Modulation Recognition Example: RML2016.10a Dataset + VT-CNN2 Mod-Rec Network

A more detailed description of this work can be found at https://arxiv.org/abs/1602.04105

A more detailed description of the RML2016.10a dataset can be found at http://pubs.gnuradio.org/index.php/grcon/article/view/11

Citation of this work is required in derivative works:

```
@article{convnetmodrec,
  title={Convolutional Radio Modulation Recognition Networks},
  author={O'Shea, Timothy J and Corgan, Johnathan and Clancy, T. Charles},
  journal={arXiv preprint arXiv:1602.04105},
  year={2016}
}
@article{rml_datasets,
  title={Radio Machine Learning Dataset Generation with GNU Radio},
  author={O'Shea, Timothy J and West, Nathan},
  journal={Proceedings of the 6th GNU Radio Conference},
  year={2016}
}
```

The RML2016.10a dataset is used for this work (https://radioml.com/datasets/)

In [ ]:

```
# Import all the things we need ---
#   by setting env variables before Keras import you can set up which backend and which GPU it uses
%matplotlib inline
import os,random
os.environ["KERAS_BACKEND"] = "theano"
#os.environ["KERAS_BACKEND"] = "tensorflow"
os.environ["THEANO_FLAGS"]  = "device=gpu%d"%(1)
import numpy as np
import theano as th
import theano.tensor as T
from keras.utils import np_utils
import keras.models as models
from keras.layers.core import Reshape,Dense,Dropout,Activation,Flatten
from keras.layers.noise import GaussianNoise
from keras.layers.convolutional import Convolution2D, MaxPooling2D, ZeroPadding2D
from keras.regularizers import *
from keras.optimizers import adam
import matplotlib.pyplot as plt
import seaborn as sns
import cPickle, random, sys, keras
```

```
Using gpu device 1: GeForce GTX TITAN X (CNMeM is disabled, cuDNN 5004)
Using Theano backend.
/usr/local/lib/python2.7/dist-packages/IPython/html.py:14: ShimWarning: The `IPython.html` package has been deprecated. You should
import from `notebook` instead. `IPython.html.widgets` has moved to `ipywidgets`.
  "`IPython.html.widgets` has moved to `ipywidgets`.", ShimWarning)
```

Dataset setup

In [ ]:

```
# Load the dataset ...
#  You will need to seperately download or generate this file
Xd = cPickle.load(open("RML2016.10a_dict.dat",'rb'))
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)
```

In [ ]:

```
# Partition the data
#  into training and test sets of the form we can train/test on
#  while keeping SNR and Mod labels handy for each
np.random.seed(2016)
n_examples = X.shape[0]
n_train = n_examples * 0.5
train_idx = np.random.choice(range(0,n_examples), size=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]
def to_onehot(yy):
    yy1 = np.zeros([len(yy), max(yy)+1])
    yy1[np.arange(len(yy)),yy] = 1
    return yy1
Y_train = to_onehot(map(lambda x: mods.index(lbl[x][0]), train_idx))
Y_test = to_onehot(map(lambda x: mods.index(lbl[x][0]), test_idx))
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:7: VisibleDeprecationWarning: using a non-integer number instead of an integer will result in an error in the future
```

In [ ]:

```
in_shp = list(X_train.shape[1:])
print X_train.shape, in_shp
classes = mods
```

```
(110000, 2, 128) [2, 128]
```

Build the NN Model

In [ ]:

```
# Build VT-CNN2 Neural Net model using Keras primitives --
#  - Reshape [N,2,128] to [N,1,2,128] on input
#  - Pass through 2 2DConv/ReLu layers
#  - Pass through 2 Dense layers (ReLu and Softmax)
#  - Perform categorical cross entropy optimization

dr = 0.5 # dropout rate (%)
model = models.Sequential()
model.add(Reshape([1]+in_shp, input_shape=in_shp))
model.add(ZeroPadding2D((0, 2)))
model.add(Convolution2D(256, 1, 3, border_mode='valid', activation="relu", name="conv1", init='glorot_unifor
model.add(Dropout(dr))
model.add(ZeroPadding2D((0, 2)))
model.add(Convolution2D(80, 2, 3, border_mode="valid", activation="relu", name="conv2", init='glorot_uniform
model.add(Dropout(dr))
model.add(Flatten())
model.add(Dense(256, activation='relu', init='he_normal', name="dense1"))
model.add(Dropout(dr))
model.add(Dense( len(classes), init='he_normal', name="dense2" ))
model.add(Activation('softmax'))
model.add(Reshape([len(classes)]))
model.compile(loss='categorical_crossentropy', optimizer='adam')
model.summary()
```

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
reshape_1 (Reshape)              (None, 1, 2, 128)     0           reshape_input_1[0][0]
____________________________________________________________________________________________________
zeropadding2d_1 (ZeroPadding2D)  (None, 1, 2, 132)     0           reshape_1[0][0]
____________________________________________________________________________________________________
conv1 (Convolution2D)            (None, 256, 2, 130)   1024        zeropadding2d_1[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 256, 2, 130)   0           conv1[0][0]
____________________________________________________________________________________________________
zeropadding2d_2 (ZeroPadding2D)  (None, 256, 2, 134)   0           dropout_1[0][0]
____________________________________________________________________________________________________
conv2 (Convolution2D)            (None, 80, 1, 132)    122960      zeropadding2d_2[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 80, 1, 132)    0           conv2[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 10560)         0           dropout_2[0][0]
____________________________________________________________________________________________________
dense1 (Dense)                   (None, 256)           2703616     flatten_1[0][0]
____________________________________________________________________________________________________
dropout_3 (Dropout)              (None, 256)           0           dense1[0][0]
____________________________________________________________________________________________________
dense2 (Dense)                   (None, 11)            2827        dropout_3[0][0]
____________________________________________________________________________________________________
activation_1 (Activation)        (None, 11)            0           dense2[0][0]
____________________________________________________________________________________________________
reshape_2 (Reshape)              (None, 11)            0           activation_1[0][0]
====================================================================================================
Total params: 2830427
____________________________________________________________________________________________________
```

In [ ]:

