# Deep Learning based Signal Modulation Identification in OFDM Systems

**Ahmet Gizik**
ag7739 – N13012289
Deep Learning / Final Project Report
Code is publicly accessible here: Github Link

## Introduction

The ability to classify signals is an important task that holds the opportunity for many different applications. Previously to classify the signal, we should decompose the signal using FT (Fourier Transform), SIFT, MFCC, or another handcrafting method using statistical modulation features. In the past five years, we have seen rapid disruption occurring based on the improved neural network architectures, algorithms, and optimization techniques collectively known as deep learning (DL). It turns out that state of the art deep learning methods can be applied to the same problem of signal classification and shows excellent results while completely avoiding the need for difficult handcrafted feature selection. Orthogonal frequency division multiplexing (OFDM) is a special kind of multi-carrier transmission technology when it comes to differentiating signal types. OFDM technology is widely applied to wideband communication systems with low complexity and is used in 4G and 5G wireless communications. It is widely predicted that OFDM will continue to be used in 6G communications. Signal Modulation Identification(SMI) plays a very important role in OFDM systems. In non-cooperative communication systems which are becoming more and more ubiquitous in various areas, the intermediate user devices do not cooperate and the data is transmitted directly to the destination. In the non-cooperative scenarios, types of the signal modulation are first required to demodulate and estimate so that the intercepted signal can be further decrypted. Hence, SMI is considered one of the most important techniques in designing non-cooperative communications systems. Therefore, the development of accurate SMI techniques is necessary for identification of OFDM signals. In the literature, feature extraction and machine learning classification algorithms are widely used in the modulation identification of OFDM signals. Due to inherent signal features of different modulation modes the recognition accuracy of traditional methods to detect the modulation scheme of OFDM signals is very limited. A proper deep learning-based method can improve SMI accuracy.

It also is known that convolutional neural networks (CNN) are used for image classification and recognition because of its high accuracy. it was proposed by computer scientist Yann LeCun in the late 90s, when he was inspired from the human visual perception of recognizing things. The CNN follows a hierarchical model which works on building a network, like a funnel, and finally gives out a fully-connected layer where all the neurons are connected to each other and the output is processed. Therefore the CNN can adapt the weights such that they extract different features from the input image. For example, one of the feature map may extract edges, and another one may extract circular shapes. This type of multi-level feature extraction allows CNN to classify images with high accuracy. For classification problems, the network is usually terminated with a fully connected layer, so it helps to reduce the dimensionality of the features maps as the network goes deep. In this work, under the affiliation of the known CNN abilities and OFDM signal's being an image like data type with the shape of 2xN, a CNN model is implemented, trained and tested to classify OFDM modulation types.

## Literature Survey

There has been some prior work into classifying OFDM modulations. Most of existing techniques on modulation identification are based on feature extraction and machine learning classification algorithms [1]. Lately, deep learning is also considered as one of effective methods for implementing SMI. In [2], authors propose a deep neural network (DNN) with an improved identification method. In [3], authors use a deep residual networks (ResNet) to identify the modulation modes of the signal and significantly reduced training time.

Authors propose an improved identification method with deep neural network (DNN) [4]. Another aproach here is using a long short-term memory (LSTM) neural network and a deep residual networks (ResNet) to identify the modulation modes of the signal and significantly reduced training time [5]. Author of the here combine genetic programming (GP) and KNN to accurately identify four modulation modes, i.e., binary phase shift keying (BPSK), quadrature phase shift keying (QPSK), 16 quadrature amplitude modulation

(16QAM) and quadrature amplitude modulation (64QAM) [6].

## Dataset Details

In this project, OFDM Modulation Classification data set [7] is used that consists of 6 modulated OFDM baseband signals with different header and payload modulations as BPSK+BPSK, BPSK+QPSK, BPSK+8PSK, QPSK+BPSK, QPSK+QPSK, QPSK+8PSK, respectively. Different header and payload modulations will make the pattern recognition harder and can provide a proper evaluation of the deep learning model. The data set also provides signals with a good SNR range so that the noise in the data differs signal by signal. The SNR range of each signal is from -10 dB to +20 dB at intervals of 2 dB. There are 2048 pieces of signals generated for each modulation type under one specific SNR level and each signal has 1024 I and 1024 Q samples. That is 6×16×2x2048 = 393216 I and Q samples in total. The input is like a 2xN image, with the first column containing 1024 I samples, and the second column containing 1024 Q samples. This way the input looks like an image with its spatial information preserved.

