Modulation Classification

Aly Swidan(3433), Mohamed Mahmoud(3431), Zeina elhayah Nada (3490)

April 2018

1 Architectures

1.1 Fully connected Network

We used a classical fully connected layer, however we added some bells and whistles to improve accuracy and speed of convergence. The thing that improved the accuracy substantially was using the BatchNormalization layer which normalizes the data by subtracting the mean and standardizing, while learning some parameters to allow the network to learn a normalization representation in between the un-normalized and the totally normalized. Figure 3 shows architecture we used as summarized by Keras.We used the graph in Figure **??** for evaluating the model. The results are summarized by the confusion matrix for the whole data which is shown in Figure 1, which shows that the networks confuses between 8PSK and QPSK which confirms with the findings of the provided paper, as for the 0 dB figure 2 shows the results, Finally 5 shows the snr vs accuracy.

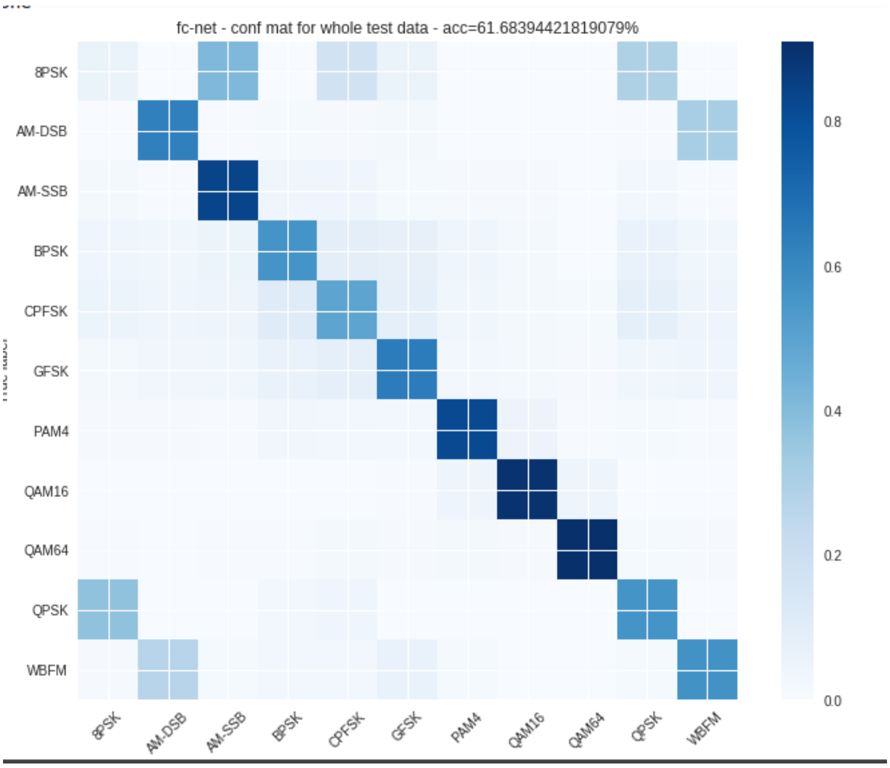


Figure 1: confusion matrix for the all the test data with an fc net

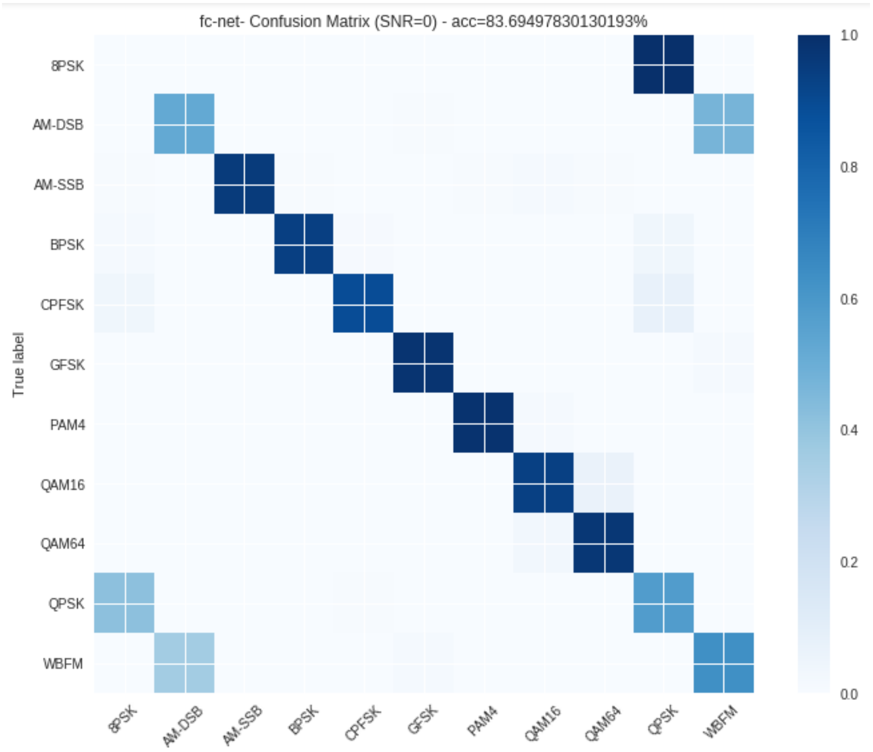
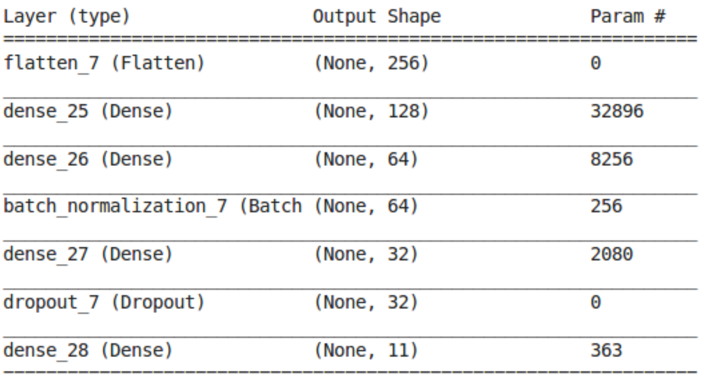


