Modulation Classification Using Neural Networks

Zhaosheng Li, *Dept. of Electrical and Computer Engineering, University of British Columbia*

Abstract

At the receiver antennas of a communication system, knowing the modulation format of incoming signals is of great importance. This can guarantee the demodulation of the signals is accurate and further preserve the signal quality. One way to let the receiver recognize the modulation format is deploying a neural network based classifier in the antenna. This paper proposes to apply and train two types of neural network architectures, namely Artificial Neural Network (ANN) and Convolutional Neural Network (CNN). Since there exists noise in the transmitted signals, this paper also compares the relationship between the classification accuracy and the signal-to-noise ratio (SNR). In addition, the fifth generation (5G) of communication systems cause a dynamically changing environment. This environment requires the modulation classifier to be trained frequently and thus have less computational complexity. In the realm of machine learning, one technique to decrease the training time is the Principal Component Analysis (PCA). This paper also implements the PCA on the aforesaid two neural networks and compares the performance for different values of dimensionality reduction.

Index Terms

Modulation classification, Artificial neural network, Convolutional neural network, Signal-to-noise ratio, Principal component analysis, Dimensionality reduction

I. Introduction

In a typical communication system, the transmitter encodes the messages into symbols and stores the symbols in a signal. After that, the transmitter superimposes that signal onto a carrier signal, which is a pure wave of constant frequency and does not carry much information itself. This process is called modulation, and in this way, the information is ready to be transmitted as electromagnetic wave in the communication channels. At the receiving side of the system, the receiver first detects the incoming signals and then recognizes the modulation scheme of the signals. Then the receiver employs the demodulation technique and decodes the demodulated symbols into messages.

In order to demodulate the receiving signals in time and accurately, the transmitter and the receiver must mutually agree to the modulation scheme of the transmitted signals. This implies the receiver needs to recognize the modulation format of the incoming signals once the signals are detected. One method to make the receiver accomplish that is applying deep learning (DL) models in the receiver. The deep learning models help the receiver classify different types of modulations and infer the modulation format solely from the received data. Recently, the development in the field of DL has urged the researchers on wireless communications field to conduct modulation classification through neural networks.

Over the years, many scholars have committed to classifying modulation format using neural networks and have made great achievements. [1] uses MATLAB to build an ANN-based modulation recognizer and classifies real modulated signals. [2] formulates a trained ANN to carry out recognition of multiple combinations of signals and modulations. [3] designs an ANN-based classifier and implements that classifier over a software defined radio testbed. [4] converts the modulated signals to signal constellation diagrams and feeds the diagrams to a CNN model to complete the classification task. [5] applies fusion CNN model to do modulation classification. [6] utilizes three models to classify modulation scheme, which includes CNN, residual network, and long short-term deep neural network. It turns out that the CNN model stands out. [7] proposes a generalized CNN-based automatic modulation recognizer, which is robust under varying noise conditions. [8] applies a CNN model to identify modulation format with promising accuracy and relatively short training time.

This paper proposes to implement ANN and CNN to conduct modulation classification. After finishing the classification task, this paper also compares the performance of those two models in terms of different values of SNRs. In addition, in order to decrease the training time on each model, the PCA technique is also involved in this work. This paper also discusses the relationship between the classification accuracy and the length of training time.

The rest of the paper is organized as follows: Section II describes the features of the dataset used in this work. Section III presents detailed information of the ANN and CNN models applied in this project. Section IV gives an overview of the dimensionality reduction, which is accomplished through PCA. Section V provides an in-depth discussion on the results. Section VI concludes the paper along with an outlook on the future work.

II. Dataset and Features

The dataset used in this project is RADIOML 2016.10A, which is provided by DeepSig Inc. and is publicly available at [9]. This dataset is widely used in the DL literature, and all the signals in the dataset are generated with GNU Radio software. The RADIOML 2016.10A consists of 220,000 entries, and each entry has a size of 2×128. The 2 refers to the in-phase and quadrature (IQ) components of each signal, whereas the 128 represents the number of samples in one sampling period, which is 1 μs.

