Spectral Correlation Function based Convolutional Neural Networks for Radio Modulation Detection

(Final Project for ECE 504, Fall 2017)

Md Jubayer Shawon*, John Chiasson, Vishal Saxena
ECE Department, University of Idaho
875 Perimeter Dr, Moscow, ID 83844

***Abstract*—In this project, a convolutional neural network has been designed to recognize modulation type in a complex-valued temporal radio signal. The network has been trained using RadioML dataset [1]. However, the data was not directly used to train the network. The data has been preprocessed by implementing Spectral Correlation Function (SCF) [2-4] and Max-normalization. Prior to proposing a final design, several network architectures have been investigated, and the architecture with the best performance on the test data has been chosen. The final design takes in 4096 inputs and has 11 outputs (corresponding to 11 modulations available in the data set).**

Keywords—Convolutional Neural Networks; Modulation Detection; Spectral Correlation Function

I. INTRODUCTION

In recent years, demand for massive data traffic in the wireless communication channels has sparked the need for better and more efficient radio spectrum analysis with high degree of automation and optimization. A small yet very important part of meeting such demands is to detect modulation type of a received signal with high level of accuracy. Therefore, for this current project, a neural network has been proposed and implemented to classify modulation schemes used in the received signal using Convolutional Neural Network (CNN) [5].

II. THE DATASET

A. RadioML 2016.10

For this project, the dataset from RadioML [1] has been used to train and test the network. Although RadioML offers several types of data packets, "RadioML 2016.10" dataset has been used in this current work. It contains complex-valued temporal radio signal corresponding to 11 different popular modulation (both analog and digital) schemes. These complex-valued temporal radio signal has undergone moderate LO drift, light fading, at different signal and noise power settings.

This dataset has 220,000 slices of temporal radio signals with SNR levels ranging from -20 dB to +18dB (with a step of 2dB). Each slice contains 2 time series, each containing 128 sampled points. They represent I and Q values in the constellation diagram of a particular modulation. I/Q values are nothing but a representation of the changes in magnitude and phase of a sine wave [6].

TABLE I. RADIOML 2016.10 DATASET CONSTITUENTS

Modulation	***SNR Range***	***No. of Data points***
8PSK	-20dB to 18dB	20,000
AM-DSB	-20dB to 18dB	20,000
AM-SSB	-20dB to 18dB	20,000
BPSK	-20dB to 18dB	20,000
CPFSK	-20dB to 18dB	20,000
GFSK	-20dB to 18dB	20,000
PAM4	-20dB to 18dB	20,000
QAM16	-20dB to 18dB	20,000
QAM64	-20dB to 18dB	20,000
QPSK	-20dB to 18dB	20,000
WBFM	-20dB to 18dB	20,000
Total		220,000

B. Data Segmentation

RadioML 2016.10 dataset has been split into 3 segments - Training, Validation and Test data. Their relative ratios (no. of data points) have been presented in the following table.

TABLE II. TRAINING, VALIDATION AND TEST DATA

Dataset	***Size***	***% of Total Data***
Training	170500	77.5
Validation	24750	11.25
Test	24750	11.25

C. Data Preprocessing

The datasets have been preprocessed before using them to train and test the network. Right after extracting the raw data from RadioML 2016.10 pack, the data have been shuffled and partitioned (into training, validation and test data). Afterwards, each temporal radio signal (I + jQ) has been fed to Spectral Correlation Function (SCF). The spectral correlation function transformed each temporal radio signal in the dataset into a 64×64 matrix. Afterwards, each element of the matrix has been

normalized by the largest value in that matrix (max-normalization). Assuming each matrix represents an image, the entire process has generated 220,000 2D-SCF patterns (images) of the size of 64 Pixels × 64 Pixels (each). The library for generating SCF patters is developed by Prof. Vishal Saxena (ECE, University of Idaho).

Fig. 1 illustrates the steps taken to process the RadioML data.

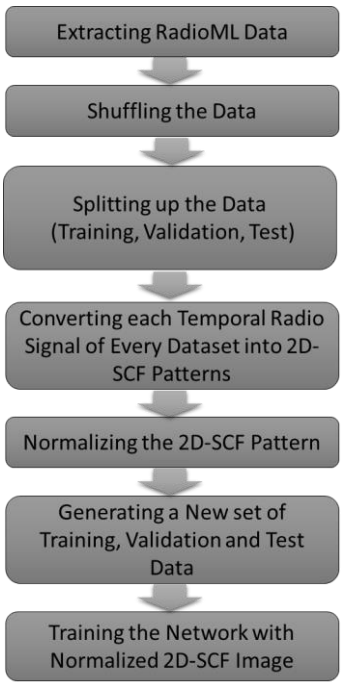

Fig. 1. *Flowchart demonstrating the data processing steps*

D. Spectral Correlation Function

In the previous section, it was mentioned that Spectral Correlation function (SCF) was used to preprocess the data. Therefore, it is important to discuss SCF to develop a better understanding of the entire process.