Figure 2: confusion matrix for the 0 dB part of the test datawith an fc net



```
Layer (type)                 Output Shape              Param #
=================================================================
flatten_7 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_25 (Dense)             (None, 128)               32896
_________________________________________________________________
dense_26 (Dense)             (None, 64)                8256
_________________________________________________________________
batch_normalization_7 (Batch (None, 64)                256
_________________________________________________________________
dense_27 (Dense)             (None, 32)                2080
_________________________________________________________________
dropout_7 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_28 (Dense)             (None, 11)                363
=================================================================
```

Figure 3: fc net architecuture

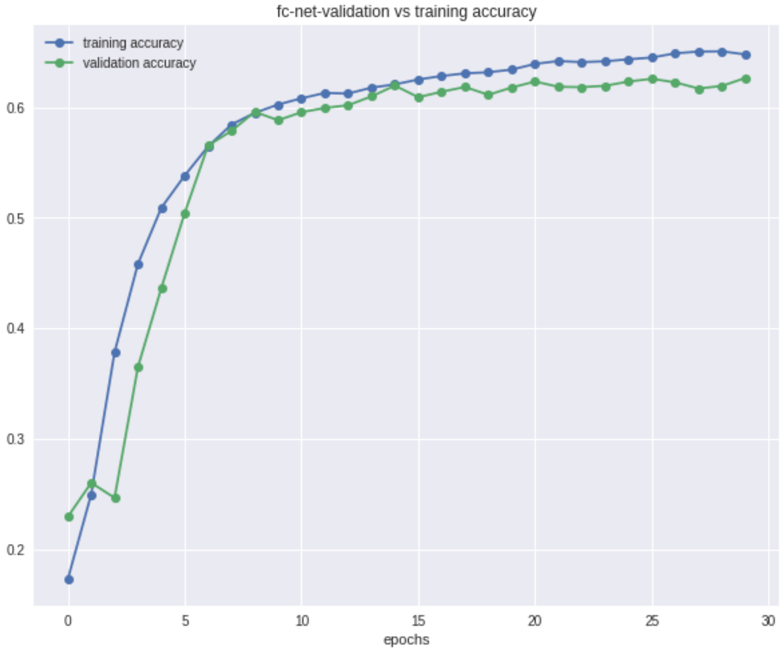


Figure 4: fc net accuracy

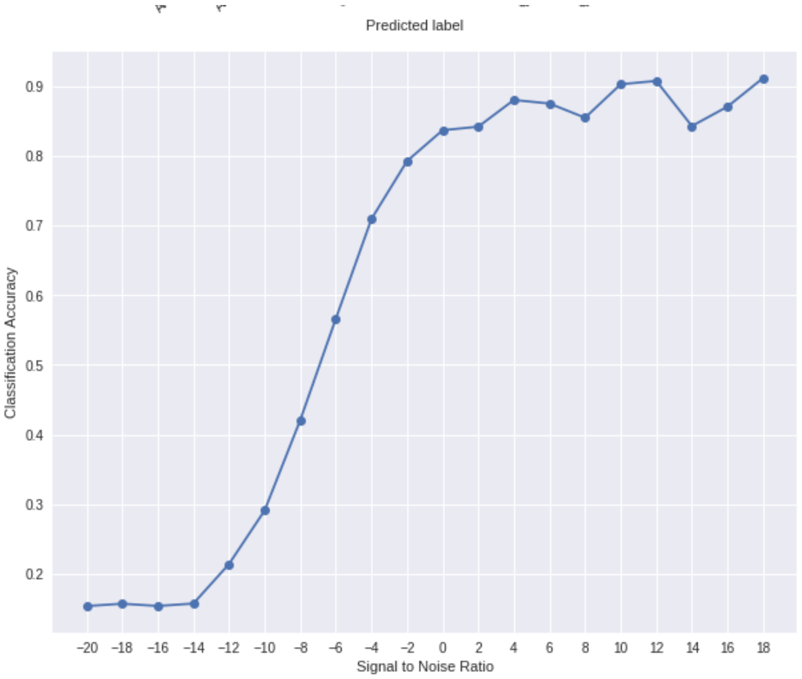
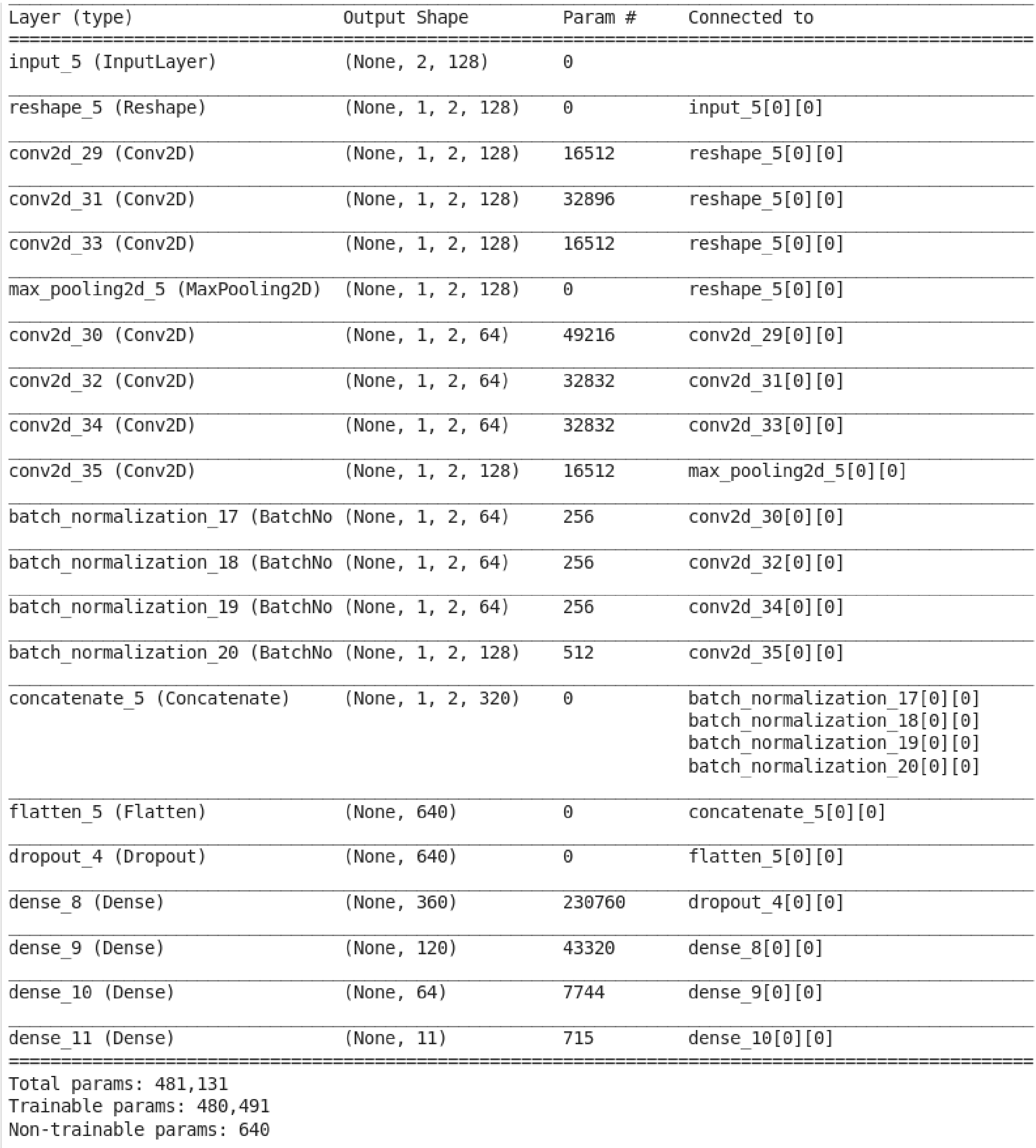