At the receiver, the time-domain signal can be written as below:

$$y(t) = exp(j\theta_t + j2\pi\epsilon t)\sum_{m=1}^{M} x(t - mT_s)h(m) + n(t)$$

where $\epsilon$ is the carrier frequency offset, $\theta_t$ is the phase offset, and $n(t)$ is the additive white Gaussian noise (AWGN). This block allows the user to simulate an AWGN noise source, a random walk process of carrier frequency drift and sampling rate offset drive. In the equation above $h(m)$ is a frequency-selective multipath channel function with m number of fading paths which is equal to 16 since the data varies in 16 different SNR levels(-10dB, -8dB, -6dB, -4dB, -2dB, 0dB, 2dB, 4dB, 6dB, 8dB, 10dB, 12dB, 14dB, 16dB, 18dB, 20dB). The author of the dataset used the equation above with the dynamic channel model block in the GNU Radio[8].

Due to GNU Radio's protocol-based encoding, each OFDM frame consists of two 64-byte preambles for synchronization, a 4-bytes header for recording the information of the current frame, a 50-byte payload for transmitting, and a 4-byte CRC correction code generated from the payload and the preamble modulation type is BPSK. Therefore GNU Radio's protocol-based coding makes the data, same as real OFDM coded signal and harder to predict due to added features that are not related to the modulation type of the encoded OFDM signal.

Before the classification with CNN, there was not any detection of the data parts as CRC, preamble, header, or payload which can be added to the model to improve the accuracy in the future. The classification of modulation types for header and payload has been done at the same time.

The data set was saved in multiple '.h5' type files as sequences(4194304 x 1 for each label) instead of matrixes with the shape 4096 x 1024 x2 per label. Moreover, the saved data for each label saved with the shape 4194304 x 1 and the given values are goes as 1024 I samples 1024 Q samples per signal. The data set does not have any label or SNR values, the author prepared it as 6 folders includes '.h5' type files for each SNR level so each folder with the name of the label had sixteen '.h5' type files. More specifically, there are six files with the names 'BPSK-8PSK', 'BPSK-BPSK', 'BPSK-QPSK', 'QPSK-8PSK', 'QPSK-BPSK', 'QPSK-QPSK' and each file has 16 '.h5' type files for every label type with the names '-10dB.h5', '-2dB.h5', '-4dB.h5', '-6dB.h5', '-8dB.h5', '0dB.h5', '10dB.h5', '12dB.h5', '14dB.h5', '16dB.h5', '18dB.h5', '20dB.h5', '2dB.h5', '4dB.h5', '6dB.h5', '8dB.h5'. I have labeled the data according to file names before training.

In many trials of mine at the beginning of the work, I have converted the dataset per label from the shape of 4194304 x 1 to the shape of 4194304 x 2 using the orthogonal property of the OFDM signal which was a silly mistake and took me several weeks to realize. Data was sequentially prepared and there was actually 2048 signals with 1024 I and 1024 Q samples.

## Model Details

Since the signal is sampled with 1024 elements, the input is a 2x1024 two-dimensional time-domain vector that contains I/Q modulated signal. One of the rows has the I samples, and another one has the Q samples. Therefore the input looks like an image with its spatial information preserved.

When the given input to CNN has zero mean and unit variance, the model has less validation loss/test loss and is more stabilized as it was stated in the paper of batch normalization [9]. Although batch normalization is suggested in the above citation, the model is trained faster and better after removing batch normalization layers. That shows that for this data it caused information loss.

Dropout is another regularization technique that is typically applied in neural networks to prevent overfitting. In the model, dropout operation is applied total of 6 times after every convolution and dense layer with a 0.5 dropping ratio. Therefore every 1 of 2 adaptive parameters, such as weights and biases are zeroed out randomly. Any dropped adaptive parameters do not affect the result and are not updated in the training iteration in which they are dropped. During the experiments, when the dropout is not applied the training accuracy isn't affected and keeps improving but the validation accuracy stops improving earlier than the model with dropout operation and cannot achieve the same accuracy level with the model in which dropout is applied. That implies that without dropout, the model overfits the data and gives poorer results.