The Spectral Correlation Function is spectral density of time-averaged correlation (covariance) [2-4]. It is defined as the Fourier Transform of the cyclic autocorrelation function (ACF), to allow the localization in the frequency domain of the amount of time-correlation between frequency-shifted versions of the discrete signal x[n] [7]. The SCF is given by [7]-

$$S_X^{\alpha}(f) = \sum_{l=-\infty}^{\infty} R_x^{\alpha}[l].\, e^{-j2\pi fl} \tag{1}$$

Where,

$$R_x^{\alpha}[l] = \langle x[n].\, x^*[n-l].\, e^{-j2\pi\alpha n} \rangle\, e^{-j2\pi\alpha l} \tag{2}$$

The reason SCF was used in this work is because it is capable of extracting the signatures of a signal embedded deep within the noise [3-5]. Therefore, instead of using the RadioML data directly, SCF was employed to increase the efficiency of the neural network. In Fig. 2, few examples of 2D-SCF patterns of signals (with noise) for different modulations are presented.

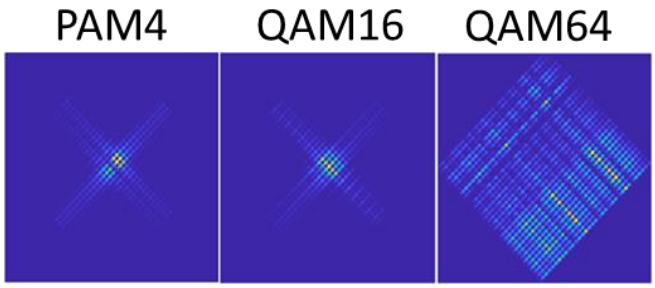

Fig. 2. *Example of SCF patterns of a PAM4 (left), QAM16 (middle) and QAM64 (right) radio signal*

III. Network Architecture

In this project, two popular neural network architectures known as Feedforward Neural networks and Convolutional Neural networks were investigated.

A. Feedforward Neural Network

Feedforward neural networks are the ones with no loops in the network. It consists of simple neuron-like units. Every unit in a particular layer is connected with the units of the previous layer. The connections between the units are usually weighted with different values. The data passes through the neural network, and the sum of the products of the weights & the inputs is calculated at each node [8]. Depending on the activation function used, if the value exceeds a certain threshold value, the neuron fires. Fig. 3 shows a schematic of a typical feedforward neural network.

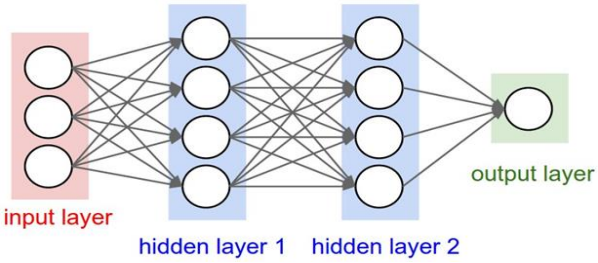

Fig. 3. *Feedforward Neural Network [9]*

B. Convolutional Neural Network

Convolutional neural network conducts a convolution operation on the input data (image) and passes it to the next layer. The next layer is usually a max or average pooling layer. This pooling layer takes the maximum or average of value of a group of neurons of the previous layer. This convolution and pooling process might occur multiple times based on the accuracy requirements. Afterwards, fully

connected layers are placed which are connected to all the neurons in the previous layer and the next layer [5].

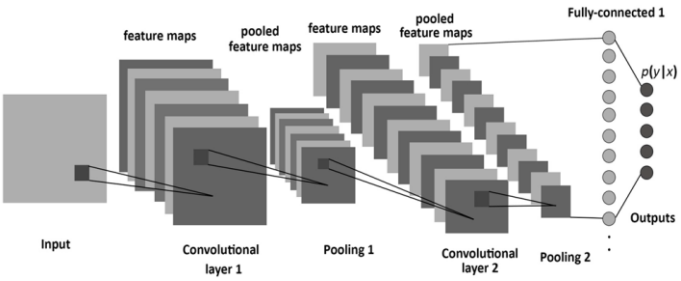

Fig. 4. *Convolutional Neural Network [8].*

IV. DESIGNING THE NETWORK

The network designing process involved two primary steps- Network Architecture Selection & Network Parameter Optimization. Performance of the network on the test data and computational complexity were taken into account to select the best candidate to take on the modulation detection problem.