Figure 5: fc net snr vs acc

1.2 CNN

1.3 Inception

Inception modules are a typical convolutional neural networks (CNNs) that is made up of stacked convolutional layers in combination with max pooling and dropout. To sum it up we could say if we make a choice to either have a stack of 3x3 filters, or a stack of 5x5 filters or a max pooling layer. In general all of these are beneficial to the modelling power of the network. The inception module suggests the use of all of them.
Figure 6 shows architecture we used as summarised by keras.



```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_5 (InputLayer)            (None, 2, 128)       0
__________________________________________________________________________________________________
reshape_5 (Reshape)             (None, 1, 2, 128)    0           input_5[0][0]
__________________________________________________________________________________________________
conv2d_29 (Conv2D)              (None, 1, 2, 128)    16512       reshape_5[0][0]
__________________________________________________________________________________________________
conv2d_31 (Conv2D)              (None, 1, 2, 128)    32896       reshape_5[0][0]
__________________________________________________________________________________________________
conv2d_33 (Conv2D)              (None, 1, 2, 128)    16512       reshape_5[0][0]
__________________________________________________________________________________________________
max_pooling2d_5 (MaxPooling2D)  (None, 1, 2, 128)    0           reshape_5[0][0]
__________________________________________________________________________________________________
conv2d_30 (Conv2D)              (None, 1, 2, 64)     49216       conv2d_29[0][0]
__________________________________________________________________________________________________
conv2d_32 (Conv2D)              (None, 1, 2, 64)     32832       conv2d_31[0][0]
__________________________________________________________________________________________________
conv2d_34 (Conv2D)              (None, 1, 2, 64)     32832       conv2d_33[0][0]
__________________________________________________________________________________________________
conv2d_35 (Conv2D)              (None, 1, 2, 128)    16512       max_pooling2d_5[0][0]
__________________________________________________________________________________________________
batch_normalization_17 (BatchNo (None, 1, 2, 64)     256         conv2d_30[0][0]
__________________________________________________________________________________________________
batch_normalization_18 (BatchNo (None, 1, 2, 64)     256         conv2d_32[0][0]
__________________________________________________________________________________________________
batch_normalization_19 (BatchNo (None, 1, 2, 64)     256         conv2d_34[0][0]
__________________________________________________________________________________________________
batch_normalization_20 (BatchNo (None, 1, 2, 128)    512         conv2d_35[0][0]
__________________________________________________________________________________________________
concatenate_5 (Concatenate)     (None, 1, 2, 320)    0           batch_normalization_17[0][0]
                                                                 batch_normalization_18[0][0]
                                                                 batch_normalization_19[0][0]
                                                                 batch_normalization_20[0][0]
__________________________________________________________________________________________________
flatten_5 (Flatten)             (None, 640)          0           concatenate_5[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 640)          0           flatten_5[0][0]
__________________________________________________________________________________________________
dense_8 (Dense)                 (None, 360)          230760      dropout_4[0][0]
__________________________________________________________________________________________________
dense_9 (Dense)                 (None, 120)          43320       dense_8[0][0]
__________________________________________________________________________________________________
dense_10 (Dense)                (None, 64)           7744        dense_9[0][0]
__________________________________________________________________________________________________
dense_11 (Dense)                (None, 11)           715         dense_10[0][0]
==================================================================================================
Total params: 481,131
Trainable params: 480,491
Non-trainable params: 640
```

Figure 6: CNN Inception Architecuture

Figure 7 shows the plotting of accuracy comparison between training and validation accuracy.