```
# Set up some params
nb_epoch = 100     # number of epochs to train on
batch_size = 1024  # training batch size
```

Train the Model

In [ ]:

```
# perform training ...
#   - call the main training loop in keras for our network+dataset
filepath = 'convmodrecnets_CNN2_0.5.wts.h5'
history = model.fit(X_train,
    Y_train,
    batch_size=batch_size,
    nb_epoch=nb_epoch,
    show_accuracy=False,
    verbose=2,
    validation_data=(X_test, Y_test),
    callbacks = [
        keras.callbacks.ModelCheckpoint(filepath, monitor='val_loss', verbose=0, save_best_only=True, mode='
        keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, verbose=0, mode='auto')
    ])
# we re-load the best weights once training is finished
model.load_weights(filepath)
```

```
Train on 110000 samples, validate on 110000 samples
Epoch 1/100
15s - loss: 2.2384 - val_loss: 2.1028
Epoch 2/100
15s - loss: 2.0282 - val_loss: 1.8806
Epoch 3/100
15s - loss: 1.8641 - val_loss: 1.7373
Epoch 4/100
16s - loss: 1.7378 - val_loss: 1.6276
Epoch 5/100
16s - loss: 1.6693 - val_loss: 1.5724
Epoch 6/100
16s - loss: 1.6102 - val_loss: 1.4985
Epoch 7/100
16s - loss: 1.5535 - val_loss: 1.4475
Epoch 8/100
16s - loss: 1.5115 - val_loss: 1.4226
Epoch 9/100
16s - loss: 1.4799 - val_loss: 1.4122
Epoch 10/100
16s - loss: 1.4569 - val_loss: 1.3884
Epoch 11/100
16s - loss: 1.4412 - val_loss: 1.3534
Epoch 12/100
16s - loss: 1.4264 - val_loss: 1.3395
Epoch 13/100
16s - loss: 1.4175 - val_loss: 1.3401
Epoch 14/100
16s - loss: 1.4080 - val_loss: 1.3408
Epoch 15/100
16s - loss: 1.4012 - val_loss: 1.3495
Epoch 16/100
16s - loss: 1.3988 - val_loss: 1.3250
Epoch 17/100
16s - loss: 1.3835 - val_loss: 1.3171
Epoch 18/100
16s - loss: 1.3775 - val_loss: 1.3095
Epoch 19/100
16s - loss: 1.3752 - val_loss: 1.3052
Epoch 20/100
16s - loss: 1.3743 - val_loss: 1.3056
Epoch 21/100
16s - loss: 1.3670 - val_loss: 1.3045
Epoch 22/100
16s - loss: 1.3643 - val_loss: 1.3231
Epoch 23/100
16s - loss: 1.3620 - val_loss: 1.3103
Epoch 24/100
16s - loss: 1.3541 - val_loss: 1.2933
Epoch 25/100
16s - loss: 1.3530 - val_loss: 1.3028
Epoch 26/100
16s - loss: 1.3479 - val_loss: 1.2945
Epoch 27/100
16s - loss: 1.3478 - val_loss: 1.2984
Epoch 28/100
16s - loss: 1.3428 - val_loss: 1.2893
Epoch 29/100
16s - loss: 1.3358 - val_loss: 1.3019
Epoch 30/100
16s - loss: 1.3333 - val_loss: 1.2903
Epoch 31/100
16s - loss: 1.3304 - val_loss: 1.2911
Epoch 32/100
16s - loss: 1.3247 - val_loss: 1.2825
Epoch 33/100
16s - loss: 1.3252 - val_loss: 1.2891
Epoch 34/100
16s - loss: 1.3207 - val_loss: 1.2846
Epoch 35/100
16s - loss: 1.3167 - val_loss: 1.2836
Epoch 36/100
16s - loss: 1.3186 - val_loss: 1.2789
Epoch 37/100
16s - loss: 1.3127 - val_loss: 1.2765
Epoch 38/100
16s - loss: 1.3101 - val_loss: 1.2802
Epoch 39/100
16s - loss: 1.3077 - val_loss: 1.2973
Epoch 40/100
```

```
16s - loss: 1.3033 - val_loss: 1.2996
Epoch 41/100
16s - loss: 1.3016 - val_loss: 1.2877
Epoch 42/100
16s - loss: 1.2993 - val_loss: 1.2769
Epoch 43/100
16s - loss: 1.2963 - val_loss: 1.3045
```

```
/usr/local/lib/python2.7/dist-packages/Keras-1.0.4-py2.7.egg/keras/models.py:391: UserWarning: The "show_accuracy" argument is
deprecated, instead you should pass the "accuracy" metric to the model at compile time:
`model.compile(optimizer, loss, metrics=["accuracy"])`
  warnings.warn('The "show_accuracy" argument is deprecated, '
```

Evaluate and Plot Model Performance

In [ ]:
```
# Show simple version of performance
score = model.evaluate(X_test, Y_test, show_accuracy=True, verbose=0, batch_size=batch_size)
print score
```