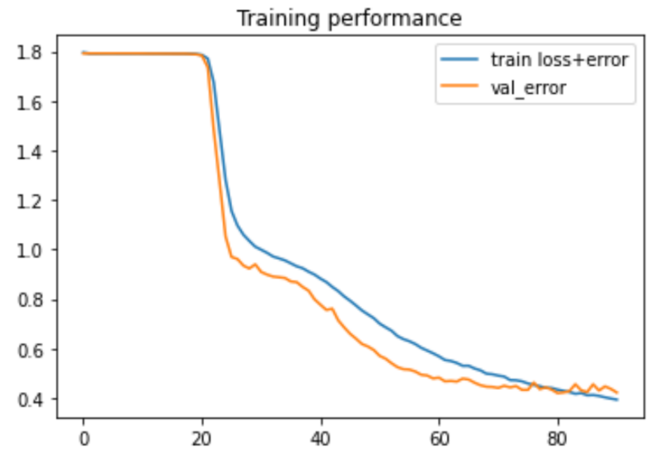
| **Layer** | **Output** |
|---|---|
| Input | 1x2×1024 |
| Conv2D(filter 128, size 1×8)+PReLU | 128x2x1024 |
| Dropout(0.5) | N/A |
| Conv2D(filter 64, size 1×4)+PReLU | 64x2x1024 |
| Dropout(0.5) | N/A |
| Conv2D(filter 32, size 1×2)+PReLU | 32x2x1024 |
| Dropout(0.5) | N/A |
| Flatten(0.5) | 65536 |
| Dense(256)+PReLU | 256 |
| Dropout(0.5) | N/A |
| Dense(128)+PReLU | 128 |
| Dropout(0.5) | N/A |
| Dense(64)+PReLU | 64 |
| Dropout(0.5) | N/A |
| Dense(modulation modes)+softmax | modes(6) |

Table 1: The network model and output dimensions

Parametric rectified linear unit (PReLU) is used as the activation function for all layers but the last fully connected layer so that gradient cannot go to zero and that prevents vanishing gradient problem. Moreover, when the calculation of the gradient error is done with PReLU, the number of computations can be less compared to other nonlinear activation functions Tanh and Sigmoid. Since this is a classification model, the softmax activation function is applied to the last fully connected layer so that there is a probability distribution matrix of the last layer. The categorical cross-entropy loss function is used which computes the cross-entropy loss between the labels and predictions.

After a few small changes the CNN model has 17,250,790 number of trainable parameters and trained with 16 different SNR levels.

For each of six different labels data set provides 2048 signals at 16 SNR levels. Total of 2048x16x6 which is 196,608 signals were ready for the training at the beginning. An early stopping was set in the case of stopping decrease in the validation loss for 10 epochs during the training. Early stopping is used because in many training sessions previously



has been done, the model did not increase its validation test performance after certain number of epochs such as 80, 82 and caused over fitting.

Right before the training data is split used with random state variable set to 42. Random state ensured that the splits that generated are reproducible. Random permutations generates the splits. The random state is used as a seed to the random number generator.

Training took 91 epochs with batch size of 512 samples and stopped due to the fact that early stopping was set. The categorical cross-entropy loss function is used which computes the cross-entropy loss between the labels and predictions which has the advantage of small computation, fast convergence, and accurate classification results. The model uses Adaptive Moment Estimation (Adam) optimizer with learning rate is set to 0.0007. Adam optimizer combines the momentum concept from SGD and adaptive learning from Ada delta optimizer. The memory requirement for Adam is low, it is also computationally efficient, straightforward to implement and it is much faster than SGD. The success metric was set to be accuracy.

