WHITE PAPER

Deep Learning Using Synthesized Data for Communications and Radar

MathWorks®

This white paper demonstrates how you can use MATLAB® for modulation identification and target classification in radar and communications applications. You will learn how to:

1. Synthesize and label radar and communications waveforms
2. Generate radar returns for moving objects
3. Train deep networks with synthetic data
4. Perform waveform modulation ID and target classification using deep learning and machine learning techniques
5. Test your systems with data collected from software defined radios and radar systems

Three application examples demonstrate possible workflows. The first two examples use deep learning to identify waveform modulation types. The third example uses radar returns to identify objects based on radar cross section (RCS) and motion characteristics.

The examples show how you can synthesize communications and radar baseband waveforms and radar reflections off objects using Phased Array System Toolbox™ and Communications Toolbox™ and configure, train, and implement learning networks using Deep Learning Toolbox™ and Statistics and Machine Learning Toolbox™.

Background

Modulation identification and target classification are important functions for intelligent receivers. These functions have numerous applications in cognitive radar, software-defined radio (SDR), and efficient spectrum management. To identify both communications and radar waveforms, it is necessary to classify them by modulation type. For this, you can extract meaningful features that can be input to a classifier. While effective, this procedure can require effort and domain knowledge to yield an accurate identification. A similar challenge exists for target classification.

This white paper shows a framework to automatically extract time-frequency features from received signals, which you can then use to perform classification with a deep learning network. You can use a range of techniques to feed signals to a deep learning network, as shown in the examples below

To support these workflows, you can generate and label synthetic, channel-impaired waveforms. These generated waveforms in turn provide the training data for a range of deep learning networks. For target classification, you can also model objects and simulate radar reflections off these targets. In both cases, you can validate the resulting system with over-the-air signals from SDRs and radars.

Figure 1 shows the workflow for modulation identification and classification. You can follow a similar workflow for object classification using radar returns.

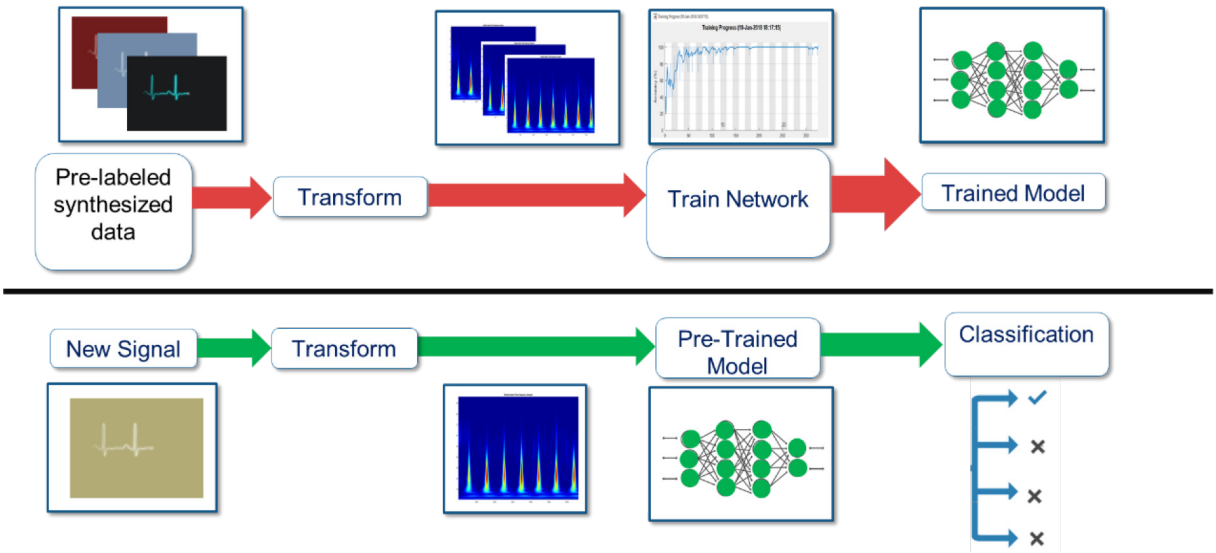

Figure 1. Modulation identification workflow with deep learning using MATLAB.

Waveform Modulation Identification

Modulation identification is challenging because of the range of waveforms that exist in any given frequency band. In addition to the crowded spectrum, the environment can be harsh in terms of propagation conditions and non-cooperative interference sources. Questions that may arise when you are doing modulation identification include:

- How will these signals present themselves to the receiver?
- How should unexpected signals, which haven't been received before, be handled?
- How do the signals interact or interfere with each other?

You can apply machine learning and deep learning techniques to help with modulation identification. To start, consider the tradeoff between the time required to manually extract features to train a machine learning algorithm versus the large datasets required to train a deep learning network. Another consideration is the computational requirements of each approach. Figure 2 shows how these tradeoffs relate to each other.

