Automatic Modulation Classification using ANN and CNN

Tarannum Nisha, *Electrical and Computer Engineering, UBC*

Abstract

Neural networks have received a lot of attention in the recent years due to its superior performance in classifying data with complex structure. In this paper, the feasibility and effectiveness of applying deep learning algorithms for automatic recognition of modulation type of the received signals is investigated. The test results show that statistical features extracted from input signal data significantly improve the performance of artificial neural networks (ANNs). Nonetheless, convolutional neural networks (CNNs) have proven to achieve performance that outperforms the expert-based approaches. Moreover, in order to reduce the training time, principal component analysis (PCA) method is applied to decrease the dimension of the training data set. Finally, the relationship between reducing the dimension of training data set and loss in classification accuracy is investigated by identifying the representative SNR values for both ANN and CNN architectures.

Index Terms

Machine learning, automatic modulation classification, artificial neural networks, convolutional networks.

I. Introduction

AUTOMATIC modulation classification (AMC) plays a critical role in the modern day wireless communication. It is a technique used in wireless communications to determine the sender's modulation format from the received signal without the knowledge of device parameters **[1]**. AMC finds its applications in many civil and military purposes, such as cognitive radio, adaptive communication, and electronic reconnaissance. Analog modulations (e.g.,AM and FM) and digital modulations (e.g., FSK and PSK) both use smoothly varying signals; the difference being that an analog-modulated signal is demodulated into an analog baseband waveform, while a digitally modulated signal is made up of distinct modulation units known as symbols that are interpreted as digital data. Furthermore hardware specifications and bandwidth allocations vary depending on the modulation. Now in order to demodulate the transmitted symbol, the receiver should know the modulation format. Although both the transmitting and receiving ends are aware of the pool of modulation types, the modulation type chosen is adaptive and may not be recognised at the receiving end. As a result, an automatic modulation classification mechanism is required at the receiving end to ensure that the correct demodulation method is selected and that the message is successfully recovered.

The following are some of the characteristics of a good modulation classifier: (i) a modulation classifier should be able to distinguish as many different forms of modulation as possible, i.e., be versatile, (ii) it should have a high accuracy of classification relative to various levels of noise, (iii) should be robust against hindrances like noise, signal fading etc to produce reliable results, i.e., be resilient and (iv) be computationally efficient since used in several real-time applications **[2]**.

Several AMC schemes have been researched for reducing training time by reducing the size of the training dataset while maintaining classification accuracy up to decent levels **[3]**. Principal component analysis (PCA) is one such method that significantly reduces training time while resulting in information loss due to compression. Even so it results in decreased classification accuracy the overall computational complexity and memory resource requirements are also reduced to a great extent in complex architectures. So far various feature-based AMC schemes have been proposed that include decision tree method which uses the cumulants and cyclic moments of time domain signal waveforms and support vector machines (SVMs). However, these methods suffer from performance degradation in real-field environments due several factors such as channel fading, noise etc. To deal with this drawback, recent deep neural network (DNN)-based AMC technique takes advantage of DNN's excellent classification accuracy **[1]**.

In this paper, appropriate ANN and CNN are designed for modulation classification task to perform quantitative comparison. PCA is applied for reduction in the dimensions of training dataset and to speed up the training. Although both architectures provide good results, factors such as run-time, accuracy, and efficiency at low SNR values are compared to determine which classifier is better for this task. The ANN is trained using selected features **[1]** derived from the high-dimensional inputs, whereas CNN uses temporal learning with deep convolutional neural networks **[4]**. The use of softmax layer at the output layer of both the classifiers, produces the probability of each modulation class at each node. Several factors such as dropout, batch size and input data dimensions are compared for accuracy and performance of the ANN and CNN models.

II. Dataset

RADIOML 2016.10A dataset prepared by Deep-Sig is used for training and testing of both ANN and CNN **[5]**. It is important and valuable to train with real data to achieve a reliable wireless communication. The dataset consists of synthetically generated radio signals in a real system, with well characterized transmit parameters identical to a real world signal. The dataset consists

of 8 digital and 3 analog modulation which sums up to 11 modulation schemes in all which are widely used in wireless communications systems. These consist of BPSK, QPSK, 8PSK, 16QAM, 64QAM, BFSK, CPFSK, and PAM4 for digital modulations, and WBFM, AM-SSB, and AM-DSB for analog modulations. The dataset has the size 220,000 x 2 x 128, consisting of 220,000 entries, each with an array of size 2 x 128. Each array represents the samples of about 128 μs of a received waveform. The samples of the signal waveforms are complex-valued, and hence they have been stored as real and imaginary parts, and therefore we have arrays of size 2 x 128 in the data set. The labels of the downloaded dataset contain two parameters: the modulation technique used from one of the 11 possible modulation techniques, and the signal-to-noise ratio (SNR) value (one of [-20;-18;-16;-14;-12;-10;-8;-6;-4;-2; 0; 2; 4; 6; 8; 10; 12; 14; 16; 18], so 20 possible SNR values).

