Signal Classification with Deep Learning

Jacob Breckenridge and Andrew Burger
Machine Learning - ECE 523
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ 85721
`jbrecken@email.arizona.edu`
`aburger@email.arizona.edu`

Abstract

The purpose of this paper is to use Deep-learning to classify signals. This has great value in the field of Cognitive radio(CR) which is a radio which can intelligently adapt in real time. Using this technique reduces the need for feature-selection entirely which is a necessity to implement many of the current CR machine learning methods today. The overall goal is to pave a path for current implementations of wireless communication systems to utilize Deep-learning.

1 Introduction

Software defined radio(SDR) and Cognitive radio(CR) are some of the newest technologies in the wireless communications field. The origin of this idea came from Joseph Mitola which was a gateway to transition away from the original radio. SDR allows a radio to change transmission parameters like bandwidth and center frequency. SDR's biggest use today amongst society is its implementation into smart phone devices.

With the current density of the overall spectrum it is necessary for a radio to understand how the spectrum is used. One method of determining whether a frequency is in use is by measuring the amount of energy present at that frequency and then comparing it to the amount of energy present when it is not in use. More significant methods are not just determining if that frequency is in use but also determining who's using it. Hence the importance of signal classification techniques. The goal is to classify signals with no a-priori information and do so in real time.

1.1 Feature-Selection Methods

The feature based methods which are primarily in use are somewhat suboptimal but greatly reduced the complexity of the problem to allow classifiers such as Support Vector Machines(SVM) and various ensembles such as bagging to be used. These features include cumulants, time and frequency domain statistics, wavelet transform coefficients, or a combination of them[1]. Other methods for modulation classification such as likelihood-based approaches fail to achieve the classification in real time due to their high computational complexity.

1.2 Deep-learning Methods

Deep-learning offers an advantage to modulation classification that the previous methods aren't able to provide. There is no need to do feature-selection at all and instead this model can learn off of raw samples containing their real and complex values. Additionally, far fewer samples per signal are needed for modulation classification using this technique. Deep-learning is showing to provide not only a far simpler problem but also far higher accuracy if given enough data. Representation learning is a set of methods that allows a machine to be fed with raw data and automatically discover representations needed for detection or classification.[3] Although it may not be entirely understood how the model is making the classification it has proven to be very effective. Deep-learning models are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that transform the representation at one level(starting with raw input) into a representation at a higher, slightly more abstract level.[3] One of the downsides to using a Multi-layer Perceptron(MLP) is performance relies heavily on the amount of data. With the use of a vector signal generator the a plentiful amount of signals and samples can be created in order to drive the MLP to a high accuracy.

2 Model structure and results

2.1 Data collection and generation

108,000 collections were used for training 27,000 for validation and 36,000 for testing. 9 different modulations were used including: BPSK, QPSK, 8PSK, 16QAM, CPFSK, GMSK, FM, AM, and OFDM.

These signals were generated by a vector signal generator and captured over the air using a Universal Software Radio Peripheral (USRP) N210 using GNU Radio to simple log complex floating-point samples to a binary file. 128 complex-signaled samples were generated for each signal and each sample has a real and complex value associated with it. Therefore, each signal used contains 256 features used for evaluation.

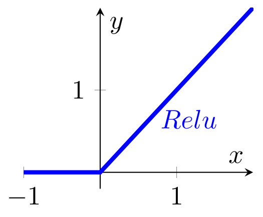
2.2 Building model

Multiple-classification can be rather cumbersome in model design. In Deep-learning, it can also be tempting to design a rather ornate model for network architecture. The model proposed fits within Occam's Razor that the simplest solution is often the best solution. It is valuable to determine the strength of various simple models in order to cross evaluate future architectures.

The multilayer perceptron is a neural network model that consists of stacked layers which train and test on data using assigned weights that are updated through repetitions of training and evaluation of the model.