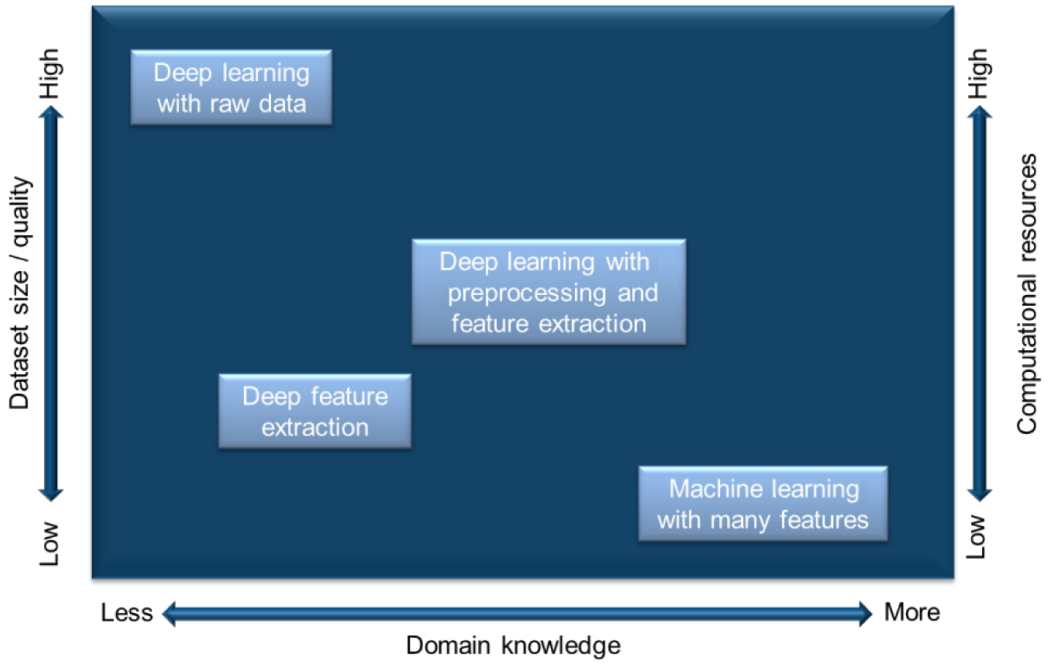

Figure 2. The tradeoffs between training dataset size, domain knowledge, and computational resources required.

Manually extracting features can take time, and the process requires detailed knowledge of the signals. On the other hand, deep learning networks require large amounts of data for training purposes to ensure the best results. One benefit of using a deep learning network is that less preprocessing work and manual feature extraction is required.

With the requirements for perception growing for autonomous driving and computer vision in general, organizations are making great investments in image and vision-based learning that other signal-based applications such as radar and communications can take advantage of. This is true whether the datasets include raw data or preprocessed data.

In the examples below, deep learning networks perform the "heavy lifting" in terms of classification, so you can focus on the best way to satisfy the dataset requirements for training and validation. Data can be generated from fielded systems, but it can be challenging to collect and label this data. Keeping track of waveforms and syncing transmit and receive systems results in large datasets that can be difficult to manage. It is also a challenge to coordinate data sources that are not geographically colocated including tests that span a wide range of conditions. In addition, labeling this data either as it is collected or after the fact requires significant effort because ground truth may not always be available or reliable.

Another option is to use synthetic data, because it can be much easier to generate, manage, and label. The question is whether the fidelity of the synthetic data is sufficient. In the use cases that follow, you will see that generating high-fidelity synthetic data is possible.

Synthesizing Radar and Communications Waveforms

As previously noted, the occupied frequency spectrum is crowded, and transmitting sources such as communications systems, radio, and navigation systems all compete for spectrum. To create a test scenario, consider the following diverse set of waveform types:

- Rectangular
- Linear frequency modulation (LFM)
- Barker code
- Gaussian frequency shift keying (GFSK)
- Continuous phase frequency shift keying (CPFSK)
- Broadcast frequency modulation (B-FM)
- Double sideband amplitude modulation (DSB-AM)
- Single sideband amplitude modulation (SSB-AM)

With these waveforms defined, you can programmatically generate large numbers of I/Q signals for each modulation type. Each waveform has unique parameters, and the resulting signals are perturbed with various impairments to increase the fidelity of the model. For each waveform, the pulse width and repetition frequency are randomly generated. For LFM waveforms, the sweep bandwidth and chirp direction are randomly generated. For Barker waveforms, the chip width and number are generated randomly. All signals are impaired with white Gaussian noise. In addition, a frequency offset with a random carrier frequency is applied to each signal. Finally, each signal is passed through a channel model. In this example, a multipath Rician fading channel is implemented, but other models are available and could be used instead.

The data is labeled as it generated in preparation to feed the training network.

Feature Extraction Using Time-Frequency Techniques

To improve the classification performance of learning algorithms, a common approach is to input extracted features in place of the original signal data. The features provide a representation of the input data that makes it easier for a classification algorithm to discriminate across the classes.

In practical applications, many signals are nonstationary. This means that their frequency-domain representation changes over time. One useful technique to extract features is the time-frequency transform, which results in an image that you can input into the classification algorithm. The time-frequency transform helps to identify if a particular frequency component or intermittent interference is present in the signal of interest.

Many automated techniques are available for time-frequency transforms including spectrogram and continuous wavelet transforms (CWT), but the Wigner-Ville distribution (WVD) is a good algorithm to start with because it provides good spectral resolution without the leakage effects present in other techniques.