III. Principal Component Analysis

Long training times in deep learning is the key barrier to using deep neural networks in real-time, where the network architecture must adjust to changing environmental conditions via online training. Machine learning algorithms would need to be trained regularly in the next generation of communication systems in order to handle new environmental changes. As a result, shortening the training time is critical for implementing these algorithms in real time. By far the most common dimensionality reduction algorithm is Principal Component Analysis (PCA) that can decrease the training time significantlty and lower the complexity of trained machine. This not only saves computational resources but also time for training the model.

PCA determines the hyperplane that is nearest to the data and then projects the data onto it. We use scikit-learn's PCA class that uses SVD decomposition to implement PCA. It automatically takes care of centering the data and the training set takes up far less space after dimensionality reduction. When PCA is applied to a dataset while keeping 95% of its variance as an example instead of the original say 784 features, we find that each instance has just over 150 features. While the majority of the variance has been retained, the dataset has shrunk to less than 20% of its original scale. Following this compression ratio in our case, it is observed that the speed of classification algorithm increased tremendously.

IV. Artificial neural networks

The dataset was trained against several hyperparameter configurations of neural network. A dense neural network was designed and experimented to reflect on the performance and select the best model configurations for modulation classification. Comparison is made in terms of overall accuracy at different SNR levels. 110,000 samples are used for both training and validation, and another 110,000 samples are used for testing the performance of the trained machine, as it has been done in the implementation provided by DeepSig. In order to prevent over-fitting, early stopping and dropout has been used. Training is conducted using a categorical cross entropy loss function and using the famous Adam optimizer. The network was implemented and trained in Keras running on top of TensorFlow on an Intel Core i7-10510U CPU @ 1.80GHz and 16 GB RAM without any GPUs.

A. Features of ANN

This subsection summarizes the features used to train the designed ANN. The features are mainly implemented in **[1]** and gave good results for more than half of the SNR values. The features are selected such that the empirical distributions of each modulation class are properly separated. We use the same set of features to classify 11 modulation types for our case. The modulated baseband signal is represented as a complex number a[i] = $a_I[n]$ + j$a_Q[n]$. The first feature is the ratio of in-phase component and quadrature component signal power given by:

$$F_1 = \frac{\sum_n a_Q^2[n]}{\sum_n a_I^2[n]}, \tag{1}$$

where $a_I[n]$ and $a_Q[n]$ are the in-phase and quadrature samples for the complex baseband signal. The second feature is standard deviation of direct instantaneous phase represented as:

$$F_2 = \sqrt{\frac{1}{c}\Big(\sum_{a_n[i]>a_t} \varphi_{NL}^2[i]\Big) - \Big(\frac{1}{c}\sum_{a_n[i]>a_t} \varphi_{NL}[i]\Big)^2}, \tag{2}$$

where $a[i] = \frac{a[i]}{mean(a[i])}$, $\varphi_{NL}[i])$ is the instantaneous phase at time instant $t = T_s i$, c is the total number of samples and a_t is the threshold. The third feature is the standard deviation of the absolute value of the non-linear component of the instantaneous phase

$$F_3 = \sqrt{\frac{1}{c}\Big(\sum_{a_n[i]>a_t} \varphi_{NL}^2[i]\Big) - \Big(\frac{1}{c}\sum_{a_n[i]>a_t} |\varphi_{NL}[i]|\Big)^2}, \tag{3}$$

The fourth feature is standard deviation of absolute value of normalized instantaneous amplitude of simulated signal

$$F_4 = \sqrt{\frac{1}{N}\Big(\sum_{i=1}^{N} a_{cn}^2[i]\Big) - \Big(\frac{1}{N}\sum_{i=1}^{N}|a_{cn}[i]|\Big)^2}, \tag{4}$$

where $a_{cn}[i] = \frac{a[i]}{mean(a[i])} - 1$ and N is the number of samples for $a_{cn}[i]$. The fifth feature is standard deviation of the absolute normalized centered instantaneous frequency for the signal segment

$$F_5 = \sqrt{\frac{1}{N_s}\Big[\sum_{i=1}^{N_s} f_N^2(i)\Big] - \left[\frac{1}{N_s}\sum_{i=1}^{N_s}|f_N(i)|\right]^2}, \tag{5}$$

where $f_N(i)$ is the centered instantaneous frequency. The sixth feature is given by:

$$F_6 = \sigma_v = \sqrt{\frac{1}{N}\Big(\sum_{i=1}^{N} a_v[i]^2\Big) - \frac{1}{N}\Big(\sum_{i=1}^{N}|a_v[i]|\Big)^2}, \tag{6}$$

where a_v is the normalized signal amplitude, i.e., $\sqrt{\frac{a[i]}{var(a[i])}} - 1$. The seventh feature is the mixed order moments expressed as:

$$F_7 = \frac{M_{4,2}(y)}{M_{2,1}^2(y)} = \frac{E(|a[i]|^4)}{E(|a[i]|^2)}, \tag{7}$$