The RADIOML 2016.10A dataset contains IQ samples of signals from 11 modulation classes over 20 SNR values ranging from -20 dB to 18 dB with step value of 2 dB. The 11 modulation classes are as follows: '8PSK', 'AM-DSB', 'AM-SSB', 'BPSK','CPFSK', 'GFSK', 'PAM4', 'QAM16', 'QAM64', 'QPSK', 'WBFM'. 8 of them are digital modulation techniques, and the rest 3 of them belong to analog modulations.

In this work, I treat all the IQ samples as input features to the DL model, while the eleven modulation formats are considered as target variables. I split the RADIOML 2016.10A dataset into training set and test set evenly. The reasoning behind that is to make both sets represent all the classes of data as equally as possible. According to [10], an imbalanced dataset is prone to degrading the model's performance. The training set, which has 110,000 data entries, is used during the training stage, while the test set is used for testing the accuracy of the trained models. The main purpose is to train the model to accurately classify the modulation scheme based on the amplitudes of each signal's IQ components.

III. Deep Learning Models

This work uses ANN and CNN as the DL models. The neural network is a widely used architecture in the domain of DL. According to [11] , in terms of solving classification problem, the neural network is capable of building arbitrary shaped classification boundaries. That means the neural network is good at capturing the non-linear relationship between the features and target. This section elaborates on the two DL models applied in this work.

A. ANN Model

The ANN model is essentially made of fully connected layers, or say dense layers. Figure 1 depicts the structure of the ANN model in this project. The model accepts the input directly from the training set and propagates the data into the hidden layers. All the hidden layers have rectified linear unit (ReLU) as the activation function. This is because the ReLU is less prone to overfitting when compared to other activation functions, such as Sigmoid and tanh. The number of neurons on each hidden layer is powers of two. This is to make the number of neuron units be coherent with the input dimension. Plus, according to [12], picking powers of two for the number of neurons is a convention. The number of hidden neurons on each layer is set after numerous trials. The main idea behind that is to decrease the number of units smoothly and ensure the model capture the patterns in the data. One layer to note is the `Flatten` layer. The function of that layer is flattening the data from 2-dimensional to 1-dimensional, since the output is 1-dimensional.

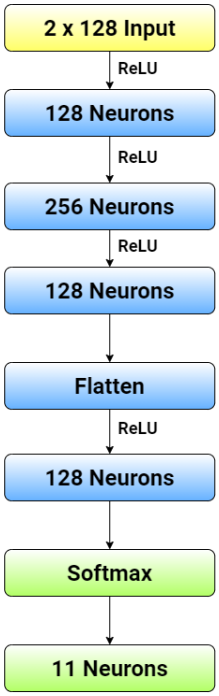

Fig. 1: ANN model

As for the output, since all the targets are categorical data, which is less helpful on training the model because the learning algorithm cannot operate on labeled data directly. Therefore, I apply one-hot encoding on the target and convert the categorical data into numerical data. After that, the target variable turns into an array of eleven binary elements. The index of the value `1` refers to the corresponding modulation scheme. In this situation, the model needs to predict the probability distribution over the eleven modulation schemes. This explains the implementation of the `Softmax` function before the output neuron.

B. CNN Model

The CNN model is a lot more complicated. Figure 2 illustrates the structure of the CNN model in this work. Instead of accepting the input directly, the CNN model reshapes the input data. Since I apply `tensorflow.keras` to build the CNN model, according to [13], the convolutional layer only accepts 4-dimensional tensor. But the input stream in this work is originally 3-dimensional. The `Reshape` layer is adding another dimension to the input tensor.

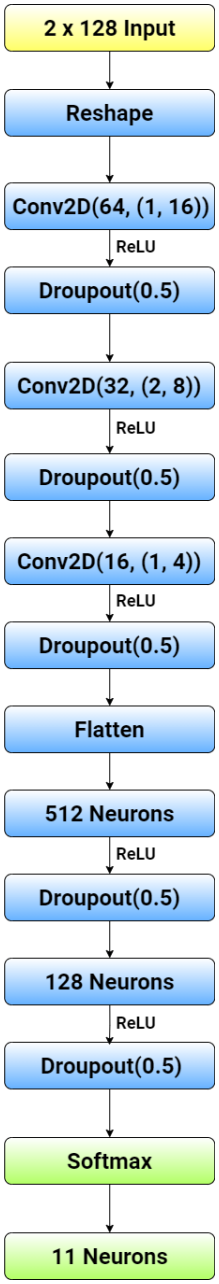

Fig. 2: CNN model

Besides the `Reshape` layer, the hidden layers contain three convolutional layers, a `Flatten` layer, and two fully connected layers. The number of filters, the kernel size in the convolutional layer, and the number of neurons in the

