Resources

Spectrum Sensing with Deep Learning to Identify 5G, LTE, and WLAN Signals

Since R2021b

This example shows how to train a semantic segmentation network using deep learning for spectrum monitoring. One of the uses of spectrum monitoring is to characterize spectrum occupancy. The neural network in this example is trained to identify 5G NR, LTE, and WLAN signals in a wideband spectrogram.

This example uses:

5G Toolbox

Communications Toolbox

Computer Vision Toolbox

Deep Learning Toolbox

LTE Toolbox

Parallel Computing Toolbox

WLAN Toolbox

Open in MATLAB Online

Copy Command

Introduction

Computer vision uses the semantic segmentation technique to identify objects and their locations in an image or a video. In wireless signal processing, the objects of interest are wireless signals, and the locations of the objects are the frequency and time occupied by the signals. In this example we apply the semantic segmentation technique to wireless signals to identify spectral content in a wideband spectrogram.

In the following, you will:

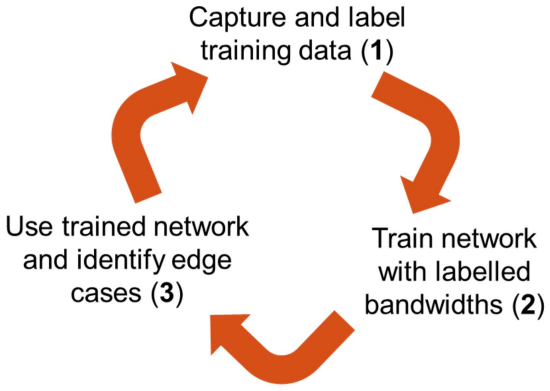
1. Generate training signals.
2. Use a semantic segmentation network to identify 5G NR, LTE, and WLAN signals in time and frequency. You have the choice of training a network from scratch or applying transfer learning to a pretrained network.
3. Test the trained network with synthetic signals.
4. Use an SDR to test the network with over the air (OTA) signals.

Associated AI for Wireless Examples

Use this example as part of a complete deep learning workflow:

1. The Capture and Label NR and LTE Signals for AI Training (Wireless Testbench) example shows how to scan, capture, and label bandwidths with 5G NR and LTE signals using an SDR.
2. The WLAN Activity Scanner (Wireless Testbench) example shows how to scan and capture WLAN signals using an SDR.
3. This example shows how to train a semantic segmentation network to identify 5G NR and LTE signals in a wideband spectrogram.
4. The Identify LTE and NR Signals from Captured Data Using SDR and Deep Learning (Wireless Testbench) example shows how to use a deep learning trained semantic segmentation network to identify NR and LTE signals from wireless data captured with a SDR.

The intended workflow of the examples above is shown in the diagram.



Generate Training Data

One advantage of wireless signals in the deep learning domain is the fact that the signals are synthesized. Also, we have highly reliable channel and RF impairment models. As a result, instead of collecting and manually labeling signals, you can generate 5G NR signals using 5G Toolbox™, LTE signals using LTE Toolbox™, and WLAN signals using WLAN Toolbox™ functions. You can pass these signals through standards-specified channel models to create the training data.

Train the network with frames that contain only 5G NR, LTE, or WLAN signals and then shift these signals in frequency randomly within the band of interest. Each frame is 40 ms long, which is the duration of 40 subframes. The network assumes that the 5G NR, LTE, or WLAN signal occupies the same band for the whole frame duration. To test the network performance, create frames that contain both 5G NR and LTE signals, 5G NR and WLAN signals, or LTE and WLAN signals on distinct random bands within the band of interest.

Use a sampling rate of 61.44 MHz. This rate is high enough to process most of the latest standard signals and several low-cost software defined radio (SDR) systems can sample at this rate providing about 50 MHz of useful bandwidth. To monitor a wider band, you can increase the sample rate, regenerate training frames and retrain the network.

Use the `helperSpecSenseTrainingData` function to generate training frames. This function generates 5G NR signals using the `helperSpecSenseNRSignal` function, LTE signals using the `helperSpecSenseLTESignal` function, and WLAN signals using `helperSpecSenseWLANSignal` function.

This table lists 5G NR variable signal parameters.

5G NR Parameter	Value	Units
Bandwidth	`[10 15 20 25 30 40 50]`	MHz
Sub-Carrier Spacing (SCS)	`[15 30]`	kHz
SSB Block Pattern	`["Case A" "Case B"]`	
SSB Period	`[20]`	ms

This table lists LTE variable signal parameters.

LTE Parameter	Value	Units
Reference Channel	`["R.2","R.6","R.8","R.9"}`	
Bandwidth	`[ 10 5 15 20]`	MHz
Duplex Mode	`FDD`	

This table lists WLAN variable signal parameters

WLAN Parameter	Value	Units
Bandwidth	[20 40]	MHz
MCS	[0 1 2 3 4 5 6 7 8 9 10 11]	

Use the `nrCDLChannel` (5G Toolbox), the `lteFadingChannel` (LTE Toolbox), and the `wlanTGaxChannel` (WLAN Toolbox) functions to add channel impairments. For details of the channel configurations, see the `helperSpecSenseTrainingData` function. This table lists channel parameters.