NOTE: In the following sections, accuracy will presented for different network architectures. Unless otherwise stated, all the accuracies are evaluated using the test data.

A. Feedforward Neural Network

A number of different combinations of parameters were investigated to design the network. The following table summarizes the parameters and the corresponding accuracy obtained.

TABLE III. ACCURACY AT DIFFERENT COMBINATIONS OF PARAMETERS

Fixed Parameters	***Epochs***	***Hidden Layer1 (no. of neurons)***	***Hidden Layer2 (no. of neurons)***	***Accuracy (%)***
Cost Function (cross-entropy) Mini-Batch (10) Regularization (10) Learning Rate (0.5)	27	500	-	40.82*
	20	200	50	42.39

*Accuracy on Validation data

B. Convolutional Neural Network

In this work, different structures and combinations of parameters were explored for convolutional neural network.

1) Number of convolution & pooling layers:

As mentioned above, one can implement multiple convolution and pooling layers for this type of network. Therefore, a quantitative analysis was carried out to find the best performing combination.

TABLE IV. ACCURACY AT DIFFERENT COMBINATIONS OF PARAMETERS

Fixed Parameters	***(Conv1) Maps1 (number) & Mask1 (size)***	***(Conv2) Maps2 (number) & Mask2 (size)***	***Fully connected 1***	***Fully connected 2***	***Accuracy (%)***
Image shape (64×64) Mini-batch (10) Activation Function (ReLU) Lambda (0.001) Learning Rate (0.03) Dropout (0.5) Pooling 1&2 (2,2)	20 & (5×5)	-	1000 neurons	-	40.67* (after 21 epochs)
	20 & (5×5)	40×20 & (3×3)	1000 neurons	1000 neurons	42.11* (after 21 epochs)

*Accuracy on Validation data

Decision: Two ConvPool and two fully connected layers are better than one ConvPool layer and one fully connected layer (for this dataset!).

2) Mask Shape:

The effect of mask shape (convolution layer) on accuracy was also investigated. The first convolution layer mask size were changed (from 5×5 to 9×9) to check its effect on accuracy.

TABLE V. ACCURACY AT DIFFERENT COMBINATIONS OF PARAMETERS

Fixed Parameters	***(Conv1) Maps1 & Mask1***	***(Conv2) Maps2 & Mask2***	***Fully connected 1***	***Fully connected 2***	***Accuracy (%)***
Image shape (64×64) Mini-batch (10) Activation Function (ReLU) Lambda (0.001) Learning Rate (0.03) Dropout (0.5) Pooling 1&2 (2,2)	20 & (5×5)	40×20 & (3×3)	1000 neurons	1000 neurons	44.49 (60 epochs)
	20 & (9×9)	40×20 & (3×3)	1000 neurons	1000 neurons	44.13 (60 epochs)

Decision: A 5×5-mask in the first convolution layer is better than a 9×9-mask (for this particular dataset).

3) Number of Neurons in the Fully-connected Layer:

The number of hidden neurons in the fully-connected layer strongly affects the accuracy. Therefore, two accuracy tests were carried out with different number of neurons keeping the other parameters constant.

TABLE VI. ACCURACY AT DIFFERENT COMBINATIONS OF PARAMETERS

Fixed Parameters	***(Conv1) Maps1 & Mask1***	***(Conv2) Maps2 & Mask2***	***Fully connected 1***	***Fully connected 2***	***Accuracy (%)***
Image shape (64×64) Mini-batch (10) Activation Function (ReLU) Lambda (0.1) Learning Rate (0.03) Dropout (0.5) Pooling 1&2 (2,2)	20 & (5×5)	40×20 & (3×3)	1000 neurons	1000 neurons	43.21* (after 23 epochs)
	20 & (5×5)	40×20 & (3×3)	100 neurons	100 neurons	38.55* (after 23 epochs)

*Accuracy on Validation data

Decision: More number of neurons results in greater accuracy (for this dataset).

4) Learning rate:

To optimize the learning rate for the best performing network, three different learning rates were investigated, keeping all other parameters constant.