dense layers are determined after numerous trials. The essential idea behind that is making the size of each hidden layer be comparable with the dimension of input tensor, which further leads the model to capture the pattern in the data stream. All the hidden layers use ReLU as activation function, which can fairly prevent the model from overfitting. The output layer is basically the same with the ANN model. I apply the `Softmax` function to predict the probability distribution over the eleven possible modulation formats.

Different from the ANN model, each convolutional layer or dense layer in the CNN model is followed by a `Dropout` layer. The `Dropout` layer ignores some fractions of neurons and helps the model avoid overfitting during the training stage. All the `Dropout` layer has a dropout rate of 0.5. This is because a dropout rate of 0.5 yields the maximum regularization and minimum training error, which has already been discussed on [14].

IV. Dimensionality Reduction

The 5G communication systems contain numerous variations. For example, the beam steering and radiation pattern are constantly changing due to the implementation of adaptive antenna arrays. When applying DL models to the communication systems, the learning algorithm needs to be trained frequently to accommodate to that time-varying condition. This implies the training time should not be tediously long. Reducing the dimension of the original data can help decrease the training time. [15] points out the PCA is a fast and flexible method for dimensionality reduction in data.

The PCA algorithm sets a designated number of principal axes and projects all the data points to those axes. The projections of each data point are the principal components of the original data. In this way, the PCA algorithm reduces the data dimension and decreases the training time simultaneously. But the drawback of this algorithm is causing some information loss in the original data and further degrading the performance of DL models.

In this work, every training input vector originally has 256 dimensions, which is quite large and yields a lengthy training time. Hence, I reduce the original data dimension by $\frac{1}{4}$, $\frac{3}{8}$, $\frac{1}{2}$, and $\frac{3}{4}$, which corresponds to 64, 96, 128, and 192 components respectively. After that, I feed the data with these four values of dimensionality reduction to both ANN and CNN models. Combining these four dimensionality reduction cases and the model with original data dimension, I compare the training time, the error on test set, and the classification accuracy among the five cases in each DL model. To visualize on the trade-off relationship between the training time and the classification accuracy, I also plot the accuracy of as a function of SNR for different values of dimension reduction.

V. Discussion

This section lists out the experimental results I get when working on this project. Feeding the original data set into the models in Section III, I compare the performance of ANN model and CNN model. Following the way in Section IV to conduct PCA on the original data set, I visualize the trade-off relationship between the training time and test accuracy. In addition, I conclude the relationship between the value of SNR and the classification accuracy.

A. ANN vs CNN

I first feed the data set with original dimension to the ANN model and CNN model respectively. The first comparison is based on the training set. Figure 3 and Figure 4 depict the loss value curve during the training phase for the two models. The validation loss is a bit lower than the training loss in CNN model, whereas the the validation loss is greater than the training loss in ANN model. This is the first indicator showing that the CNN model outperforms and is more well-trained compared to the ANN model.

