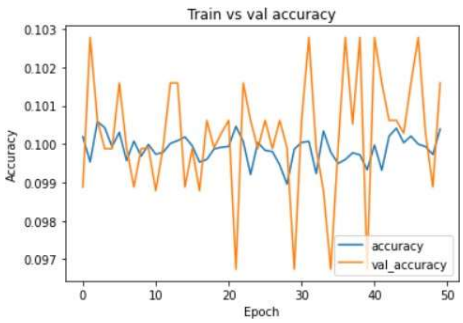
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 reshape (Reshape)           (None, 2, 128, 1)         0

 zero_padding2d (ZeroPadding  (None, 2, 132, 1)        0
 2D)

 conv1 (Conv2D)              (None, 2, 130, 64)        256

 dropout (Dropout)           (None, 2, 130, 64)        0

 zero_padding2d_1 (ZeroPaddi  (None, 2, 134, 64)       0
 ng2D)

 conv2 (Conv2D)              (None, 1, 132, 16)        6160

 dropout_1 (Dropout)         (None, 1, 132, 16)        0

 flatten (Flatten)           (None, 2112)              0

 dense1 (Dense)              (None, 128)               270464

 dropout_2 (Dropout)         (None, 128)               0

 dense2 (Dense)              (None, 10)                1290

 activation (Activation)     (None, 10)                0

 reshape_1 (Reshape)         (None, 10)                0

=================================================================
Total params: 278,170
Trainable params: 278,170
Non-trainable params: 0
```

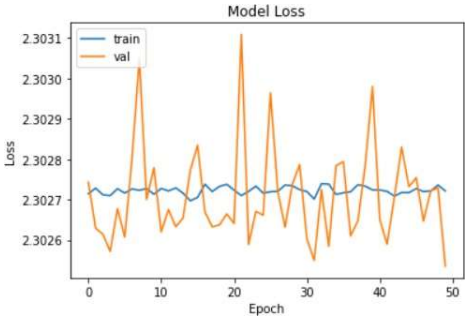
Training/testing results

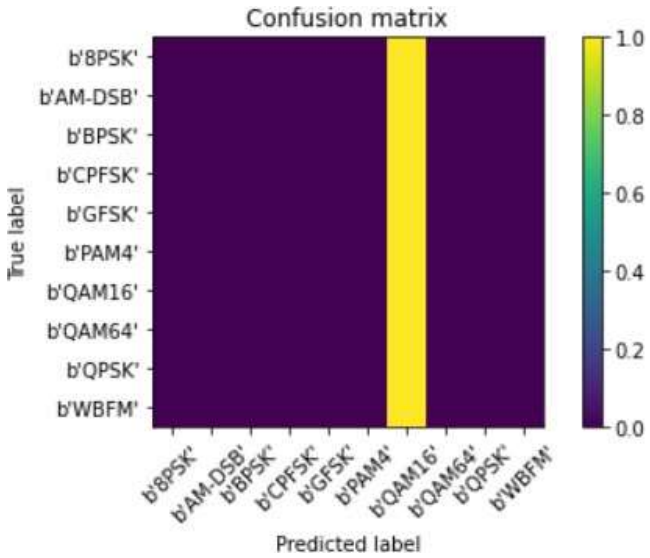
```
[ ]  score1 = model1.evaluate(X_test_driv, Y_test_driv)
     print(model1.metrics_names)
     print (score1)
```

```
11250/11250 [==============================] - 30s 3ms/step - loss: 2.3027 - accuracy: 0.0997
['loss', 'accuracy']
[2.302719831466675, 0.09968888759613037]
```

Training/testing visuals







Model design

CNN model for integrals of data

```
[ ] import keras.models as models

    dr = 0.1
    model2 = keras.models.Sequential()
    model2.add(Reshape(list(X_train_int.shape[1:])+[1], input_shape=list(X_train_int.shape[1:])))
    model2.add(ZeroPadding2D((0, 2)))
    model2.add(Conv2D(64, (1, 3),padding='valid', activation="relu", name="conv1",kernel_initializer='glorot_uniform',data_format="channels_last"))
    model2.add(Dropout(dr))
    model2.add(ZeroPadding2D((0, 2)))
    model2.add(Conv2D(16,(2, 3), padding='valid', activation="relu", name="conv2", kernel_initializer='glorot_uniform',data_format="channels_last"))
    model2.add(Dropout(dr))
    model2.add(Flatten())
    model2.add(Dense(128, activation='relu', kernel_initializer='he_normal', name="dense1"))
    model2.add(Dropout(dr))
    model2.add(Dense( 10, kernel_initializer='he_normal', name="dense2" ))
    model2.add(Activation('softmax'))
    model2.add(Reshape([10]))
    model2.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    model2.summary()
```

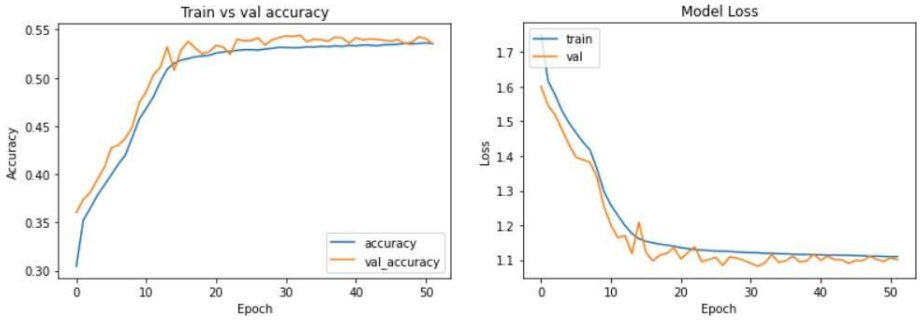
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 reshape (Reshape)           (None, 2, 128, 1)         0

 zero_padding2d (ZeroPadding  (None, 2, 132, 1)        0
 2D)

 conv1 (Conv2D)              (None, 2, 130, 64)        256

 dropout (Dropout)           (None, 2, 130, 64)        0

 zero_padding2d_1 (ZeroPaddi  (None, 2, 134, 64)       0
 ng2D)

 conv2 (Conv2D)              (None, 1, 132, 16)        6160

 dropout_1 (Dropout)         (None, 1, 132, 16)        0

 flatten (Flatten)           (None, 2112)              0

 dense1 (Dense)              (None, 128)               270464

 dropout_2 (Dropout)         (None, 128)               0

 dense2 (Dense)              (None, 10)                1290

 activation (Activation)     (None, 10)                0

 reshape_1 (Reshape)         (None, 10)                0

=================================================================
Total params: 278,170
Trainable params: 278,170
Non-trainable params: 0
```

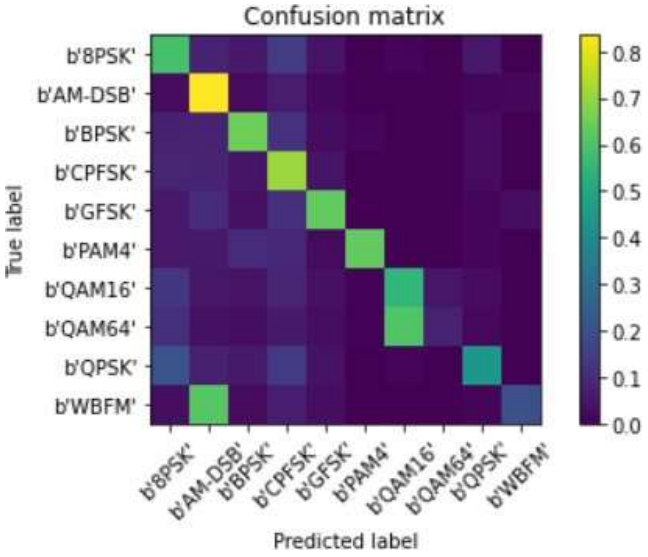
Training/testing results

```
[ ]  score2 = model2.evaluate(X_test_int, Y_test_int)
     print(model2.metrics_names)
     print (score2)
```

```
11250/11250 [==============================] - 29s 3ms/step - loss: 1.1058 - accuracy: 0.5338
['loss', 'accuracy']
[1.1057634353637695, 0.5337861180305481]
```

Training/testing visuals





3.3 RNN model

3.3.1 Model design

RNN model for raw data

```
[ ]  from keras.models import Sequential
     from keras.layers import Dense, SimpleRNN
     from sklearn.preprocessing import MinMaxScaler

     model3 = Sequential()
     model3.add(SimpleRNN(64, input_shape=list(X_train.shape[1:]), activation='relu'))
     model3.add(Dense(128, activation='relu', kernel_initializer='he_normal'))
     model3.add(Dense(units=10, activation='softmax'))
     model3.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
     model3.summary()
```

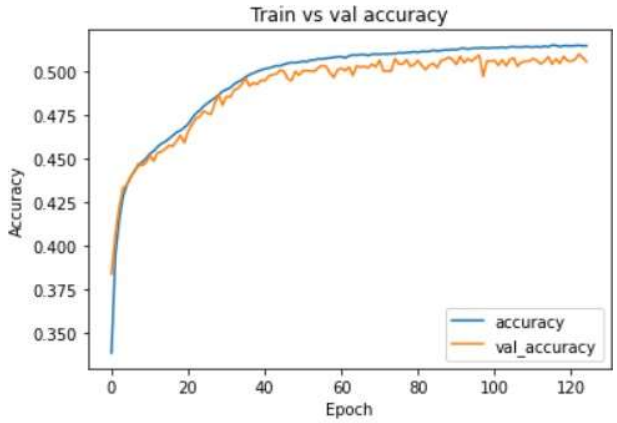
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 64)                12352

 dense (Dense)               (None, 128)               8320

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 21,962
Trainable params: 21,962
Non-trainable params: 0
_________________________________________________________________
```

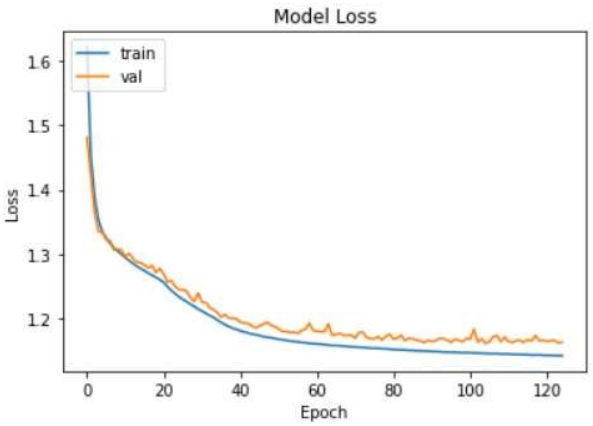
3.3.2 Training/testing results

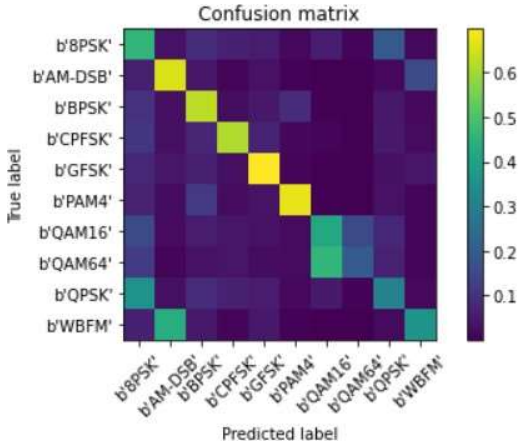
```
score3 = model3.evaluate(X_test, Y_test)
print(model3.metrics_names)
print (score3)
```

```
11250/11250 [==============================] - 29s 3ms/step - loss: 1.1706 - accuracy: 0.5035
['loss', 'accuracy']
[1.1705892086029053, 0.5034944415092468]
```

3.3.3 Training/testing visuals







Model design

RNN model for integrals of data

```
[ ]  from keras.models import Sequential
     from keras.layers import Dense, SimpleRNN
     from sklearn.preprocessing import MinMaxScaler

     model5 = Sequential()
     model5.add(SimpleRNN(64, input_shape=list(X_train_int.shape[1:]), activation='relu'))
     model5.add(Dense(128, activation='relu', kernel_initializer='he_normal'))
     model5.add(Dense(units=10, activation='softmax'))
     model5.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
     model5.summary()
```

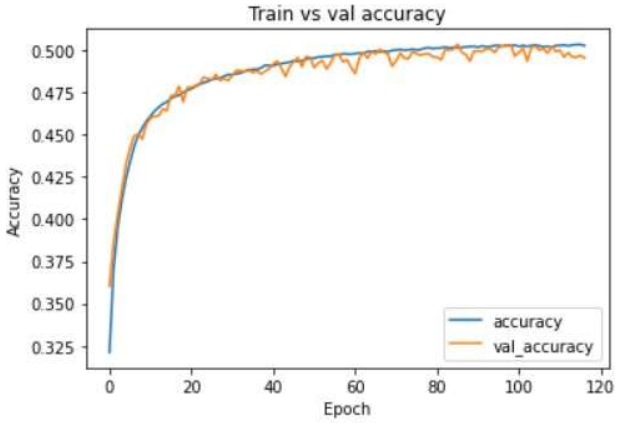
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 64)                12352

 dense (Dense)               (None, 128)               8320

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 21,962
Trainable params: 21,962
Non-trainable params: 0
_________________________________________________________________
```

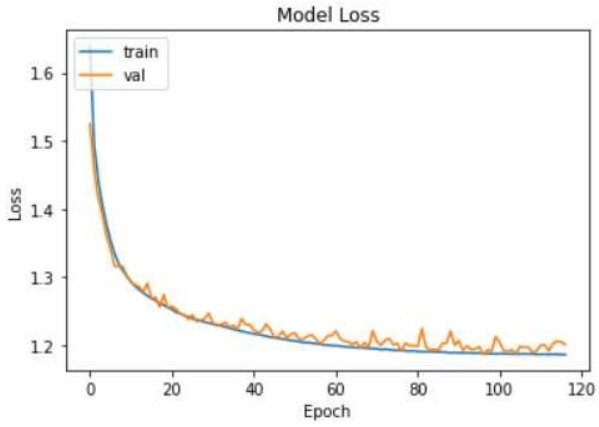
Training/testing results

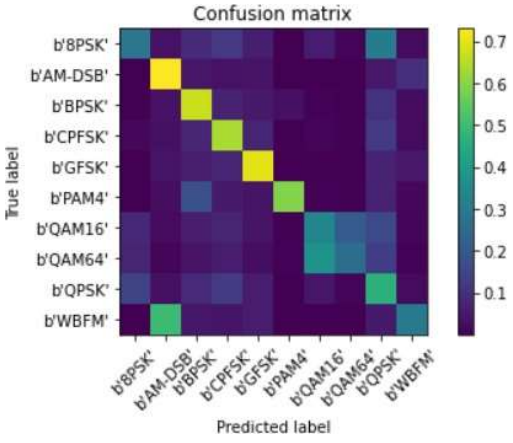
```
score5 = model5.evaluate(X_test_int, Y_test_int)
print(model5.metrics_names)
print (score5)
```

```
11250/11250 [==============================] - 26s 2ms/step - loss: 1.1971 - accuracy: 0.4984
['loss', 'accuracy']
[1.197102427482605, 0.49838611483573914]
```

Training/testing visuals







Model design

RNN model for derivatives of data

```
[ ] from keras.models import Sequential
    from keras.layers import Dense, SimpleRNN
    from sklearn.preprocessing import MinMaxScaler

    model7 = Sequential()
    model7.add(SimpleRNN(64, input_shape=list(X_train_driv.shape[1:]), activation='relu'))
    model7.add(Dense(128, activation='relu', kernel_initializer='he_normal'))
    model7.add(Dense(units=10, activation='softmax'))
    model7.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    model7.summary()
```

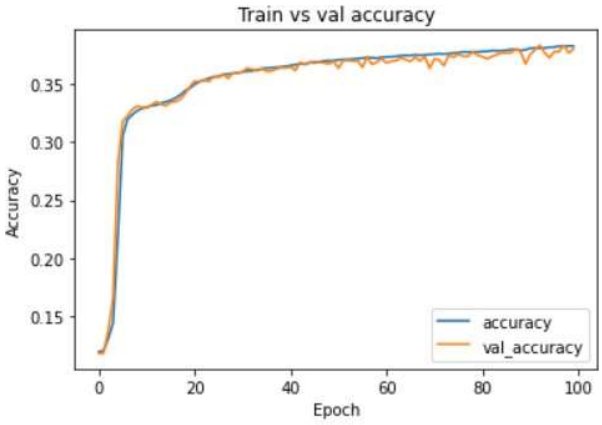
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 64)                12352

 dense (Dense)               (None, 128)               8320

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 21,962
Trainable params: 21,962
Non-trainable params: 0
_________________________________________________________________
```

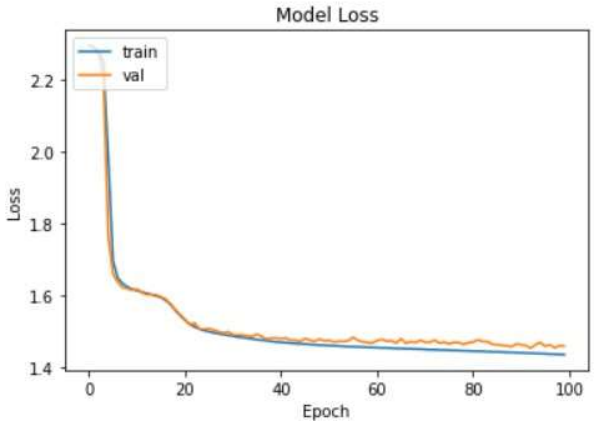
Training/testing results

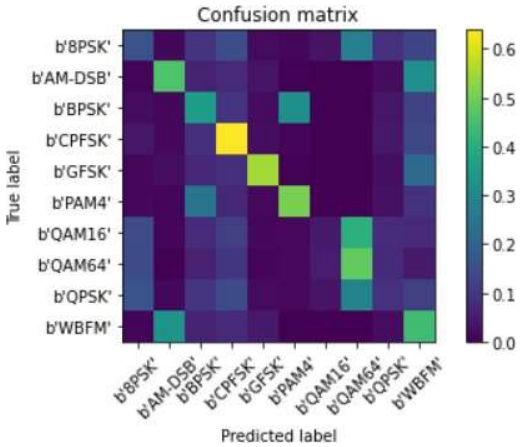
```
score7 = model7.evaluate(X_test_driv, Y_test_driv)
print(model7.metrics_names)
print (score7)
```