Mean value of the signal magnitude is the eighth feature

$$F_8 = \frac{1}{N}\sum_{n=1}^{N}|a[i]|, \tag{8}$$

where $a[i]$ is the instantaneous amplitude. The ninth feature is the normalized square root value of sum of amplitude of signal samples

$$F_9 = \frac{\sqrt{\sum_{i=1}^{N}|a[i]|}}{N}, \tag{9}$$

and the maximum value of power spectral density (PSD) of the normalized signal samples is the tenth feature

$$F_{10} = \frac{1}{N}\max|DFT(a_{cn}[i])|^2, \tag{10}$$

Features eleven to seventeen are cumulants calculated using the equations below:

$$F_{11} = E[a^2[n]], \tag{11}$$

$$F_{12} = E[|a[n]|^2], \tag{12}$$

$$F_{13} = M_{41} - 3M_{20}^2, \tag{13}$$

$$F_{14} = M_{41} - 3M_{20}M_{21}, \tag{14}$$

$$F_{15} = M_{42} - |M_{20}|^2 - 2M_{21}^2, \tag{15}$$

$$F_{16} = M_{63} - 6M_{20}M_{40} - 9M_{42}M_{21} + 18M_{20}^2M_{21} + 12M_{21}, \tag{16}$$

$$F_{17} = M_{80} - 35M_{40}^2 - 28M_{60}M_{20} + 420M_{40} - 630M_{20}^4, \tag{17}$$

where $M_{p+q,p}$ denotes $E\big[a[n]^p a[n]^{*q}\big]$. The eighteenth feature measures the tailedness of the distribution while the nineteenth feature represents the skewness, quantifying the asymmetry of the distribution

$$F_{18} = \left|\frac{E\big[(a - E[a])^4\big]}{E\big[(a - E[a])^2\big]^2}\right|, \tag{18}$$

$$F_{19} = \left|\frac{E\big[(a - E[a])^3\big]}{E\big[(a - E[a])^2\big]^{3/2}}\right|, \tag{19}$$