The Wigner-Ville distribution represents a time-frequency view of the original data that is useful for time-varying signals. The high resolution and locality in both time and frequency provide good features to help differentiate modulation types that are similar. However, the WVD performance degrades due to cross terms from multiple frequency components. Alternatively, you can compute the smoothed pseudo WVD for each of the modulation types.. Figure 3 shows the downsampled images for one set of data.

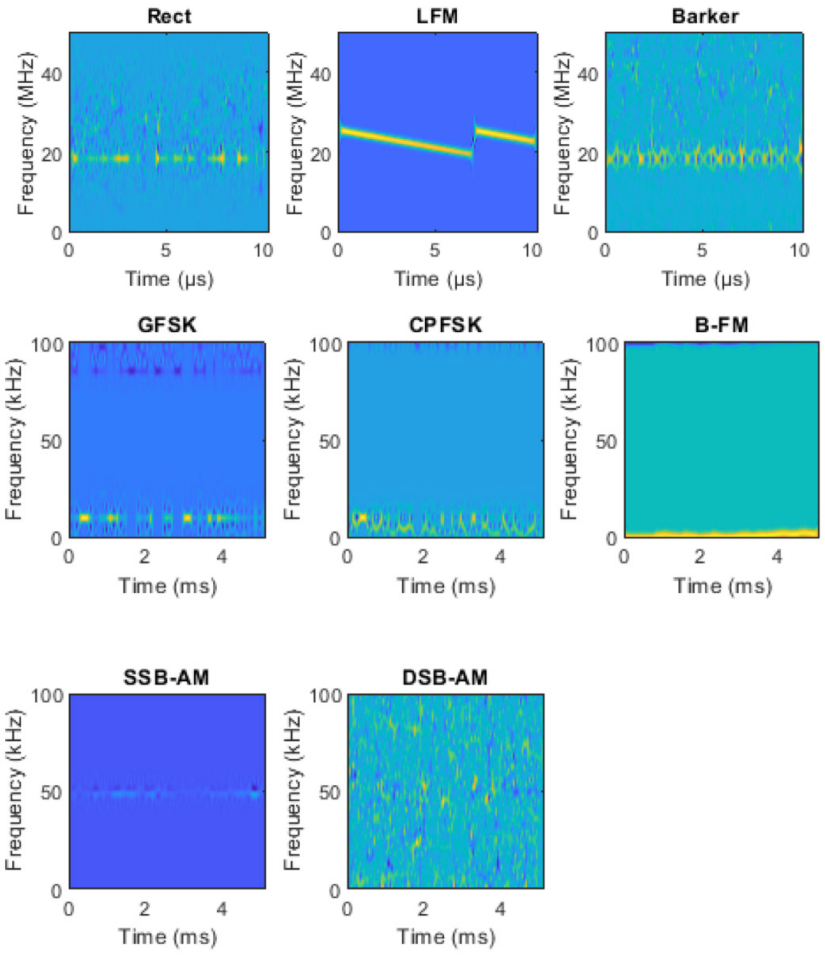

Figure 3. Time-frequency representations of radar and communications waveforms.

You can use these images to train a deep convolutional neural network (CNN). The network uses 80% of the dataset for training, 10% for testing, and 10% for validation.

Set Up and Train the Deep Learning Network

Before you can train the deep learning network, you must define the network architecture. The results for this example were obtained using transfer learning with AlexNet, which is a deep CNN created for image classification. Transfer learning is used to retrain the existing neural network to classify new targets. AlexNet performs classification of 1000 categories in its default configuration. To tune AlexNet for this dataset, we modify the final three classification layers so that they classify only our eight modulation types. This workflow can also be accomplished with other smaller-footprint networks such as SqueezeNet.

Once you have created the CNN, you can begin training. Due to the dataset's large size, it may be best to accelerate the work with either a GPU or multicore processor. Figure 4 shows the training progress as a function of time using a GPU to accelerate the training. Training progress is expressed as accuracy as a function of the number of iterations. The validation accuracy is over 97%.

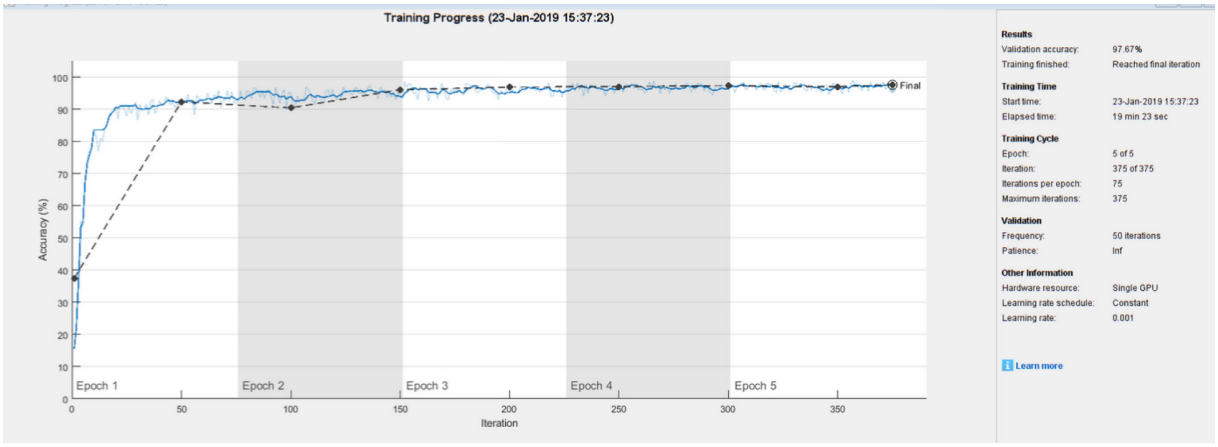

Figure 4. Training progress using GPU to accelerate process.