Channel Parameter	Value Range	Units
SNR	[0 40]	dB
Doppler	[0 500]	Hz

The `helperSpecSenseTrainingData` function uses the `helperSpecSenseSpectrogramImage` function to create spectrogram images from complex baseband signals. Calculate the spectrograms using an FFT length of 4096. Generate 256 by 256 RGB images. This image size allows a large enough batch of images to fit in memory during training while providing enough resolution in time and frequency. If your GPU does not have sufficient memory, you can resize the images to smaller sizes or reduce the training batch size.

The `trainingDataSource` variable determines whether training data is to be downloaded or generated. Choosing "Use downloaded data" downloads training data. Choosing "Generate training data" generates the training data from

scratch. Data generation may take several hours depending on the configuration of your computer. Using a PC with Intel® Xeon® W-2133 CPU @ 3.60GHz and creating a parallel pool with six workers with the Parallel Computing Toolbox™, training data generation takes about an hour. Choose "Train network now" to train the network. This process takes about 8 minutes with the same PC and NVIDIA® Titan V GPU. Choose "Use trained network" to skip network training. Instead, the example downloads the trained network.

Use 900 frames from each set of signals: 5G NR only, LTE only, WLAN only, and combination of two of the three possible types of signals. If you increase the number of possible values for the system parameters, increase the number of training frames.

You generate a set of training images for specified `imageSize` size.

The downloaded training data also includes captured, preprocessed, and labeled data for LTE, 5G, WLAN, and unknown signals over a wideband. For more information, see the Capture and Label NR and LTE Signals for AI Training (Wireless Testbench) and WLAN Activity Scanner (Wireless Testbench) examples.

```
imageSize = [256 256];                                  % pixels
sampleRate = 61.44e6;                                   % Hz
numSubFrames = 40;                                      % corresponds to 40 ms
frameDuration = numSubFrames*1e-3;                      % seconds
trainDirRoot = fullfile(pwd,"TrainingData");
classNames = ["Noise" "NR" "LTE" "WLAN" "Unknown"];
trainingDataSource = Use downloaded data ;
trainNow = Use trained network ;
useCapturedData = ☑ ;
if trainingDataSource == "Generated data"
  numFramesPerStandard = 900;
  saveChannelInfo = false;
  helperSpecSenseTrainingData(numFramesPerStandard,classNames,imageSize, ...
      trainDirRoot,numSubFrames,sampleRate,saveChannelInfo);
end
```

Choose Deep Neural Network

You have the choice of training a semantic segmentation network from scratch, or applying transfer learning to a pretrained semantic segmentation network.

- To apply transfer learning, set `baseNetwork` to the desired pretrained network architecture.
- To train a custom network from scratch, set `baseNetwork` to `"custom"`.

The `baseNetwork` is set to `resnet18` (Deep Learning Toolbox). If the Deep Learning Toolbox™ Model *for ResNet-18 Network* support package is not installed, then the function provides a link to the required support package in the Add-On Explorer. To install the support package, click the link, and then click **Install**. Check that the installation is successful by typing `resnet18` at the command line. If the required support package is installed, then the function returns a dlnetwork object. For more information, seeMake Predictions Using dlnetwork Object (Deep Learning Toolbox).

```
baseNetwork = resnet18 ;
trainDir = fullfile(trainDirRoot,'256x256');
imageSize = [256 256];
```

Based on selections, download training data and/or trained network.

```
helperSpecSenseDownloadData(trainingDataSource,trainNow,useCapturedData, ...
    baseNetwork,imageSize)
```

```
Starting download of data files from:
        https://www.mathworks.com/supportfiles/spc/SpectrumSensing/SpectrumSensingTrainingData25
```

```
Extracting files.
Extract complete.
Starting download of data files from:
        https://www.mathworks.com/supportfiles/spc/SpectrumSensing/SpectrumSensingTrainedResnet1
Extracting files.
Extract complete.
Starting download of data files from:
        https://www.mathworks.com/supportfiles/spc/SpectrumSensing/SpectrumSensingCapturedData25
Extracting files.
Extract complete.
```

Load Training Data

Use the `imageDatastore` function to load training images with the spectrogram of 5G NR, LTE, and WLAN signals. The `imageDatastore` function enables you to efficiently load a large collection of images from disk. Spectrogram images are stored in .png files.

```
folders = trainDir;
if useCapturedData
  folders = [folders,fullfile(trainDir,"captured")];
end
imds = imageDatastore(folders,FileExtensions=".png");
```