```
1.27649493232
```

```
/usr/local/lib/python2.7/dist-packages/Keras-1.0.4-py2.7.egg/keras/models.py:432: UserWarning: The "show_accuracy" argument is
deprecated, instead you should pass the "accuracy" metric to the model at compile time:
`model.compile(optimizer, loss, metrics=["accuracy"])`
  warnings.warn('The "show_accuracy" argument is deprecated, '
```

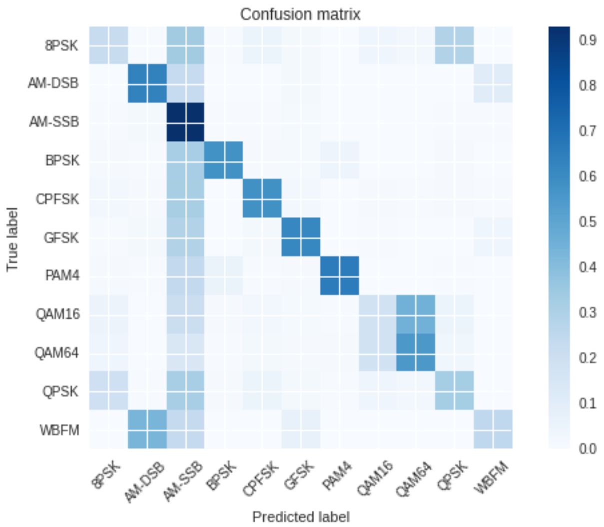
In [ ]:
```
# Show loss curves
plt.figure()
plt.title('Training performance')
plt.plot(history.epoch, history.history['loss'], label='train loss+error')
plt.plot(history.epoch, history.history['val_loss'], label='val_error')
plt.legend()
```

Out [9]:
```
&lt;matplotlib.legend.Legend at 0x7f089a6d7ad0&gt;
```

In [ ]:
```
def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.Blues, labels=[]):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    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')
```

In [ ]:
```
# Plot confusion matrix
test_Y_hat = model.predict(X_test, batch_size=batch_size)
conf = np.zeros([len(classes),len(classes)])
confnorm = np.zeros([len(classes),len(classes)])
for i in range(0,X_test.shape[0]):
    j = list(Y_test[i,:]).index(1)
    k = int(np.argmax(test_Y_hat[i,:]))
    conf[j,k] = conf[j,k] + 1
for i in range(0,len(classes)):
    confnorm[i,:] = conf[i,:] / np.sum(conf[i,:])
plot_confusion_matrix(confnorm, labels=classes)
```

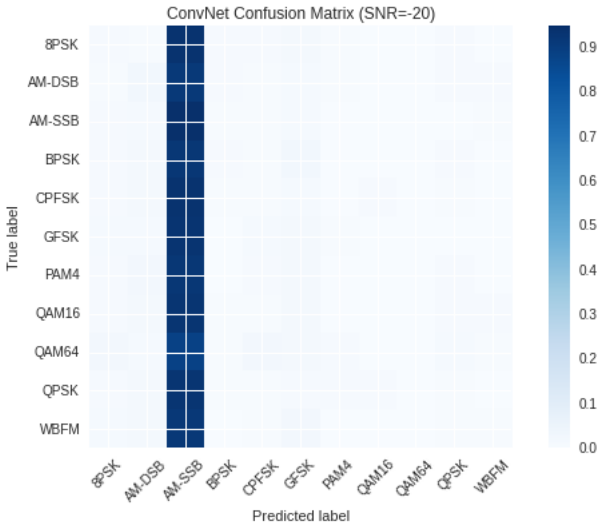
```
/usr/lib/pymodules/python2.7/matplotlib/collections.py:548: FutureWarning: elementwise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
  if self._edgecolors == 'face':
```

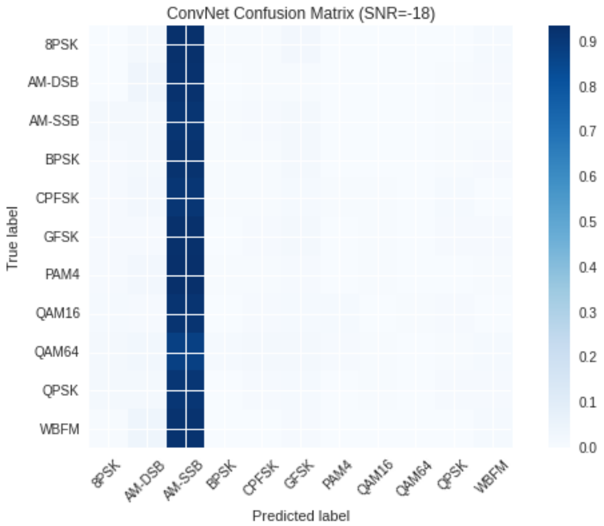


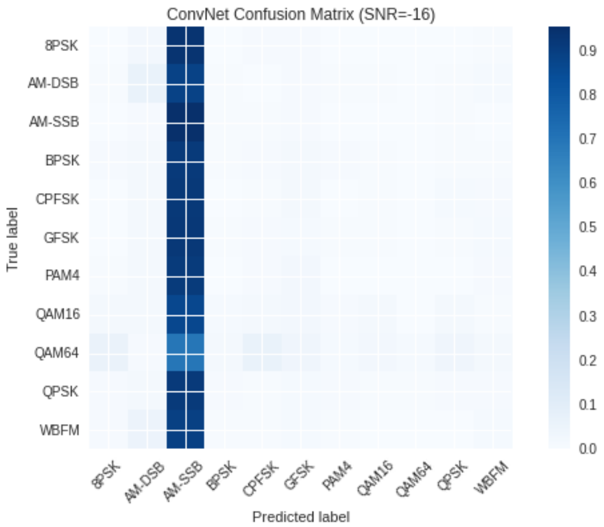
In [ ]:

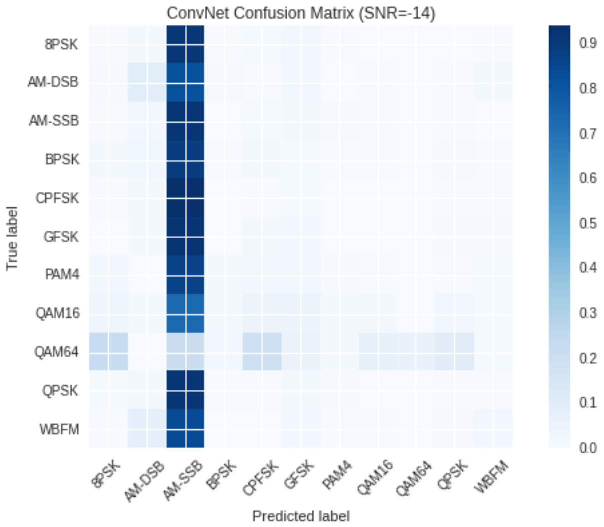
```
# Plot confusion matrix
acc = {}
for snr in snrs:

    # extract classes @ SNR
    test_SNRs = map(lambda x: lbl[x][1], test_idx)
    test_X_i = X_test[np.where(np.array(test_SNRs)==snr)]
    test_Y_i = Y_test[np.where(np.array(test_SNRs)==snr)]

    # estimate classes
    test_Y_i_hat = model.predict(test_X_i)
    conf = np.zeros([len(classes),len(classes)])
    confnorm = np.zeros([len(classes),len(classes)])
    for i in range(0,test_X_i.shape[0]):
        j = list(test_Y_i[i,:]).index(1)
        k = int(np.argmax(test_Y_i_hat[i,:]))
        conf[j,k] = conf[j,k] + 1
    for i in range(0,len(classes)):
        confnorm[i,:] = conf[i,:] / np.sum(conf[i,:])
    plt.figure()
    plot_confusion_matrix(confnorm, labels=classes, title="ConvNet Confusion Matrix (SNR=%d)"%(snr))

    cor = np.sum(np.diag(conf))
    ncor = np.sum(conf) - cor
    print "Overall Accuracy: ", cor / (cor+ncor)
    acc[snr] = 1.0*cor/(cor+ncor)
```

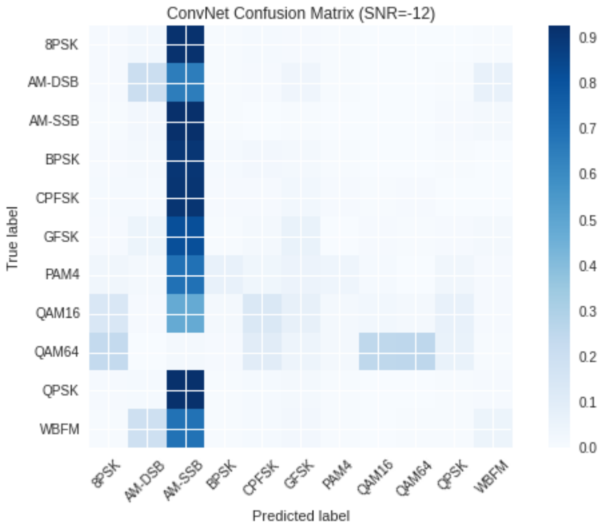
```
Overall Accuracy:  0.095516925892
Overall Accuracy:  0.0919164396004
Overall Accuracy:  0.10003617945
Overall Accuracy:  0.104474918802
Overall Accuracy:  0.150045578851
Overall Accuracy:  0.227652390261
Overall Accuracy:  0.348704758405
Overall Accuracy:  0.493401759531
Overall Accuracy:  0.589463955638
Overall Accuracy:  0.649310595065
Overall Accuracy:  0.704197080292
Overall Accuracy:  0.708833151581
Overall Accuracy:  0.722834067548
Overall Accuracy:  0.72761732852