Data is During the training sessions I have experienced several system crashes due to Colab's GPU limitations. Therefore model is saved during the training with saving best model method.**??**

## Results

As it shown in the Figure 2, model gives 0.92 accuracy ratio for whole SNR levels and as it can be seen from confusion matrixes accuracy ratios are higher for higher SNR levels. That is as expected since signals with higher SNR levels are less distorted. In Figure 2 confusion matrix for whole SNR levels shows that the predictions for the label BPSK-8PSK achived highest score with 0.96, labels BPSK-BPSK, BPSK-QPSK and QPSK-8PSK prediction accuracies are measured as 0.92. Modulation type QPSK-8PSK is predicted with 0.89 accuracy. Lastly modulation type QPSK-QPSK is predicted with 0.93 accuracy. Here the signals are modulated different

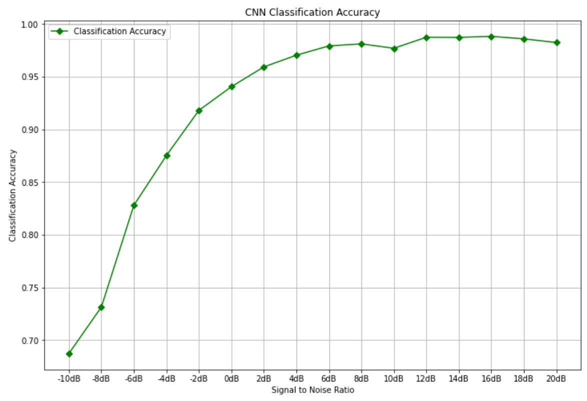


Figure 1: CNN Classification Accuracy

at the header and information parts and signals with very low SNR levels such as -10dB and -8dB are fed to predict. Therefore results show the power of the CNN when it comes to classification and CNNs can definitely be used for blind classification of signal when they are trained well.

The obvious drawback to using the CNN model is huge number of weights that is used(17,250,790). The biggest number of weights is between the third convolution layer(65536) and the first fully connected layer. It should be possible to reduce the number of weights, by adding more sub sampling layers like max pooling, without losing the classification accuracy. This is something worth investigating in future. On the other hand, the prediction accuracy in this work is measured frame by frame, but in a real blind application there most likely be an continuous signal and many frames with same modulation type. SNR also doesn't change much in short duration which also makes classification easier. Therefore even with a much smaller CNN model(much less parameters) modulation classification can be detected.

There are possible future works such that before the classification with CNN, there is no detection of the data parts as CRC, preamble, header, or payload which can be added to the model to improve the accuracy in the future. Moreover, the classification of modulation types for header and payload can be done at separate times to improve the model.

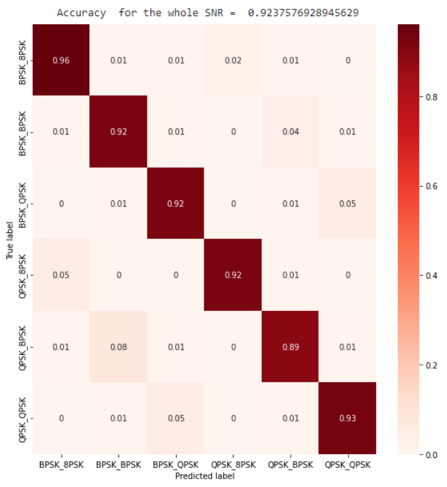


Figure 2: Whole SNR confusion matrix

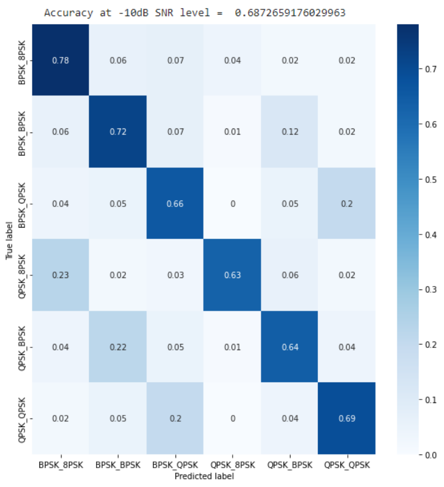


Figure 3: -10dB SNR

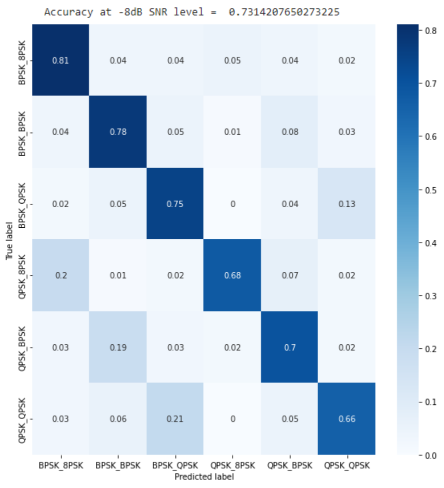


Figure 4: -8dB SNR

## References

[1] Y. Sun et al. "Template Matching Based Method for Intelligent Invoice Information Identification". In: *IEEE Access* 7 (2019), pp. 28392–28401.