Peak-to-rms ratio PR and peak-to-average ratio PA are also used as features

$$PR = F_{20} = \frac{\max|a|^2}{\frac{1}{N}\sum_{i=1}^{N}(a[i])^2}, \tag{20}$$

$$PA = F_{21} = \frac{\max|a|}{\frac{1}{N}\sum_{i=1}^{N}a[i]}, \tag{21}$$

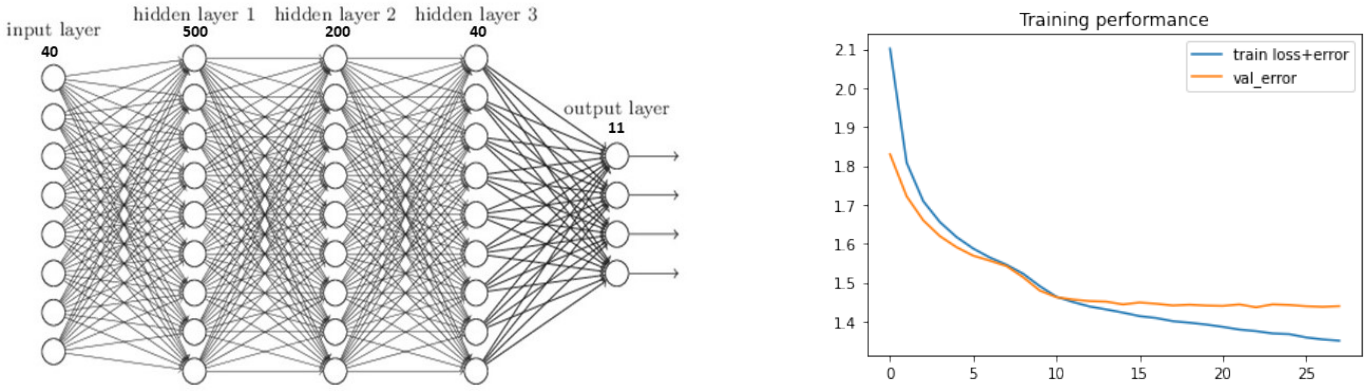


Fig. 1: ANN structure and training loss for PCA = 40

B. Structure and configurations of ANN

Figure 1 depicts the structure of the artificial neural network used in this paper after applying PCA. There are several dimensionality reductions performed in the input layer to study the effect of compression against accuracy. All the models contain 3 hidden layers each of size 500, 200 and 40 with ReLu as the activation function. Bias is used in both input and hidden layers. All the labels are one-hot encoded and since the number of classes is 11, the output layer has the same size with SoftMax activation function. The class with highest score in soft metric is chosen as a final decision. Early stopping is used to stop training when the model starts to overfit the data. The input layer has 220, 160, 80, 40, 20 and 10 neurons (using PCA method for reduction) which correspond to flattened vector of each input data. The in-phase and quadrature components are merged into a single vector. All these features are fed into the input layer after l2 normalization for each feature.

Metric		**Without PCA (# of components = 256)**	**PCA = 220**	**PCA = 160**	**PCA = 80**	**PCA = 40**	**PCA = 20**	**PCA = 10**
Training data	Loss	1.45	1.42	1.47	1.51	1.54	1.66	1.82
	Accuracy	0.44	0.44	0.42	0.42	0.41	0.37	0.31
Validation data	Loss	1.68	1.67	1.63	1.58	1.58	1.69	1.83
	Accuracy	0.35	0.37	0.37	0.39	0.39	0.35	0.31
Test data	Loss	1.68	1.67	1.63	1.59	1.58	1.69	1.83
	Accuracy	0.37	0.37	0.38	0.39	0.40	0.35	0.31

Fig. 2: Dimensionality reduction: Summary of performance metrics (dr = 0.6, batch size = 5000)

From table in figure 2 we observe that ANN with 80 and 40 features in the input layer achieves higher classification accuracy and least validation error than the other networks over all scenarios considered. Indeed, principal component analysis maps data to a lower-dimensional space in such a way that the variance of data in the low-dimensional representation is maximised. They achieve an overall validation accuracy of 39% as compared to the other configurations. In terms of categorical cross entropy loss, they achieve the lowest value at 1.58. However the training time for ANN with 80 features is longer (4.3 min) as opposed to ANN with 40 features (2 min). Hence ANN with 40 features is selected as the candidate network for modulation classification. Furthermore, training parameters such as batch size and dropout rate were experimented on the selected ANN to further improve the performance.

C. Effect of Dropout and Batch size

Dropout forces a neural network to learn more versatile features that can be used with a variety of random subsets of other neurons. From figure 3 we see that change in dropout rate has significant effect on the performance of the dense neural network as it prevents neural network from overfitting. The network with dropout rate 0.1 performs the best at high SNR regions. The final validation loss for networks with dropout rates 0.1, 0.4 and 0.6 are 1.43, 1.45 and 1.57 respectively. Overall, the network with 0.1 dropout rate has best performance and least validation loss than the other dropout rates and hence is chosen as the final dropout value for the ANN network with PCA = 40. The network was then experimented with batch sizes 256, 1024 and 5000. The validation losses for the networks were 1.44, 1.44 and 1.43 respectively. To better interpret the results the test accuracies of the models with different batch sizes were compared in Figure 3. In contrast to dropout rate effects, change in batch sizes has minor effect on performance. But the time taken for training varies. The network with batch size 1024 takes the least amount of time to train. Clearly, the best choice of batch size would be the one with least training time and hence 1024 is decided as the final batch size for the ANN network with dropout rate 0.1 and PCA = 40.