Evaluating the Performance

Recall that 10% of the generated data was reserved for testing. For the eight modulation types input to the network, over 99% of B-FM, CPFSK, GFSK, Barker, Rectangular, and LFM modulation types were correctly classified. On average, over 85% of AM signals were correctly identified. As shown in the confusion matrix in Figure 5, a high percentage of DSB-AM signals were misclassified as SSB-AM and SSB-AM as DSB-AM.

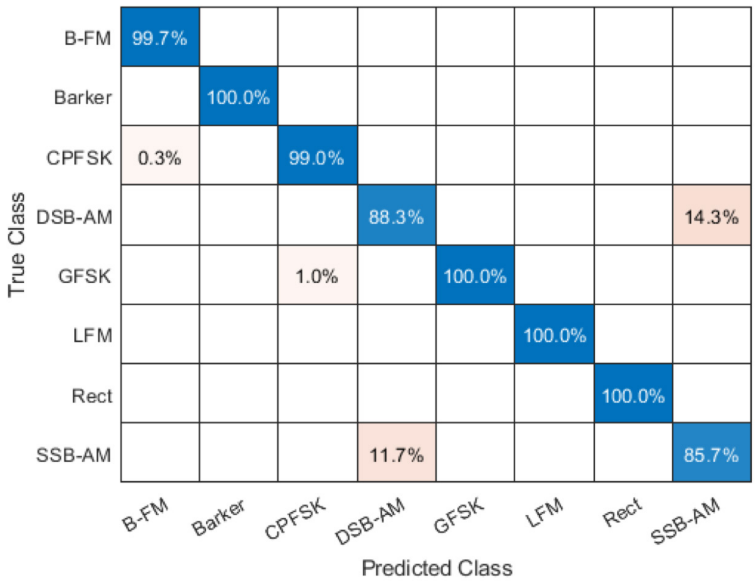

Figure 5. Confusion matrix with the results of the classification.

The framework for this workflow enables the investigation of the misclassifications to gain insight into the network's learning process. Since DSB-AM and SSB-AM signals have a very similar signature, this explains in part the network's difficulty in correctly classifying these two types. Further signal processing such as the continuous wavelet transform (CWT) could make the differences between these two modulation types clearer to the network and result in improved classification. Specifically, the CWT provides better multiresolution analysis, which provides the network with better information related to abrupt changes in the frequency of the signals.

In the next example, you will see an alternate technique to input data into the network. In addition, you will see how you can use data from a radio for the testing phase.

Alternate Approaches Using I/Q Data

To start, a new dataset is generated that includes the following 11 modulation types (8 digital and 3 analog):

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

Here, 10,000 frames for each modulation type are generated. Again, the network uses 80% of the data for training, 10% for validation, and 10% for testing, as shown in Figure 6.

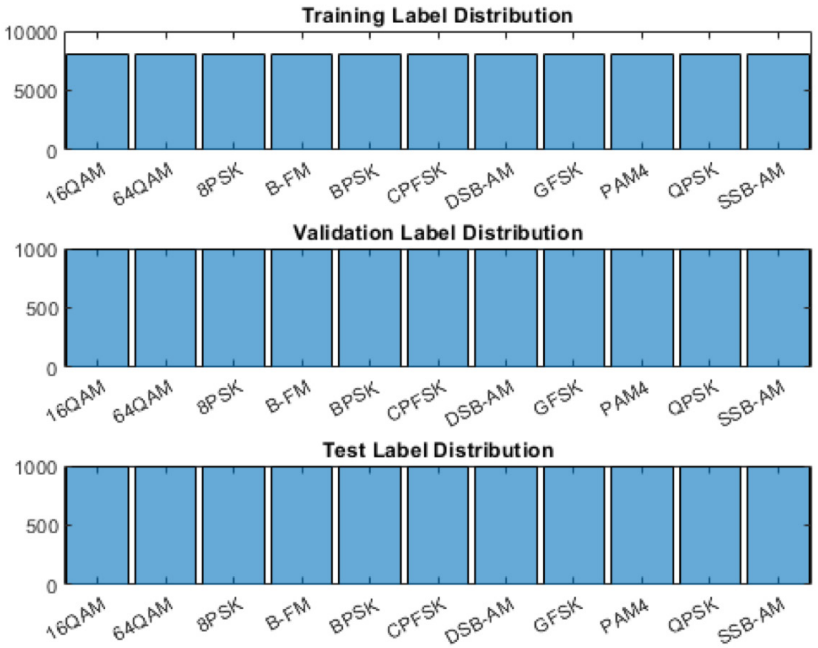

Figure 6. Distribution of labeled dataset by waveform type.