TABLE VII. ACCURACY AT DIFFERENT COMBINATIONS OF PARAMETERS

Fixed Parameters	***(Conv1) Maps1 (number) & Mask1***	***(Conv2) Maps2 (number) & Mask2***	***Learning Rate***	***Accuracy (%)***
Image shape (64×64) Mini-batch (10) Activation Function (ReLU) Lambda (0.001) Dropout (0.5) Pooling 1&2 (2,2) Fully connected 1&2 (1000)	20 & (5×5)	40×20 & (3×3)	0.005	41.97 (70 epochs)
	20 & (5×5)	40×20 & (3×3)	0.03	44.49 (60 epochs)
	20 & (5×5)	40×20 & (3×3)	0.3	9.11 (constant for all epochs)

Decision: A learning rate of 0.03 gives the best result for our dataset compared to the other learning rates tested.

5) Regularization Parameter:

Two different regularization parameters were tried on this network.

TABLE VIII. ACCURACY AT DIFFERENT COMBINATIONS OF PARAMETERS

Fixed Parameters	***(Conv1) Maps 1 & Mask1***	***(Conv2) Maps 2 & Mask2***	***Regularization Parameter***	***Accuracy (%)***
Image shape (64×64) Mini-batch (10) Activation Function (ReLU) Learning Rate (0.03) Dropout (0.5) Pooling 1&2 (2,2) Fully connected 1&2 (1000)	20 & (5×5)	40×20 & (3×3)	0.1	43.96* (51 epochs)
	20 & (5×5)	40×20 & (3×3)	0.001	44.49 (60 epochs)

* Accuracy on Validation data.

Decision: A smaller lambda (0.001) will be chosen for our network.

6) Mini-batch:

Two different mini-batch size were tried on this network.

TABLE IX. ACCURACY AT DIFFERENT COMBINATIONS OF PARAMETERS

Fixed Parameters	***(Conv1) Maps1 & Mask1***	***(Conv2) Maps2 & Mask2***	***Mini-batch Size***	***Accuracy (%)***
Image shape (64×64) Lambda (0.001) Activation Function (ReLU) Learning Rate (0.03) Dropout (0.5) Pooling 1&2 (2,2) Fully connected 1&2 (1000)	20 & (5×5)	40×20 & (3×3)	10	44.49 (60 epochs)
	20 & (5×5)	40×20 & (3×3)	30	43.42 (57 epochs)

Decision: A smaller mini-batch size (10) will be chosen for our network.

V. FINAL DESIGN AND RESULTS

Based on the decisions made in the previous sections, the convolutional neural network was chosen as it outperformed feedforward neural network. The following tables describes the final design.

TABLE X. FINAL DESIGN (INPUT & OUTPUT)

Input	***Output***
64×64	11

TABLE XI. FINAL DESIGN (CONV1 LAYER)

Conv1		
Mask (Size)	***Maps***	
	No.	***Dimension***
(5×5)	20	60×60

TABLE XII. FINAL DESIGN (POOLING1 LAYER)

Pooling layer1		
Size	***Pooled Maps (max pooling)***	
	No.	***Dimension***
2×2	20	30×30

TABLE XIII. FINAL DESIGN (CONV2 LAYER)

Conv2		
Mask (Size)	***Maps***	
	No.	***Dimension***
3×3	40×20	28×28

TABLE XIV. FINAL DESIGN (POOLING2 LAYER)

Pooling layer2		
Size	***Pooled Maps (max pooling)***	
	No.	***Dimension***
2×2	40×20	14×14

TABLE XV. FINAL DESIGN (OTHER PARAMETERS)

Fixed Parameters	**Fully connected 1&2**	***Learning Rate***	***Regularization Parameter***	***Cost Function***
Mini-batch (10) Activation Function (ReLU) Pooling 1&2 (2,2) Softmax output layer Dropout (0.5)	1000 neurons	0.03	0.001	Log-likelihood cost function

The schematics of the network is presented in the Fig. 5.

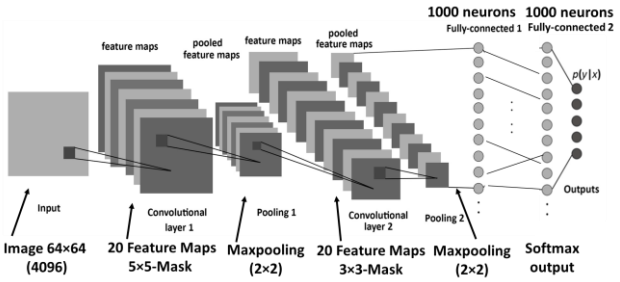

Fig. 5. *Schematics of the final design*

Using the abovementioned parameters, a training-run was carried out for 60 epochs. The entire result of this training has been provided at the end of this report. For this training run, the epoch vs accuracy plot is shown in Fig. 6. The final accuracy on the test data was found to be **44.49%**. This is very close to the result reported in [11]. In [11], the accuracy on the test data was **50.33%**.