Overall Accuracy:  0.717583408476
Overall Accuracy:  0.732786885246
Overall Accuracy:  0.723881948217
Overall Accuracy:  0.729470381989
Overall Accuracy:  0.725067873303
Overall Accuracy:  0.723852385239
```

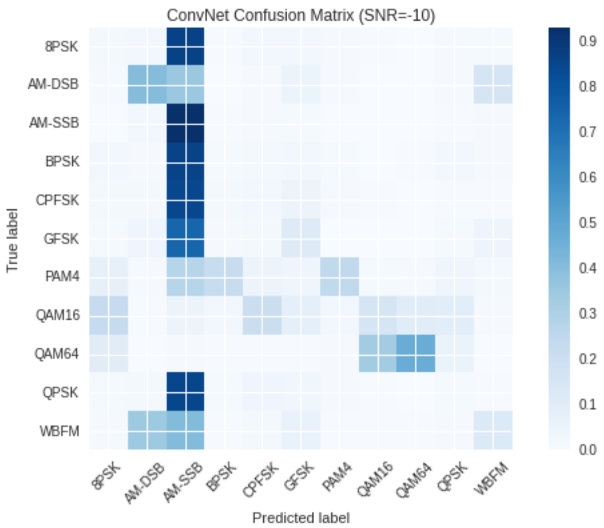

ConvNet Confusion Matrix (SNR=-20)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

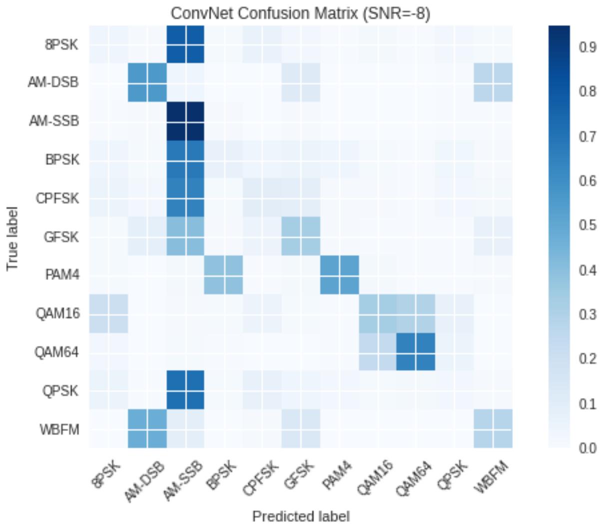

ConvNet Confusion Matrix (SNR=-18)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

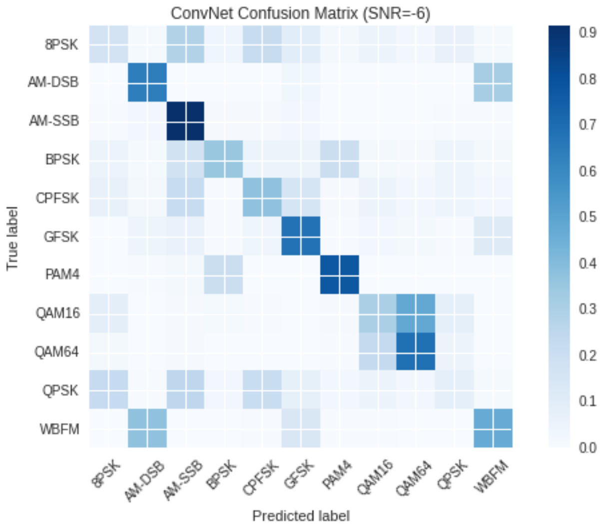

ConvNet Confusion Matrix (SNR=-16)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

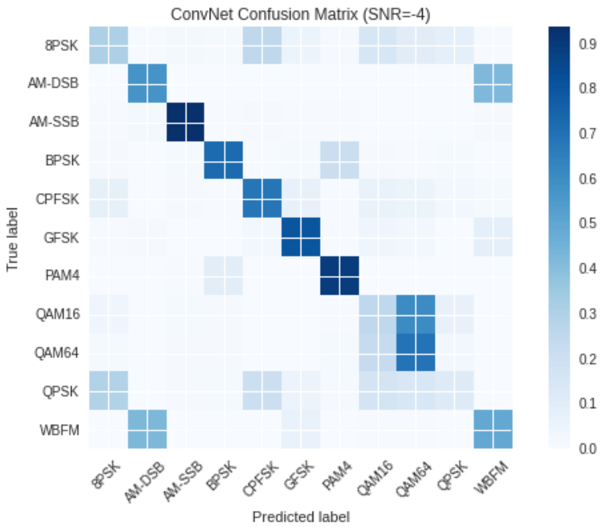

ConvNet Confusion Matrix (SNR=-14)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

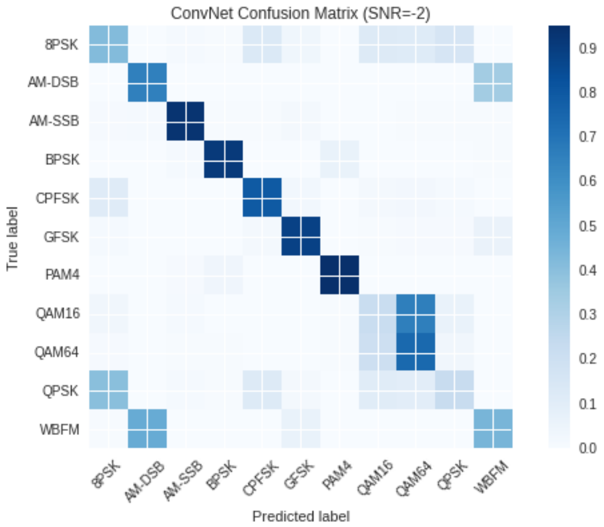

ConvNet Confusion Matrix (SNR=-12)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

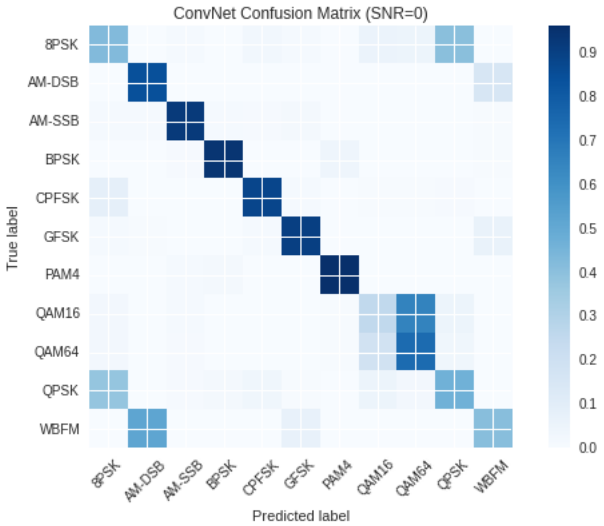

ConvNet Confusion Matrix (SNR=-10)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

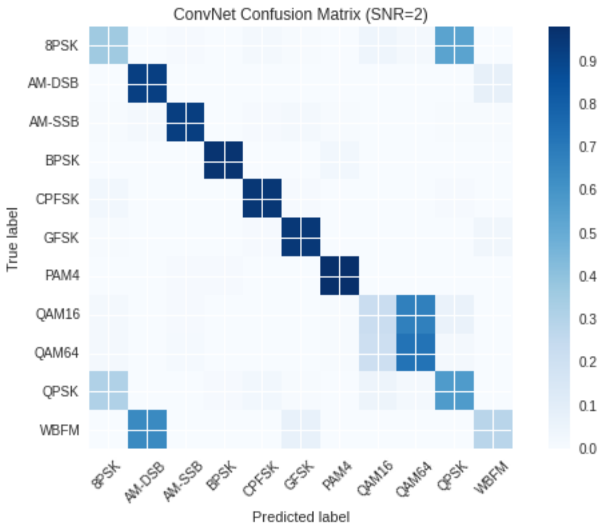

ConvNet Confusion Matrix (SNR=-8)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

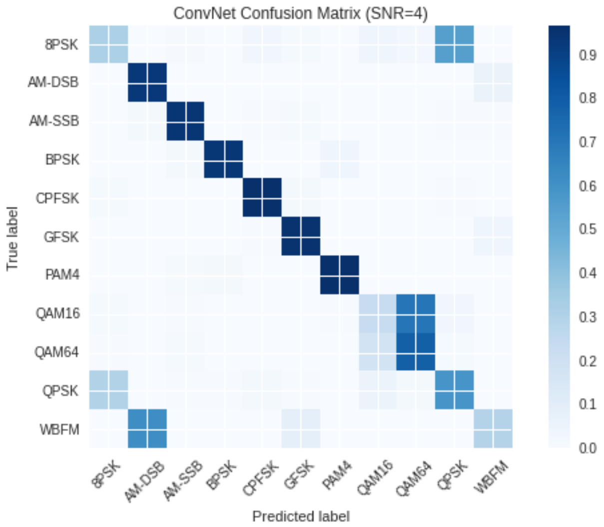

ConvNet Confusion Matrix (SNR=-6)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

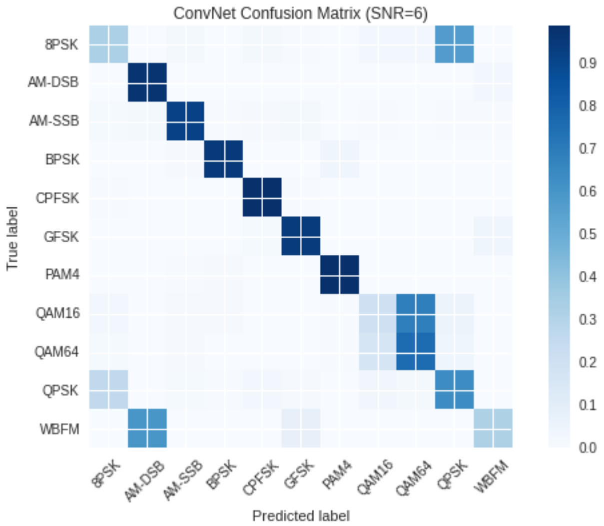

ConvNet Confusion Matrix (SNR=-4)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

