CPE 646 Pattern Recognition and Classification Final Project Report

Modulation Classificaton Using CNNs

May 14, 2022

Joshua Hwang
Tim Demetriades
Gianna Miggins

Abstract

Radio frequency signals are modulated when traveling from the transmitter to the receiver. In order to demodulate the signal, the receiver must know what kind of modulation was used. By grouping the unique characteristics of each waveform together, the model should be able to interpret a snapshot of the signal and place it into the correct category to be properly demodulated. A software defined radio system implements modulating/demodulating by connecting to an embedded system where signal processing occurs. The goal of this project is to classify different types of modulations from simulated sampled data, as well as images of the sampled data. Utilizing python libraries and artificially generated waveforms using MatLab led to two working models that combined the successes of previously made models. The proposed models using the waveforms I/Q components with Convolutional Neural Networks did not turn out to be the most accurate model but they explored a new way of examining the data that can be tweaked in the future to obtain more desirable results.

Introduction

As our use of technology on the go continues to expand, radio systems must also be improved to better suit our needs. The future of things like military communication, GPS usage, and the listening of podcasts will rely on software-defined radio (SDR) systems to adapt to bandwidth availability, improve sound quality, and transmit a wide variety of waveforms. This means that the signal receiver must be able to identify the modulation type and demodulate any input. Modulation can be analog or digital and the most common analog modulation types are amplitude modulation (AM) and frequency modulation (FM).

In a study done at The Pennsylvania State University, the modulation classification focused on analog modulation only because it is less complex and easier to identify [*Bilen*]. However, digital modulation is very widely used today and is more secure than analog. An article published in the Mathematical Problems in Engineering focused on only digital modulation and concluded that their proposed method reached a 93.76% accuracy and outperformed other successful models such as LSTM, CNN, MentorNet and ResNet [*Wu*].

Model

The sample data for this project was found in a MATLAB example, “Modulation Classification with Deep Learning[MATLAB].” In the example, a model exists, having an accuracy estimated at 94%. While using the same dataset, two approaches were done with the goal of beating these results. One model was based off of the MATLAB example’s model, utilizing a Convolutional Neural Network. The second model utilizes a CNN as well, but on image data. The data utilized in this model used only two classes rather than the 11 that the first model had, as well as much less data (1,000 samples rather than the original 110,000).

The sample data consists of 10,000 frames for each modulation type (110,000 frames total for the 11 classes). Each frame is made up of 1024 in-phase and quadrature samples, two rows and 1,024 columns. The data downloaded from MATLAB for our use was structures already split into training, validation, and testing, 80:10:10. SciPy, an open-source library with the capability to interpret .mat files, was used to import the sample data. After manipulating the data which was a dictionary object, the already separated data could be set up for training the model.

For the second dataset (second approach / Model 1), much less data was used. Now of the 11 different modulation techniques, two were chosen, “Broadcast-FM (B-FM)” and “Double sideband amplitude modulation (DSB-AM)” so that binary classification could be conducted. Five hundred frames from each modulation type were taken from the sample pool (compared to the original 10,000 frames per class). These frames were plotted (Amplitude vs Time for both the real and imaginary parts of the waveforms) in MATLAB and saved as .jpeg files for image classification.

Below are the 11 categories of modulation included in the study (8 digital and 3 analog).