```
11250/11250 [==============================] - 23s 2ms/step - loss: 1.4691 - accuracy: 0.3751
['loss', 'accuracy']
[1.4691197872161865, 0.37514999508857727]
```

Training/testing visuals







3.4 LSTM model

3.4.1 Model design

LSTM model for raw data

```
[ ] from keras.layers import Dense, Dropout, Embedding, LSTM, GlobalMaxPooling1D

    model4 = Sequential()
    model4.add(LSTM(64, dropout = 0.3, return_sequences = True, recurrent_dropout = 0.3, input_shape = list
    model4.add(Flatten())
    model4.add(Dense(128, activation='relu', kernel_initializer='he_normal'))
    model4.add(Dense(10, activation = 'softmax'))
    model4.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    model4.summary()
```

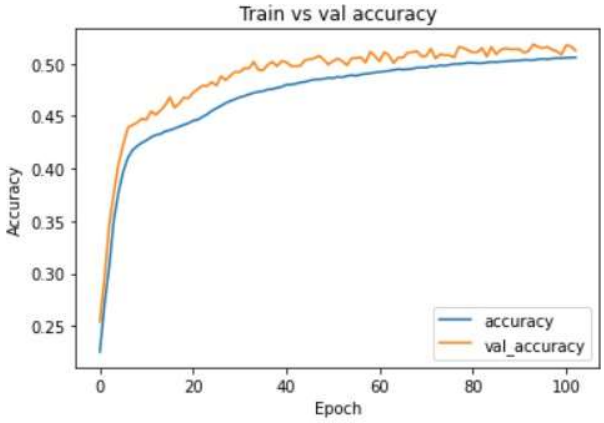
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 2, 64)             49408

 flatten (Flatten)           (None, 128)               0

 dense_2 (Dense)             (None, 128)               16512

 dense_3 (Dense)             (None, 10)                1290

=================================================================
Total params: 67,210
Trainable params: 67,210
Non-trainable params: 0
_________________________________________________________________
```

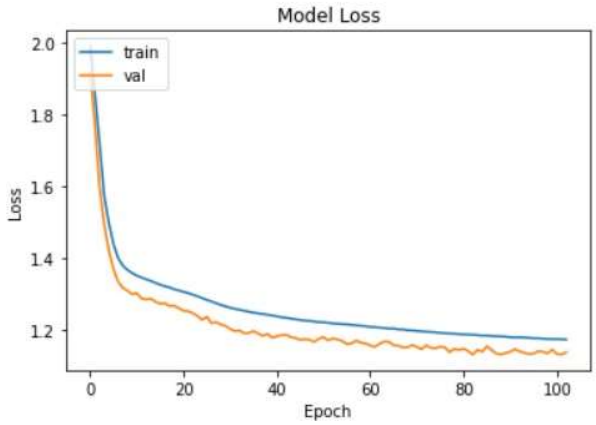
3.4.2 Training/testing results

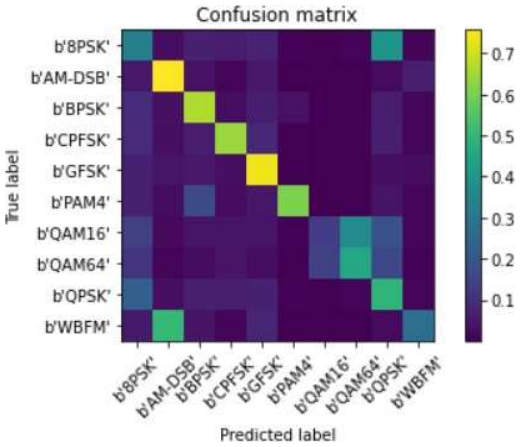
```
[ ]  score4 = model4.evaluate(X_test, Y_test)
     print(model4.metrics_names)
     print (score4)
```

```
11250/11250 [==============================] - 35s 3ms/step - loss: 1.1426 - accuracy: 0.5109
['loss', 'accuracy']
[1.1425615549087524, 0.5108555555343628]
```

3.4.3 Training/testing visuals







Model design

LSTM model for integrals of data

```
[ ]  from keras.layers import Dense, Dropout, Embedding, LSTM, GlobalMaxPooling1D

     model6 = Sequential()
     model6.add(LSTM(64, dropout = 0.3, return_sequences = True, recurrent_dropout = 0.3, input_shape = list
     model6.add(Flatten())
     model6.add(Dense(128, activation='relu', kernel_initializer='he_normal'))
     model6.add(Dense(10, activation = 'softmax'))
     model6.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
     model6.summary()
```

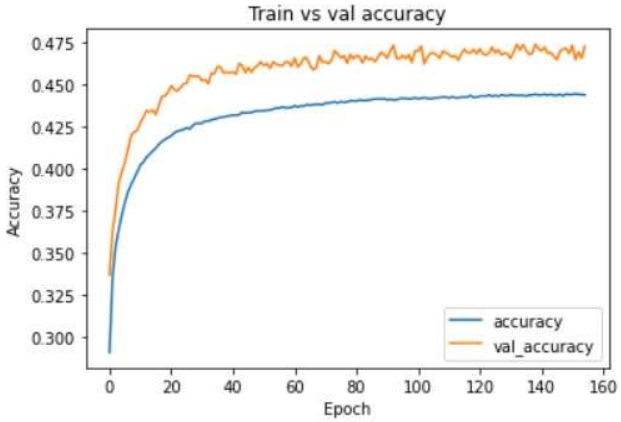
```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_1 (LSTM)               (None, 2, 64)             49408

 flatten_1 (Flatten)         (None, 128)               0

 dense_4 (Dense)             (None, 128)               16512

 dense_5 (Dense)             (None, 10)                1290

=================================================================
Total params: 67,210
Trainable params: 67,210
Non-trainable params: 0
_________________________________________________________________
```

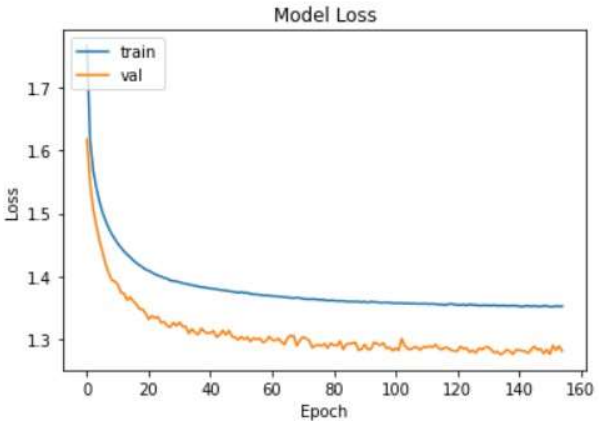
Training/testing results

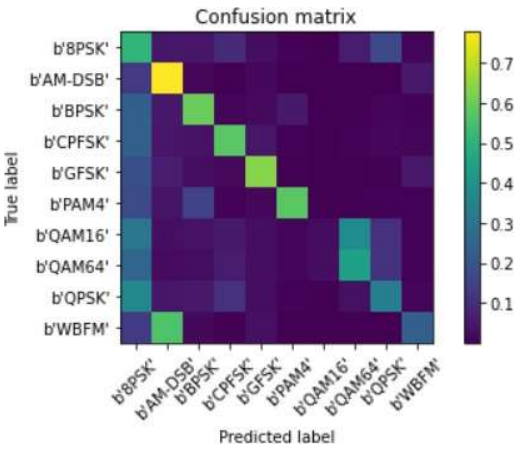
```
[ ]  score6 = model6.evaluate(X_test_int, Y_test_int)
     print(model6.metrics_names)
     print (score6)
```

```
11250/11250 [==============================] - 33s 3ms/step - loss: 1.2822 - accuracy: 0.4721
['loss', 'accuracy']
[1.2822120189666748, 0.4721166789531708]
```

Training/testing visuals







Model design

LSTM model for derivatives of data

```
[ ] from keras.layers import Dense, Dropout, Embedding, LSTM, GlobalMaxPooling1D

    model8 = Sequential()
    model8.add(LSTM(64, dropout = 0.3, return_sequences = True, recurrent_dropout = 0.3, input_shape = list(
    model8.add(Flatten())
    model8.add(Dense(128, activation='relu', kernel_initializer='he_normal'))
    model8.add(Dense(10, activation = 'softmax'))
    model8.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    model8.summary()
```

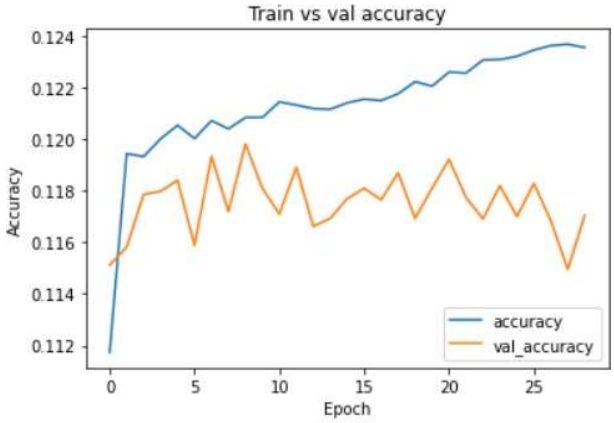
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 2, 64)             49408

 flatten (Flatten)           (None, 128)               0

 dense_2 (Dense)             (None, 128)               16512

 dense_3 (Dense)             (None, 10)                1290

=================================================================
Total params: 67,210
Trainable params: 67,210
Non-trainable params: 0
_________________________________________________________________
```

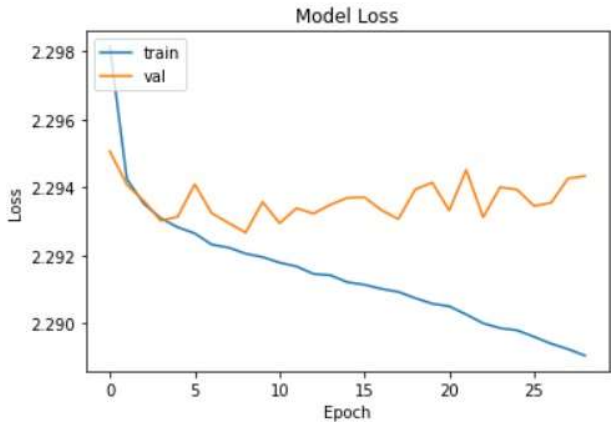
Training/testing results

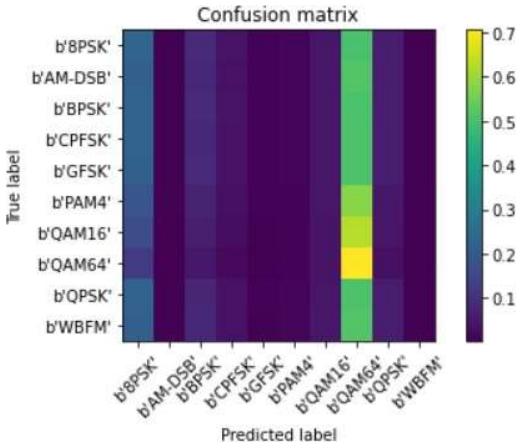
```
[ ]  score8 = model8.evaluate(X_test_driv, Y_test_driv)
     print(model8.metrics_names)
     print (score8)
```

```
11250/11250 [==============================] - 28s 2ms/step - loss: 2.2946 - accuracy: 0.1201
['loss', 'accuracy']
[2.294595241546631, 0.12007778137922287]
```

Training/testing visuals





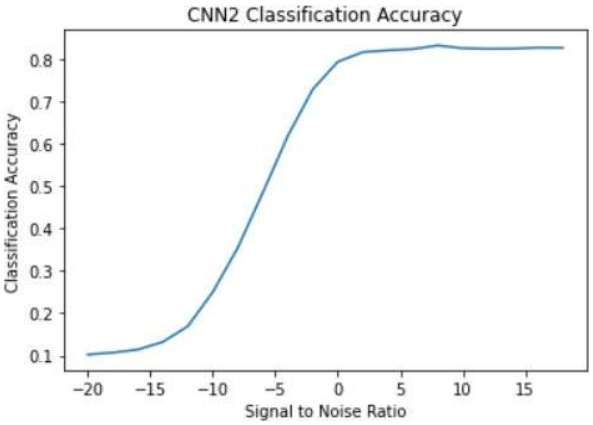


Note: Hyper parameter tuning on learning rate has been made and it was found that the best results are with ADAM's default learning rate 0.001.

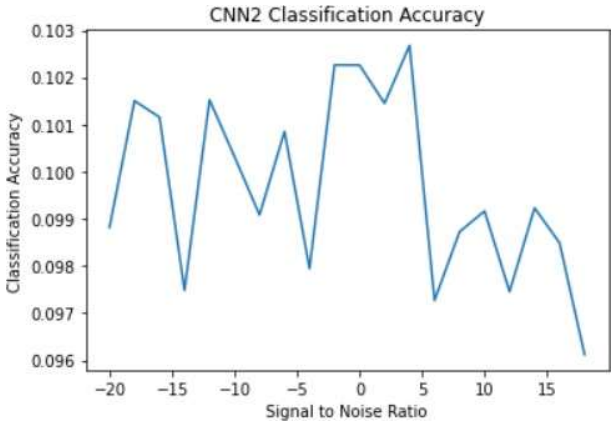
4 Big picture

4.1 Plot of accuracy against SNR

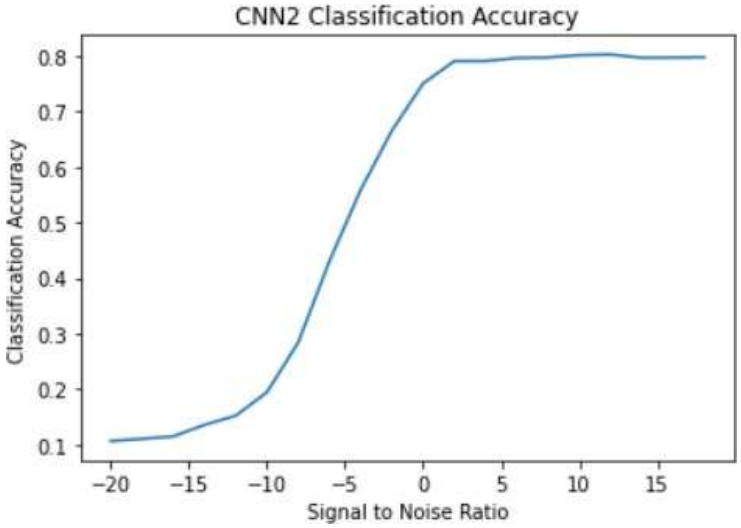
CNN with raw data



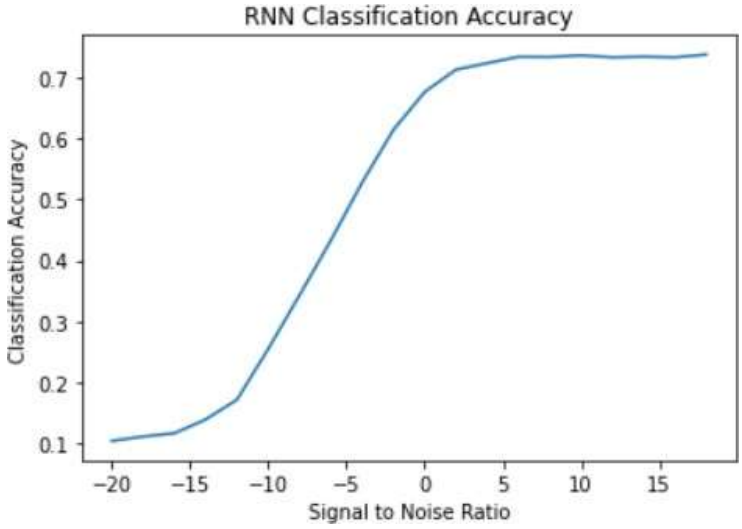
CNN with derivatives



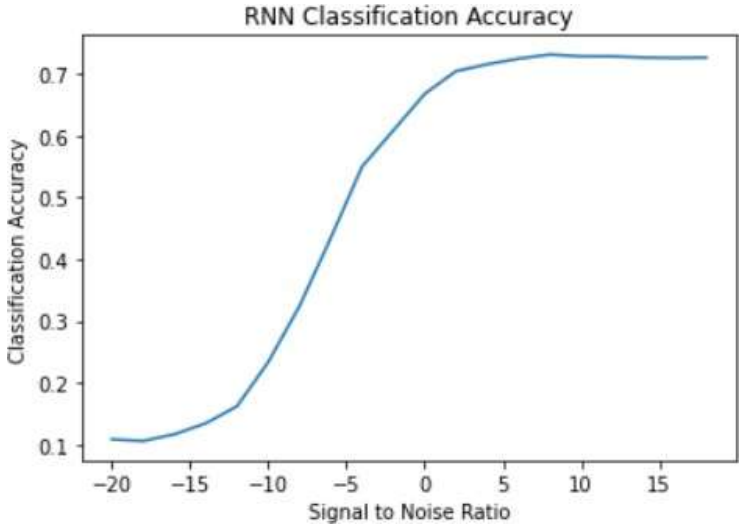
CNN with integrals



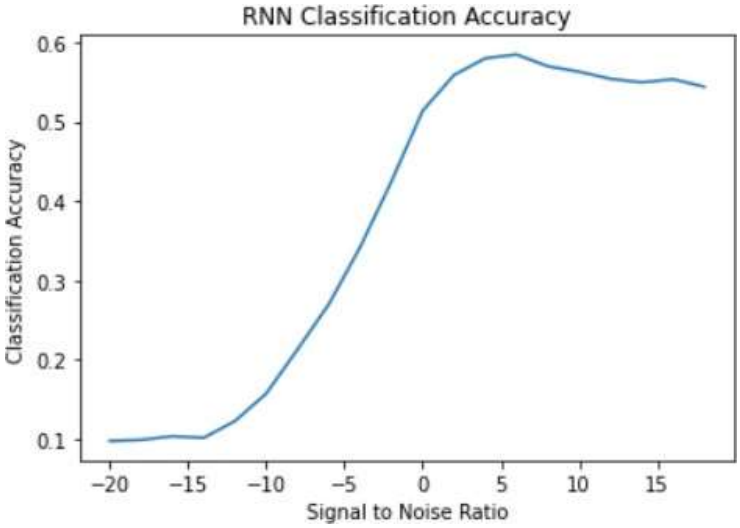
RNN with raw data



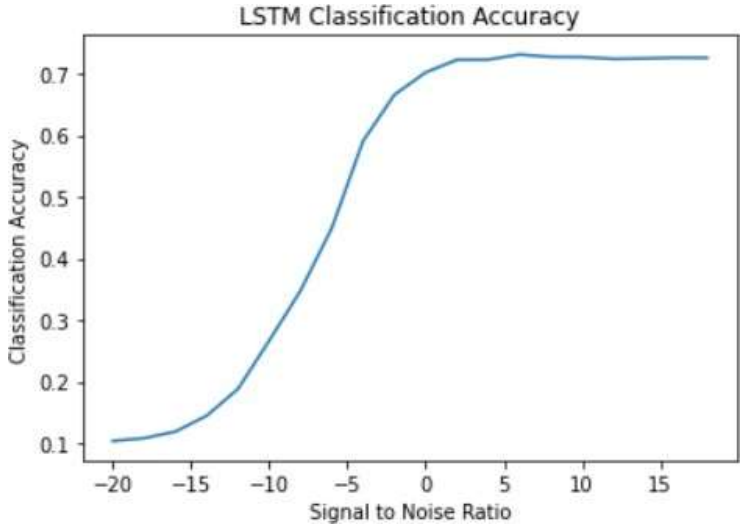
RNN with integrals



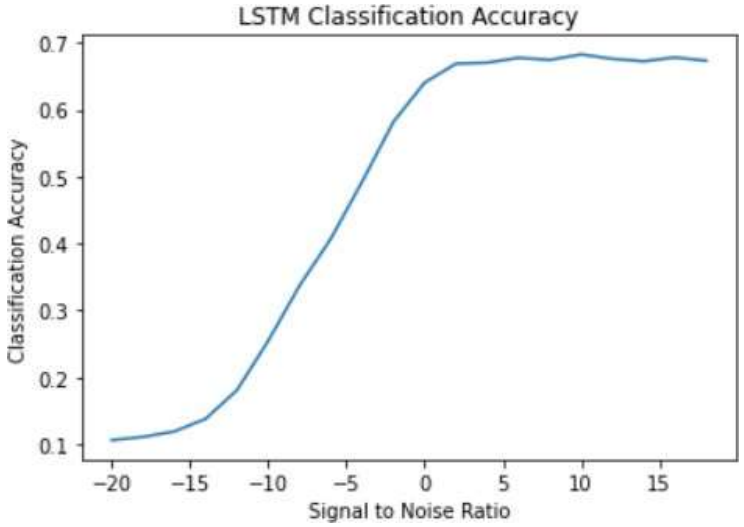
RNN with derivatives



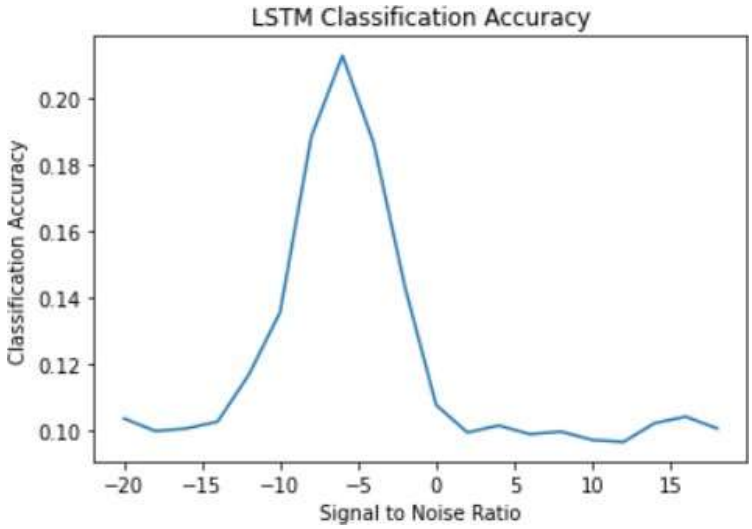
LSTM with raw data



LSTM with integrals



LSTM with derivatives



4.2 Report average overall acc

CNN with raw data