For digital modulation types, eight samples are used to represent a symbol. The network makes each decision based on single frames rather than on multiple consecutive frames. Similar to our first example, each signal passes through a channel with additive white Gaussian noise (AWGN), Rician multipath fading, and a random clock offset. The resulting channel-impaired frames for each modulation type are stored with their corresponding labels. To make the scenario more realistic, a random number of samples are removed from the beginning of each frame to remove transients and to make sure that the frames have a random starting point with respect to the symbol boundaries. Figure 7 shows the time and time-frequency representations of each waveform type.

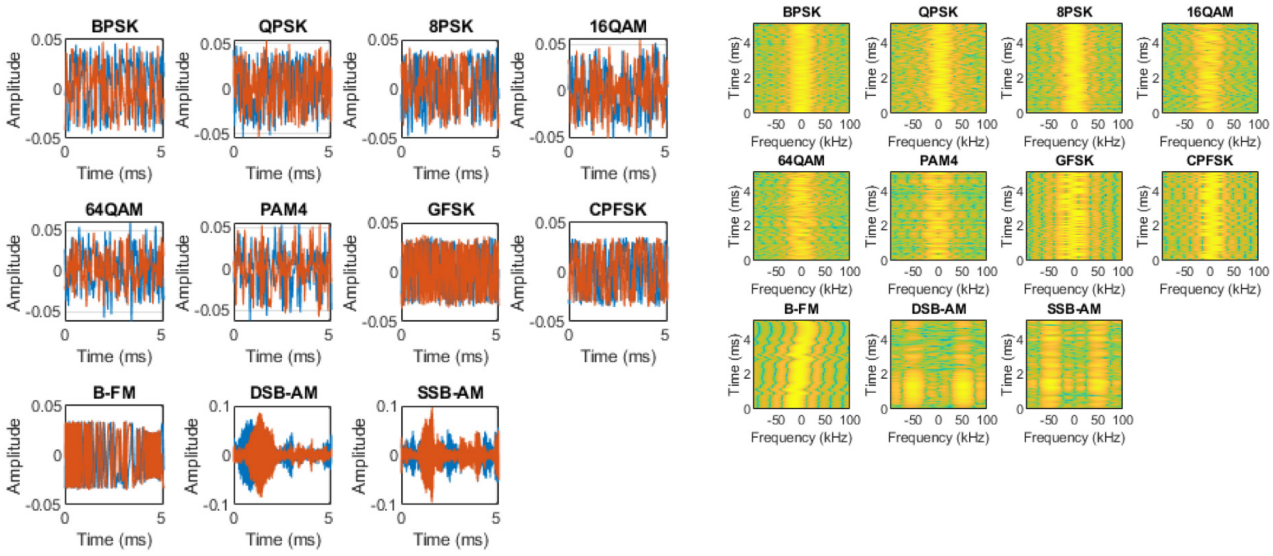

Figure 7. Example of time representation of generated waveforms (left) and corresponding time-frequency representations (right).

Train the CNN and Evaluate the Results

This example uses a CNN that consists of six convolution layers and one fully connected layer. Each convolution layer except the last is followed by a batch normalization layer, rectified linear unit (ReLU) activation layer, and max pooling layer. In the last convolution layer, the max pooling layer is replaced with an average pooling layer. To train the network, a GPU again is used to accelerate the process.

In the previous example, each image was transformed to an image. This example implements an alternate approach in which the I/Q baseband samples are used directly without further preprocessing.

To do this, you can use the I/Q baseband samples in rows as part of a 2D array. Here, the convolutional layers process in-phase and quadrature components independently. Only in the fully connected layer is information from the in-phase and quadrature components combined. This approach yields a 90% accuracy.

A variant on this approach is to use the I/Q samples as a 3D array where in-phase and quadrature components are part of the third dimension (pages). This approach mixes the information in the I and Q evenly in the convolutional layers and makes better use of the phase information.

As the confusion matrix in Figure 8 shows, representing I/Q components as pages instead of rows dramatically increases the ability of the network to accurately differentiate 16-QAM and 64-QAM frames and QPSK and 8-PSK frames. The variant yields an overall accuracy of 95%, an increase of about 5% over the previous approach.