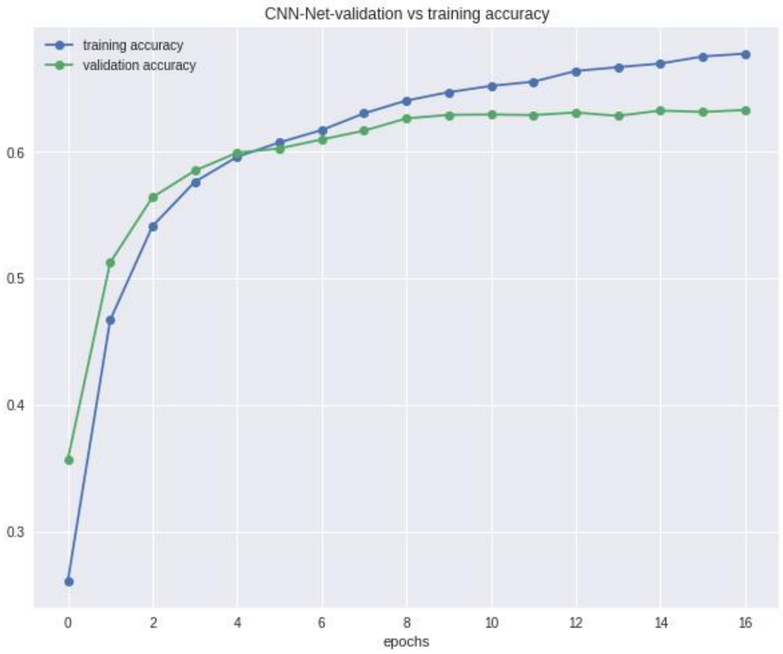


Figure 7: CNN Inception val vs training acc

Figure 8 shows the plotting of confusion matrix of the whole test data.