[2] W. Xie et al. "Deep Learning in Digital Modulation Recognition Using High Order Cumulants". In: *IEEE Access* 7 (2019), pp. 63760–63766.

[3] Y. Wang et al. "Data-Driven Deep Learning for Automatic Modulation Recognition in Cognitive Radios". In: *IEEE Trans. Veh. Technol.* 68 (2019), pp. 4074–4077.

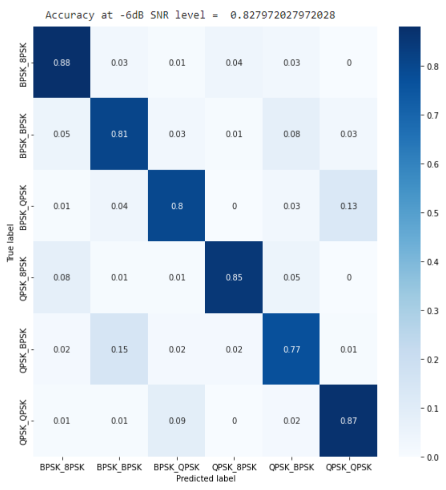


Figure 5: -6dB SNR

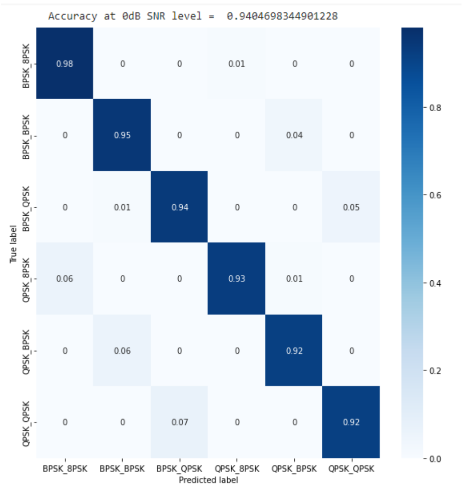