```
Overall Accuracy:  0.1023517721099702
Overall Accuracy:  0.10685787623458512
Overall Accuracy:  0.11403557686945855
Overall Accuracy:  0.13204176204567025
Overall Accuracy:  0.16911154165513423
Overall Accuracy:  0.24984724768094208
Overall Accuracy:  0.35441778170960964
Overall Accuracy:  0.4844501622105381
Overall Accuracy:  0.618258411396264
Overall Accuracy:  0.7283039955294776
Overall Accuracy:  0.7930861723446894
Overall Accuracy:  0.8157428919128222
Overall Accuracy:  0.8200947449597885
Overall Accuracy:  0.8230083534226608
Overall Accuracy:  0.8315935591338145
Overall Accuracy:  0.8249958451055344
Overall Accuracy:  0.823549032688459
Overall Accuracy:  0.823979167820932
Overall Accuracy:  0.8264931598264932
Overall Accuracy:  0.8261640798226164
```

CNN with derivatives

```
Overall Accuracy:  0.09881859335320746
Overall Accuracy:  0.10150103230846493
Overall Accuracy:  0.10115429654826298
Overall Accuracy:  0.0974819943051756
Overall Accuracy:  0.1015222806531968
Overall Accuracy:  0.10031661389768372
Overall Accuracy:  0.09908216299900476
Overall Accuracy:  0.10085020695827274
Overall Accuracy:  0.09794357297267335
Overall Accuracy:  0.10226320201173512
Overall Accuracy:  0.10226007570696949
Overall Accuracy:  0.10144927536231885
Overall Accuracy:  0.10267709595681393
Overall Accuracy:  0.09726972024443573
Overall Accuracy:  0.09872293170460855
Overall Accuracy:  0.09916348124757632
Overall Accuracy:  0.09745385812764065
Overall Accuracy:  0.09922987423125935
Overall Accuracy:  0.09848737626515404
Overall Accuracy:  0.09611973392461197
```

CNN with integrals

```
Overall Accuracy:  0.10687865739207243
Overall Accuracy:  0.11054070643379275
Overall Accuracy:  0.11509507611665644
Overall Accuracy:  0.1358383116520574
Overall Accuracy:  0.15222806531967895
Overall Accuracy:  0.1944675887352108
Overall Accuracy:  0.28386597368130045
Overall Accuracy:  0.43041727262557333
Overall Accuracy:  0.5586719139737265
Overall Accuracy:  0.6654372729812796
Overall Accuracy:  0.7502783344466711
Overall Accuracy:  0.7906848102666224
Overall Accuracy:  0.7906797400022034
Overall Accuracy:  0.7961540617816898
Overall Accuracy:  0.797223764575236
Overall Accuracy:  0.8013406459475929
Overall Accuracy:  0.8024238381142984
Overall Accuracy:  0.7966092304282786
Overall Accuracy:  0.7970192414636859
Overall Accuracy:  0.7975609756097561
```

RNN with raw data

```
Overall Accuracy:  0.10384233189797946
Overall Accuracy:  0.11104291055186652
Overall Accuracy:  0.11643339095522222
Overall Accuracy:  0.13907654513985818
Overall Accuracy:  0.17149183504013285
Overall Accuracy:  0.2549019607843137
Overall Accuracy:  0.342917173504368
Overall Accuracy:  0.43276652869448484
Overall Accuracy:  0.527742364613935
Overall Accuracy:  0.6139703827884884
Overall Accuracy:  0.6762413716321531
Overall Accuracy:  0.7128000885053657
Overall Accuracy:  0.7233667511292278
Overall Accuracy:  0.7338678028816505
Overall Accuracy:  0.7335369239311493
Overall Accuracy:  0.7361365021328459
Overall Accuracy:  0.7329330664887703
Overall Accuracy:  0.7341126932240013
Overall Accuracy:  0.7329551773996218
Overall Accuracy:  0.7375277161862528
```

RNN with integrals

```
Overall Accuracy:  0.10814839350778403
Overall Accuracy:  0.10557446571061883
Overall Accuracy:  0.11632186471867507
Overall Accuracy:  0.13410753168443973
Overall Accuracy:  0.1617492388596734
Overall Accuracy:  0.23362772871188137
Overall Accuracy:  0.32433926794205464
Overall Accuracy:  0.4351717194317038
Overall Accuracy:  0.5504683775843912
Overall Accuracy:  0.6092763341715564
Overall Accuracy:  0.6683366733466933
Overall Accuracy:  0.7051665007191061
Overall Accuracy:  0.7163159634240388
Overall Accuracy:  0.7254022537422212
Overall Accuracy:  0.7320932815102721
Overall Accuracy:  0.7291562794305024
Overall Accuracy:  0.7290971758950411
Overall Accuracy:  0.72729791124162
Overall Accuracy:  0.7266155043932822
Overall Accuracy:  0.7272727272727273
```

RNN with derivatives

```
Overall Accuracy:  0.09754885723749586
Overall Accuracy:  0.09887841080296858
Overall Accuracy:  0.10344058439747951
Overall Accuracy:  0.10144603874713863
Overall Accuracy:  0.12261278715748686
Overall Accuracy:  0.1573071154807532
Overall Accuracy:  0.21303770872498065
Overall Accuracy:  0.26988477458328675
Overall Accuracy:  0.3417770633556898
Overall Accuracy:  0.4246437552388935
Overall Accuracy:  0.5138053885548876
Overall Accuracy:  0.5587454364420843
Overall Accuracy:  0.5798722044728435
Overall Accuracy:  0.5845713965352918
Overall Accuracy:  0.5697390338700722
Overall Accuracy:  0.5629051022104039
Overall Accuracy:  0.5539248387814099
Overall Accuracy:  0.5494487229209375
Overall Accuracy:  0.5533867200533867
Overall Accuracy:  0.5440133037694014
```

LSTM with raw data

```
Overall Accuracy:  0.10373192006183063
Overall Accuracy:  0.10819708721611518
Overall Accuracy:  0.11888696815925946
Overall Accuracy:  0.14465970632572162
Overall Accuracy:  0.18820924439523942
Overall Accuracy:  0.26667777592623454
Overall Accuracy:  0.3476722326661506
Overall Accuracy:  0.4493791251817877
Overall Accuracy:  0.5907100493320769
Overall Accuracy:  0.6656607991058955
Overall Accuracy:  0.7019594745045646
Overall Accuracy:  0.7224250470184755
Overall Accuracy:  0.7226506555029195
Overall Accuracy:  0.7308964511969501
Overall Accuracy:  0.7270405330372015
Overall Accuracy:  0.7267741399368456
Overall Accuracy:  0.723815877251501
Overall Accuracy:  0.7245276746634163
Overall Accuracy:  0.7257813368924481
Overall Accuracy:  0.7254988913525499
```

LSTM with integrals

```
Overall Accuracy:  0.10605056862095617
Overall Accuracy:  0.11076390826404776
Overall Accuracy:  0.11899849439580662
Overall Accuracy:  0.1379599129026855
Overall Accuracy:  0.18073623027954608
Overall Accuracy:  0.25456868299727825
Overall Accuracy:  0.3370009952449408
Overall Accuracy:  0.4070365812730731
Overall Accuracy:  0.49232304195998006
Overall Accuracy:  0.5817267393126572
Overall Accuracy:  0.6405032286795814
Overall Accuracy:  0.6687133532470406
Overall Accuracy:  0.6702104219455768
Overall Accuracy:  0.677468184111678
Overall Accuracy:  0.6743475846751804
Overall Accuracy:  0.6825106642291285
Overall Accuracy:  0.6759506337558372
Overall Accuracy:  0.672281012798493
Overall Accuracy:  0.677955733511289
Overall Accuracy:  0.6731707317073171
```

LSTM with derivatives

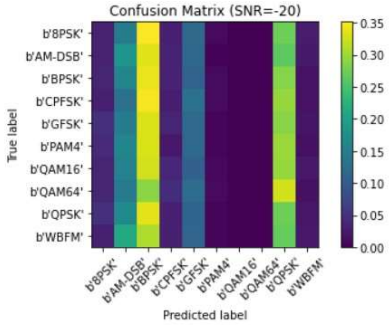
```
Overall Accuracy:  0.10384233189797946
Overall Accuracy:  0.10005022041180738
Overall Accuracy:  0.10081971783862154
Overall Accuracy:  0.10289766065546313
Overall Accuracy:  0.11702186548574592
Overall Accuracy:  0.1357551519191246
Overall Accuracy:  0.1885436249032401
Overall Accuracy:  0.21294328224633627
Overall Accuracy:  0.18651959425752454
Overall Accuracy:  0.14339200894104498
Overall Accuracy:  0.10788243152972612
Overall Accuracy:  0.09962385219603938
Overall Accuracy:  0.10174066321471852
Overall Accuracy:  0.09911980714245669
Overall Accuracy:  0.09988895058300944
Overall Accuracy:  0.0973907262755526
Overall Accuracy:  0.09678674672003558
Overall Accuracy:  0.10244334866197573
Overall Accuracy:  0.10438215993771549
Overall Accuracy:  0.10083148558758315
```

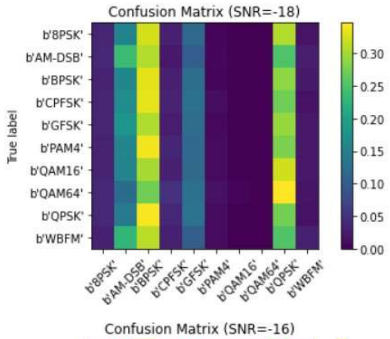
Note: These overall accuracies and confusion matrices are calculated for each SNR individually (-20 SNR to 18 SNR) with the following code.

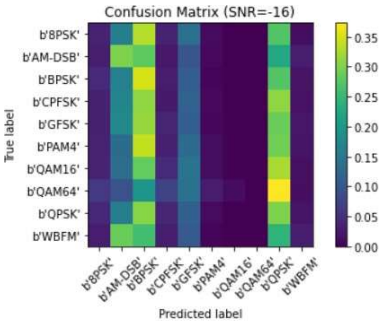
```
[ ]  acc8 = {}
     for snr in snrs:

         test_SNRs_driv = list(map(lambda x: lbl[x][1], test_idx_driv))
         test_X_i_driv = X_test_driv[np.where(np.array(test_SNRs_driv)==snr)]
         test_Y_i_driv = Y_test_driv[np.where(np.array(test_SNRs_driv)==snr)]

         test_Y_i_hat_driv = model8.predict(test_X_i_driv)
         conf8 = np.zeros([10,10])
         confnorm8 = np.zeros([10,10])
         for i in range(0,test_X_i_driv.shape[0]):
             j = list(test_Y_i_driv[i,:]).index(1)
             k = int(np.argmax(test_Y_i_hat_driv[i,:]))
             conf8[j,k] = conf8[j,k] + 1
         for i in range(0,10):
             confnorm8[i,:] = conf8[i,:] / np.sum(conf8[i,:])
         plt.figure()
         plot_confusion_matrix(confnorm8, labels=mods, title="Confusion Matrix (SNR=%d)"%(snr))