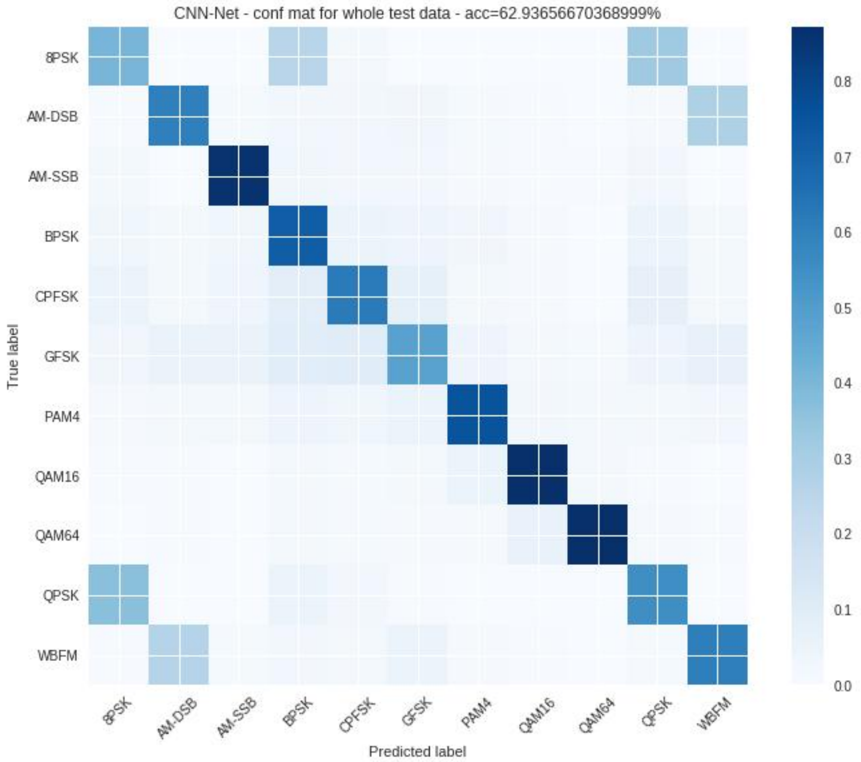


Figure 8: CNN Inception Confusion matrix and accuracy of whole test data

Figure 9 shows the plotting of confusion matrix of snr = 0

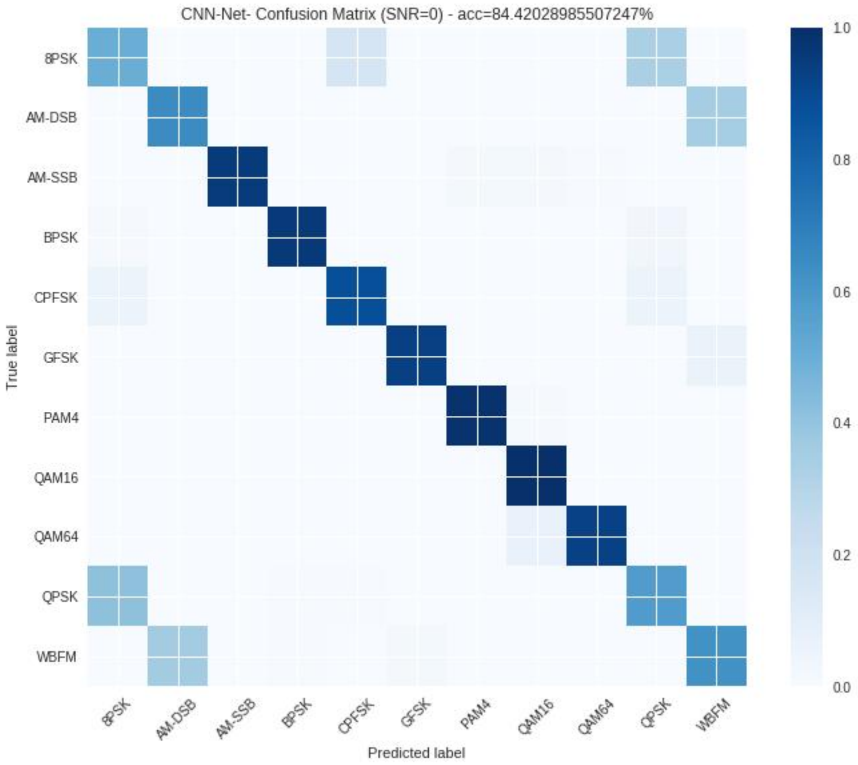


Figure 9: CNN Inception Confusion Matrix of SNR = 0

Figure 10 shows the plotting of accuracy against the SNR

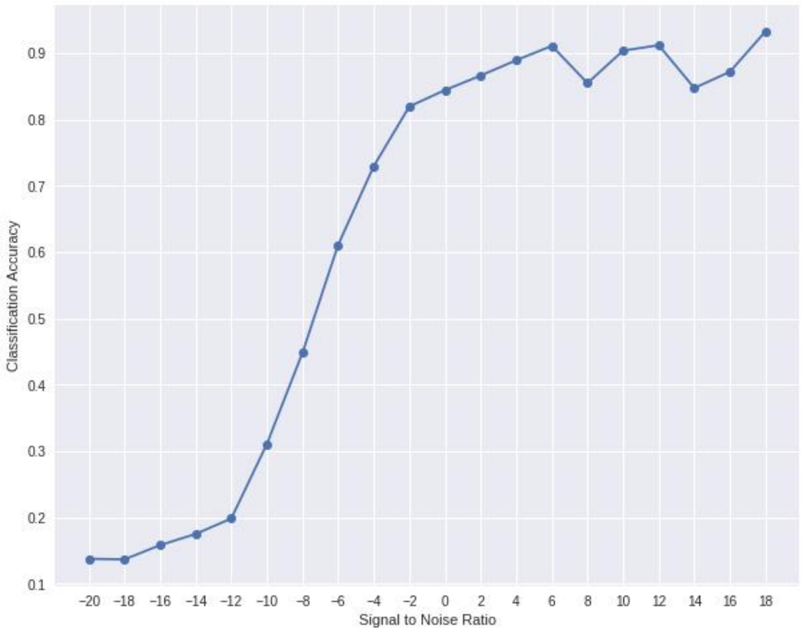


Figure 10: CNN Inception Confusion Matrix of SNR = 0

2 Feature

2.1 Derivative

Calculated derivative of each sample and created fully connected model and CNN (inception) model with these samples