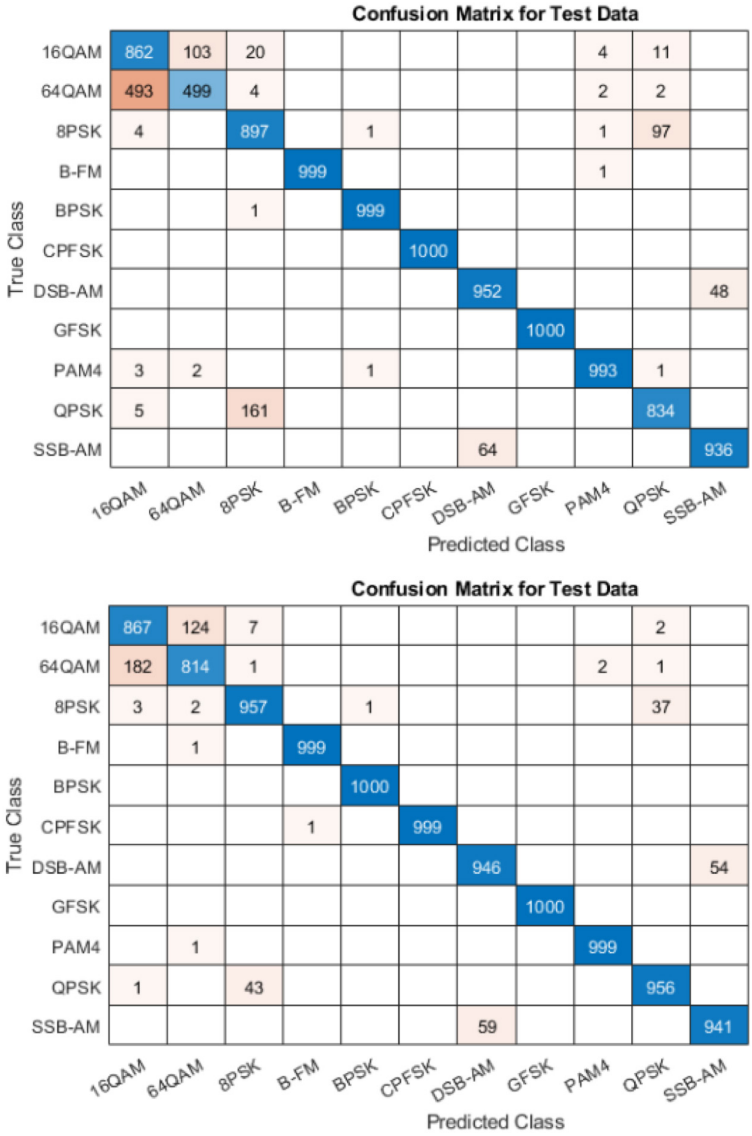

Figure 8. Results with I/Q components represented as rows (top) and as pages (bottom).

Testing with SDR and Radar Systems

You can test your system using only synthesized data. You can also use over-the-air signals to test your system. To demonstrate this, communications waveforms were generated from two ADALM-PLUTO software defined radios that are stationary and configured on a desktop. The network achieves 99% overall accuracy as shown in Figure 9. This is better than the results obtained for synthetic data because of the simple nature of the desktop configuration. The workflow can be extended for radar and radio data collected in more realistic scenarios.

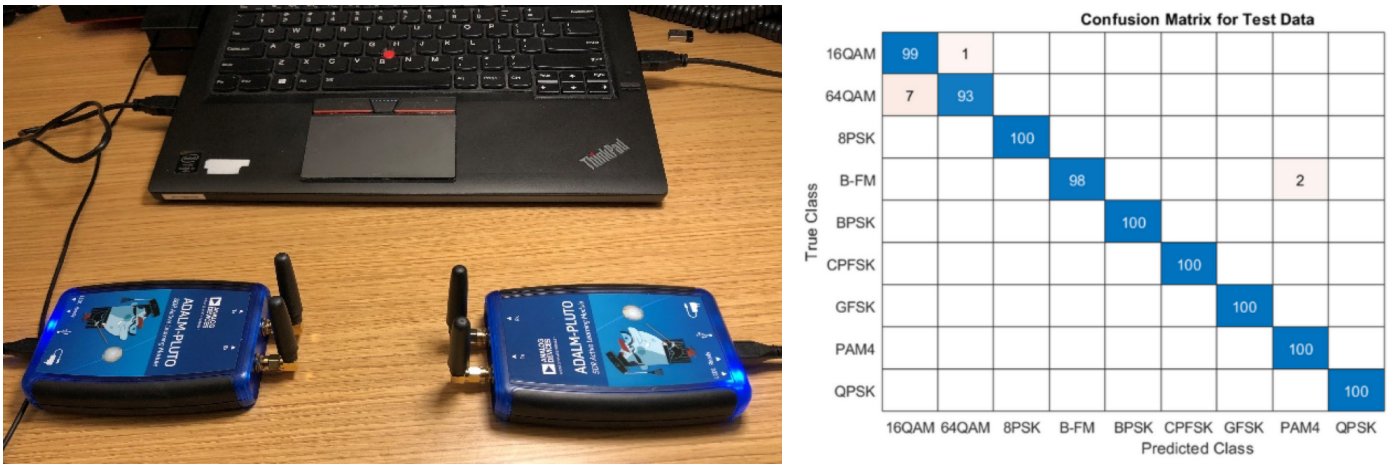

Figure 9. SDR configuration using ADALM-PLUTO radios (left) and corresponding confusion matrix (right).

Figure 10 shows the DemoRad platform from Analog Devices, which also interfaces directly to MATLAB. You can collect data and process baseband data from this software-defined radar. You can use this type of radar system to collect data and identify it by extending the example described above.

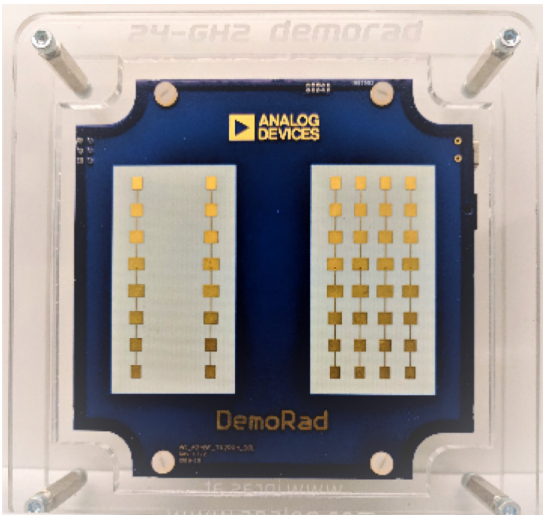

Figure 10. The Analog Devices DemoRad radar evaluation board.