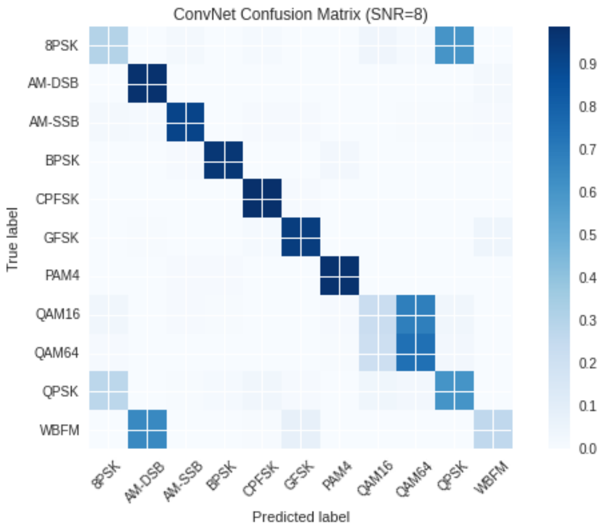

ConvNet Confusion Matrix (SNR=-2)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

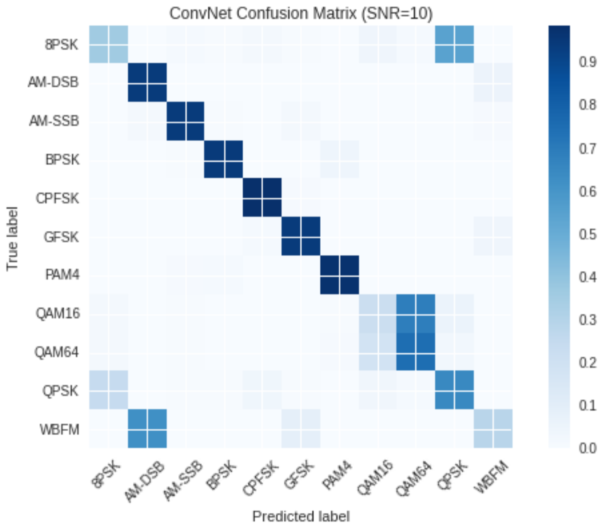

ConvNet Confusion Matrix (SNR=0)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


ConvNet Confusion Matrix (SNR=2)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

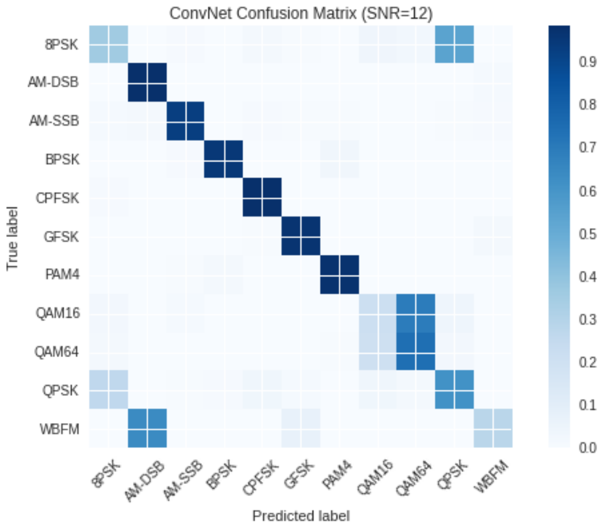

ConvNet Confusion Matrix (SNR=4)
True label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
Predicted label
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0


ConvNet Confusion Matrix (SNR=6)
True label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
Predicted label
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

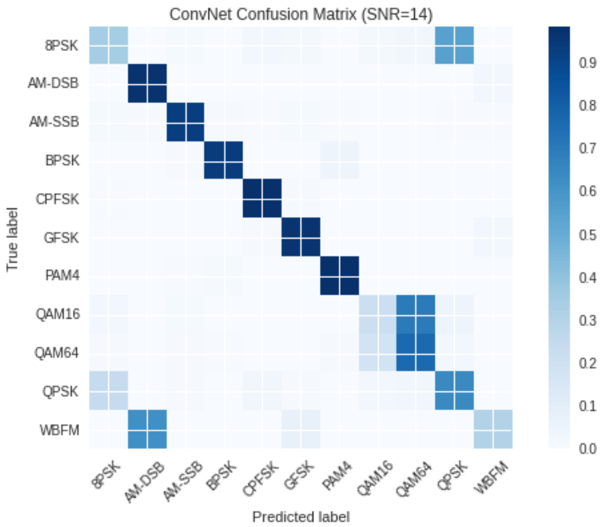

ConvNet Confusion Matrix (SNR=8)
True label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
Predicted label
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0



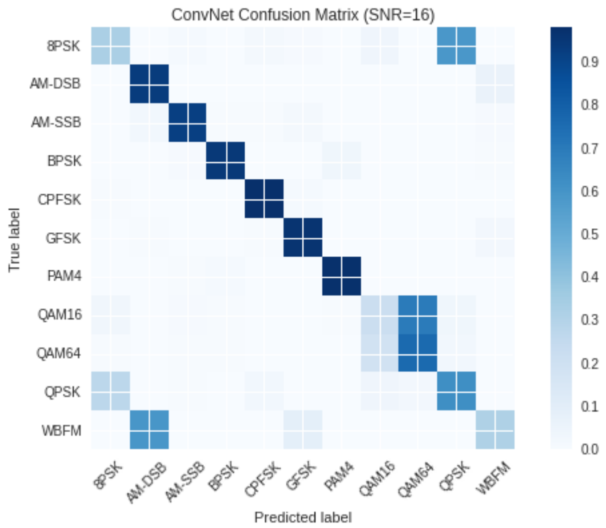
ConvNet Confusion Matrix (SNR=10)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

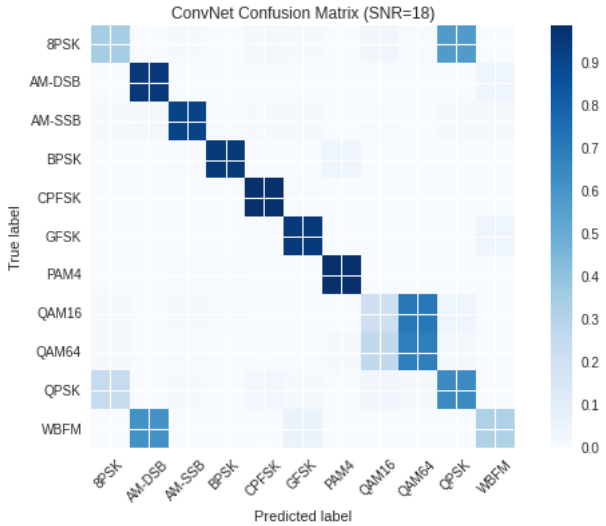


ConvNet Confusion Matrix (SNR=12)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0



ConvNet Confusion Matrix (SNR=14)
True label
Predicted label
8PSK
AM-DSB
AM-SSB
BPSK
CPFSK
GFSK
PAM4
QAM16
QAM64
QPSK
WBFM
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0





In [ ]:

```
# Save results to a pickle file for plotting later
print acc
fd = open('results_cnn2_d0.5.dat','wb')
cPickle.dump( ("CNN2", 0.5, acc) , fd )
```

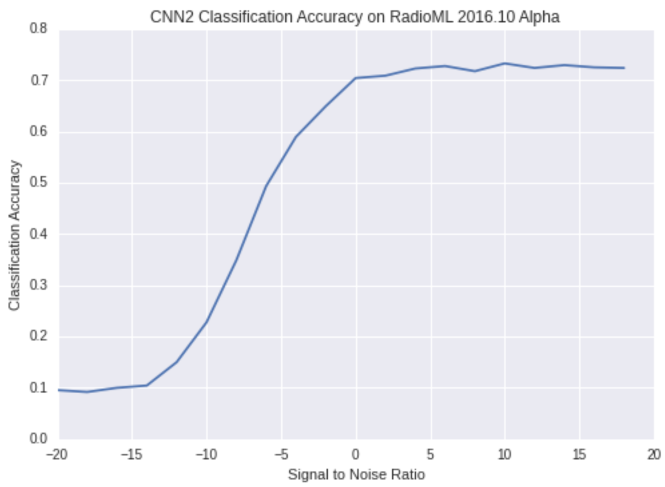
```
{0: 0.70419708029197081, 16: 0.72506787330316746, 2: 0.70883315158124316, 4: 0.72283406754772395, 6: 0.72761732851985561, 8: 0.71758340847
```

In [ ]:

```
# Plot accuracy curve
plt.plot(snrs, map(lambda x: acc[x], snrs))
plt.xlabel("Signal to Noise Ratio")
plt.ylabel("Classification Accuracy")
plt.title("CNN2 Classification Accuracy on RadioML 2016.10 Alpha")
```

Out [14]:

```
<matplotlib.text.Text at 0x7f089771fb10>
```



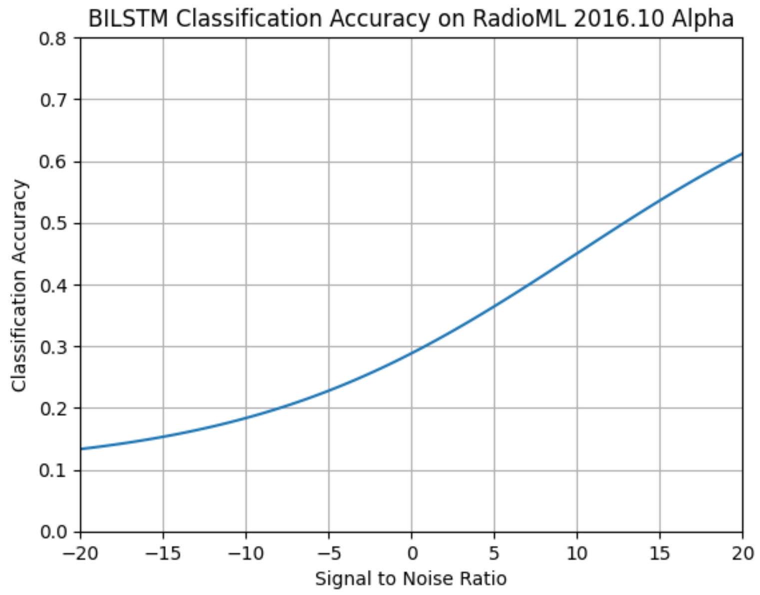
BILSTM code

In [ ]:

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Bidirectional, LSTM, Dense


# Define the model
dr=0.5
model = Sequential()
model.add(Reshape([1]+in_shp, input_shape=in_shp))
model.add(Bidirectional(LSTM(128, return_sequences=True), input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Bidirectional(LSTM(64)))
model.add(Dropout(dr))
model.add(Flatten())
model.add(Dense(256, activation='relu', init='he_normal', name="dense1"))
model.add(Dropout(dr))
model.add(Dense( len(classes), init='he_normal', name="dense2" ))
model.add(Activation('softmax'))
model.add(Reshape([len(classes)]))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()

nb_epoch = 100
batch_size = 1024
# Train the model
history = model.fit(X_train,
    Y_train,
    batch_size=batch_size,
    nb_epoch=nb_epoch,
    show_accuracy=False,
    verbose=2,
    validation_data=(X_test, Y_test),
    callbacks = [
        keras.callbacks.ModelCheckpoint(filepath, monitor='val_loss', verbose=0, save_best_only=True, mode='
        keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, verbose=0, mode='auto')
    ])

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss:.4f}")
print(f"Test Accuracy: {accuracy:.4f}")

# Make predictions
predictions = model.predict(X_test)
```

In [6]:

```
plt.plot(predictions, map(lambda x: acc[x], predictions))
plt.xlabel('Signal to Noise Ratio')
plt.ylabel('Classification Accuracy')
plt.title('BILSTM Classification Accuracy on RadioML 2016.10 Alpha')
plt.show()
```


BILSTM Classification Accuracy on RadioML 2016.10 Alpha
Classification Accuracy
Signal to Noise Ratio
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
−20
−15
−10
−5
0
5
10
15
20