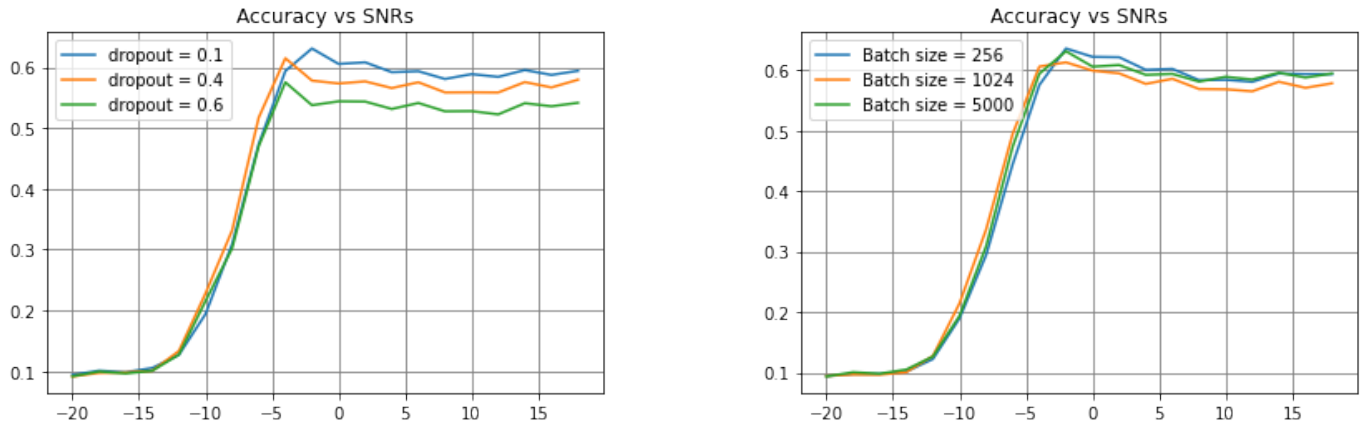
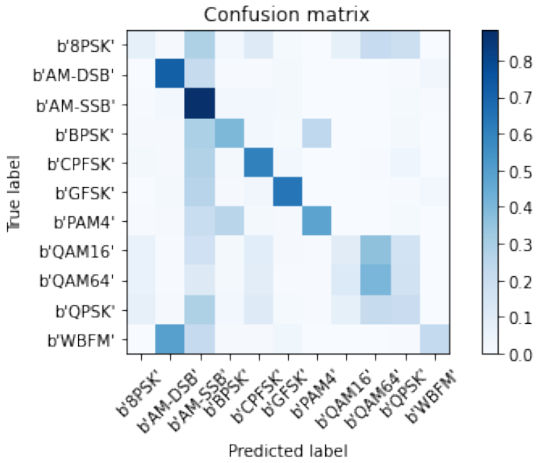
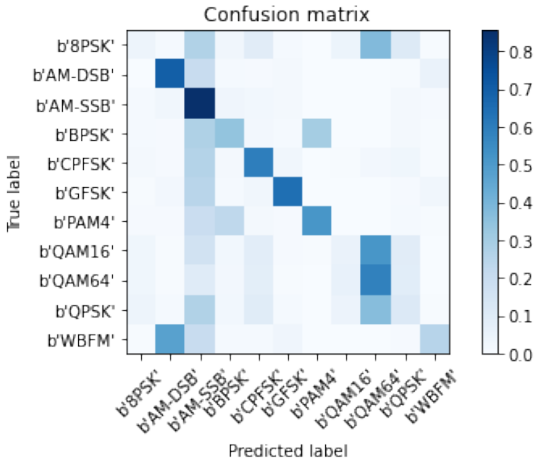


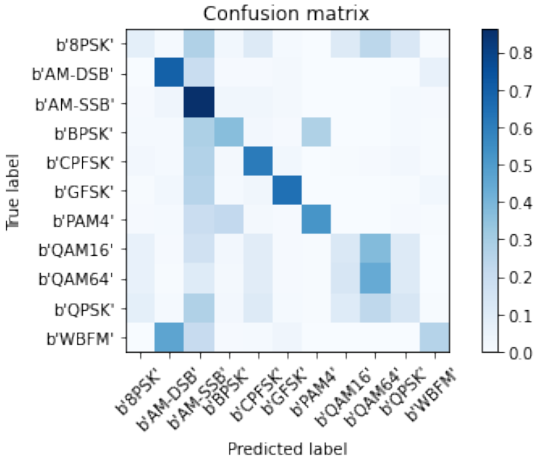
Fig. 3: Effect of dropout rate and batch size on performance of ANN, PCA = 40



(a) Batch size = 256



(b) Batch size = 1024



(c) Batch size = 5000

Fig. 4: Confusion matrix of ANN, PCA = 40

D. Results for ANN

After carefully choosing training parameters, ANN with PCA components of 40 features is selected with dropout rate of 0.1 and batch size of 1024. The model gives greater accuracy at high SNR values with less computational resources and lesser