Object Classification

As described above, radar waveforms can be identified using deep learning techniques, but they can also be used as part of a larger radar system model. In this type of simulation, waveforms are transmitted out an array, passed through a propagation channel, reflected off objects in the radar field of view, and received back at the radar. In this next example, you will see how you can apply machine learning techniques to classify an object in the radar field of view.

Objects can be modeled with an RCS pattern that varies with angle frequency. In this simple example, a cylinder and cone represent objects in the radar field of view. Figure 11 shows the RCS of a cylinder over different aspect angles. Also shown is an example of a motion model profile for each object and the corresponding radar return for a single dataset. In this case, it is assumed that the shapes undergo a motion that causes small vibrations around bore sight; as a result, the aspect angle changes from one sample to the next.

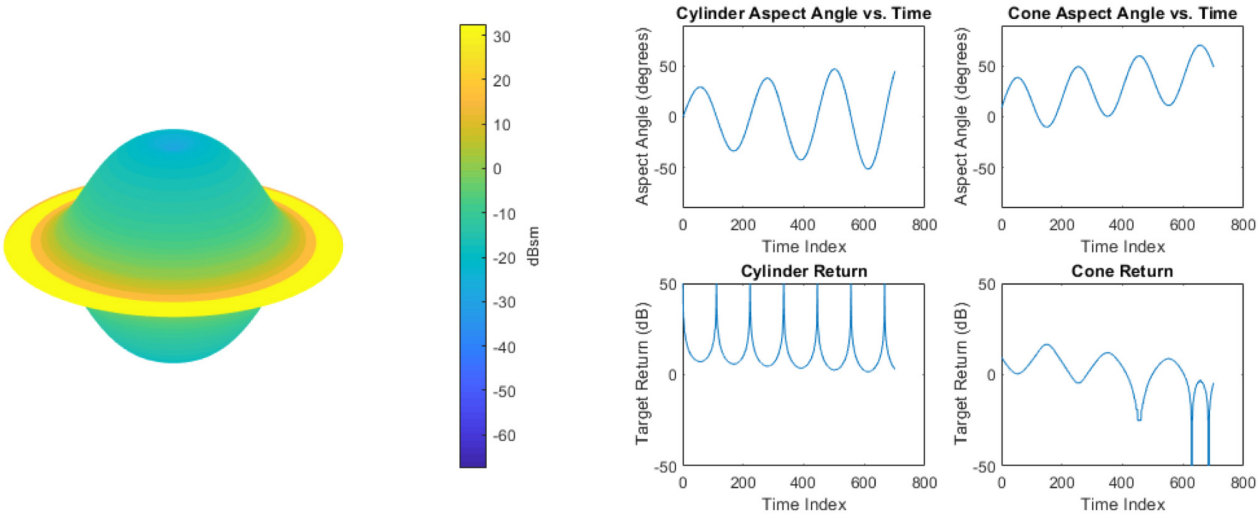

Figure 11. RCS of cylinder (left) and example of motion model and radar return for a cylinder and cone.

You can generate the radar returns off the cone, along with many other shapes, similarly. Using this modeling and simulation as a basis, you can generate many datasets to train a network with different shape characteristics and motion variants. This can be a more efficient way to generate data before moving to your hardware in the field and facing the same collection and labeling challenges discussed in the earlier sections.

To improve the matching performance of learning algorithms, you can use feature extraction to make it easier for the classification algorithm to discriminate between returns from different targets. In addition, the features are often smaller than the original signal so fewer computational resources are required for learning.

The time-frequency signature of the signal in this example uses a wavelet packet representation of the signal. Figure 12 shows the wavelet packet signatures for both the cone and the cylinder. These signatures provide some insight that the learning algorithms will be able to distinguish between the two. Specifically, there is separation between the frequency content over time between the two signatures.