         cor8 = np.sum(np.diag(conf8))
         ncor8 = np.sum(conf8) - cor8
         print ("Overall Accuracy: ", cor8 / (cor8+ncor8))
         acc8[snr] = 1.0*cor8/(cor8+ncor8)
```

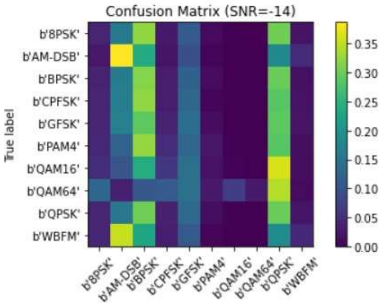
4.3 Show confusion matrices

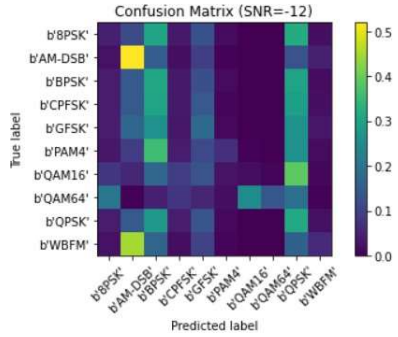
CNN with raw data

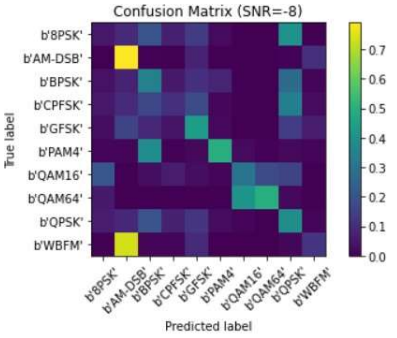


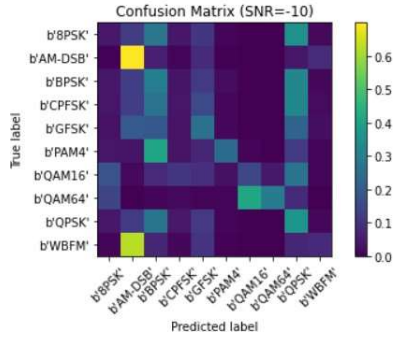


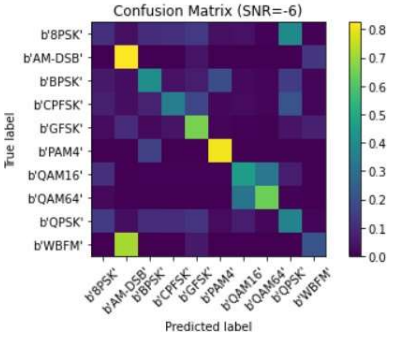


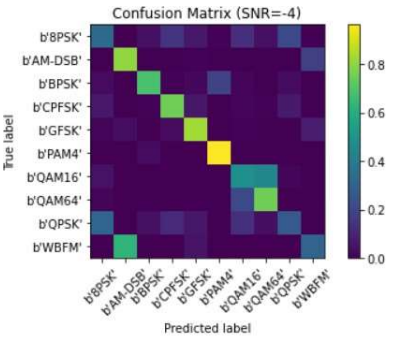


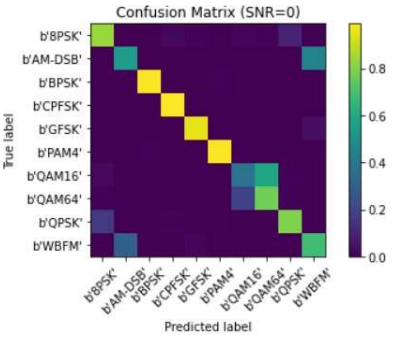

Confusion Matrix (SNR=-12)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.1
0.2
0.3
0.4
0.5

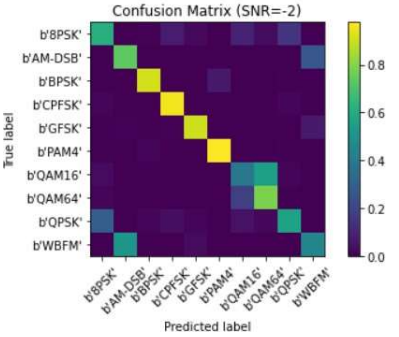

Confusion Matrix (SNR=-8)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

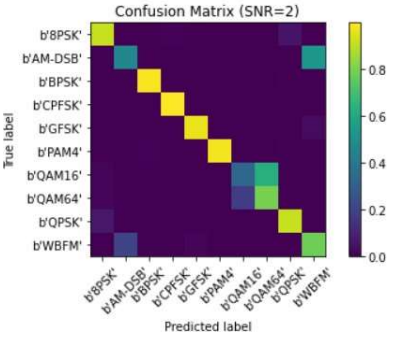

Confusion Matrix (SNR=-10)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.1
0.2
0.3
0.4
0.5
0.6

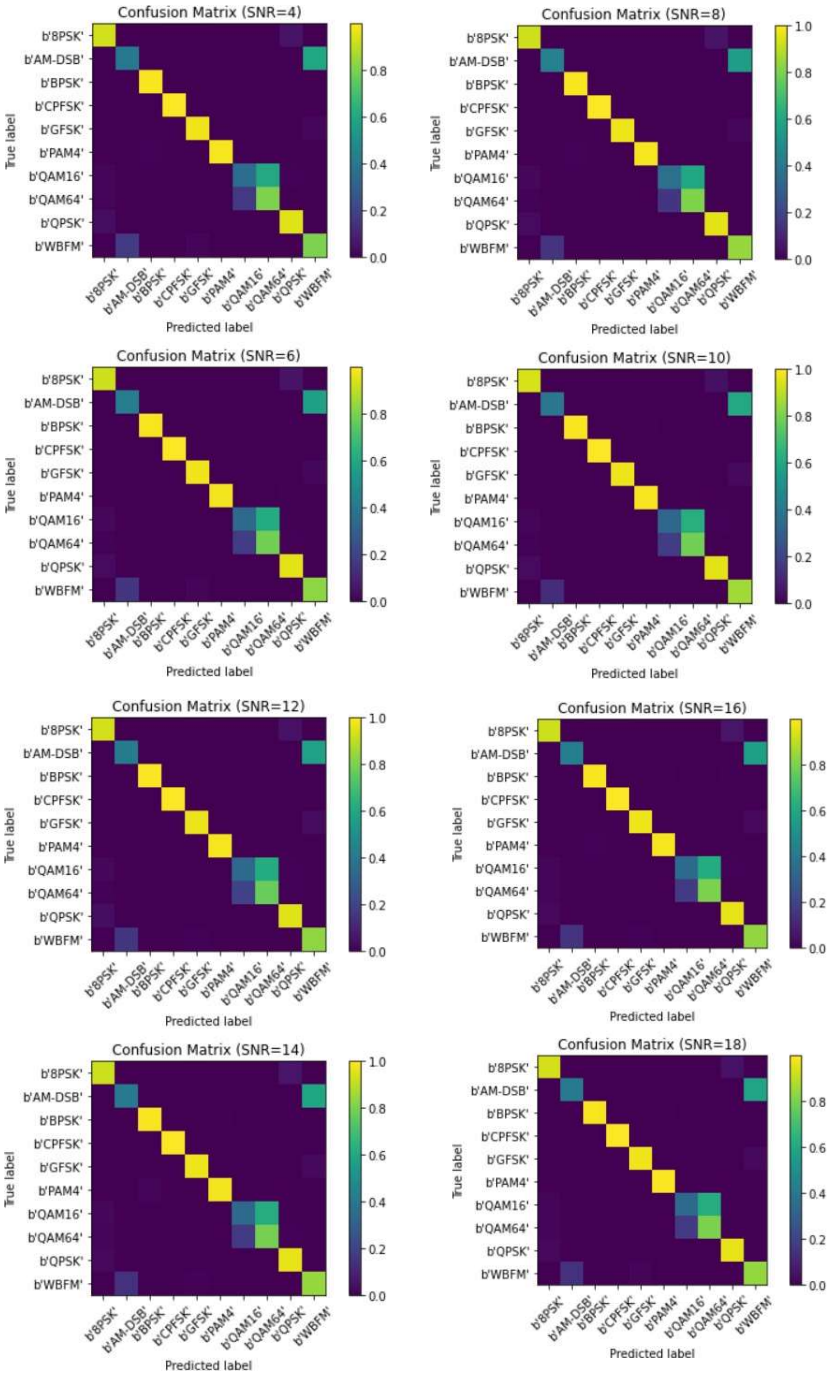

Confusion Matrix (SNR=-6)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


Confusion Matrix (SNR=-4)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.2
0.4
0.6
0.8

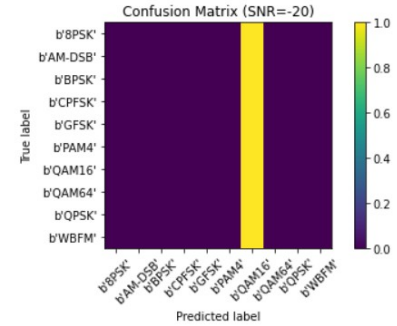

Confusion Matrix (SNR=0)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.2
0.4
0.6
0.8

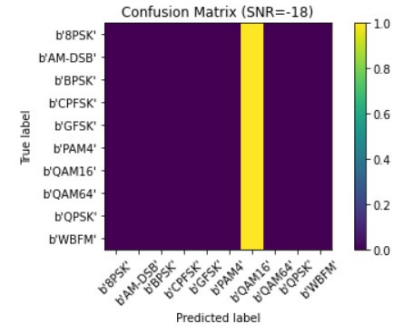

Confusion Matrix (SNR=-2)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.2
0.4
0.6
0.8

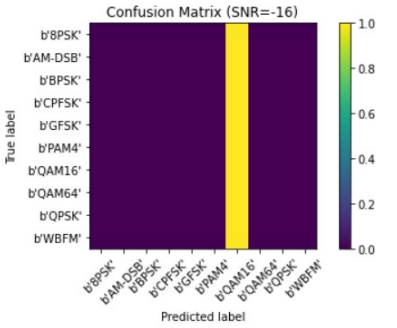

Confusion Matrix (SNR=2)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.2
0.4
0.6
0.8

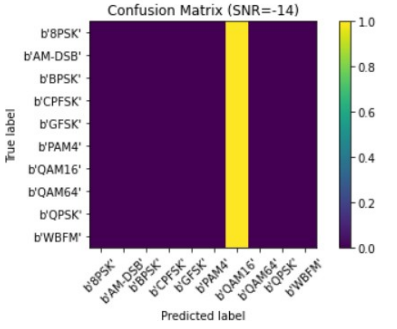

Confusion Matrix (SNR=4)
Confusion Matrix (SNR=8)
Confusion Matrix (SNR=6)
Confusion Matrix (SNR=10)
Confusion Matrix (SNR=12)
Confusion Matrix (SNR=16)
Confusion Matrix (SNR=14)
Confusion Matrix (SNR=18)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'

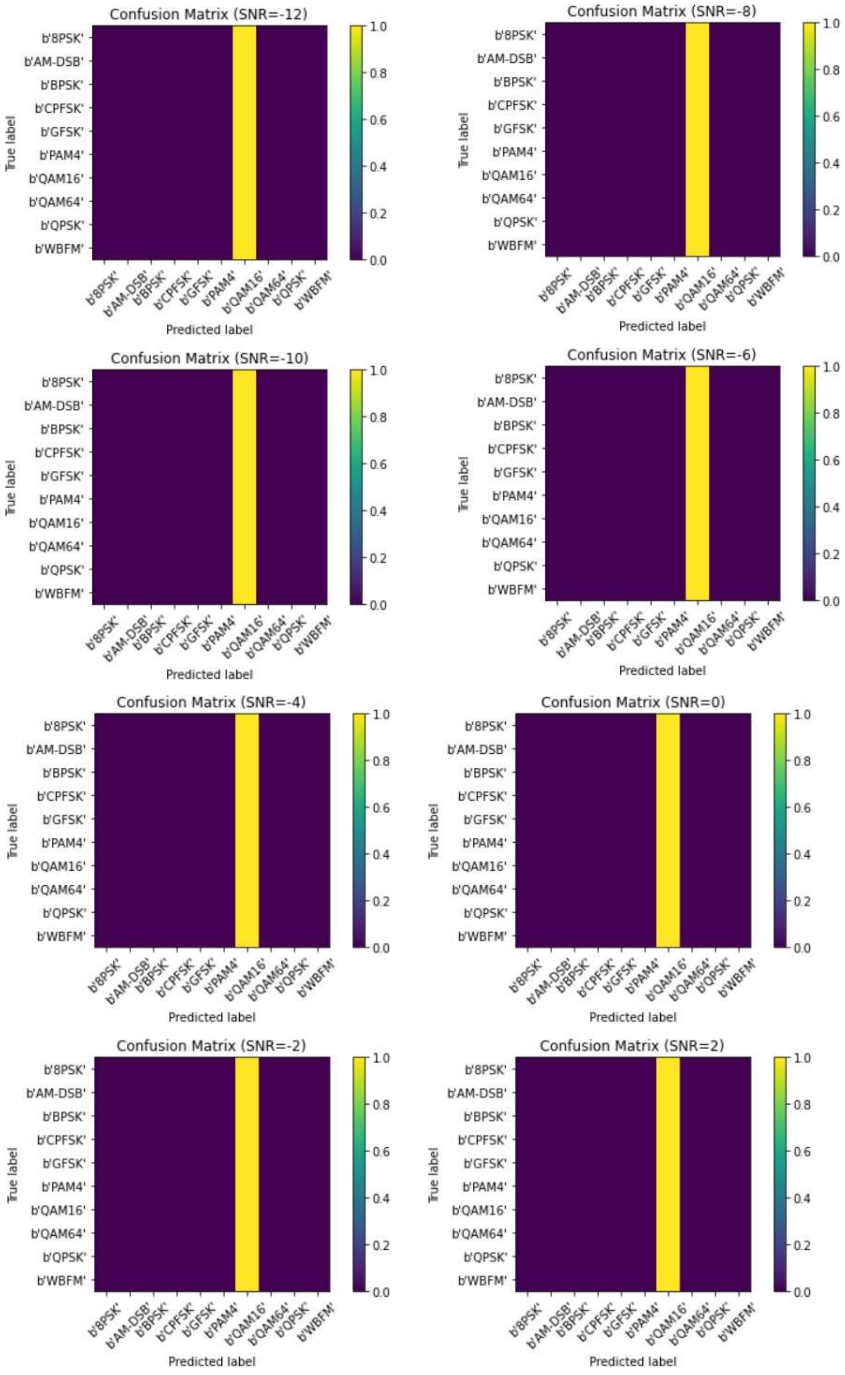
CNN with derivatives

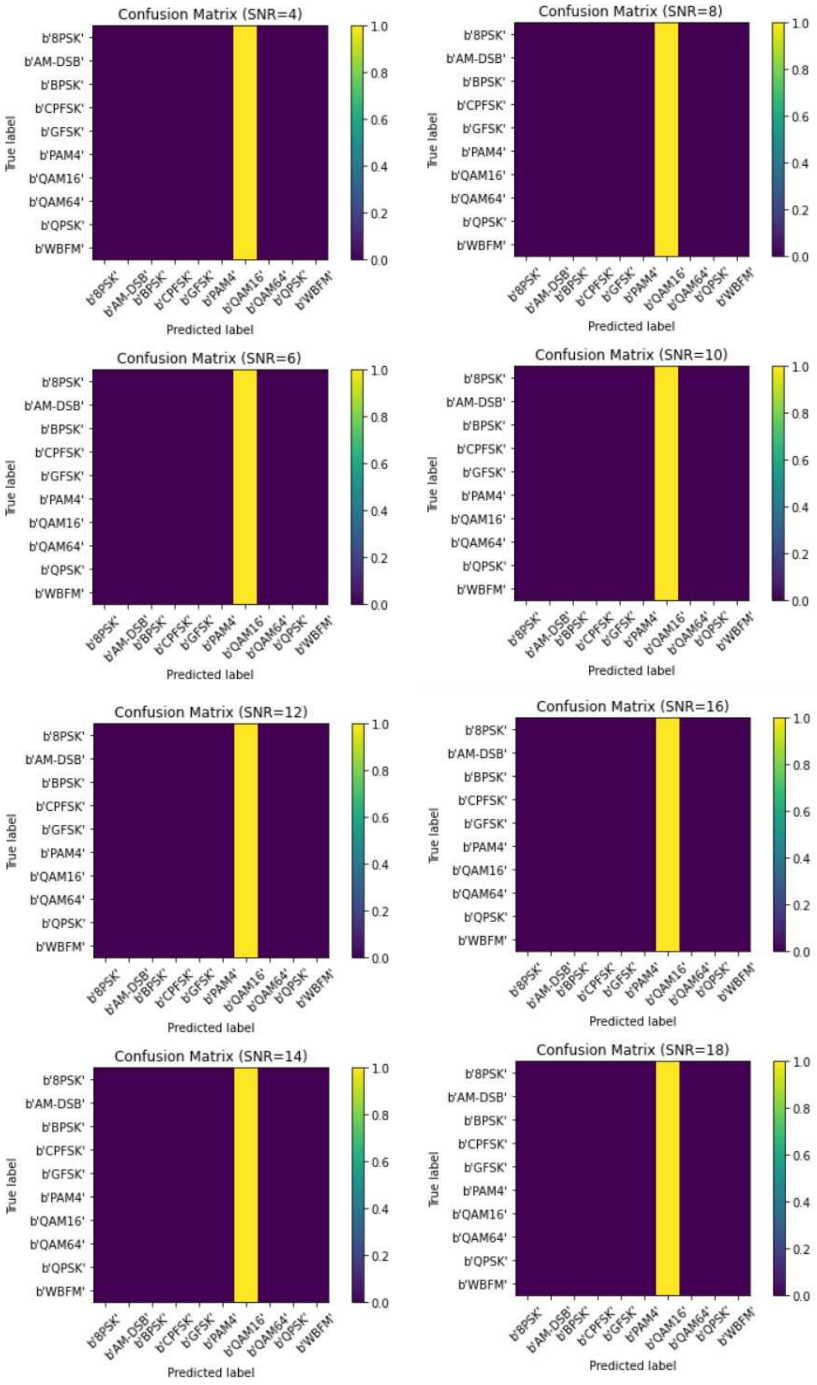




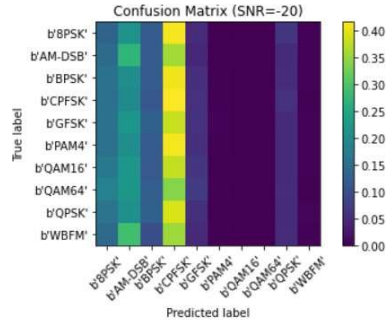


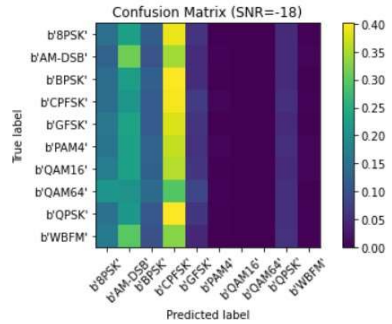


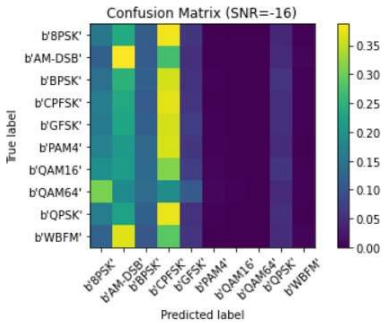

Confusion Matrix (SNR=-12)
Confusion Matrix (SNR=-8)
Confusion Matrix (SNR=-10)
Confusion Matrix (SNR=-6)
Confusion Matrix (SNR=-4)
Confusion Matrix (SNR=0)
Confusion Matrix (SNR=-2)
Confusion Matrix (SNR=2)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.2
0.4
0.6
0.8
1.0

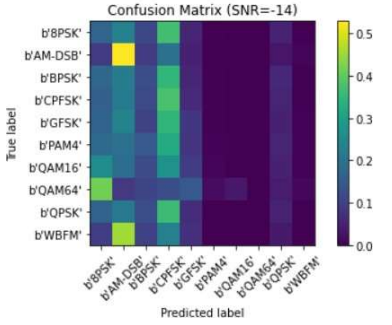

Confusion Matrix (SNR=4)
Confusion Matrix (SNR=8)
Confusion Matrix (SNR=6)
Confusion Matrix (SNR=10)
Confusion Matrix (SNR=12)
Confusion Matrix (SNR=16)
Confusion Matrix (SNR=14)
Confusion Matrix (SNR=18)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

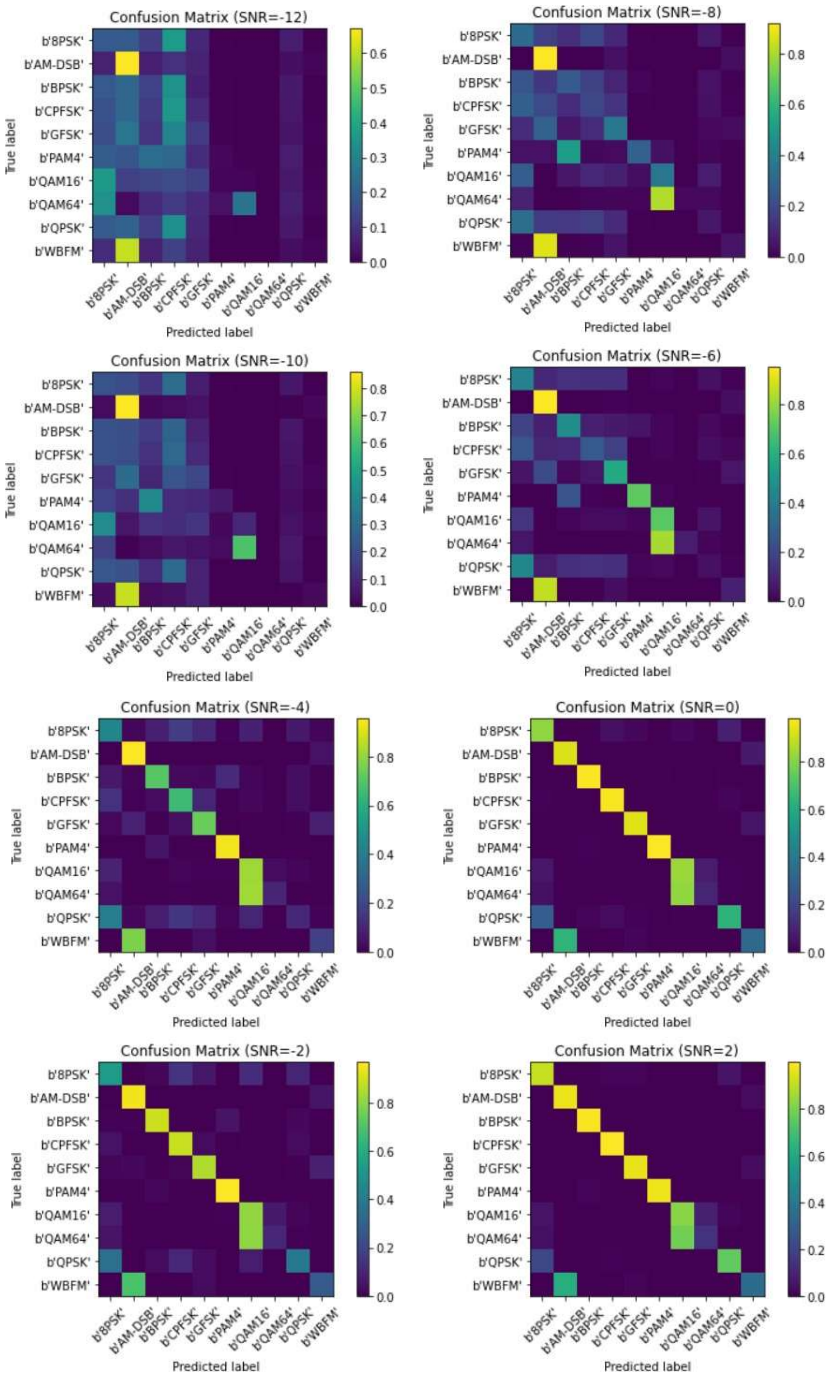
CNN with integrals

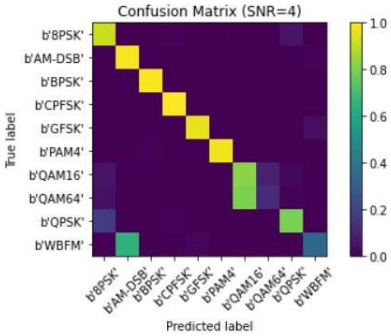