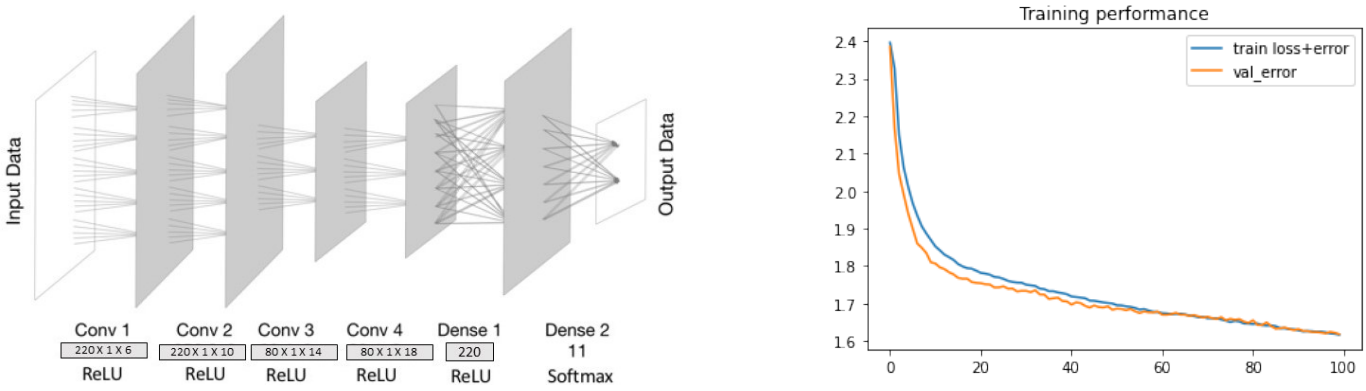


Fig. 5: CNN structure **[3]** and training loss for PCA = 220

training time. From figure 4, we can see the different confusion matrix plots with different batch size values. Clearly, batch size of 1024 gives the better accuracy plot among others.

V. Convolutional Neural Networks

Unlike artificial neural networks, which rely on features to learn, convolutional neural networks use temporal learning to find the required features on their own. The complex valued signal is treated as an input of real values and use $r(t)$ as a set of 2xN vectors into a narrow 2D Convolutional Network where the orthogonal synchronously sampled In-Phase and Quadrature (I & Q) samples make up this 2-wide dimension **[4]**. In order to select the best model for classification, several candidate convolutional neural networks were trained and tested with different PCA component values. All the CNNs have a similar pattern, each network contains convolutional layers and dense fully connected layers at the end. Regularization is used to prevent over-fitting. The convolutional layers of the CNN use Dropout and early stopping is used to prevent over-fitting and PCA to save time. Training is conducted using a categorical cross entropy loss function and using the famous Adam optimiser with batch size 5000 to begin with. The network was implemented and trained in Keras running on top of TensorFlow on an Intel Core i7-10510U CPU @ 1.80GHz and 16 GB RAM without any GPUs. The best model for modulation classification was chosen after comparing the performance of several models at different settings. 110,000 samples are used for both training and validation, and another 110,000 samples are used for the testing the performance of the trained machine, as it has been done in the implementation provided by DeepSig. These samples are uniformly distributed in SNR from -20 dB to +18 dB and tagged so that we can evaluate performance on specific subsets. Classification accuracy across different SNR values of the different training examples were inspected to evaluate the performance of the different CNN models.

A. Structure and configurations of CNN

The architecture of CNN is identical to the one used in **[3]**, which contains 4 convolutional layers and 2 dense layers as depicted in the Figure 5. The first parameter below each convolutional layer in the figure 5 represents the number of filters in that layer, while the second and third numbers show the size of each filter. The network has two convoluional layers each with 220 filters and another two convoluional layers with 80 filters. The dense layers of CNN has 220 and 11 neurons.

Metric		**Without PCA (# of components = 256)**	**PCA = 220**	**PCA = 160**	**PCA = 80**	**PCA = 40**	**PCA = 20**	**PCA = 10**
Training data	Loss	1.59	1.76	1.77	1.87	1.95	2.02	2.16
	Accuracy	0.39	0.33	0.32	0.29	0.27	0.26	0.22
Validation data	Loss	1.63	1.79	1.80	1.88	1.95	2.03	2.16
	Accuracy	0.38	0.32	0.31	0.29	0.27	0.25	0.22
Test data	Loss	1.63	1.79	1.80	1.87	1.96	2.03	2.17
	Accuracy	0.38	0.33	0.31	0.29	0.27	0.26	0.22

Fig. 6: Dimensionality reduction: Summary of performance metrics (dr = 0.6, batch size = 5000)

From table in figure 6 we observe that CNN with 220 features in the input layer achieves higher classification accuracy and least validation error than the other networks over all scenarios considered. Indeed, compression resulting from using principal component analysis in case of CNN resulted in decreased classification accuracy due to data leakage. It achieves an overall validation accuracy of 32% as compared to the other configurations. In terms of categorical cross entropy loss, it achieves the