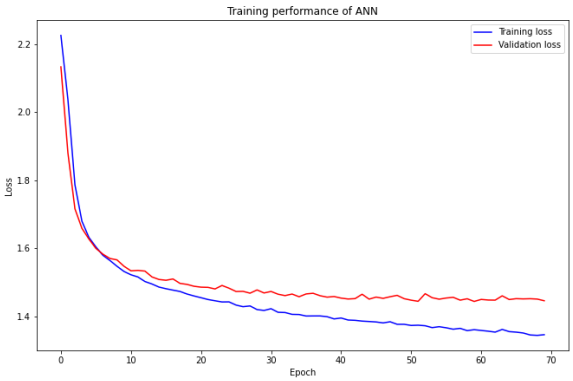

Fig. 3: ANN training loss curve

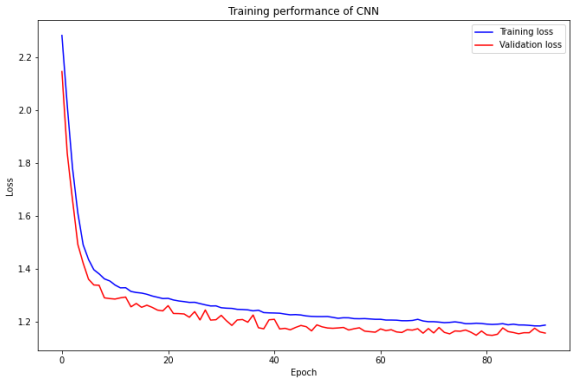

Fig. 4: CNN training loss curve

After that I compare the training time for each of those two models. The training time for the CNN model is nearly 45 times longer than that for the ANN model. But this longer training time does guarantee a great performance of the model. Implemented with the original test set, the CNN model has lower error and greater accuracy than the ANN model. This reveals that the CNN model spends longer training time on figuring out the pattern between the input feature and output target. Table I illustrates the training time, test loss, and test accuracy for the two models.

Model	Training time	Test loss	Test accuracy
ANN	281.324 s	1.4437	44.55%
CNN	12741.907 s	1.1487	55.60%

TABLE I: Comparison between ANN and CNN

I also import the `classification_report` function from `sklearn`, which is a popular Python library. That function helps me produce some statistics of classification results for the two models. In this way, I can further compare the performance of the ANN and CNN models. The `precision` column shows the percent of correct predictions for the classifier. Figure 5 and Figure 6 demonstrate that the CNN model mostly has a significantly higher precision than the ANN model. Plus, [16] points out that the average of `f1-score` should be used to compare classifier models. Equation 1 shows that `f1-score` takes both `precision` and `recall` into account. The CNN model has 0.56 for the average `f1-score`, whereas the ANN model has 0.45, which further proves that the CNN model outperforms.

$$f1\ score = \frac{2 \times recall \times precision}{recall + precision} \tag{1}$$

	precision	recall	f1-score	support
0	0.29	0.23	0.26	9991
1	0.59	0.46	0.52	10041
2	0.27	0.89	0.42	9915
3	0.52	0.44	0.48	9841
4	0.66	0.58	0.61	10076
5	0.80	0.63	0.71	10067
6	0.76	0.49	0.60	10086
7	0.29	0.33	0.31	10024
8	0.47	0.18	0.26	9920
9	0.31	0.19	0.24	9998
10	0.54	0.48	0.51	10041
accuracy			0.45	110000
macro avg	0.50	0.45	0.45	110000
weighted avg	0.50	0.45	0.45	110000

Fig. 5: ANN classification report

	precision	recall	f1-score	support
0	0.73	0.54	0.62	9991
1	0.51	0.67	0.58	10041
2	0.26	0.93	0.41	9915
3	0.89	0.58	0.71	9841
4	0.73	0.65	0.69	10076
5	0.83	0.63	0.72	10067
6	0.91	0.68	0.78	10086
7	0.34	0.04	0.08	10024
8	0.51	0.71	0.59	9920
9	0.82	0.53	0.65	9998
10	0.65	0.16	0.25	10041
accuracy			0.56	110000
macro avg	0.65	0.56	0.55	110000
weighted avg	0.65	0.56	0.55	110000

Fig. 6: CNN classification report

To further improve data visualization, I also adopt and alter the `confusion_matrix` and `plot_confusion_matrix` functions from [17]. Those two functions help me plot the confusion matrix for both ANN model and CNN model. Figure 7 and Figure 8 show that the diagonal of the CNN confusion matrix has more darker cells, which implies the CNN model makes more precise predictions than the ANN model.