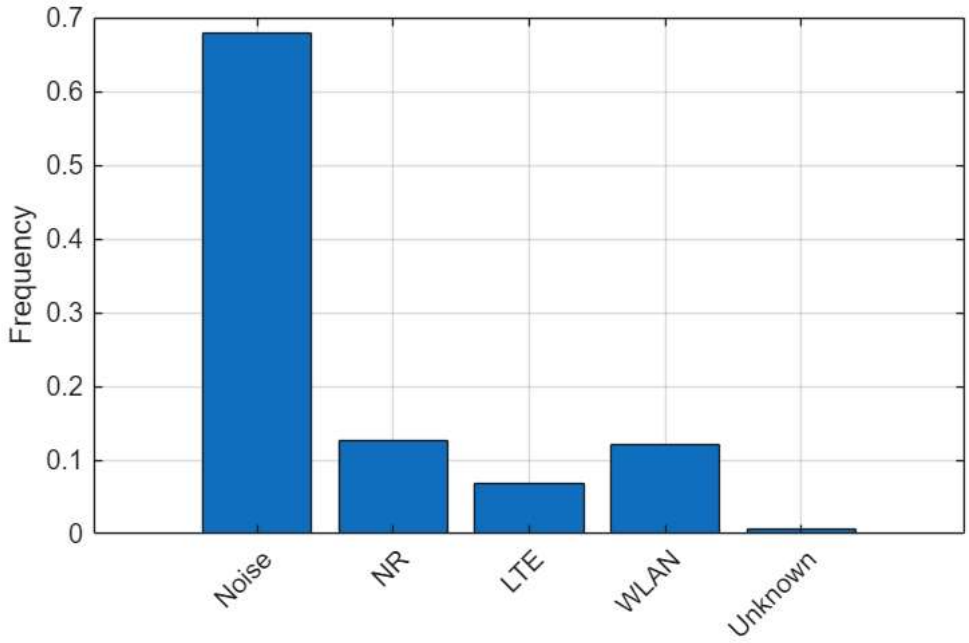
Use the `pixelLabelDatastore` (Computer Vision Toolbox) function to load spectrogram pixel label image data. Each pixel is labeled as one of "NR", "LTE", "Noise", "WLAN" or "Unknown". A pixel label datastore encapsulates the pixel label data and the label ID to a class name mapping. Pixel labels are stored in .hdf files.

```
numClasses = length(classNames);
pixelLabelID = floor((0:numClasses-1)/(numClasses-1)*255);
pxdsTruthLTENRWLAN = pixelLabelDatastore(folders,classNames,pixelLabelID,...
                                  FileExtensions=".hdf");
```

Analyze Dataset Statistics

To see the distribution of class labels in the training dataset, use the `countEachLabel` (Computer Vision Toolbox) function to count the number of pixels by class label, and plot the pixel counts by class.

```
tbl = countEachLabel(pxdsTruthLTENRWLAN);
frequency = tbl.PixelCount/sum(tbl.PixelCount);
figure
bar(1:numel(classNames),frequency)
grid on
xticks(1:numel(classNames))
xticklabels(tbl.Name)
xtickangle(45)
ylabel("Frequency")
```



Ideally, all classes would have an equal number of observations. However, with wireless signals it is common for the classes in the training set to be imbalanced. 5G NR and WLAN signals may have larger bandwidth than LTE signals, and noise fills the background. Because the learning is biased in favor of the dominant classes, imbalance in the number of observations per class can be detrimental to the learning process. In the Balance Classes Using Class Weighting section, class weighting is used to mitigate bias caused by imbalance in the number of observations per class.

Prepare Training, Validation, and Test Sets

The deep neural network uses 80% of the signal images from the dataset for training, 10% of the images for validation, and 10% of the images for testing. The `helperSpecSensePartitionData` function randomly splits the image and pixel label data into training,validation, and test sets.

```
[imdsTrain,pxdsTrain,imdsVal,pxdsVal,imdsTest,pxdsTest] = ...
  helperSpecSensePartitionData(imds,pxdsTruthLTENRWLAN,[80 10 10]);
cdsTrain = combine(imdsTrain,pxdsTrain);
cdsVal = combine(imdsVal,pxdsVal);
cdsTest = combine(imdsTest,pxdsTest);
```

Get

Train Deep Neural Network

You have the choice of training a network from scratch or applying transfer learning.

Use a Pretrained Network for Transfer Learning

Apply transfer learning to a popular semantic segmentation network.

To apply transfer learning, use the `deeplabv3plus` (Computer Vision Toolbox) function to create a semantic segmentation neural network. Choose `resnet18` as the base network (by setting the value of `baseNetwork`) and specify the input image size (number of pixels used to represent time and frequency axes) and the number of classes.

```
if ~strcmp(baseNetwork,"custom")
    layers = deeplabv3plus([256 256],numel(classNames),baseNetwork);
end
```

Get

Design a Simple Semantic Segmentation Network

Instead of transfer learning, you can design a simple semantic segmentation network.

A common pattern in semantic segmentation networks requires the downsampling of an image between convolutional and ReLU layers, and then upsampling the output to match the input size. During this process, a network performs the operations using non-linear filters optimized for a specific set of classes that you want to segment.

```
if strcmp(baseNetwork,"custom")
    layers = helperSpecSenseCustomNetwork(imageSize,numClasses);
end
```

Balance Classes Using Class Weighting

To improve training when classes in the training set are not balanced, you can use class weighting to balance the classes. Use the pixel label counts computed earlier with the `countEachLabel` function and calculate the median frequency class weights.

```
imageFreq = tbl.PixelCount ./ tbl.ImagePixelCount;
imageFreq(isnan(imageFreq)) = [];
classWeights = median(imageFreq) ./ imageFreq;
classWeights = classWeights/(sum(classWeights)+eps(class(classWeights)));
if length(classWeights) < numClasses
    classWeights = [classWeights; zeros(numClasses - length(classWeights),1)];
end
```