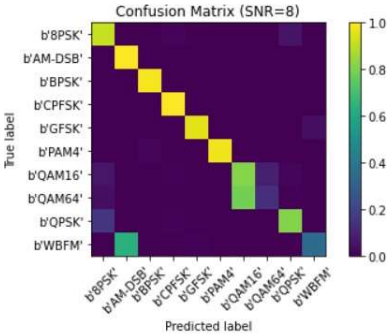


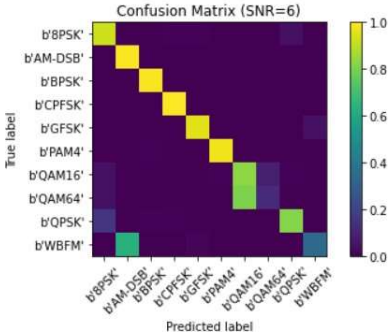


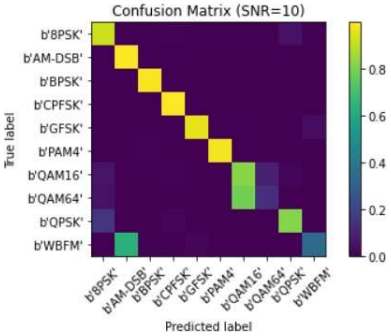


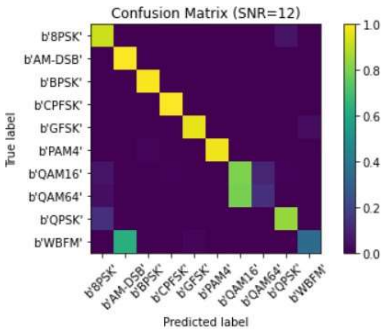

Confusion Matrix (SNR=-12)
Confusion Matrix (SNR=-8)
Confusion Matrix (SNR=-10)
Confusion Matrix (SNR=-6)
Confusion Matrix (SNR=-4)
Confusion Matrix (SNR=0)
Confusion Matrix (SNR=-2)
Confusion Matrix (SNR=2)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'

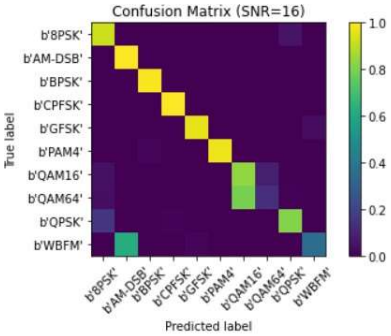

Confusion Matrix (SNR=4)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

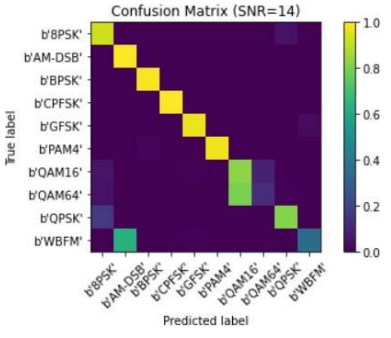

Confusion Matrix (SNR=8)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

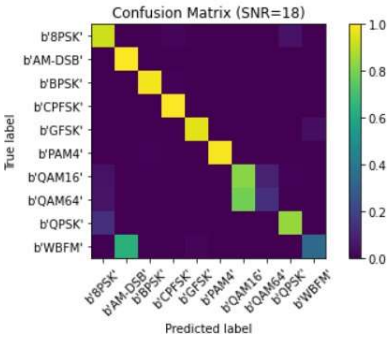

Confusion Matrix (SNR=6)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0


Confusion Matrix (SNR=10)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.8
0.6
0.4
0.2
0.0

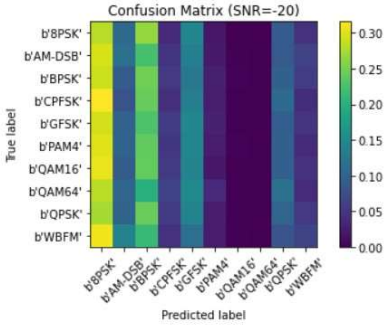

Confusion Matrix (SNR=12)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

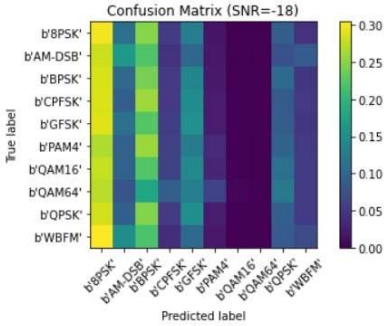

Confusion Matrix (SNR=16)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

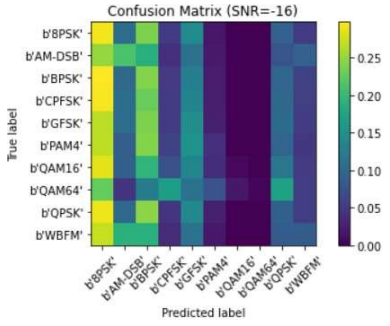

Confusion Matrix (SNR=14)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

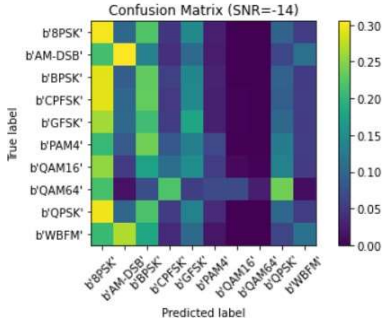

Confusion Matrix (SNR=18)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

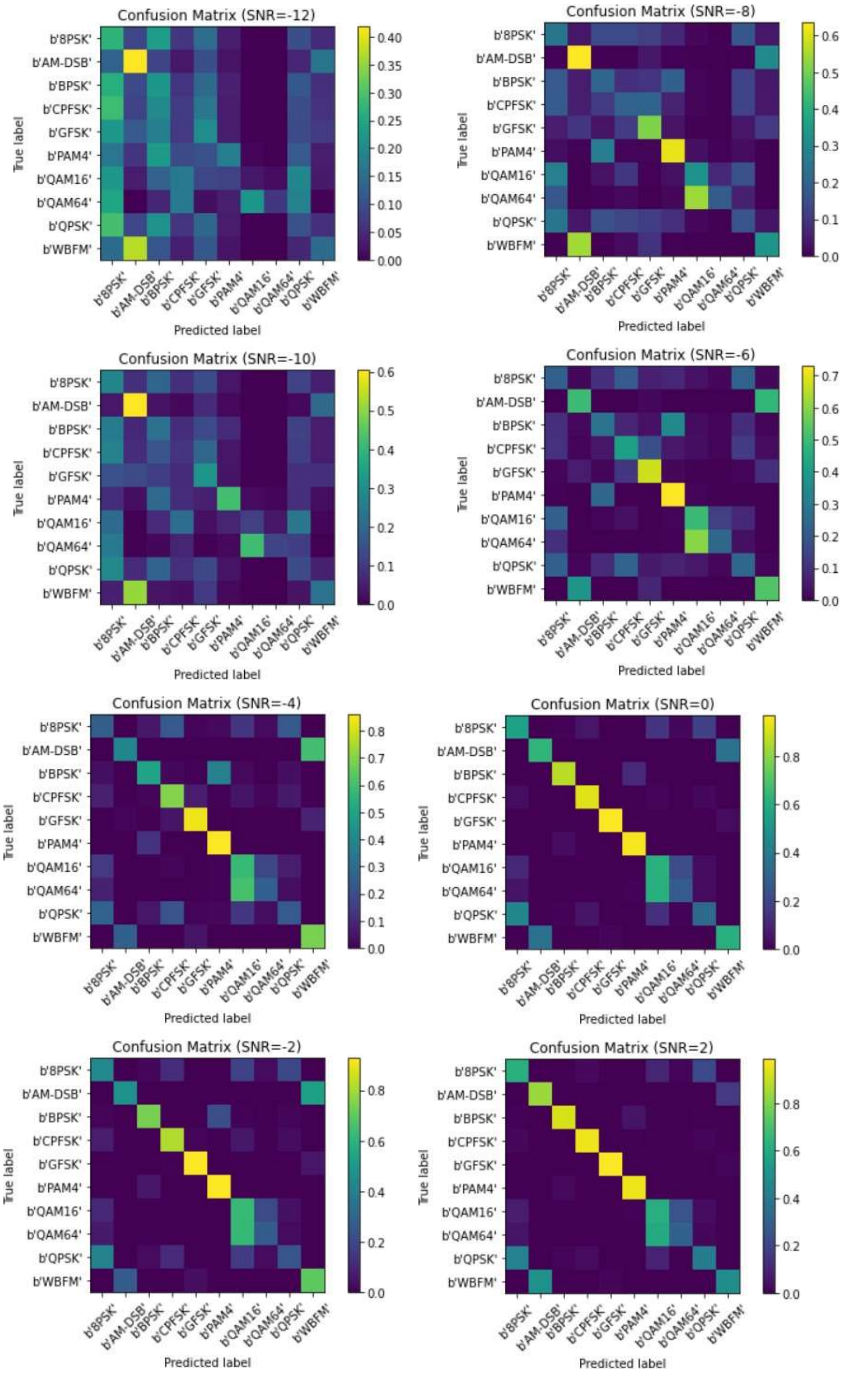
RNN with raw data

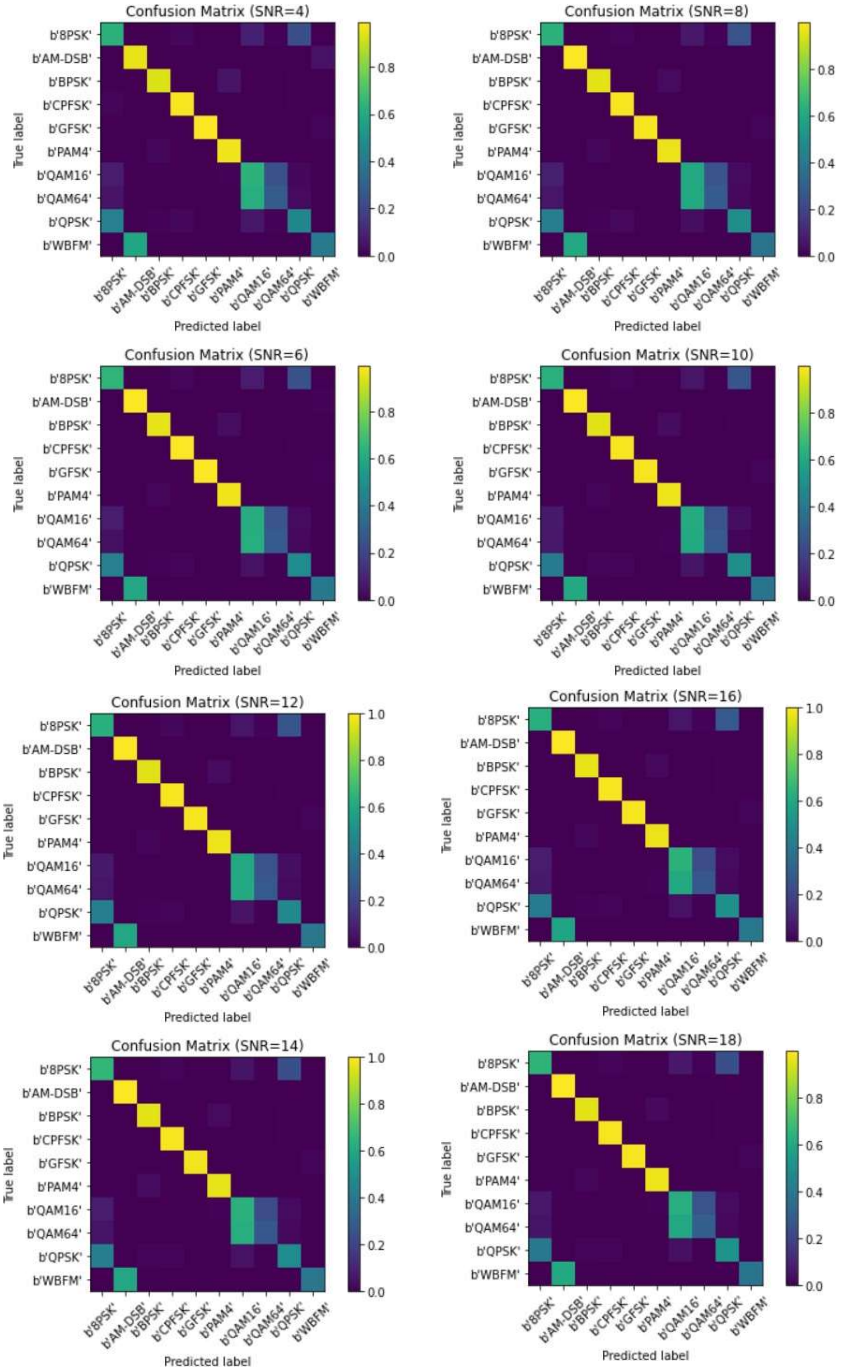




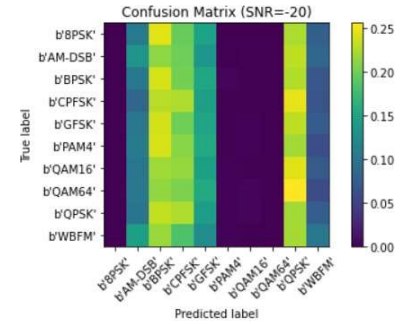


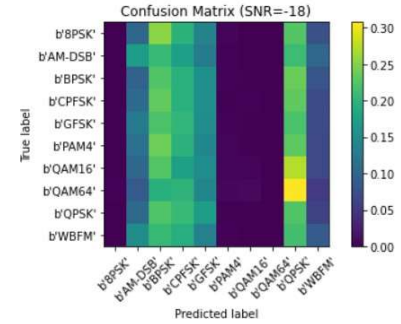


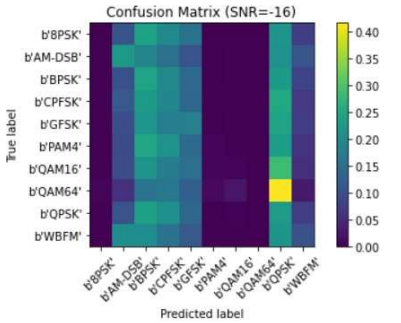

Confusion Matrix (SNR=-12)
Confusion Matrix (SNR=-8)
Confusion Matrix (SNR=-10)
Confusion Matrix (SNR=-6)
Confusion Matrix (SNR=-4)
Confusion Matrix (SNR=0)
Confusion Matrix (SNR=-2)
Confusion Matrix (SNR=2)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'

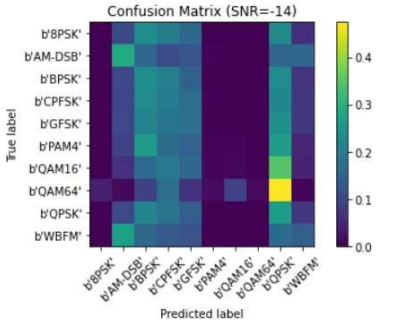

Confusion Matrix (SNR=4)
Confusion Matrix (SNR=8)
Confusion Matrix (SNR=6)
Confusion Matrix (SNR=10)
Confusion Matrix (SNR=12)
Confusion Matrix (SNR=16)
Confusion Matrix (SNR=14)
Confusion Matrix (SNR=18)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'

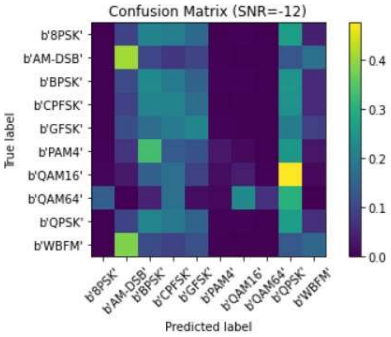
RNN with integrals

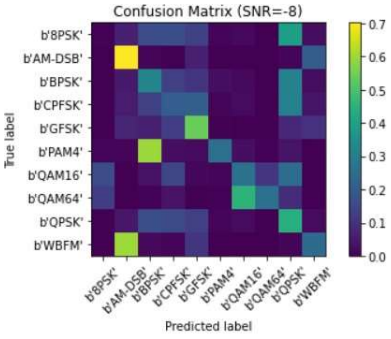


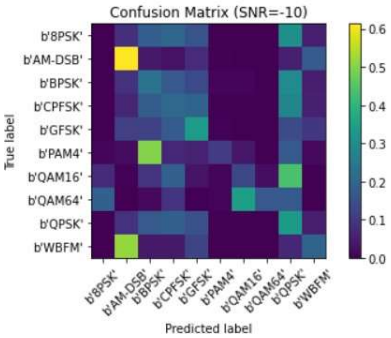


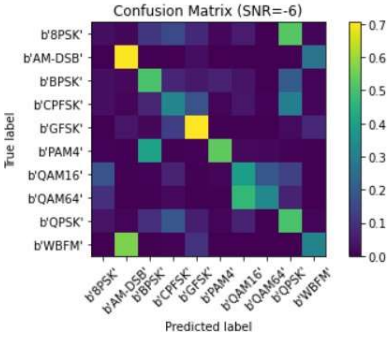


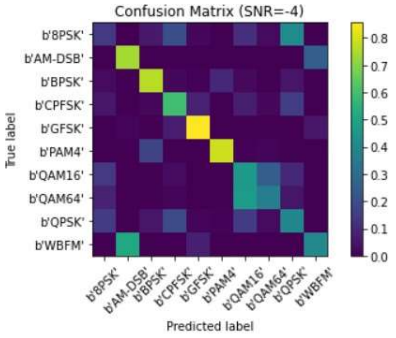


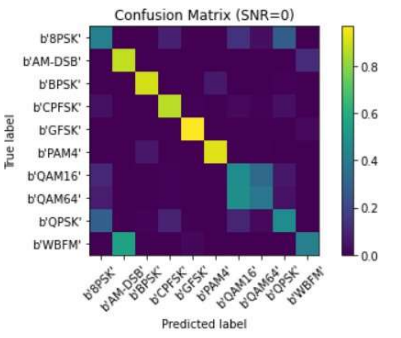

Confusion Matrix (SNR=-12)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.1
0.2
0.3
0.4

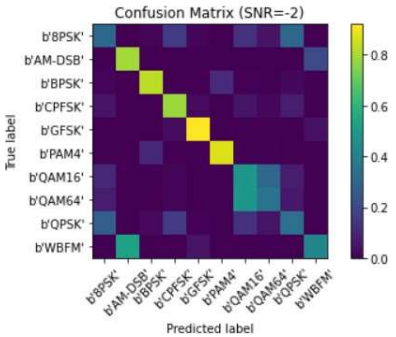

Confusion Matrix (SNR=-8)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

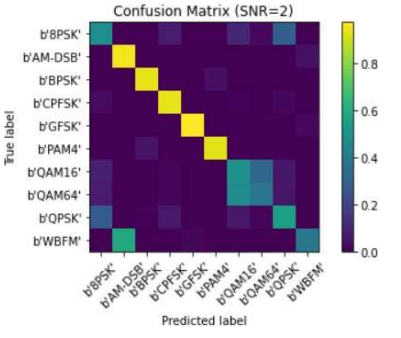

Confusion Matrix (SNR=-10)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.1