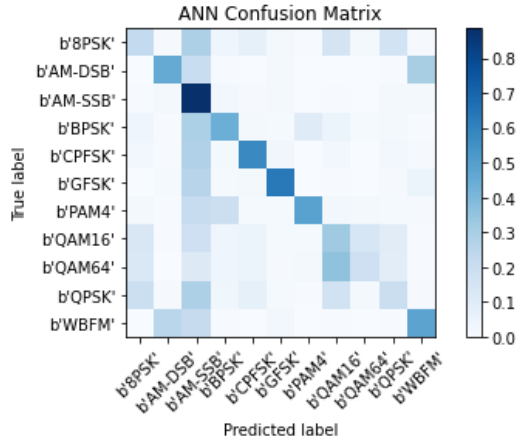

Fig. 7: ANN confusion matrix

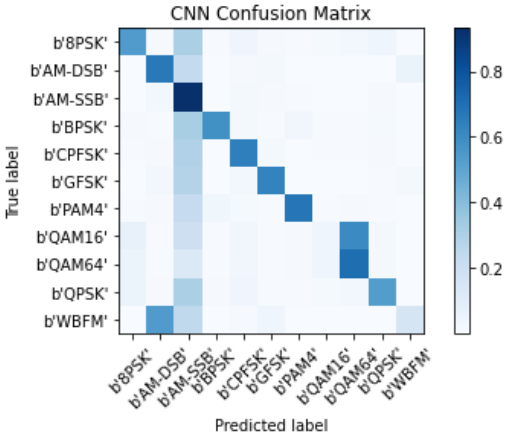

Fig. 8: CNN confusion matrix

At last, I compare the classification accuracy for each single value of SNR. Figure 9 illustrates the accuracy-SNR curve for both ANN and CNN models. There is a tie when starting the SNR from $-20dB$, but as the SNR increases, the classification accuracy for the CNN model gradually takes the lead. When the SNR becomes positive, the CNN classifier accuracy is even nearly 20% more than the ANN classifier accuracy. This result implies when the SNR of a signal is negative, namely the noise in the transmitted signal has significantly larger power, the modulation classification accuracy is low. This is because the noise in the signals prevents the model from recognizing the modulation scheme and degrades the model performance. Once again, this result proves that the CNN model outperforms in this work.

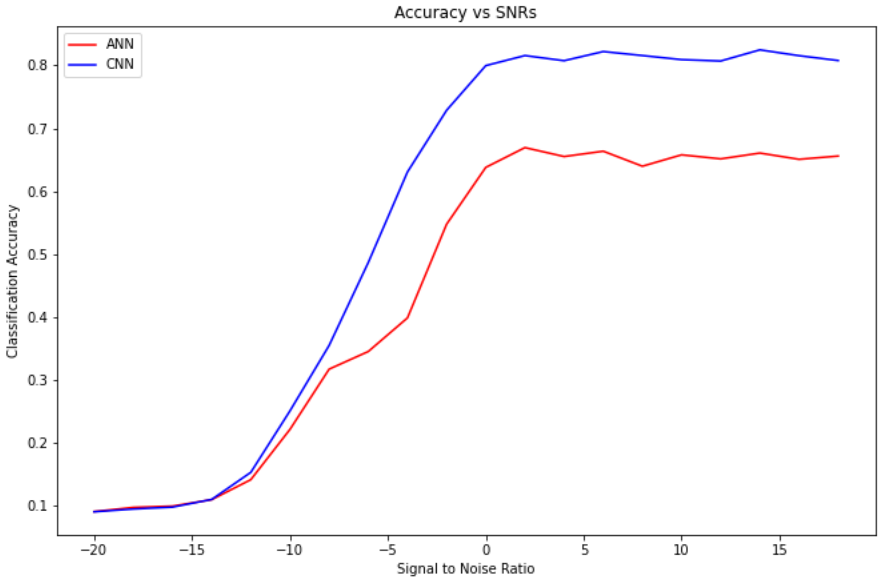

Fig. 9: Comparison of classification accuracy between ANN and CNN

In summary, the CNN model performs better than the ANN model. This is because the ANN model conducts forward propagation and backpropagation and changes the weight according to the error gradient. In other words, the ANN uses weights to learn. Whereas the CNN model applies filters on the data and spawns feature maps. Those feature maps dramatically helps the model understand the pattern in the data and guarantees a better performance. [18] concludes the CNN model is generally a more powerful and accurate way of solving classification problem, while the ANN model is considered as an optimal solution when the data has limited reliance.