Select Training Options

Configure training using the `trainingOptions` (Deep Learning Toolbox) function to specify the stochastic gradient descent with momentum (SGDM) optimization algorithm and the hyper-parameters used for SGDM. To get the best performance from the network, you can use the Experiment Manager (Deep Learning Toolbox) to optimize training options.

```
mbs = 40;
opts = trainingOptions("sgdm",...
  MiniBatchSize = mbs,...
  MaxEpochs = 20, ...
  LearnRateSchedule = "piecewise",...
  InitialLearnRate = 0.02,...
  LearnRateDropPeriod = 10,...
  LearnRateDropFactor = 0.1,...
  ValidationData = cdsVal,...
  ValidationPatience = 5,...
  Shuffle="every-epoch",...
  OutputNetwork = "best-validation-loss",...
  Plots = 'training-progress');
```

Train the network using the combined training data store, `cdsTrain`. The combined training data store contains single signal frames and true pixel labels. Use weighted cross-entropy loss together with a custom normalization to update the network during training. Define a custom loss function, lossFunction, using the `crossentropy` (Deep Learning Toolbox) loss function and apply custom normalization.

```
if trainNow
    [net,trainInfo] = trainnet(cdsTrain,layers, ...
        @(ypred,ytrue) lossFunction(ypred,ytrue,classWeights),opts); %#ok
    save(sprintf('myNet_%s_%s',baseNetwork, ...
        datetime('now',format='yyyy_MM_dd_HH_mm')), 'net')
else
    net = loadNetworkFromMATFile(baseNetwork);
end
```

Test Deep Neural Network

Test the network signal identification performance using signals that contain both 5G NR, LTE, and WLAN signals. Use the `semanticseg` (Computer Vision Toolbox) function to get the pixel estimates of the spectrogram images in the test data set. Use the `evaluateSemanticSegmentation` (Computer Vision Toolbox) function to compute various metrics to evaluate the quality of the semantic segmentation results.

```
dataDir = fullfile(trainDir,"LTE_NR_WLAN");
imdsLTENRWLAN = imageDatastore(dataDir,FileExtensions=".png");
pxdsResultsLTENRWLAN = semanticseg(imdsLTENRWLAN,net,MinibatchSize=mbs,WriteLocatior
    Classes=classNames);
```

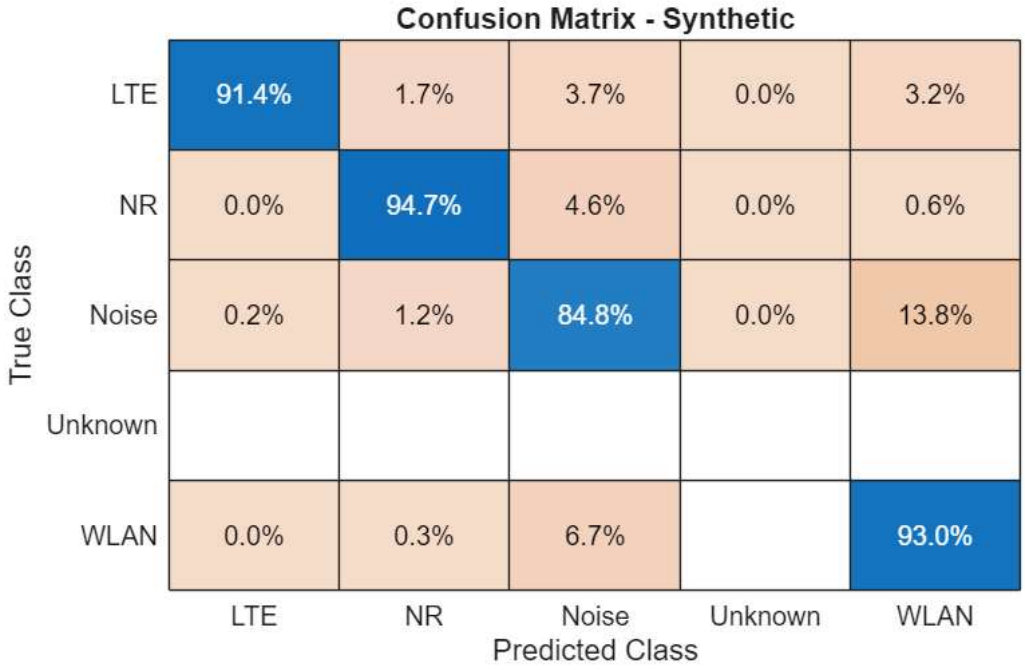
```
Running semantic segmentation network
-------------------------------------
* Processed 2294 images.
```

```
pxdsTruthLTENRWLAN = pixelLabelDatastore(dataDir,classNames,pixelLabelID,...
  FileExtensions=".hdf");
metrics = evaluateSemanticSegmentation(pxdsResultsLTENRWLAN,pxdsTruthLTENRWLAN);
```

```
Evaluating semantic segmentation results
----------------------------------------
* Selected metrics: global accuracy, class accuracy, IoU, weighted IoU, BF score.
* Processed 2294 images.
* Finalizing... Done.
* Data set metrics:

    GlobalAccuracy    MeanAccuracy    MeanIoU    WeightedIoU    MeanBFScore
    ______________    ____________    _______    ___________    ___________

        0.90116             NaN        0.67109      0.82466        0.64699
```