The model used is a multilayer perceptron consisting of four layers: An input layer with 256 nodes; two dense layers consisting of 750 inputs outputs nodes and a final dense layer with 9 input nodes serving as classification. Dimensions of the input layer should have a 1 to 1 ratio to the number of samples per signal in your dataset.

Each layer uses a 'ReLu' activation for feeding forward the weights as they are updated. 'ReLu'(Recitified linear unit) is an important activation function, because it prevents the vanishing effect on your local gradient. However, you will still get some diminishing on the gradient at the output layer because it uses softmax for classification.



2.3 Training model

Training the model was performed over 20 epochs. Various tests were performed evaluating the number of epochs and options for regularization during training. After about 20 epochs, the model had very little distinctive results. Regularization often seemed to stunt the model and results often went down. The default learning rate of 0.001 was used along with batch sizes of 1024, L2 bias regularization with a constant coefficient of 0.01 was also used for this model to prevent over fitting.

Table 1: Model with optimal results

Batch	Opt	Reg	LR	Network Shape	Network Shape	Epochs	Loss	Acc
1024	Adam	L2(0.01)	0.001	——- Dense Layer ——-		10	0.57	0.80
				Input Dim: 256	Output Dim: 750			
				——- Dense Layer ——-				
				Input Dim: 750	Output Dim: 750	100	1.86	.81
				——- Dense Layer ——-				
				Input Dimensions: 9				

2.3.1 Optimization

For this model the categorical cross-entropy loss function with L2 weight decay regularization[4]:

$$E = -\sum_{i=1}^{N_{samples}} crossEntropy(x_i, y_i) + \lambda \sum_{j=1}^{N_{layers}} \sum_{k=1}^{N^j_{units}} \sum_{l=1}^{N^{j+1}_{units}} (w^j_{k,l})^2$$

Choosing the optimizer is important this can greatly change the overall accuracy.

> The Adam optimizer is a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement. Multi-layer neural networks have non-convex objective functions. It has been empirically found that Adam often outperforms other methods. The method is straightforward to implement, and is computationally efficient. The hyper parameters have intuitive interpretations and typically require little tuning[2]

The default parameters as recommended by the initial paper were implemented. See Table 1

3 Results

The Adam Optimizer was able to correct the model rather swiftly. *Figure 1* shows the training and validation acurracy for the model for 120 consecutive epochs. It shows from *Figure 1* That within about 20 the network reaches its plateau of around 79%-81% accuracy for testing. Before 10 epochs are complete the training passes the validation accuracy, at which point the model is overfitting.