B. ANN with PCA

Applying PCA on the original data set, I feed the training data with designated dimension (number of components) to the ANN model. Table II demonstrates the training time, test loss, and the test accuracy for each situation. The last entry is the model with the original data dimension. It is surprised to see the training time does not monotonically decline as the number of components decreases. Also, the test loss and the test accuracy are not monotonically changing either. This result implies the PCA has make the training data become unrepresentative to the ANN model. That means the training set does not provide enough information for the ANN model to learn. The reason for this result is the PCA reduces too much information from the original data set and makes it hard for the ANN model to figure out the pattern in the training set. That makes the model cannot recognize the modulation scheme on validation set with great performance. Since I set an early stop condition on the training phase, the training process needs to be shut down if the validation loss does not decrease within 10 epochs. This explains the training time is the shortest when the number of components is half of the original data dimension.

`Number of components`	Training time	Test loss	Test accuracy
`64`	77.844 s	1.7883	33.31%
`96`	65.634 s	1.8439	31.23%
`128`	58.939 s	1.8711	28.89%
`192`	101.014 s	1.8397	32.44%
`256`	160.204 s	1.4509	44.91%

TABLE II: Statistics of ANN with PCA

In order to look into this result clearly, I plot the training loss curve for all the scenarios with reduced-dimension data, which are depicted on Figure 10, Figure 11, Figure 12, and Figure 13.

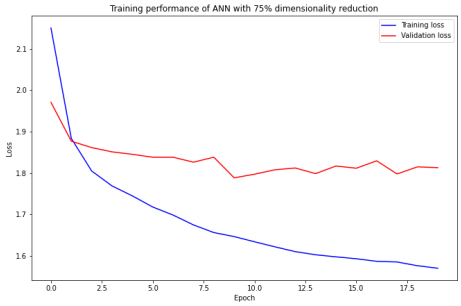

Fig. 10: ANN training loss curve 64 dimensions

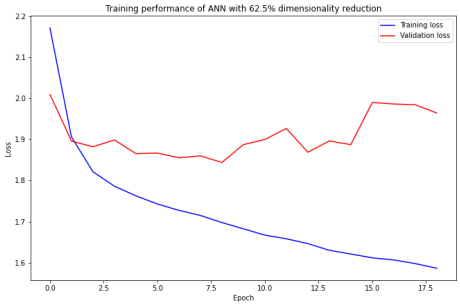

Fig. 11: ANN training loss curve with 96 dimensions

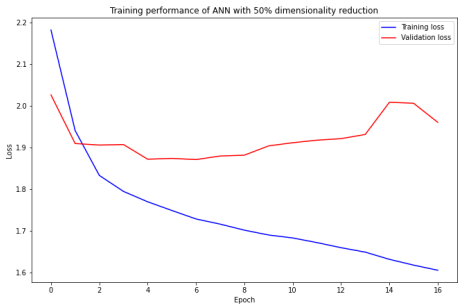

Fig. 12: ANN training loss curve with 128 dimensions

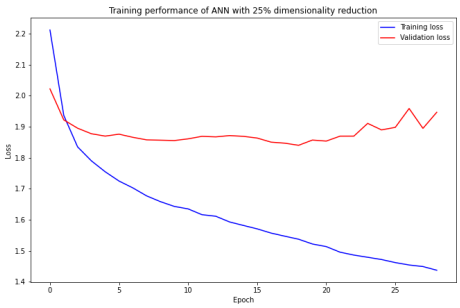

Fig. 13: ANN training loss curve with 192 dimensions

After that, I extract the accuracy for each value of SNR and plot the accuracy-SNR curve for each scenario on Figure 14. Even though the PCA reduces the training time for the ANN model, but the learning performance is highly degraded. From the five curves below, the best performance happens with the data without any dimensionality reduction.