Plot the normalized confusion matrix for all test frames.

```
cm = confusionchart(metrics.ConfusionMatrix.Variables, ...
  classNames, Normalization='row-normalized');
cm.Title = 'Confusion Matrix - Synthetic';
```



If you chose to use captured data in the Generate Training Data section, test with just captured data.

```
if useCapturedData
  capturedIdx = contains(imdsTest.Files,'captured');
  imdsTestCaptured = subset(imdsTest,capturedIdx);
  pxdsTestCaptured = subset(pxdsTest,capturedIdx);
```

Repeat the same process, considering only the frames with captured data in the test set.

```
pxdsResultsCaptured = semanticseg(imdsTestCaptured,net,MinibatchSize=mbs,WriteLoca
  Classes=classNames);
metrics = evaluateSemanticSegmentation(pxdsResultsCaptured,pxdsTestCaptured);
```

Replot the normalized confusion matrix.

```
cm = confusionchart(metrics.ConfusionMatrix.Variables, ...
  classNames, Normalization="row-normalized");
cm.Title = "Normalized Confusion Matrix";
```

The confusion matrix shows that the network confuses NR signals with Noise or Unknown signals. Examining the captured signals reveals that the captured signals with file prefix CF3550 has very low SNR and the network is having a hard time to identify signals correctly.

```
CF3550Indices = contains(imdsTestCaptured.Files,'CF3550');
idx = find(CF3550Indices,1);
rcvdSpectrogram = readimage(imdsTestCaptured,idx);
trueLabels = readimage(pxdsTestCaptured,idx);
predictedLabels = readimage(pxdsResultsCaptured,idx);
figure
helperSpecSenseDisplayResults(rcvdSpectrogram,trueLabels,predictedLabels, ...
  classNames,250e6,0,frameDuration)
```

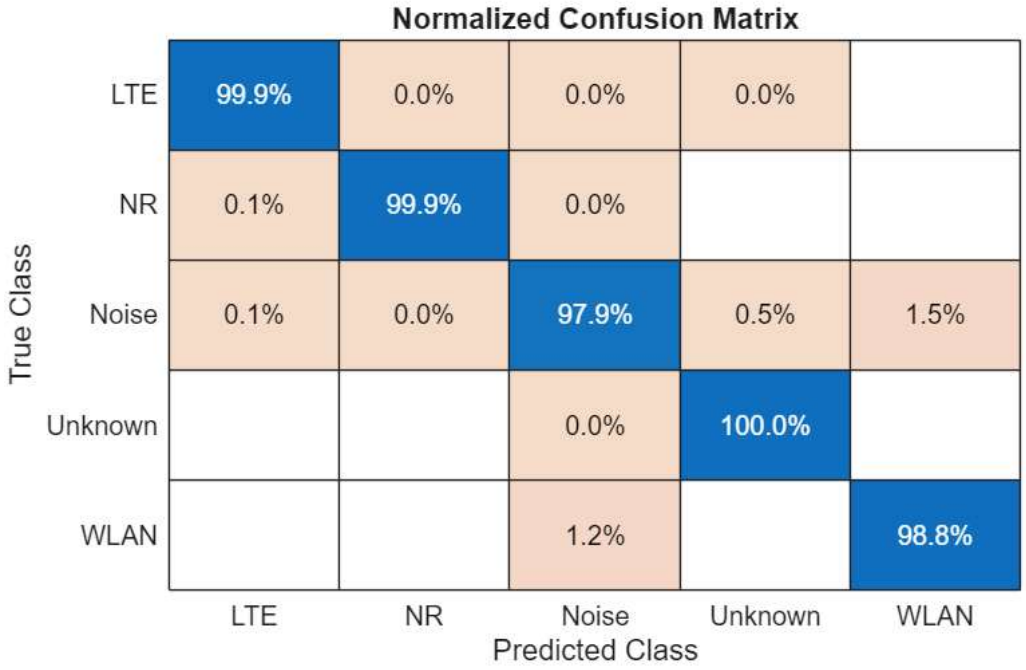
Test with captured data but exclude CF3550 frames.