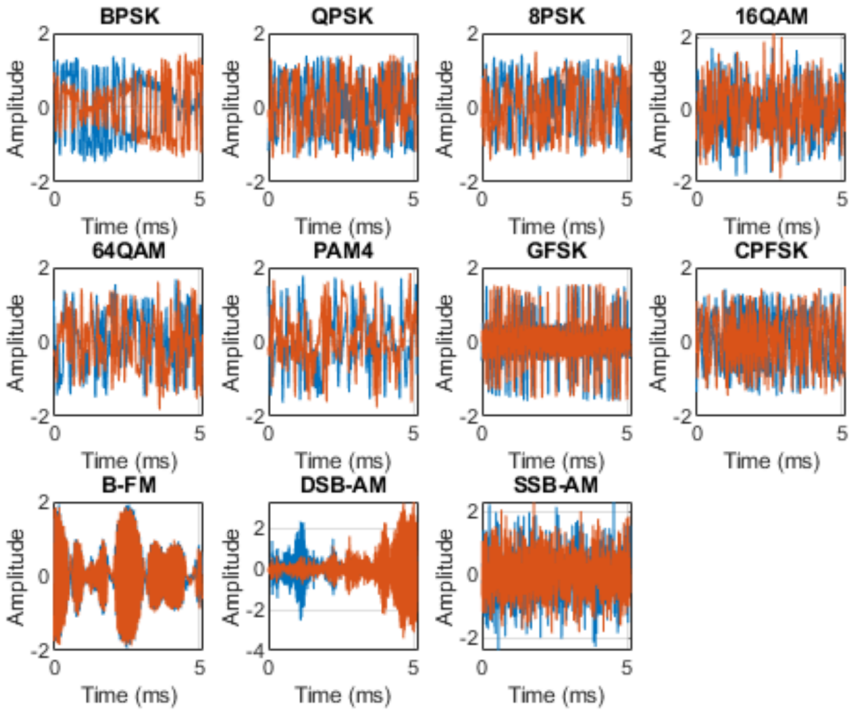
Digital

- Binary phase shift keying (BPSK)
- Quadrature phase shift keying (QPSK)
- 8-ary phase shift keying (8-PSK)
- 16-ary quadrature amplitude modulation (16-QAM)
- 64-ary quadrature amplitude modulation (64-QAM)
- 4-ary pulse amplitude modulation (PAM4)
- Gaussian frequency shift keying (GFSK)
- Continuous phase frequency shift keying (CPFSK)

Analog

- Broadcast FM (B-FM)
- Double sideband amplitude modulation (DSB-AM)
- Single sideband amplitude modulation (SSB-AM)

Below are plots of the amplitude of both the real and imaginary part of one random waveform of each modulation type. These plots are what was used for the image classification approach (Model 1).



Model 1

Since we are dealing with image data, it makes sense to make use of Convolutional Neural Networks (CNNs) for the classification of the modulations. CNNs are great at extracting local features from images through the use of convolution operations, which involves sliding a filter over the image and calculating the convolution product of each portion of the image. This results in a feature map that is used to determine where the features are in the image.

Following the convolution layers are max pooling layers, which essentially shrinks the feature maps received from the convolution layers in order to reduce computational load, memory usage, and the number of parameters in the model. This allows for the model to be trained much faster while still preserving important characteristics. Finally, the last few layers of a CNN consist of fully connected layers. This is where the relationship between the position of features in the image and the class are determined for classification.

As mentioned, the data fed to this CNN model was images, specifically 256x256 color images of the Amplitude vs Time plots for the real and imaginary parts of the modulated waveforms. To simplify things, two modulation types were selected, B-FM and DSB-AM, so that binary classification could be done. These are both analog modulations and were chosen based on how different their plots look which would allow the model to have an easier time distinguishing them.

For this model, 500 images for each class were used, with 90% being designated for training (and 10% of that being designated for validation) and the other 10% being used for testing. The model that gave the group the best results consisted of 4 convolution layers (ranging from 16 to 64 filters each), with each followed by a relu activation layer, a max pooling layer, and a batch normalization layer. After this the data was flattened and fed to a fully-connected dense layer of 512 nodes. A dropout layer was then added to help prevent overfitting. Finally, the model ended with a 1 node dense layer with sigmoid activation, since the data was binary. The results for this model can be seen in the following section.

Model 2

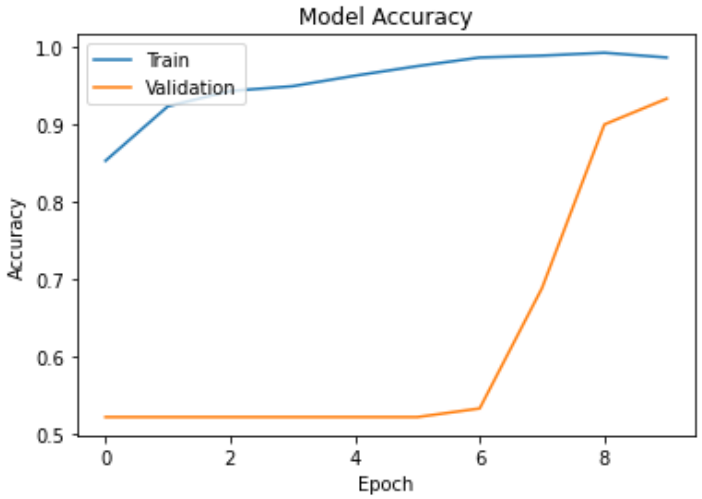
As for the second approach, each input of the model was a 2x1024 (with 2 rows corresponding to the real and imaginary part of the waverforms) matrix having an output vector with 11 elements, determining which modulation the input can be classified as. One-hot-encoding was used to binarize the output labels, which were originally integers between 1 and 11. The second model used a similar method as the one used in the