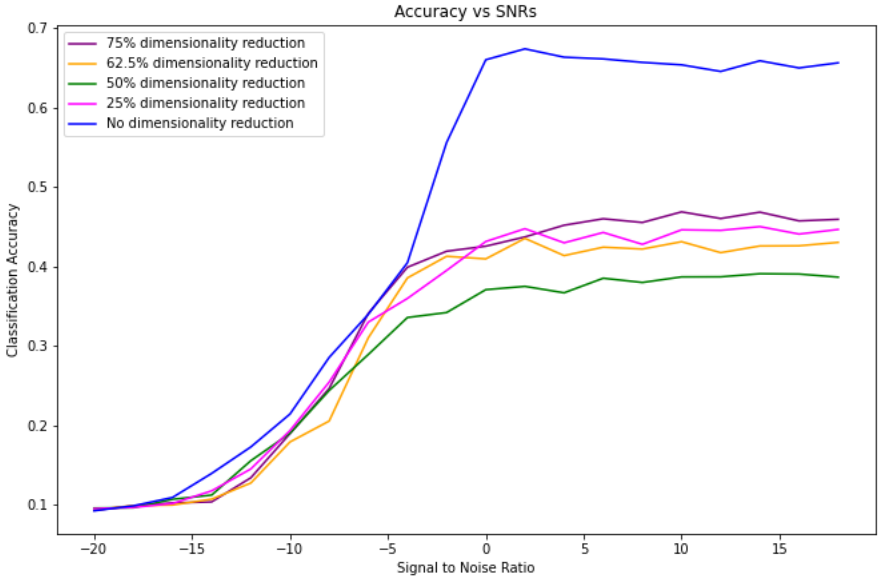

Fig. 14: ANN with various PCA

C. CNN with PCA

I also feed the data with dimensionality reduction to the CNN model. The results on the CNN model look more encouraging. Table III shows the training time, test loss, and the test accuracy for different input data dimensions. The training time is decreasing along with the reducing dimensionality. And the test loss and test accuracy is monotonically changing, even though there is a tiny fluctuation when the data dimension is one half of the original dimension.

`Number of components`	Training time	Test loss	Test accuracy
`64`	727.873 s	1.7058	35.85%
`96`	1992.885 s	1.6263	39.03%
`128`	2868.902 s	1.5968	40.41%
`192`	7646.649 s	1.6258	39.41%
`256`	10880.318 s	1.1498	55.45%

TABLE III: Statistics of CNN with PCA

I also plot the accuracy-SNR curve for various data dimensions on Figure 15. It is still the model with the original data dimension has the best performance. Although applying PCA helps the CNN model reduce the training time, the high accuracy cannot be guaranteed due to the loss of some information in the original data set.

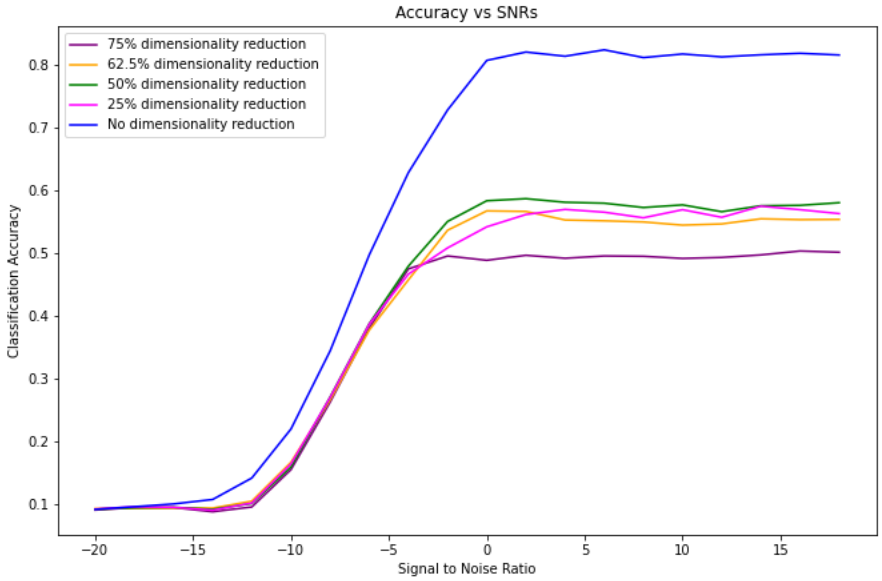

Fig. 15: CNN with various PCA

To sum up, implementing the PCA reduces the data dimension and increases the speed of training. But the price for that is the deduction of classification accuracy and the degradation of model performance.

VI. Conclusion

This paper proposes to apply ANN and CNN models to conduct modulation classification. And the experimental results show that the CNN has a better performance in this work. In addition to that, the PCA technique is applied to reduce the data dimension and increase the speed of training for the two DL models. But the classification accuracy is significantly decreasing as a price. This result proves the trade-off relationship between the training time and the prediction accuracy.