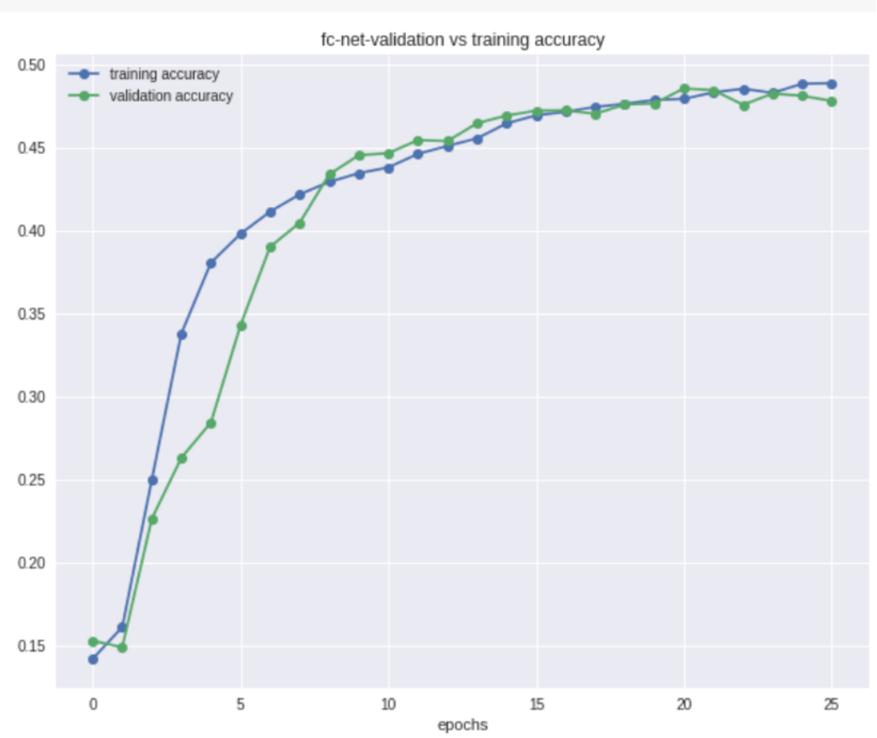


Figure 11: fully connected val vs training accuracy with derivative feature only

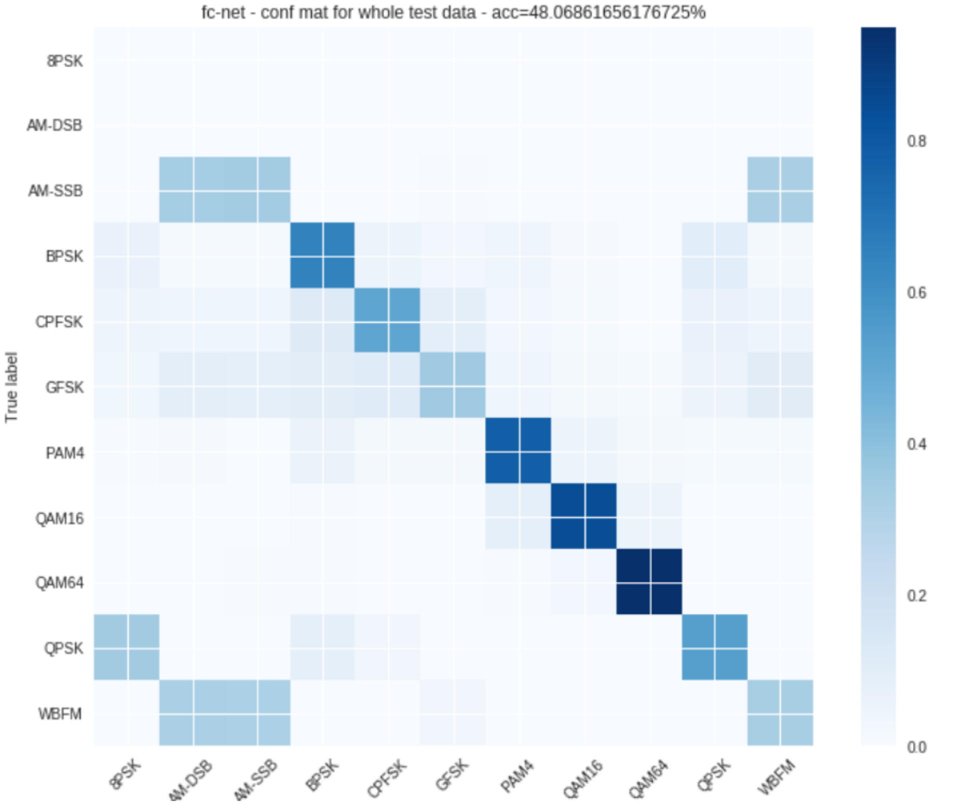


Figure 12: fully connected derivative feature only, Confusion matrix and accuracy of whole test data

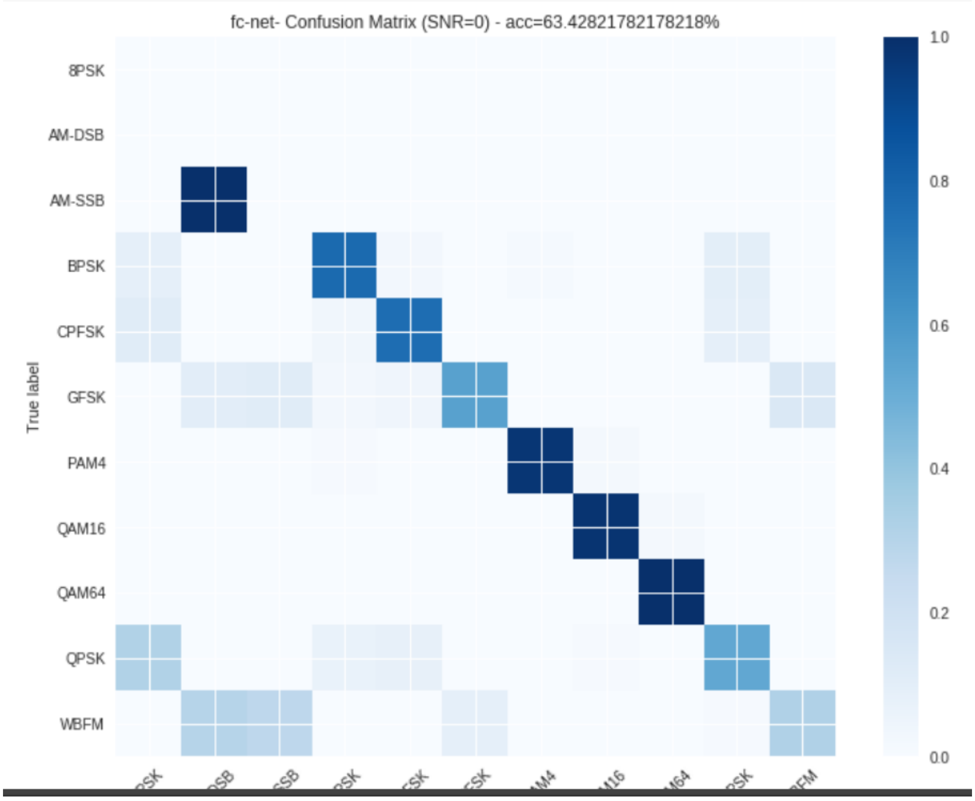


Figure 13: fully connected derivative feature only, Confusion Matrix of SNR = 0

2.2 FFT

Calculated Fourier transform of each sample and created fully connected model and CNN (inception) model with these samples