0.2
0.3
0.4
0.5
0.6

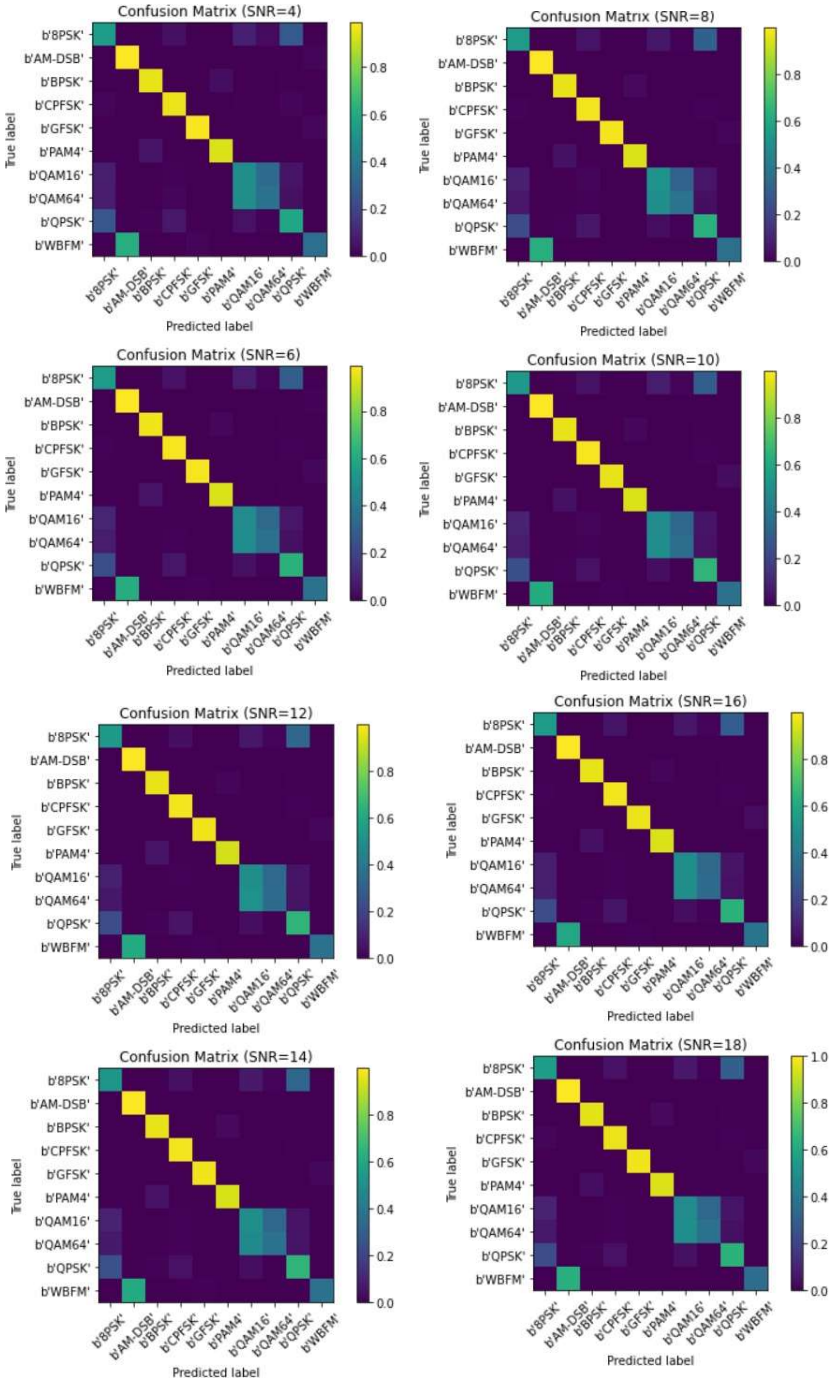

Confusion Matrix (SNR=-6)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7


Confusion Matrix (SNR=-4)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

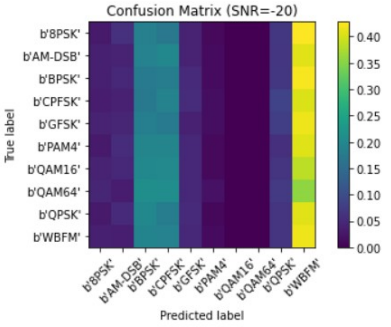

Confusion Matrix (SNR=0)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.2
0.4
0.6
0.8

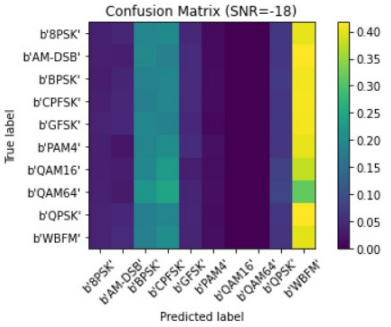

Confusion Matrix (SNR=-2)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.2
0.4
0.6
0.8

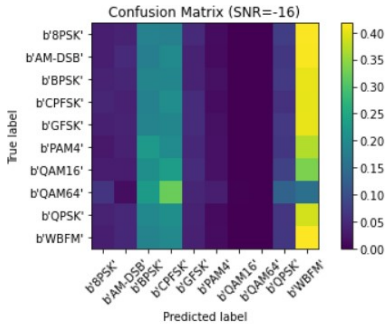

Confusion Matrix (SNR=2)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
0.0
0.2
0.4
0.6
0.8

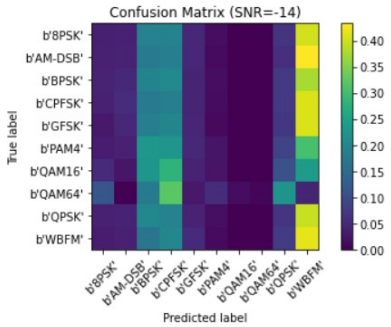

Confusion Matrix (SNR=4)
Confusion Matrix (SNR=8)
Confusion Matrix (SNR=6)
Confusion Matrix (SNR=10)
Confusion Matrix (SNR=12)
Confusion Matrix (SNR=16)
Confusion Matrix (SNR=14)
Confusion Matrix (SNR=18)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'

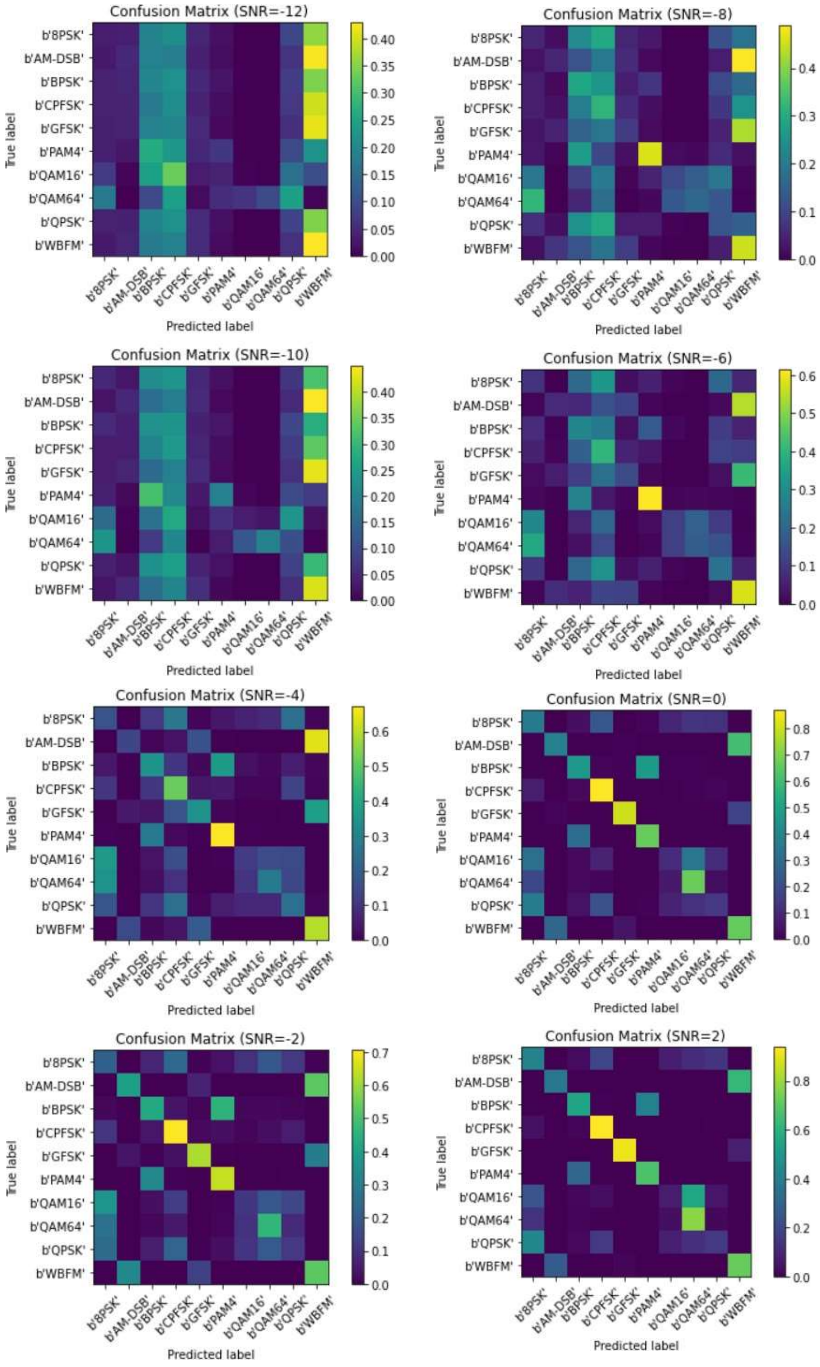
RNN with derivatives

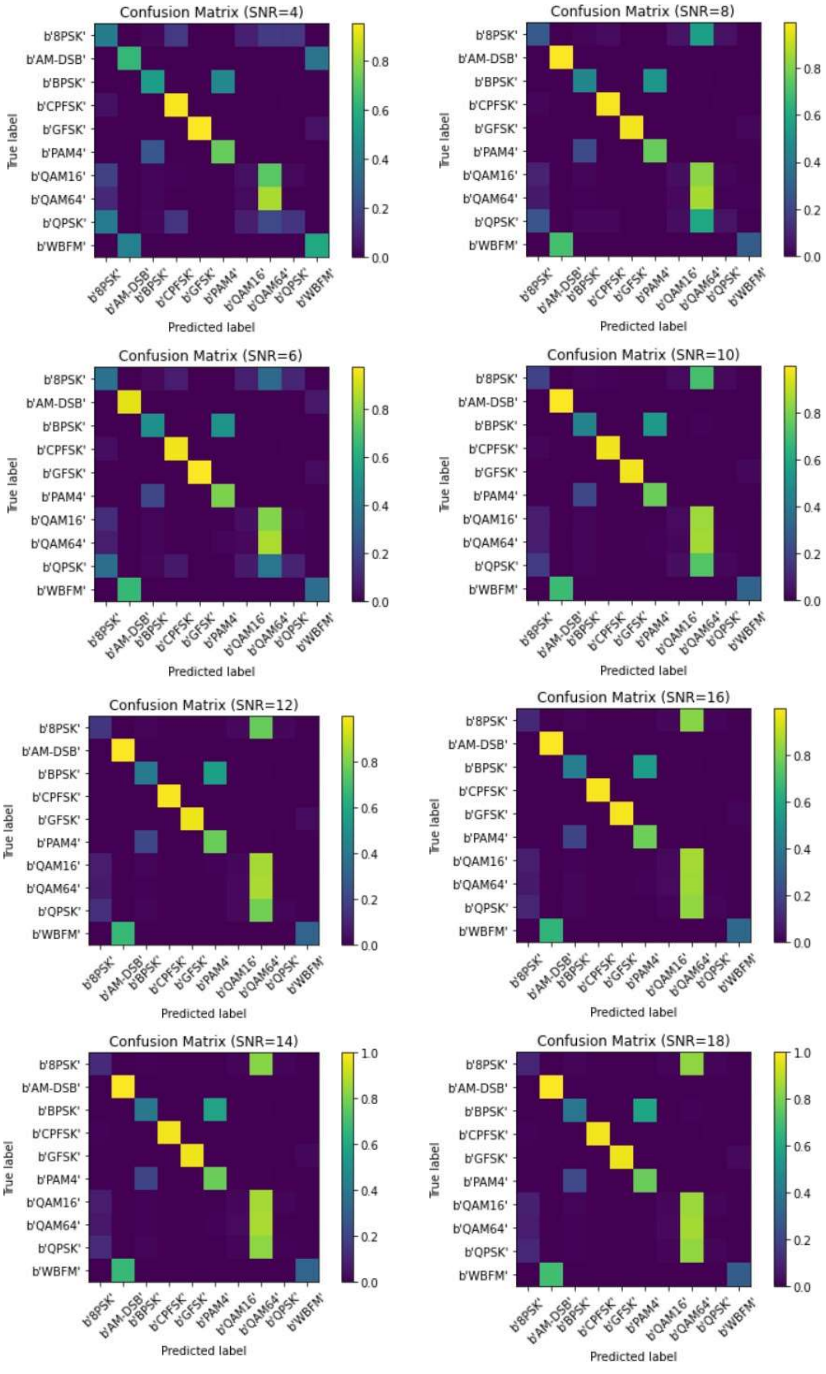




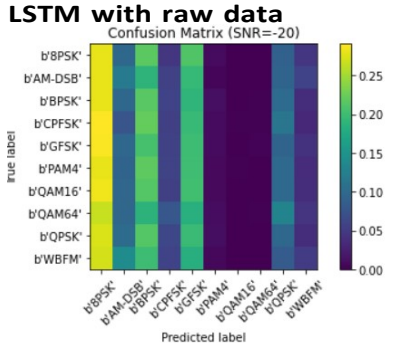


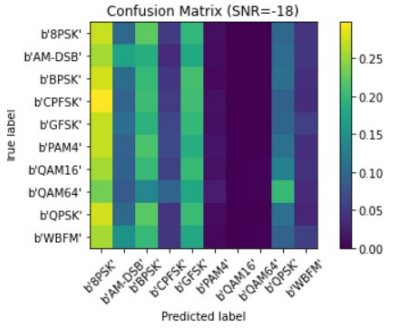


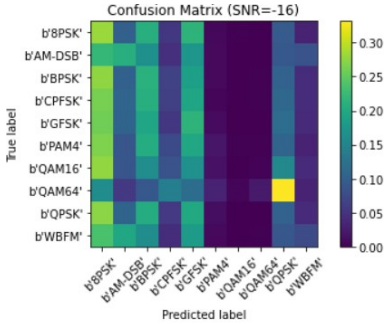

Confusion Matrix (SNR=-12)
Confusion Matrix (SNR=-8)
Confusion Matrix (SNR=-10)
Confusion Matrix (SNR=-6)
Confusion Matrix (SNR=-4)
Confusion Matrix (SNR=0)
Confusion Matrix (SNR=-2)
Confusion Matrix (SNR=2)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'

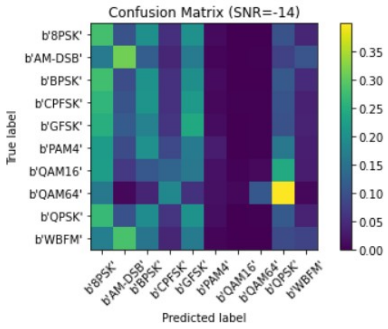

Confusion Matrix (SNR=4)
Confusion Matrix (SNR=8)
Confusion Matrix (SNR=6)
Confusion Matrix (SNR=10)
Confusion Matrix (SNR=12)
Confusion Matrix (SNR=16)
Confusion Matrix (SNR=14)
Confusion Matrix (SNR=18)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'

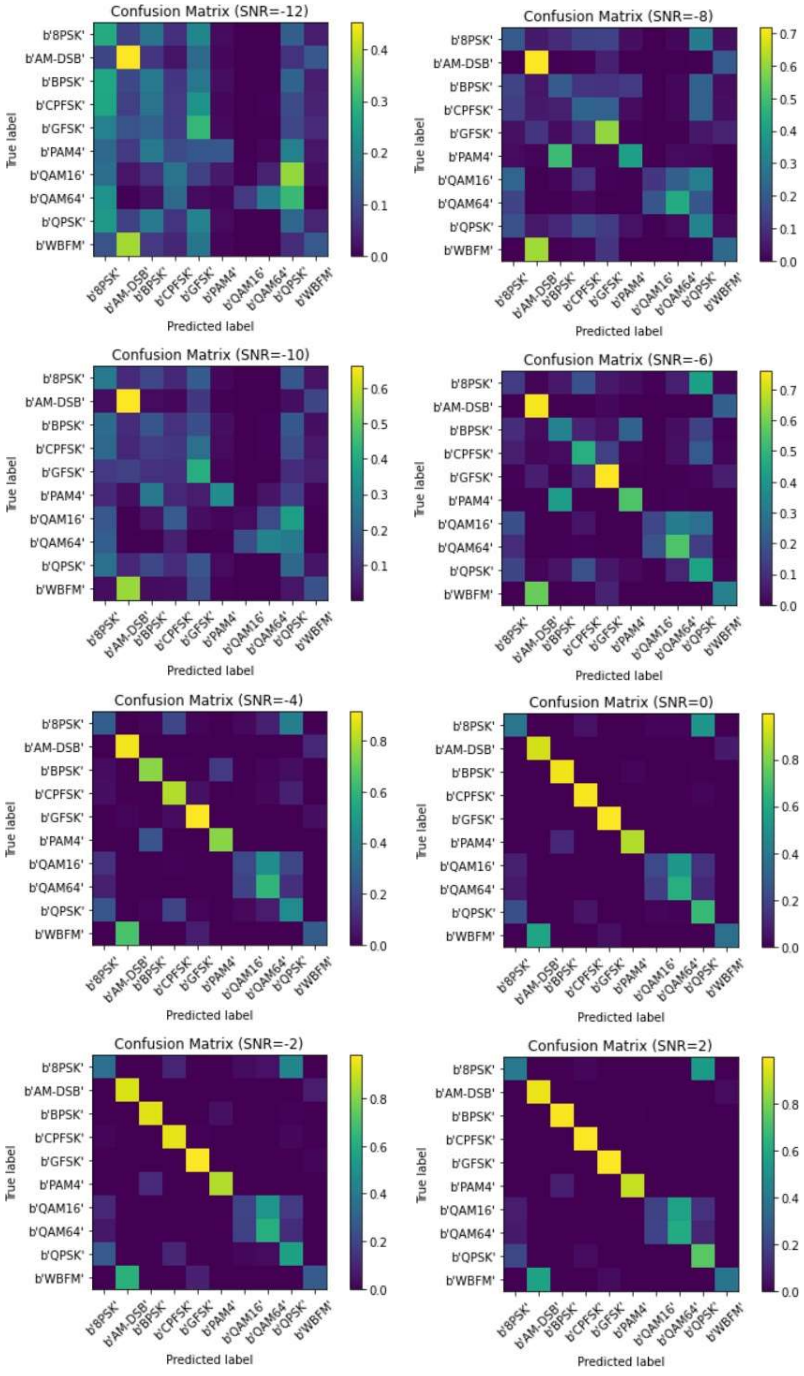
LSTM with raw data

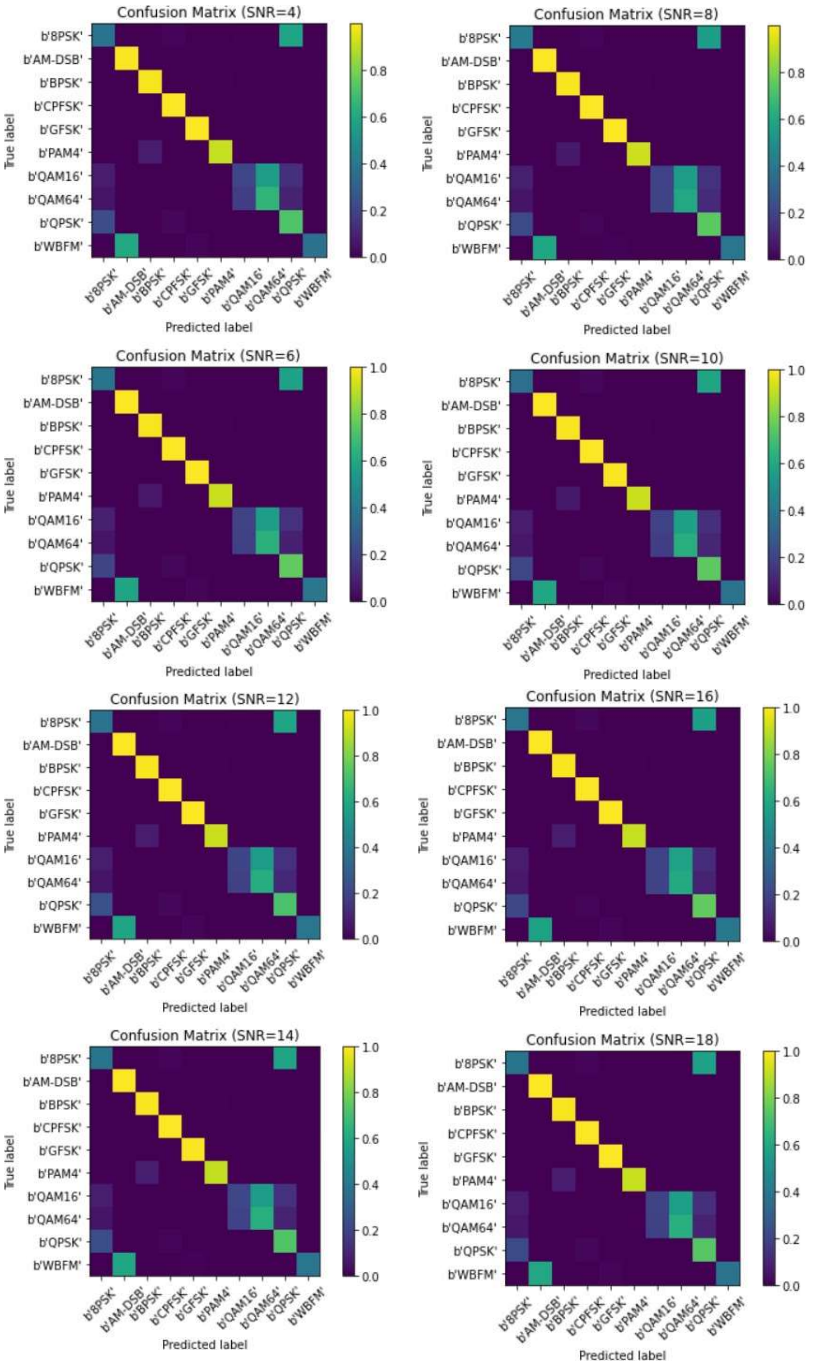




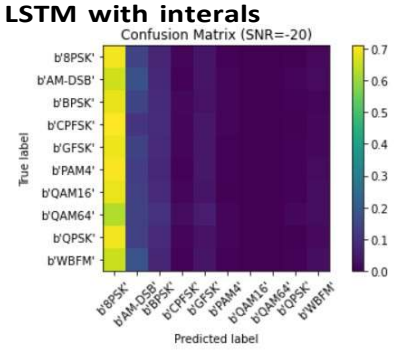


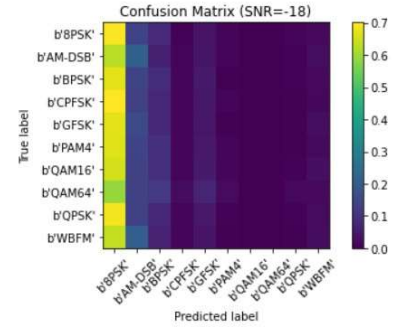


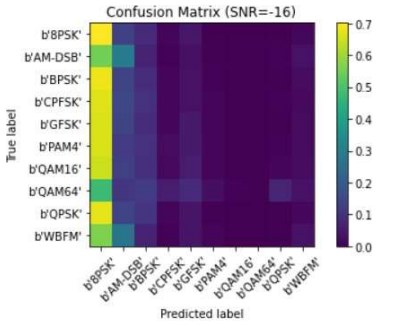

Confusion Matrix (SNR=-12)
Confusion Matrix (SNR=-8)
Confusion Matrix (SNR=-10)
Confusion Matrix (SNR=-6)
Confusion Matrix (SNR=-4)
Confusion Matrix (SNR=0)
Confusion Matrix (SNR=-2)
Confusion Matrix (SNR=2)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'

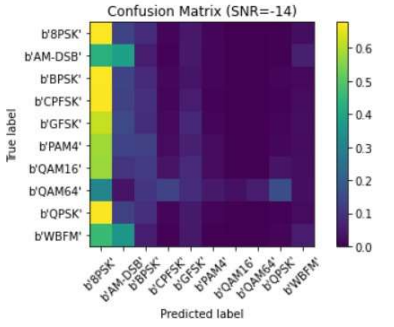

Confusion Matrix (SNR=4)
Confusion Matrix (SNR=8)
Confusion Matrix (SNR=6)
Confusion Matrix (SNR=10)
Confusion Matrix (SNR=12)
Confusion Matrix (SNR=16)
Confusion Matrix (SNR=14)
Confusion Matrix (SNR=18)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'

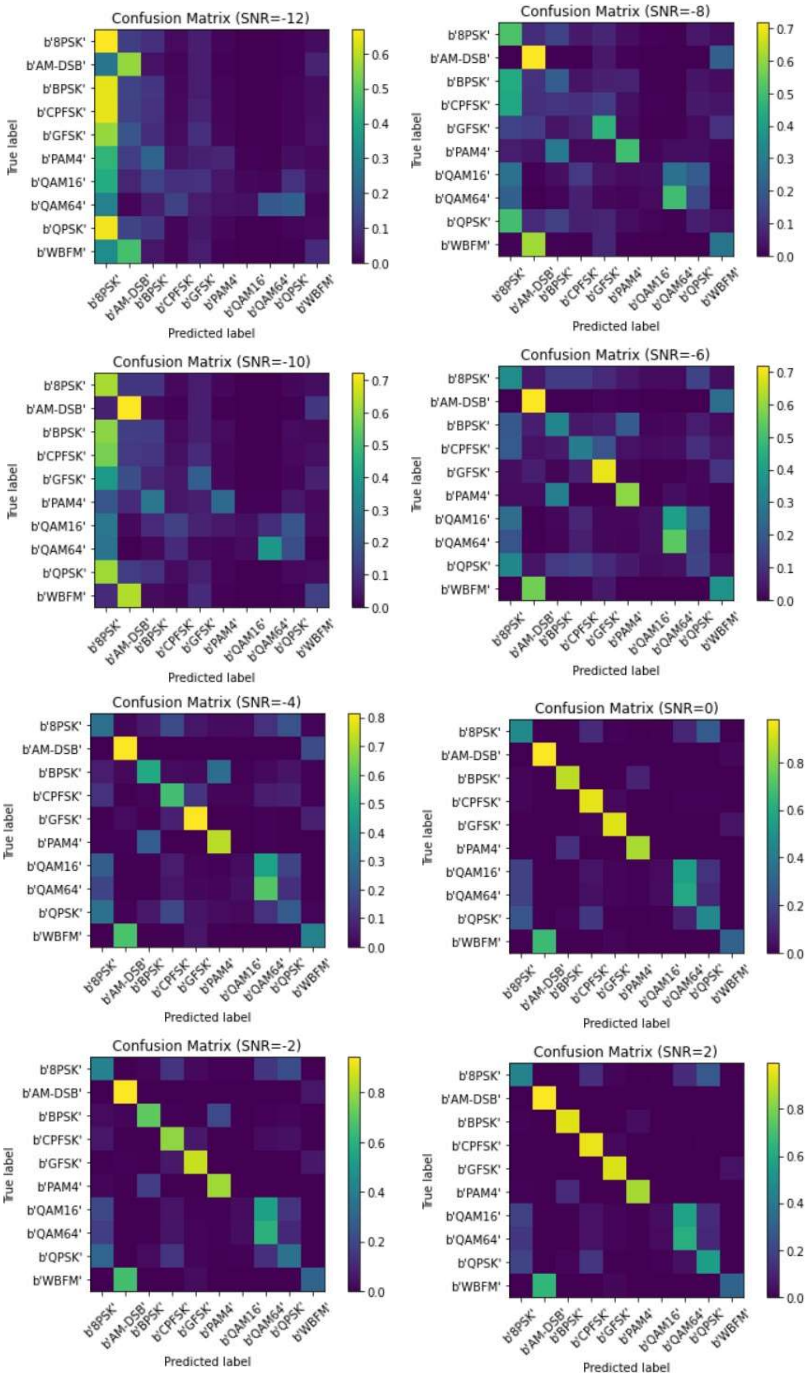
LSTM with interals

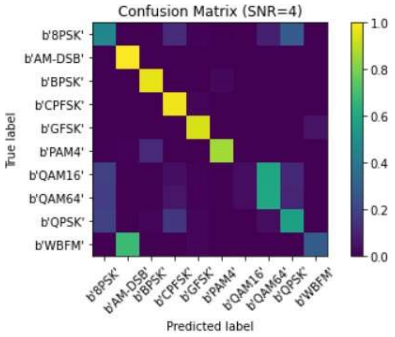


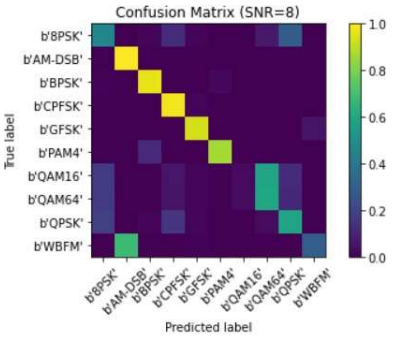


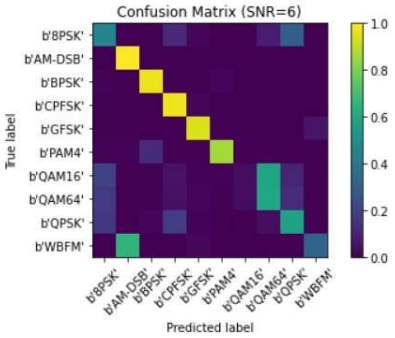