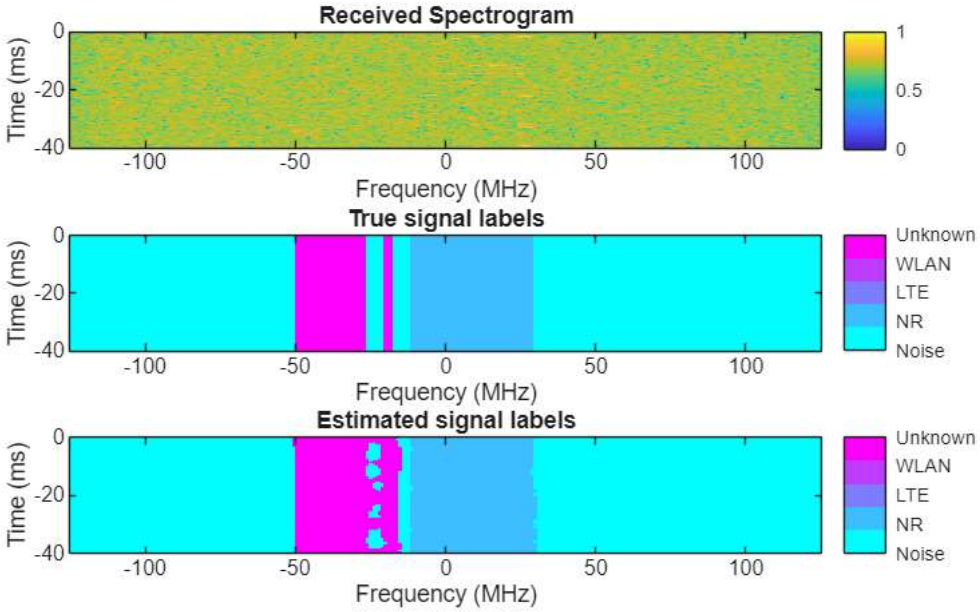
```
imdsTestCaptured2 = subset(imdsTestCaptured,~CF3550Indices);
pxdsTestCaptured2 = subset(pxdsTestCaptured,~CF3550Indices);
pxdsResultsCaptured2 = semanticseg(imdsTestCaptured2,net,MinibatchSize=mbs,WriteLc
  Classes=classNames);
metrics = evaluateSemanticSegmentation(pxdsResultsCaptured2,pxdsTestCaptured2);
```

Replot the normalized confusion matrix.

```
figure
cm = confusionchart(metrics.ConfusionMatrix.Variables, ...
  classNames, Normalization="row-normalized");
cm.Title = "Normalized Confusion Matrix";
end
```

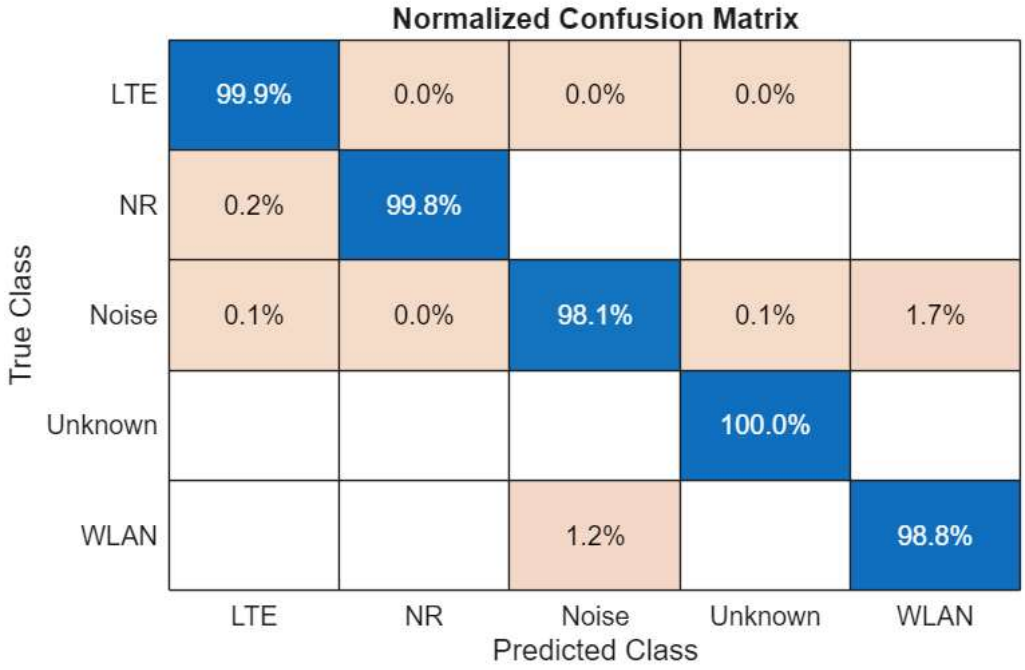
```
Running semantic segmentation network
-------------------------------------
* Processed 76 images.
Evaluating semantic segmentation results
----------------------------------------
* Selected metrics: global accuracy, class accuracy, IoU, weighted IoU, BF score.
* Processed 76 images.
* Finalizing... Done.
* Data set metrics:

    GlobalAccuracy    MeanAccuracy    MeanIoU    WeightedIoU    MeanBFScore
    ______________    ____________    _______    ___________    ___________

       0.98271           0.99308       0.93931      0.96778        0.89497
```






```
Running semantic segmentation network
-------------------------------------
* Processed 66 images.
Evaluating semantic segmentation results
----------------------------------------
* Selected metrics: global accuracy, class accuracy, IoU, weighted IoU, BF score.
* Processed 66 images.
* Finalizing... Done.
* Data set metrics:
```

GlobalAccuracy	MeanAccuracy	MeanIoU	WeightedIoU	MeanBFScore
0.98404	0.99344	0.95623	0.97037	0.89122