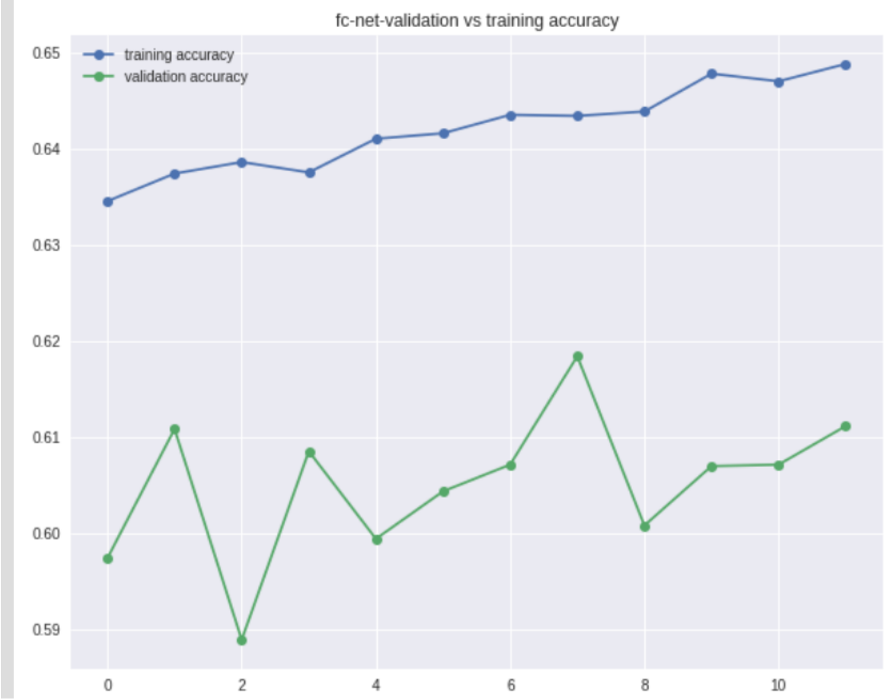


Figure 14: fully connected val vs training accuracy with fft feature only

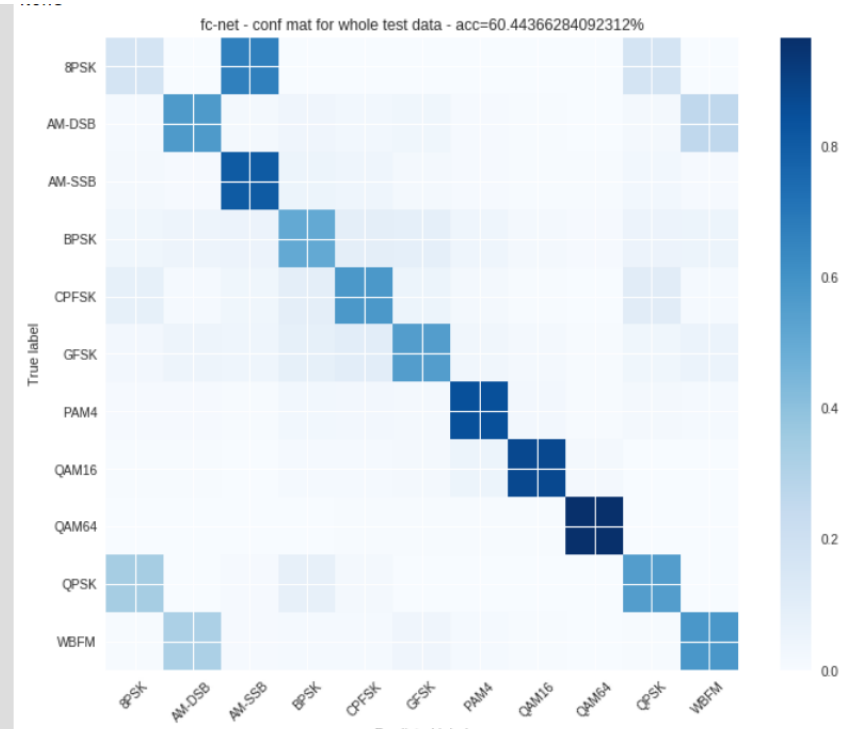


Figure 15: fully connected fft feature only, Confusion matrix and accuracy of whole test data

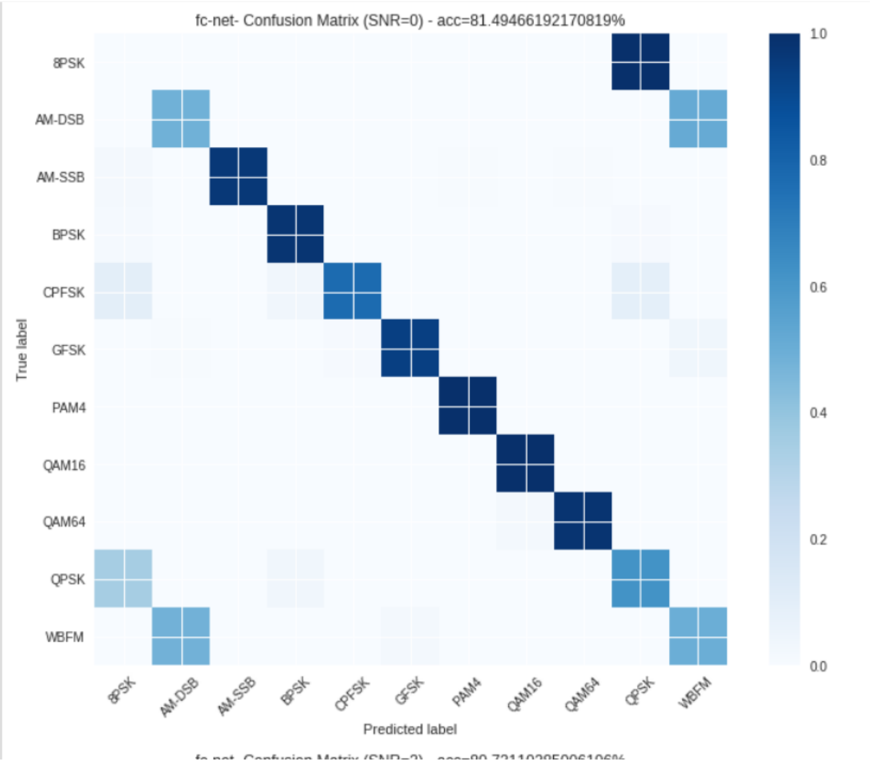


Figure 16: fully connected fft feature only, Confusion Matrix of SNR = 0

2.3 combined features

Combined above features with raw data and created fully connected model and CNN (inception) model with these samples