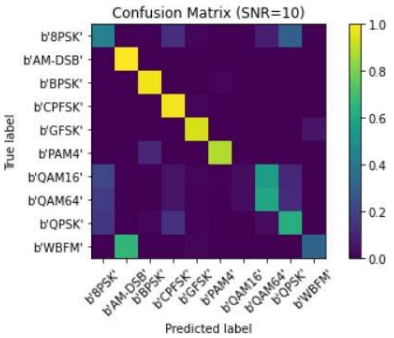


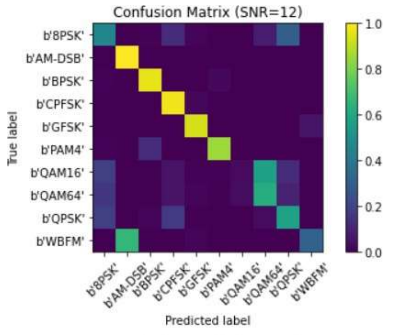

Confusion Matrix (SNR=-12)
Confusion Matrix (SNR=-8)
Confusion Matrix (SNR=-10)
Confusion Matrix (SNR=-6)
Confusion Matrix (SNR=-4)
Confusion Matrix (SNR=0)
Confusion Matrix (SNR=-2)
Confusion Matrix (SNR=2)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'

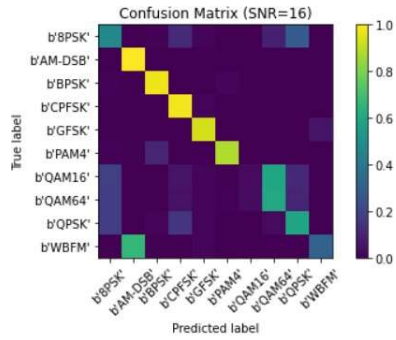

Confusion Matrix (SNR=4)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

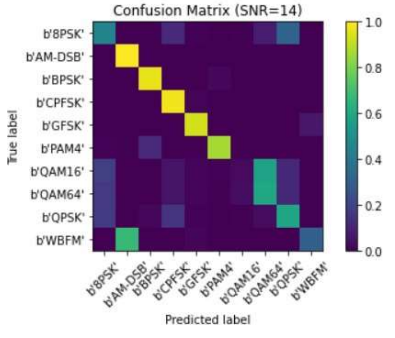

Confusion Matrix (SNR=8)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

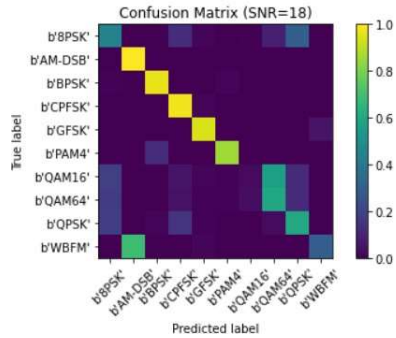

Confusion Matrix (SNR=6)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0


Confusion Matrix (SNR=10)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

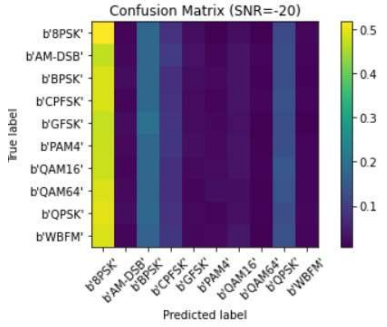

Confusion Matrix (SNR=12)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

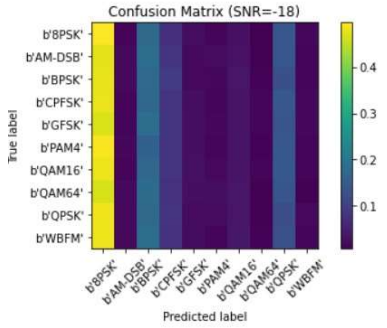

Confusion Matrix (SNR=16)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

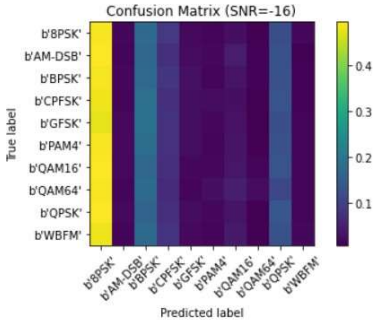

Confusion Matrix (SNR=14)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

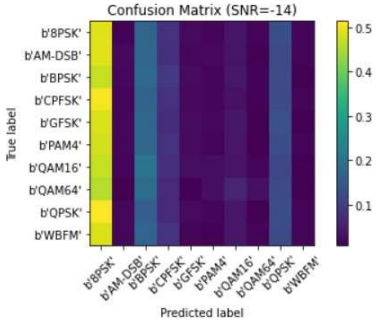

Confusion Matrix (SNR=18)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

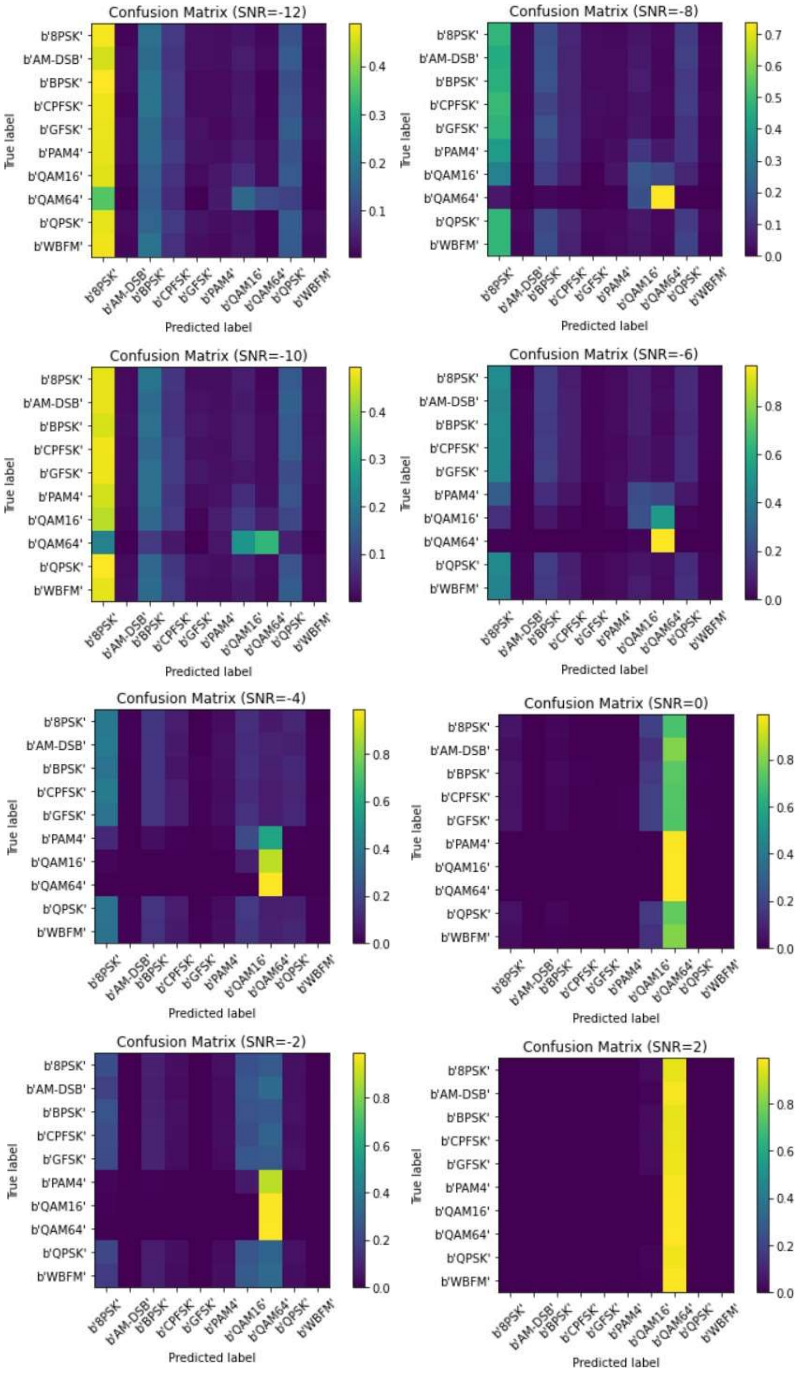
LSTM with iderivatives

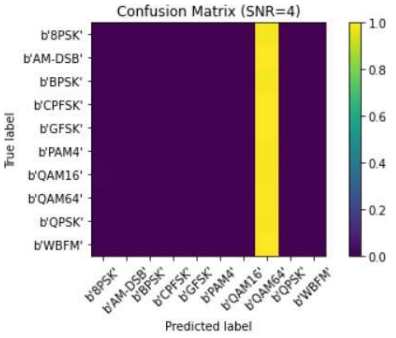


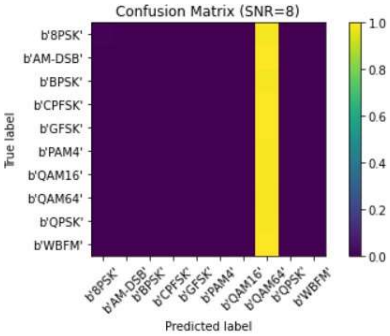


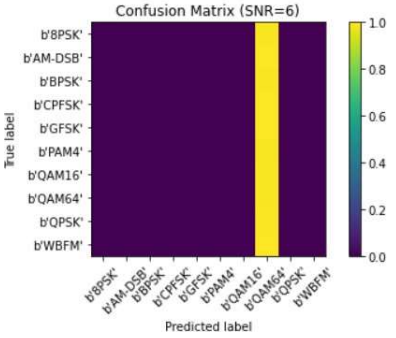


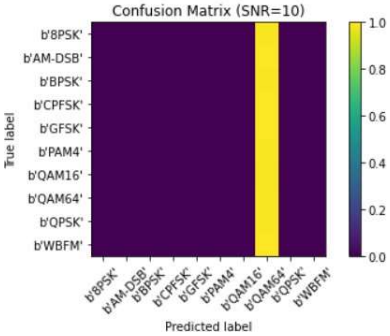


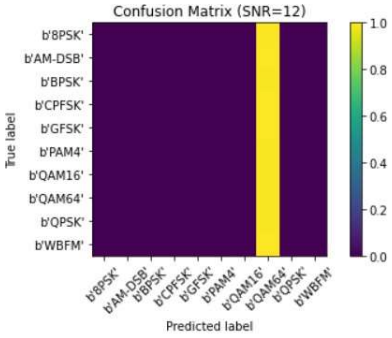

Confusion Matrix (SNR=-12)
Confusion Matrix (SNR=-8)
Confusion Matrix (SNR=-10)
Confusion Matrix (SNR=-6)
Confusion Matrix (SNR=-4)
Confusion Matrix (SNR=0)
Confusion Matrix (SNR=-2)
Confusion Matrix (SNR=2)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'

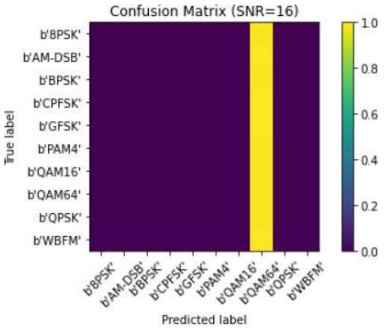

Confusion Matrix (SNR=4)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

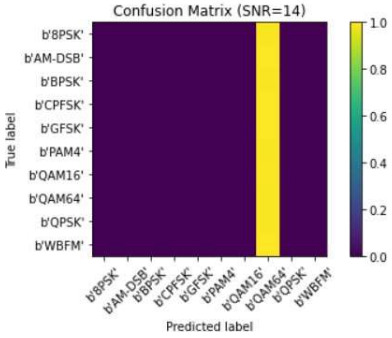

Confusion Matrix (SNR=8)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

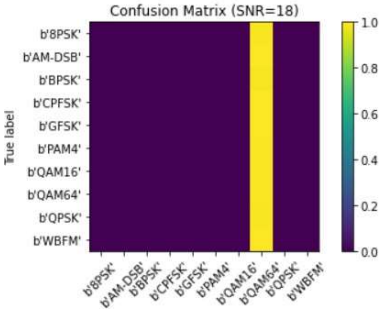

Confusion Matrix (SNR=6)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0


Confusion Matrix (SNR=10)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0


Confusion Matrix (SNR=12)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0


Confusion Matrix (SNR=16)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0


Confusion Matrix (SNR=14)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0


Confusion Matrix (SNR=18)
True label
Predicted label
b'8PSK'
b'AM-DSB'
b'BPSK'
b'CPFSK'
b'GFSK'
b'PAM4'
b'QAM16'
b'QAM64'
b'QPSK'
b'WBFM'
1.0
0.8
0.6
0.4
0.2
0.0

5 Bonus

5.1.1 Model design

Fixing data dimensions