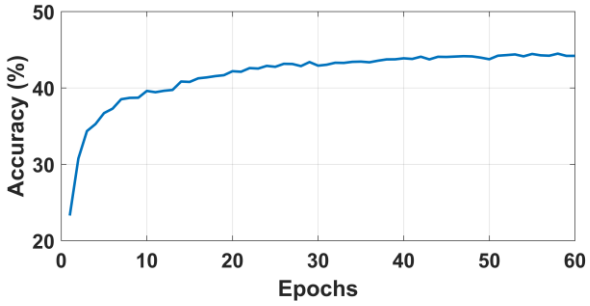

Fig. 6. *Epochs vs Accuracy (%) during the training of the final network.*

The first conv layer of the final network has 20 feature maps. These feature maps are shown in Fig. 7.

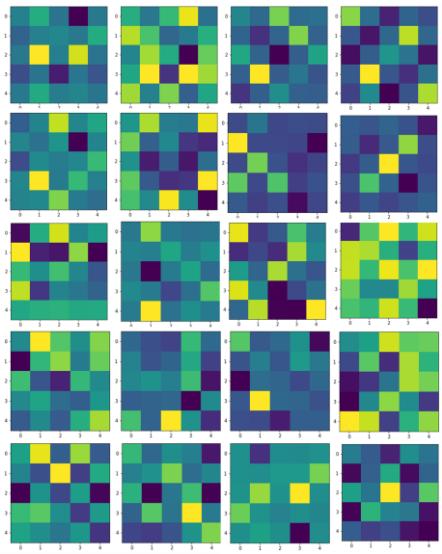

Fig. 7. *Feature maps in the first conv layer.*

VI. Discussion

A. Limitations of this work

Due to the long computational time required, several aspects of the design optimization could not be investigated. They are as follows-

1) *Dropout parameter*
2) *Activation function: ReLU vs Sigmoid*
3) *Mask shape of the Conv2 layer*
4) *Downsampling schemes: max vs average pooling*
5) *2+ ConvPool layers with 2+ Fully connected layers*
6) *Comparison of the cost functions: squared error, cross-entropy, maximum likelihood*

B. Challenges Encountered

This biggest challenge of this particular project was the implementation of Spectral correlation function. Extensive literature review was carried out to implement the SCF in python.

Another challenge of this project was the computational complexity introduced by SCF. For RadioML dataset, the SCF produced 24.7 gigabytes of data which severely impacted the number of trial-and-error steps that could be carried out with the limited computational resources.

C. Further Comments

Although the convolutional neural network was implemented and selected as our final design, it is important to note that the simple feedforward neural network was performing almost equally well. For this network, the highest accuracy on the test data was found to be 42.39%. However, this accuracy was obtained after running only 20 epochs without optimizing any hyper parameters. Due to the limitations of the computational resources, further exploration into optimizing this network was not carried out. It is highly probable that, for this particular dataset (RadioML), feedforward network may outperform the sophisticated convolutional neural network given that the proper optimization of the parameters are carried out.

Acknowledgements

Support from Nvidia GPU donation program is gratefully acknowledged. The author would like to thank Prof. Vishal Saxena for developing the SCF library and Prof. John Chiasson for the frequent project related discussions.

References

[1] https://radioml.org/

[2] https://cyclostationary.blog/2015/09/28/the-spectral-correlation-function/

[3] Mendis, G. J., Wei, J., and Madanayake, A., "Deep learning-based automated modulation classification for cognitive radio," in Communication Systems (ICCS), 2016 IEEE International Conference on (pp. 1-6), December, 2016.

[4] Mendis, G. J., Wei, J., and Madanayake, A. "Deep belief network for automated modulation classification in cognitive radio," in IEEE Workshop on Cognitive Communications for Aerospace Applications (CCAA), pp. 1-5, June 2017.

[5] https://en.wikipedia.org/wiki/Convolutional_neural_network

[6] http://www.ni.com/tutorial/4805/en/

[7] Costa, Evandro Luiz da, Thesis title - "Detection and identification of cyclostationary signals," Naval Postgraduate School., https://calhoun.nps.edu/handle/10945/8241.

[8] https://en.wikipedia.org/wiki/Feedforward_neural_network

[9] https://www.datasciencecentral.com/profiles/blogs/a-simple-neural-network-with-python-and-keras

[10] Albelwi, Saleh; Mahmood, Ausif. 2017. "A Framework for Designing the Architectures of Deep Convolutional Neural Networks." Entropy 19, no. 6: 242.

[11] https://github.com/radioML/examples/blob/master/modulation_recognition/RML2016.10a_VTCNN2_example.ipynb