Figure 8: 0dB SNR

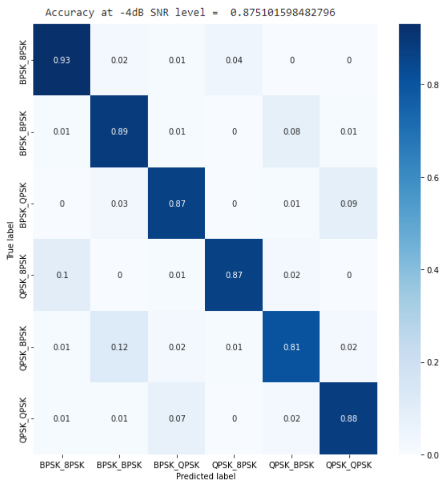


Figure 6: -4dB SNR

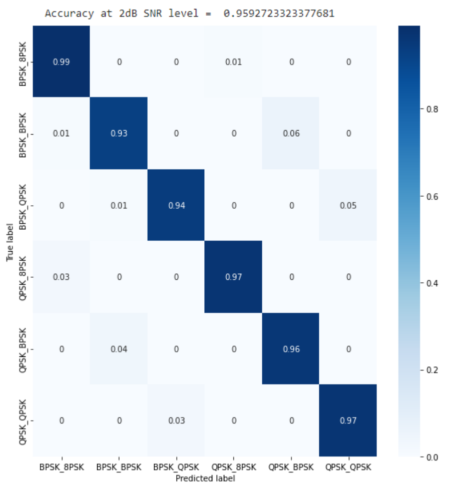


Figure 9: 2dB SNR

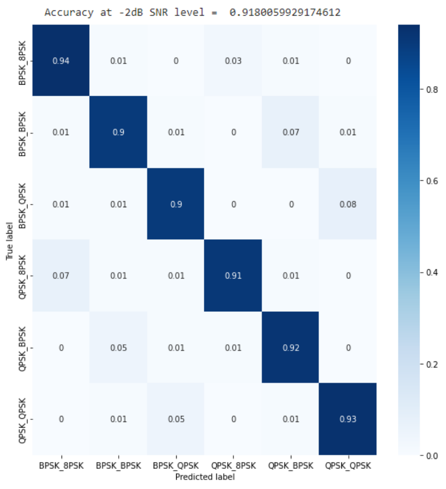


Figure 7: -2dB SNR

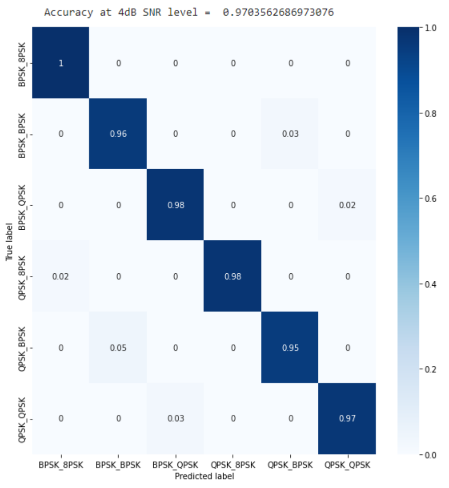


Figure 10: 4dB SNR

Accuracy at 6dB SNR level = 0.9791779340183883

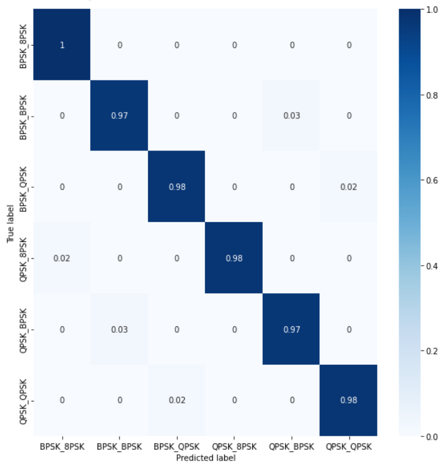
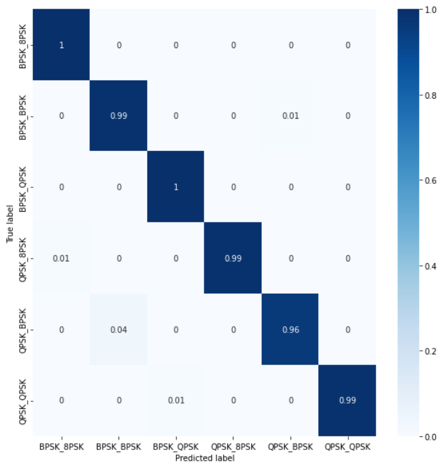