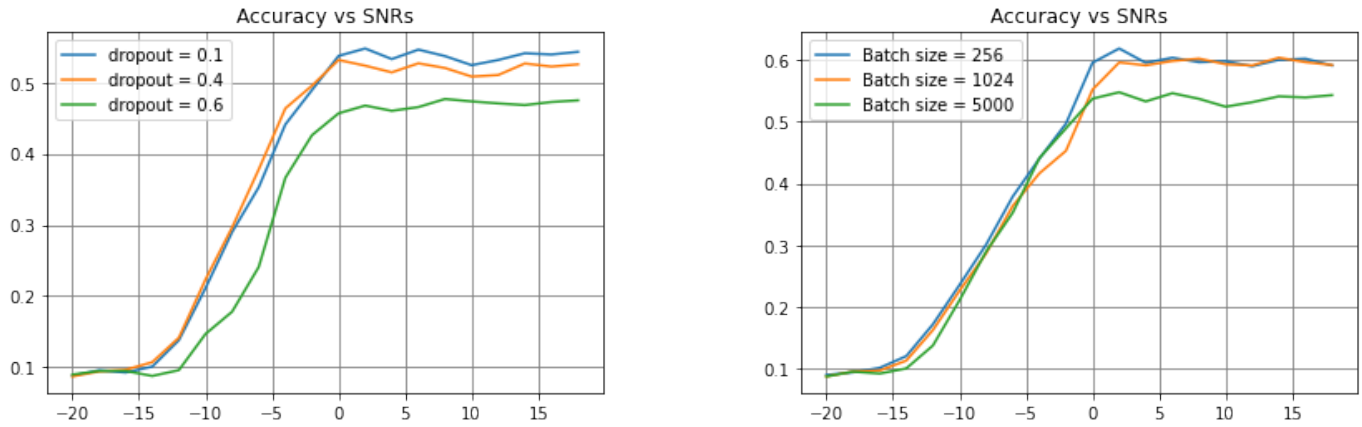


Fig. 7: Effect of dropout rate and batch size on performance of CNN, PCA = 220

lowest value at 1.79 when PCA applied. Hence CNN with 220 features is selected as the candidate network for modulation classification. Furthermore, training parameters such as batch size and dropout rate were experimented on the selected CNN to further improve the performance.

B. Effect of Dropout and Batch size

The effect of change in dropout rate was experimented on the designed CNN. Interestingly, varying dropout has different effects at different SNR levels as shown in Figure 7. The network when trained with low dropout rate of 0.1 has better performance at low SNR levels. In order to select the optimum dropout rate, the validation losses were compared. The final loss of the networks with dropout rates of 0.1, 0.4 and 0.6 are 1.61, 1.62 and 1.79 respectively. Overall, the network with 0.1 dropout rate has better performance and less validation loss than the other dropout rates and hence is chosen as the final dropout value for the network. As the complexity of CNN model increases from two convolutional layers (demo implementation for project reference) to four, we see decent improvement in the CNN classifier even after applying PCA to reduce training time.

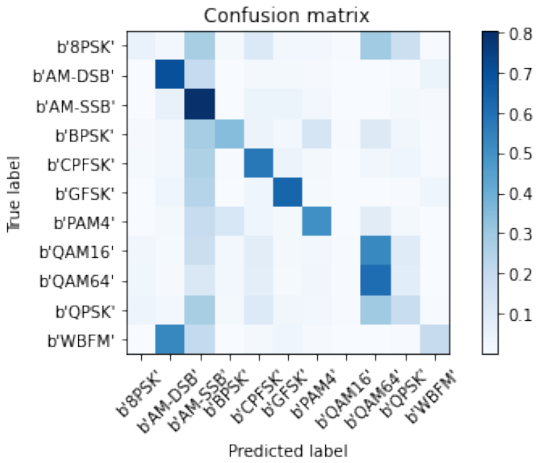
The network was then experimented with batch sizes 256, 1024 and 5000. The validation losses for the networks were 1.56, 1.54 and 1.61 respectively. To better interpret the results, the test accuracies of the models with different batch sizes were compared in Figure 7. We see significant effects on performance of the network with changes in dropout rates and batch sizes. But the time taken for training varies. The network with batch size 1024 takes the least amount of time to train despite slighlty lower accuracies with batch size 256. Clearly, the best choice of batch size would be the one with least training time and hence 1024 is decided as the final batch size for the CNN network with dropout rate 0.1 and PCA = 220.