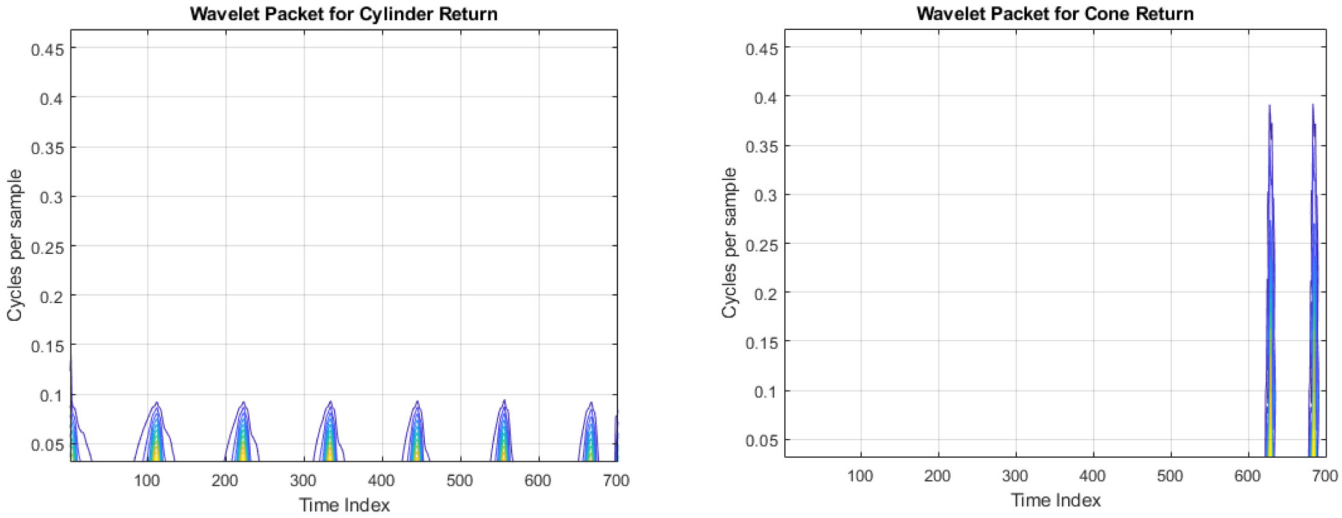

Figure 12. Wavelet transform representation for a cylinder and cone.

The apparent frequency separation between the cylinder and cone returns suggests using a frequency-domain measure to classify the signals.

The extracted feature set consists of 8 predictors per target return. Compared with the original time domain signal of 701 points, this is a significant reduction in the data. You can tune the number of levels for the wavelet transform to improve performance of the classification algorithms.

You can use the Classification Learner app, shown in Figure 13, to train the classifier. Once the training data is loaded, the app can help apply different learning algorithms against the dataset and report back the classification accuracy.

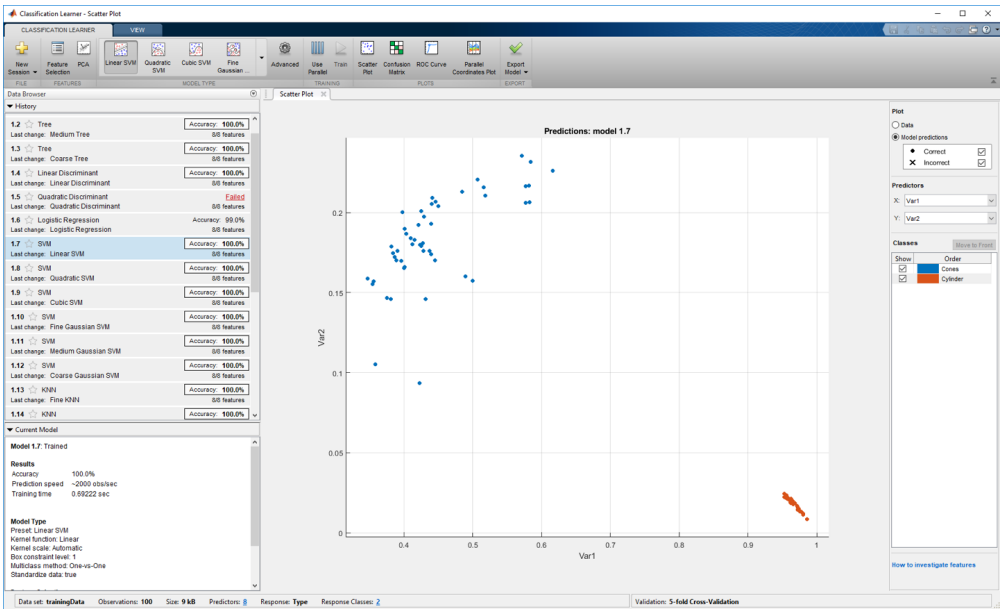

Figure 13. The Classification Learner app.

Once the model is ready, the network can process the received target return and perform classification. This data is passed through the derived classifier to see if it can correctly classify the two shapes. The test data contains 25 cylinder returns and 25 cone returns. These cylinders and cones consist of 5 sizes for each shape and 5 motion profiles for each size. The generation process is the same as the training data, but the specific values are slightly different due to randomness of the size values and incident angle values. The total number of samples for each shape is 701x25 and the resulting accuracy is 82%.

You can improve the performance of the classifier by increasing the quality and quantity of the training data. In addition, the feature extraction process can be improved to further distinguish characteristics of each target within the classification algorithm. Note that different features may have different optimal classification algorithms.

For more complex datasets, a deep learning workflow using a convolutional neural network and a long short-term memory (LSTM) recurrent neural network will also improve performance. These workflows are demonstrated in more detail at the links below.

Summary

MATLAB enables you to automatically extract time-frequency features from signals. You can use these features to perform modulation and target classification with a deep learning network. You can generate and label synthetic, channel-impaired waveforms and radar returns, which can augment or replace live data for training purposes. You can then validate these types of systems with over-the-air signals from software-defined radios (SDR) and radars.

Learn More

For more detailed descriptions and MATLAB scripts for the above examples, see:

- ***Radar Waveform Classification Using Deep Learning*** - Example
- ***Modulation Classification with Deep Learning*** - Example
- ***Radar Target Classification Using Machine Learning*** - Example