As for the future development on this work, some various techniques can be implemented. `Scikit-Learn` contains a couple of variants on PCA, such as RandomizedPCA and SparsePCA. Instead of the regular PCA, those advanced PCA techniques are worth trying. Instead of the deep learning models, some other machine learning models can also be considered as potential solutions to the modulation classification problems, such as Support Vector Machine or Decision Trees. Even some advanced machine learning algorithms, including Reinforcement Learning and Generative Adversarial Network can be applied to recognize the modulation scheme in the signals.

Acknowledgment

The author would like to thank Dr. Lutz Lampe and his Ph.D. students for offering this project. Also, the author is grateful to DeepSig Inc. for making RADIOML 2016.10A dataset available to the public.

References

[1] M. Richterova, D. Juracek, and A. Mazalek, "Modulation classifier of analogue modulated signals based on method of artificial neural networks," pp. 145–147, 2006.
[2] R. Roy, B. Saikia, and K. Sarma, "Modulation recognition using artificial neural network(ann)," 2011.
[3] J. Jagannath, N. Polosky, D. O'Connor, L. N. Theagarajan, B. Sheaffer, S. Foulke, and P. K. Varshney, "Artificial neural network based automatic modulation classification over a software defined radio testbed," pp. 1–6, 2018.
[4] S. Peng, H. Jiang, H. Wang, H. Alwageed, Y. Zhou, M. M. Sebdani, and Y. Yao, "Modulation classification based on signal constellation diagrams and deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 3, pp. 718–727, 2019.
[5] S. Zheng, P. Qi, S. Chen, and X. Yang, "Fusion methods for cnn-based automatic modulation classification," *IEEE Access*, vol. 7, pp. 66 496–66 504, 2019.
[6] A. Jolly, P. Khorramshahi, and T. Saeed, "Improvements to modulation classification techniques using deep learning," 2020.
[7] T. Zhang, C. Shuai, and Y. Zhou, "Deep learning for robust automatic modulation recognition method for iot applications," *IEEE Access*, vol. 8, pp. 117 689–117 697, 2020.
[8] K. Tekbıyık, A. R. Ekti, A. Görçin, G. K. Kurt, and C. Keçeci, "Robust and fast automatic modulation classification with cnn under multipath fading channels," pp. 1–6, 2020.
[9] D. Inc., "Rf datasets for machine learning." [Online]. Available: https://www.deepsig.ai/datasets
[10] J. T. Raj, "What to do when your classification dataset is imbalanced." Sep 2019. [Online]. Available: https://towardsdatascience.com/what-to-do-when-your-classification-dataset-is-imbalanced-6af031b12a36
[11] M. Haldar, "How do neural networks work?" May 2020. [Online]. Available: https://medium.com/machine-intelligence-report/how-do-neural-networks-work-57d1ab5337ce
[12] N. Slater, "Why should the number of neurons in a hidden layer be a power of 2?" Feb 2018. [Online]. Available: https://ai.stackexchange.com/a/5401
[13] "tf.keras.layers.conv2d : Tensorflow core v2.4.1." [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D
[14] C. Ranjan, "Simplified math behind dropout in deep learning," May 2019. [Online]. Available: https://towardsdatascience.com/simplified-math-behind-dropout-in-deep-learning-6d50f3f47275
[15] J. T. VanderPlas, *Python data science handbook: essential tools for working with data.* Beijing etc.: O'Reilly, 2017, ch. In Depth: Principal Component Analysis, p. 433–445.
[16] S. Kohli, "Understanding a classification report for your machine learning model," Nov 2019. [Online]. Available: https://medium.com/@kohlishivam5522/understanding-a-classification-report-for-your-machine-learning-model-88815e2ce397
[17] Ubcyinjia, "ubcyinjia/radioml_example," Jan 2020. [Online]. Available: https://github.com/ubcyinjia/RadioML_Example
[18] V. Meel, "Ann and cnn: Analyzing differences and similarities," Feb 2021. [Online]. Available: https://viso.ai/deep-learning/ann-and-cnn-analyzing-differences-and-similarities/