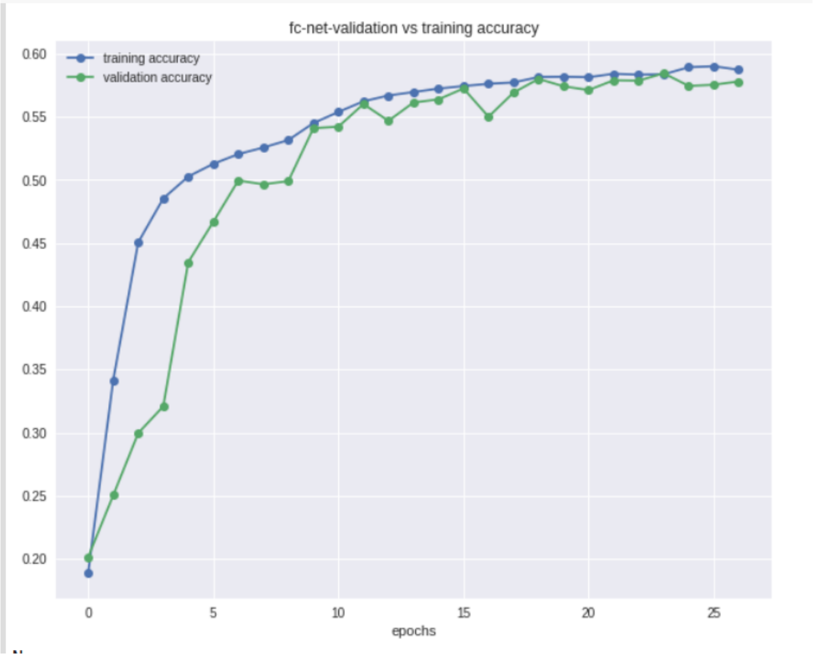


Figure 17: fully connected val vs training accuracy with all features

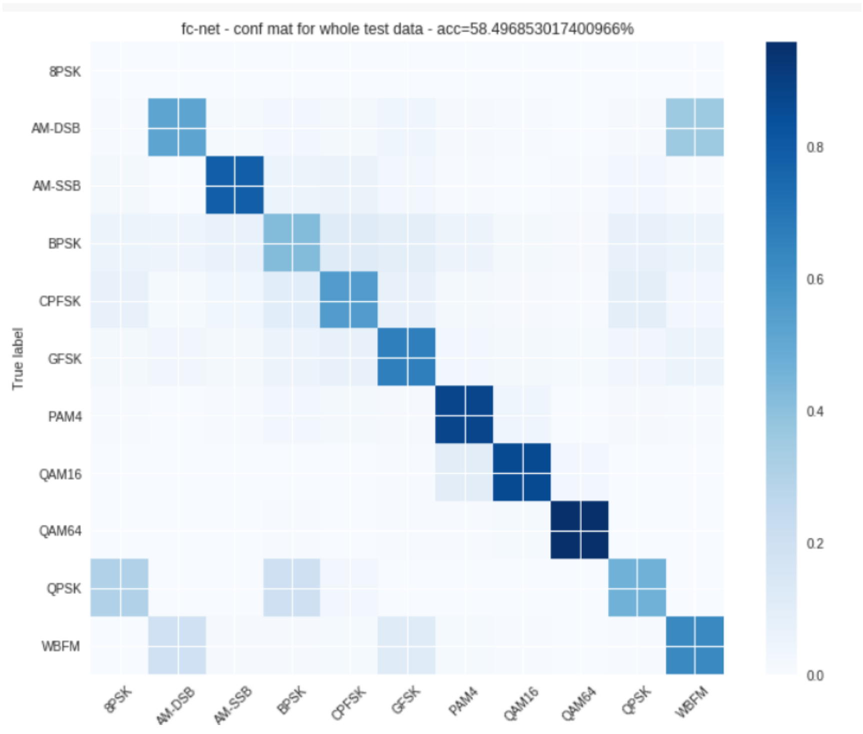


Figure 18: fully connected with all features, Confusion matrix and accuracy of whole test data

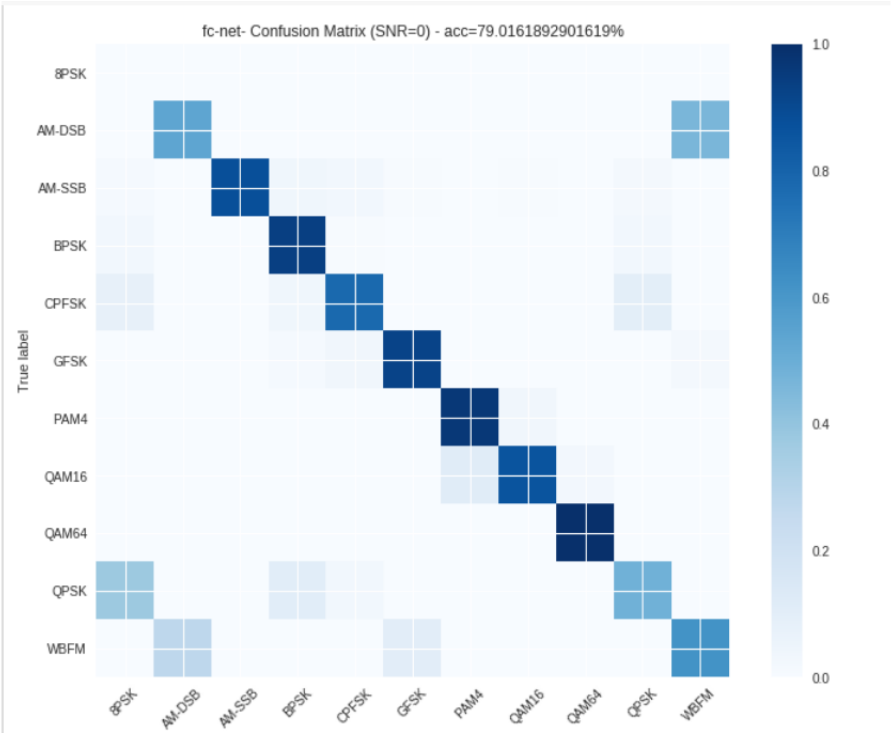


Figure 19: fully connected with all features, Confusion Matrix of SNR = 0

3 Conclusion

3.1 First Comparison

Comparison of Average Accuracies		
Features	Fully Connected	Inception Module (CNN)
Raw data	61.68	62.0
Derivative	48.07	15.22
Raw & FFT	60.44	60.63
Raw & Derivative & FFT	58.49	61.24

3.2 Second Comparison

Comparison of Accuracies at SNR = 0		
Features	Fully Connected	Inception Module (CNN)
Raw data	83.69	86.05
Derivative	63.43	15.62
Raw & FFT	81.49	83.28
Raw & Derivative & FFT	79.02	85.21

3.3 Third Comparison

Comparison of Best Accuracies		
Features	Fully Connected	Inception Module (CNN)
Raw data	93.57	93.1
Derivative	76.35	17.31
Raw & FFT	87.19	88.56
Raw & Derivative & FFT	82.04	92.19