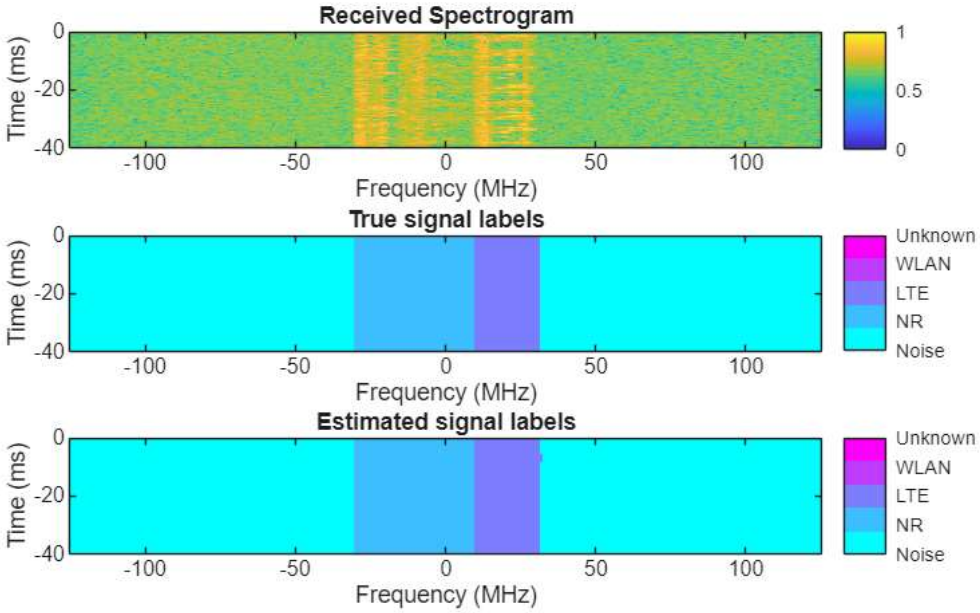


The NR detection rate increases to more than 99%.

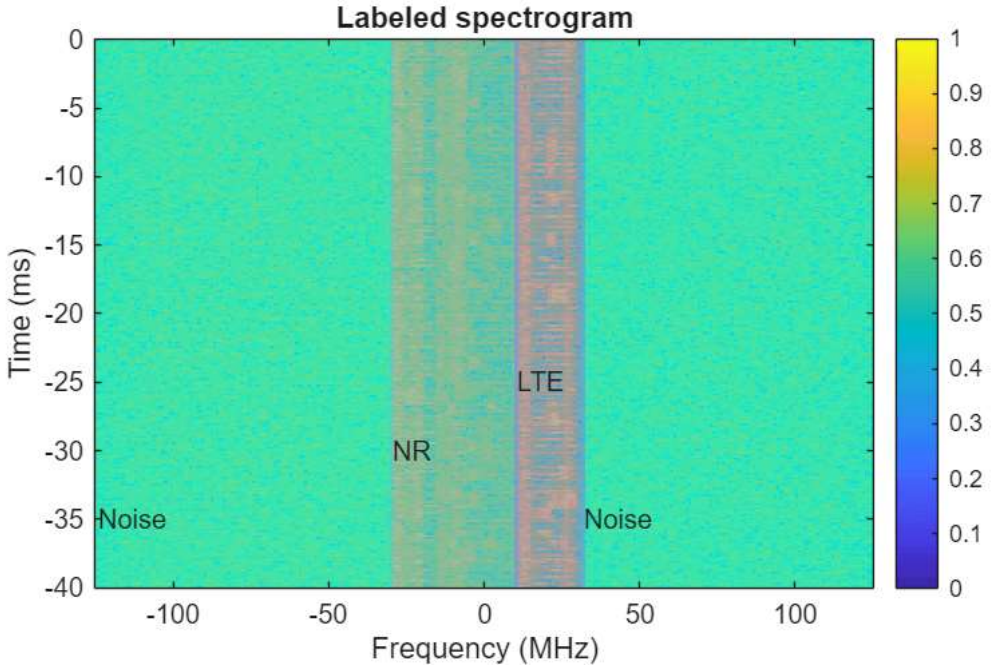
Identify 5G NR and LTE Signals in Spectrogram

Visualize the received spectrum, true labels, and predicted labels for a captured signal.