```
[37] from keras.utils.np_utils import to_categorical

     X_train = X_train.reshape(840000,2,1,128)
     Y_train = to_categorical(Y_train[:,0], num_classes=10)
     Y_train = Y_train.astype(int);

     X_test = X_test.reshape(360000,2,1,128)
     Y_test = to_categorical(Y_test[:,0], num_classes=10)
     Y_test = Y_test.astype(int);

     X_train = X_train[:,:,np.newaxis]
     X_test = X_test[:,:,np.newaxis]
```

CONV-LSTM using raw data

```
[48] from keras.layers.convolutional import Conv3D
     from keras.layers.convolutional_recurrent import ConvLSTM2D

     input_shape = (2,1,1,128)
     model9 = keras.models.Sequential()
     model9.add(ConvLSTM2D(64, (1, 3), activation="relu",input_shape=input_shape,strides=1,padding='
     model9.add(keras.layers.Flatten())
     model9.add(Dense(256, activation='relu'))
     model9.add(keras.layers.Dropout(0.3))
     model9.add(Dense(11, activation="relu"))
     model9.add(Dense(10, activation="softmax"))
     model9.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
     model9.summary()
```

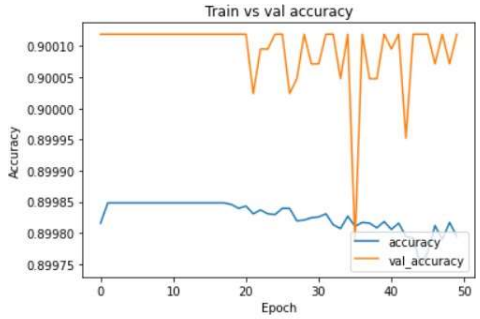
```
Model: "sequential_30"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv_lstm2d_27 (ConvLSTM2D)  (None, 1, 1, 64)          147712

 flatten_1 (Flatten)          (None, 64)                0

 dense_20 (Dense)             (None, 256)               16640

 dropout_1 (Dropout)          (None, 256)               0

 dense_21 (Dense)             (None, 11)                2827

 dense_22 (Dense)             (None, 10)                120

=================================================================
Total params: 167,299
Trainable params: 167,299
Non-trainable params: 0
_________________________________________________________________
```

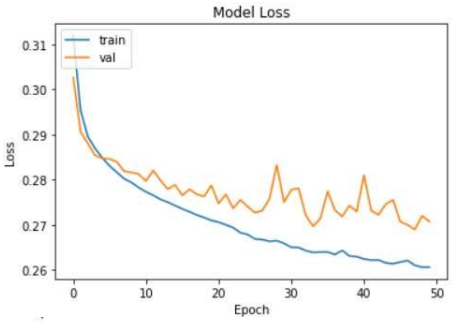
5.1.2 Training/testing results

```
[50] score9 = model9.evaluate(X_test, Y_test)
     print(model9.metrics_names)
     print (score9)
```

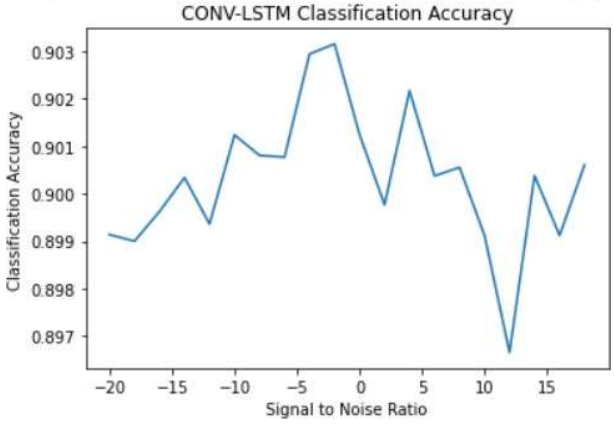
```
11250/11250 [==============================] - 33s 3ms/step - loss: 0.2892 - accurac
['loss', 'accuracy']
[0.28916457295417786, 0.9003194570541382]
```

5.1.3 Training/testing visuals






```
Overall Accuracy:  0.8991387876780391
Overall Accuracy:  0.8990011718096088
Overall Accuracy:  0.899626387107567
Overall Accuracy:  0.9003405728323377
Overall Accuracy:  0.8993634099086631
Overall Accuracy:  0.9012386824418153
Overall Accuracy:  0.900807254229791
Overall Accuracy:  0.9007718984226424
Overall Accuracy:  0.9029432958261737
Overall Accuracy:  0.9031573065101984
Overall Accuracy:  0.9012469383210866
Overall Accuracy:  0.8997676734152008
Overall Accuracy:  0.9021703205905035
Overall Accuracy:  0.9003756237035376
Overall Accuracy:  0.9005552470849528
Overall Accuracy:  0.8991191623732757
Overall Accuracy:  0.8966533244385145
Overall Accuracy:  0.9003822926477921
Overall Accuracy:  0.899121343565788
Overall Accuracy:  0.900609756097561
```



Conclusion

All the detailed training and validation accuracies are present in the google co-lab notebook.
The models couldn't be trained on more than one feature space together (raw + derivative + integral), the session crashed due to using the whole available RAM.
Raw data gave the best results, followed by data integrals and finally data derivatives.