C. Results for CNN

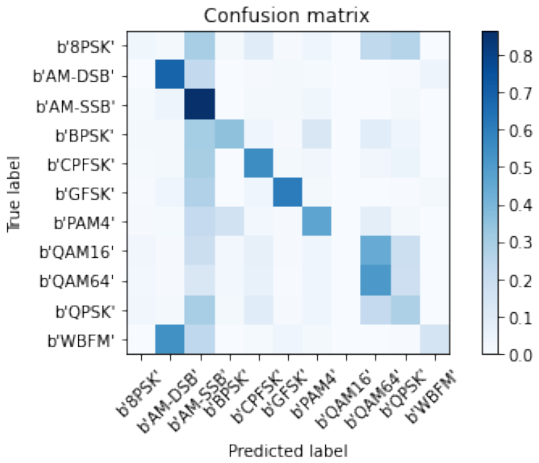
After carefully choosing training parameters, CNN with PCA components of 220 is selected with dropout rate of 0.1 and batch size of 1024. The model gives greater accuracy at low SNR values with less computational resources and lesser training time. From figure 8, we can see the different confusion matrix plots with different batch size values. Although batch size of 256 gives better SNR accuracy among others, to overcome the main challenge of training time, batch size 1024 is selected as the final model parameter.

VI. Comparison

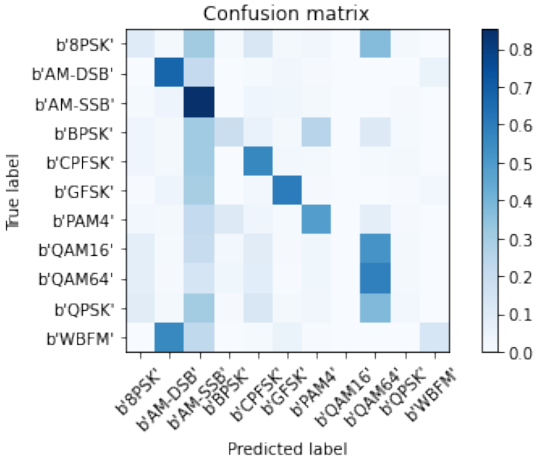
After applying PCA, ANN performs better than CNN in terms of test accuracy and test losses. The maximum test accuracy obtained with an ANN was 40% whereas CNN has about 33% accuracy. Also, it is interesting to note that ANN with 40 features at high SNR values outperformed CNN with 220 features in terms of overall accuracy. This indicates that carefully selected features result in good performance with ANN which has comparatively very less model complexity. On the contrary, we obtain significantly better low-SNR classification accuracy performance from convolutional neural networks than the feature based ANNs. This is crucial as performance at low SNR impacts range and coverage area over which it can be effectively used as a classifier. This is a significant performance improvement, as this could double effective coverage area of a sensing system **[4]**. To be able to deploy such a wireless communication system in real time, it is crucial to take into account the training and classification run time due to computational complexity. Hence, using PCA for dimension reduction surely plays a significant role in overcoming the above mentioned challenge. It is observed that the runtime of ANN is far less compared to CNN model. In sum, the trade-off between the computation time and accuracy is a crucial while deciding the model for the classifier.



(a) Batch size = 256



(b) Batch size = 1024



(c) Batch size = 5000

Fig. 8: Confusion matrix of CNN, PCA = 220

VII. Conclusion

In this work, we proposed using fast deep learning algorithms to differentiate between 11 different modulation types, with high classification accuracy over a wide range of SNR values. ANN and CNN architecture networks were studied in a variety of settings. The results demonstrate that compared to the well known expert feature based ANNs, CNNs are work quite well in low levels. Hence for robust classification at low SNR, CNNs are a preferred choice to rely on. One major challenge on using CNN is the long training time. This creates a bottleneck towards the feasibility of applying such algorithms in real-time, where online training is needed to adapt the network architecture to changing environmental conditions. As a result, both ANN and CNN can be implemented and trained to provide good performance, the option, however, is dependent on the specifications of the classifier and other constraints.

References

[1] B. Kim, J. Kim, H. Chae, D. Yoon and J. W. Choi, “Deep neural network-based automatic modulation classification technique,” *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, 2016, pp. 579-582.

[2] Z. Zhu, A. K. Nandi (2015). “Automatic Modultaion Classification: Principles, Algorithms and Applications,” London : Wiley.

[3] S. Ramjee, S. Ju, D. Yang, X. Liu, A. E. Gamal, and Y. C. Eldar, ”Fast deep learning for automatic modulation classification,” arXiv preprint arXiv:1901.05850, 2019.

[4] T. J. O’Shea, J. Corgan, and T. C. Clancy, ”Convolutional radio modulation recognition networks,” *International conference on engineering applications of neural networks*, pp. 213-226, Springer, 2016.

[5] T. J. O’Shea and N. West, ”Radio machine learning dataset generation with gnu radio,” in *Proceedings of the GNU Radio Conference*, vol. 1, 2016.

[6] Géron A. ”Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow,” 2nd Edition. : O’Reilly Media, Inc.; 2019.

[7] F. Chollet, Keras, https://github.com/fchollet/keras, 2020