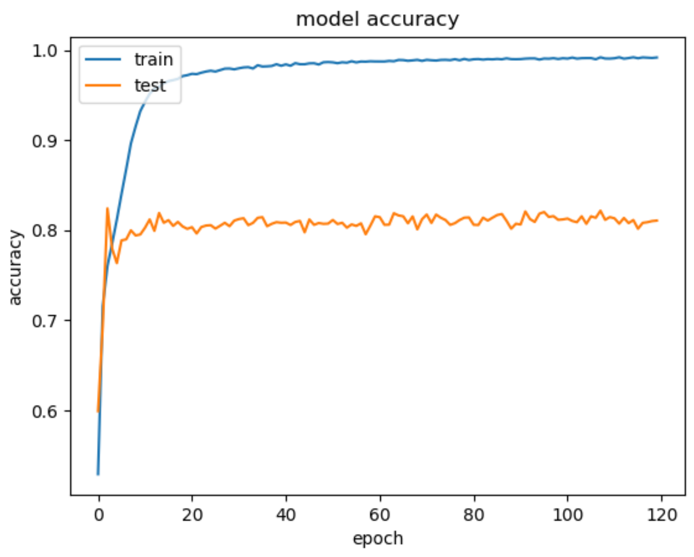


Figure 1: Accuracy Over Epochs

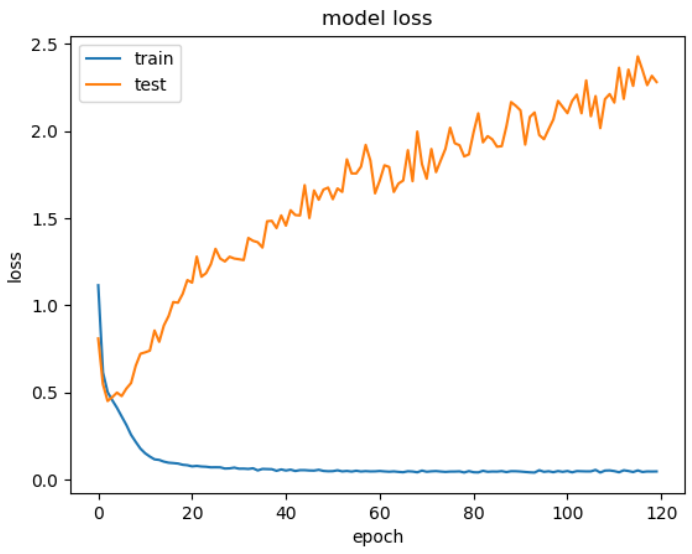
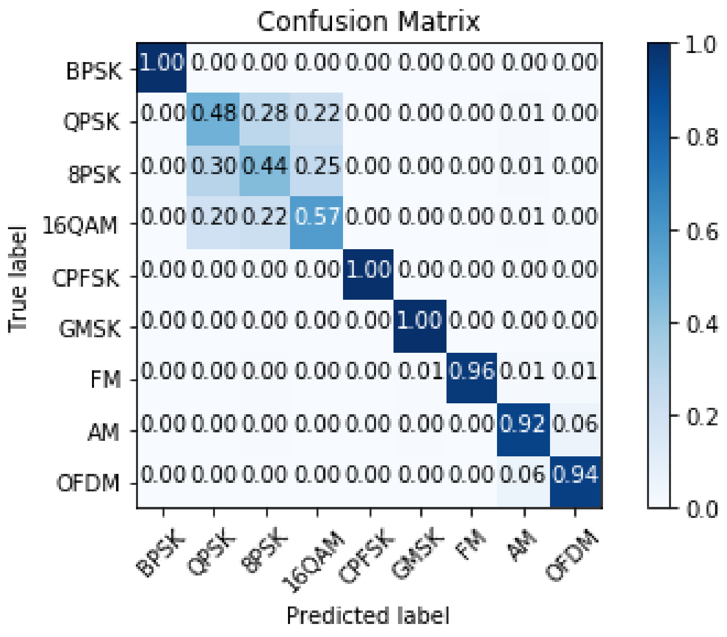


Figure 2: Loss Over Epochs

Figure 2 Shows the model training loss over epochs. Within 10 epochs, the loss begins to diverge between training and validation. As this continues, the loss becomes a greater and greater factor of the accuracy of the model as it grows. From these figures, we can see that training for more than about 20 epochs has little additional value to the accuracy, while it continues to increase in loss. This may be a result of training and testing on short signals. There is a great amount of similarity in the signals at a length of 128 samples and the model may be overfitting after just a short period of training. Curiously, however, results of extending the number of samples for each signal decreased the accuracy of the model.



	BPSK	QPSK	8PSK	16QAM	CPFSK	GMSK	FM	AM	OFDM
BPSK	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
QPSK	0.00	0.48	0.28	0.22	0.00	0.00	0.00	0.01	0.00
8PSK	0.00	0.30	0.44	0.25	0.00	0.00	0.00	0.01	0.00
16QAM	0.00	0.20	0.22	0.57	0.00	0.00	0.00	0.01	0.00
CPFSK	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
GMSK	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
FM	0.00	0.00	0.00	0.00	0.00	0.01	0.96	0.01	0.01
AM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.06
OFDM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.94

Figure 2 Shows the confusion matrix of the results from testing the model. The model was tested on a batchsize of 1024 signals. The horizontal axis represents the predicted label and the vertical axis represents the true label. The outputs represent the weighted number of times the model predicted labe 'X' while label 'Y' was the true label. For example, the model predicted the label 'AM' 6% of the time when the true label was 'OFDM', the model predicted 'OFDM' 94% of the time when 'OFDM' was the true label.