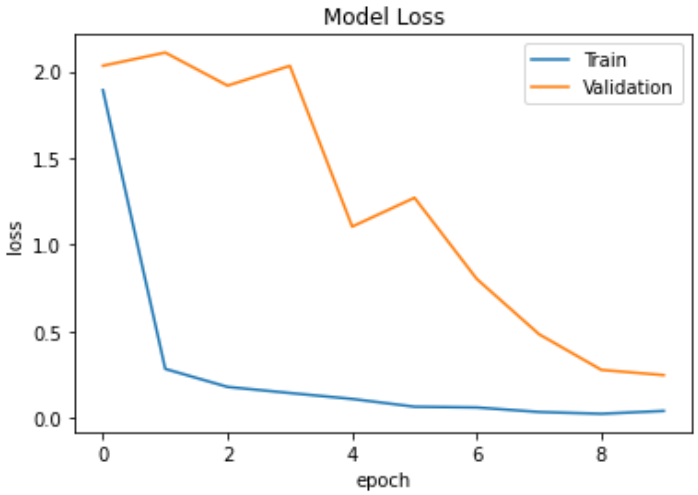
MATLAB example, but with fewer convolutional layers. This model consisted of four convolutional layers, max pooling layers, batch normalization layers, and a fully connected-layer. While the first model's input expected an image (represented as pixel values), this second model's input took vectors. Along with the convolutional layers, activation functions were set to "ReLU," nulling negative inputs.

Results

Model 1 performed surprisingly well with a test accuracy of 94% which was comparable with the CNN model in the MATLAB example. As seen in the images below, both the training and validation accuracy can be seen increasing over time and converging, which shows the validity of the high accuracy as there is no sign of overfitting (the validation accuracy never decreases). The model's loss can also be seen decreasing over time, with the training and validation components converging. This shows the model has been trained well with little error at the eighth epoch.

Below are plots of the training and validation accuracy and loss over time for Model 1.





The second model, however, classifying 11 modulation types and using a different form of the data, performed considerably worse. This is likely a result of the complexity of this model and its data. If given more time, the team could have attempted to play around with the data to get it into a form that the model would have better been able to handle. Regardless, the team was satisfied with the results of the first model, as they managed to compare with the results from the MatLab example (test accuracy of 94%).

Running the Model

How to compile and run code

The code was written in Google Colab notebooks in order to take advantage of its GPU runtime, as CNNs can take very long to train when just using your computers CPU, especially if large images are being used. The data was stored as .jpg files for Model 1 and .mat files for Model 2. Simply unzip the project file and save all of the contents to one folder, making sure the data folders match the paths they are expected to be in within the code. Open the Python program in either a Jupyter Notebook or Google Colab and run the code blocks in order.

Contribution Breakdown

The team communicated virtually about the project ideas and updated all shared files as progress was made. Completing the report was a collaborative effort where everyone took part in writing and proofreading.

Gianna researched previous applications of this idea to gather attempted approaches. This gave the team ideas of how to improve previously used models by

incorporating the best practices and experimenting with new workflows. She compared the team's results with the previous study results.

Josh found the available dataset of signals since we could not collect this information on our own. He made sure the dataset was complete and did not consist of any null values. He worked on Model 2.

Tim compiled most of the code and trained the model with various combinations of parameters to optimize the testing results. He worked on Model 1. He also created visual representations of the results from the model.

Resources

“Modulation Classification with Deep Learning.” *MATLAB & Simulink*, https://www.mathworks.com/help/deeplearning/ug/modulation-classification-with-deep-learning.html.

Peng Wu, Bei Sun, Shaojing Su, Junyu Wei, Jinhui Zhao, Xudong Wen, "Automatic Modulation Classification Based on Deep Learning for Software-Defined Radio", Mathematical Problems in Engineering, vol. 2020, Article ID 2678310, 2020. https://doi.org/10.1155/2020/2678310

Sven G. Bilén, Andrew Price, Okhtay Azarmanesh, Julio Urbina, “Modulation Classification for Radio Interoperability Via SDR”, SDR Forum Technical Conference 2007. https://www.wirelessinnovation.org/assets/Proceedings/2007/2007-sdr07-1.2-4-bilen.pdf