Figure 11: 6dB SNR

Accuracy at 14dB SNR level = 0.987292817679558



Accuracy at 16dB SNR level = 0.9882449425915801

Figure 14: 14dB SNR

Accuracy at 8dB SNR level = 0.9811170212765957

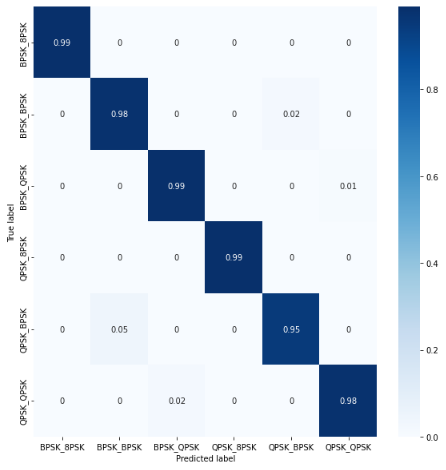


Figure 12: 8dB SNR

Accuracy at 16dB SNR level = 0.9882449425915801

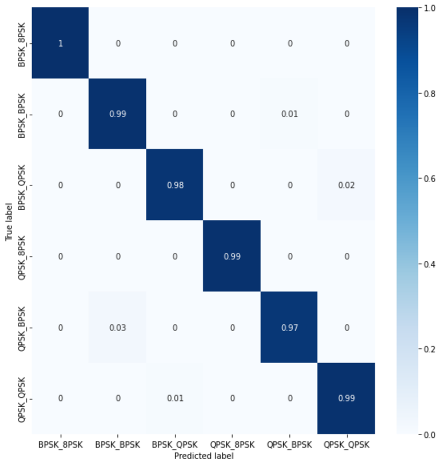


Figure 15: 16dB SNR

Accuracy at 10dB SNR level = 0.9769484083424808

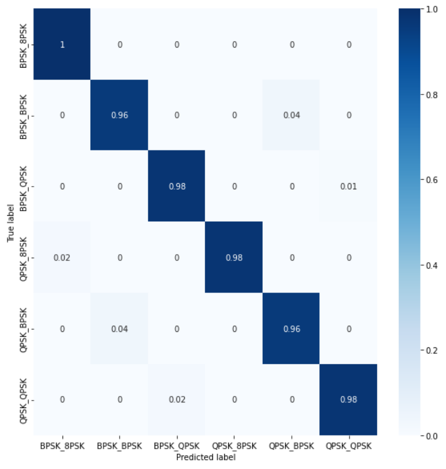


Figure 13: 10dB SNR

Accuracy at 18dB SNR level = 0.9858892438764644

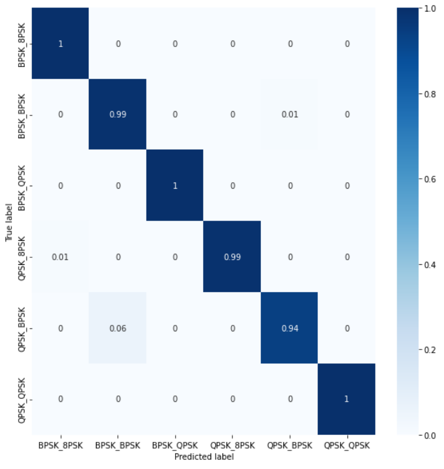


Figure 16: 18dB SNR

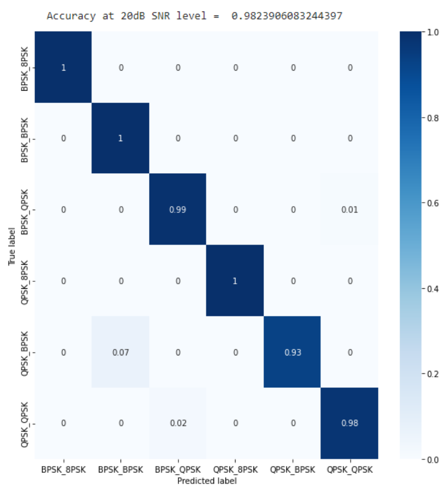


Figure 17: 20dB SNR

[4] WENWU XIE et al. “Deep Learning in Digital Modulation Recognition Using High Order Cumulants”. In: *IEEE Access* abs/1502.03167 (2019). URL: `https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8715390`.

[5] Yu Wang et al. “Data-Driven Deep Learning for Automatic Modulation Recognition in Cognitive Radios”. In: *IEEE Transactions on Vehicular Technology* 68.4 (2019), pp. 4074–4077. DOI: `10.1109/TVT.2019.2900460`.

[6] Muhammad Waqar Aslam, Zhechen Zhu, and Asoke Kumar Nandi. “Automatic Modulation Classification Using Combination of Genetic Programming and KNN”. In: *IEEE Transactions on Wireless Communications* 11.8 (2012), pp. 2742–2750. DOI: `10.1109/TWC.2012.060412.110460`.

[7] Chong Lin. *OFDM MODULATION CLASSIFICATION DATASET*. `https://ieee-dataport.org/open-access/ofdm-modulation-classification-dataset`. Accessed on 2021-11-24. Mar. 2021.

[8] *Dynamic Channel Model*. `https://wiki.gnuradio.org/index.php/Dynamic_Channel_Model`. Accessed on 2021-11-24.

[9] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *CoRR* abs/1502.03167 (2015). arXiv: `1502.03167`. URL: `http://arxiv.org/abs/1502.03167`.