4 Non-Deep Learning Experiments

To show the significance of using Deep-learning the following are results from other traditional machine learning models.

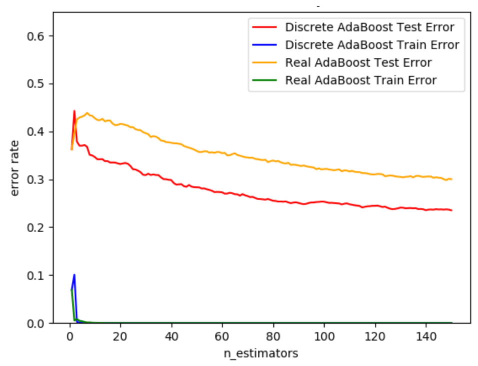


Figure 3: Decision Trees Train/Validation Error Rate with and without Adaboost

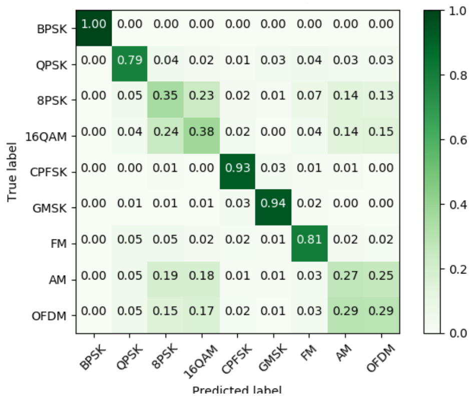


Figure 4: Testing Accuracy = 60%

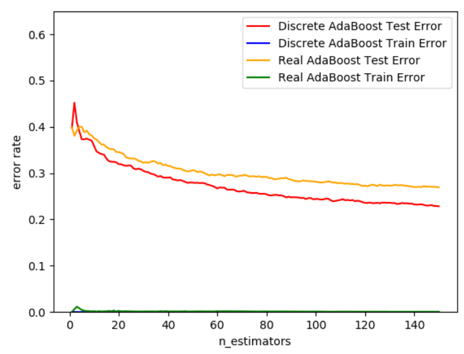


Figure 5: Extra Trees Trees Train/Validation Error Rate with and without Adaboost

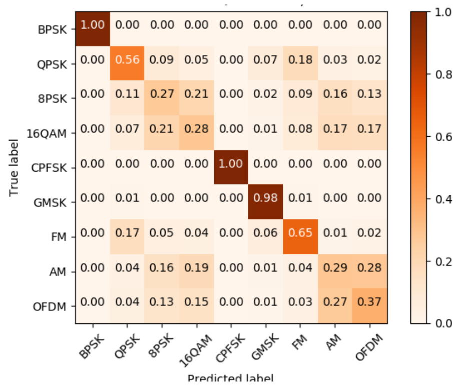


Figure 6: Testing Accuracy = 60%

5 Conclusion

Due to the similarities between modulations such as 8PSK and QPSK the confidence rating significantly decreases. However, when comparing vastly different methods of modulation the ability to intelligently differentiate and learn is evident. Using the Adam optimizer, the model described can learn rather quickly, which is crucial to the real-time learning problem. With relatively simple models, implementation of Deep-learning into wireless communication systems is achievable.

References

[1] A. Hazza and M. Shoaib, "An Overview of Feature-Based Methods for Digital Modulation Classification," *Conference on Communications, Signal Processing, and their applications, vol. 1, no. 8,2013*

[2] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[3] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436.

[4] Bengio, Yoshua. "Practical recommendations for gradient-based training of deep architectures." Neural networks: Tricks of the trade. Springer, Berlin, Heidelberg, 2012. 437-478.