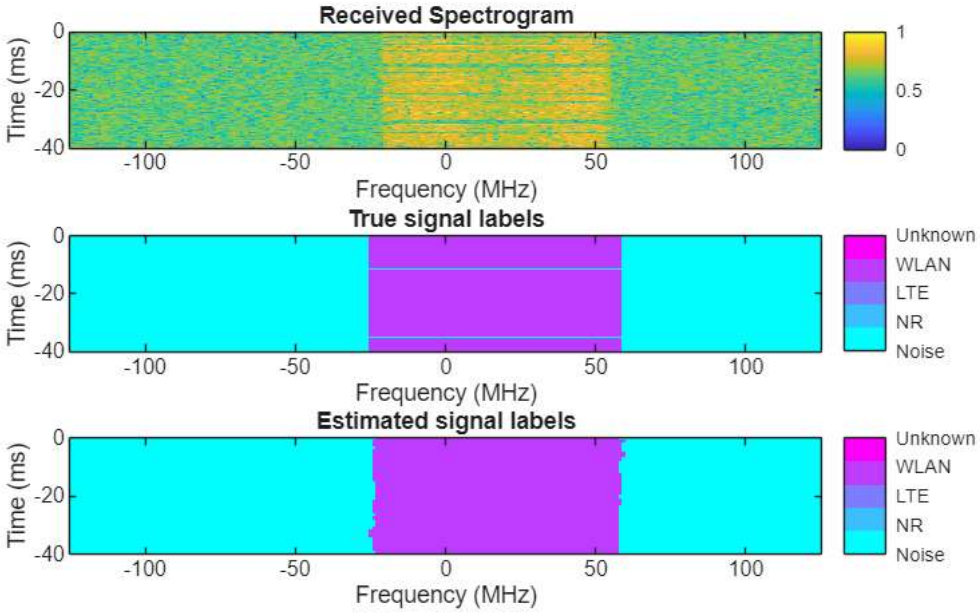
```
if useCapturedData
  signals = find(~CF3550Indices);
  numSignals = length(signals);
  idx = 8;
  rcvdSpectrogram1 = readimage(imdsTestCaptured,signals(idx));
  trueLabels1 = readimage(pxdsTestCaptured,signals(idx));
  predictedLabels1 = readimage(pxdsResultsCaptured,signals(idx));
  idx = 1;
  rcvdSpectrogram2 = readimage(imdsTestCaptured,signals(idx));
  trueLabels2 = readimage(pxdsTestCaptured,signals(idx));
  predictedLabels2 = readimage(pxdsResultsCaptured,signals(idx));
else
  numSignals = length(imdsLTENRWLAN.Files);
  idx = 3;
  rcvdSpectrogram1 = readimage(imdsLTENRWLAN,idx);
  trueLabels1 = readimage(pxdsTruthLTENRWLAN,idx);
  predictedLabels1 = readimage(pxdsResultsLTENRWLAN,idx);
  idx = length(imdsLTENRWLAN.Files);
  rcvdSpectrogram2 = readimage(imdsLTENRWLAN,idx);
  trueLabels2 = readimage(pxdsTruthLTENRWLAN,idx);
  predictedLabels2 = readimage(pxdsResultsLTENRWLAN,idx);
end
figure
helperSpecSenseDisplayResults(rcvdSpectrogram1,trueLabels1,predictedLabels1, ...
  classNames,250e6,0,frameDuration)
```



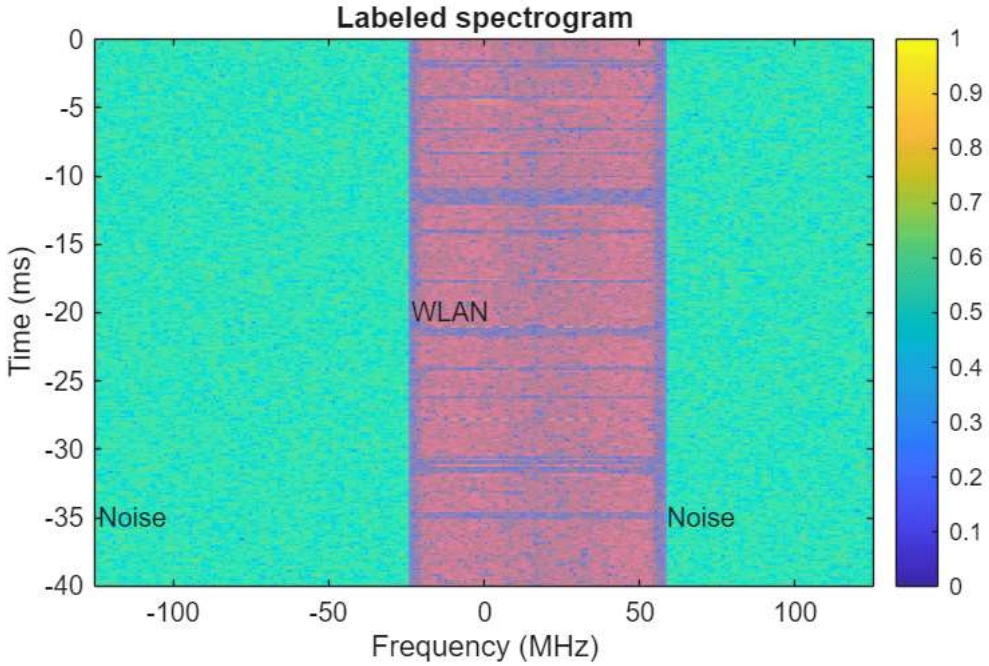

```
figure
helperSpecSenseDisplayIdentifiedSignals(rcvdSpectrogram1,predictedLabels1, ...
  classNames,250e6,0,frameDuration)
```




```
figure
helperSpecSenseDisplayResults(rcvdSpectrogram2,trueLabels2,predictedLabels2, ...
  classNames,250e6,0,frameDuration)
```




```
figure
helperSpecSenseDisplayIdentifiedSignals(rcvdSpectrogram2,predictedLabels2, ...
  classNames,250e6,0,frameDuration)
```



Test with Captured Data using SDR

Test the performance of the trained network using over-the-air signal captures using SDR. Find a nearby base station and tune the center frequency of your radio to cover the band of the signals you want to identify. This example sets the center frequency to 2.35 GHz. If you have at least one ADALM-PLUTO radio and have installed Communication Toolbox Support Package for ADALM-PLUTO Radio, you can run this section of the code. In case you do not have access to an ADALM-PLUTO radio, this example shows results of a test conducted using captured signals and a trained network.

Use Wireless Testbench example Identify LTE and NR Signals from Captured Data Using SDR and Deep Learning (Wireless Testbench) to test with wideband signals.

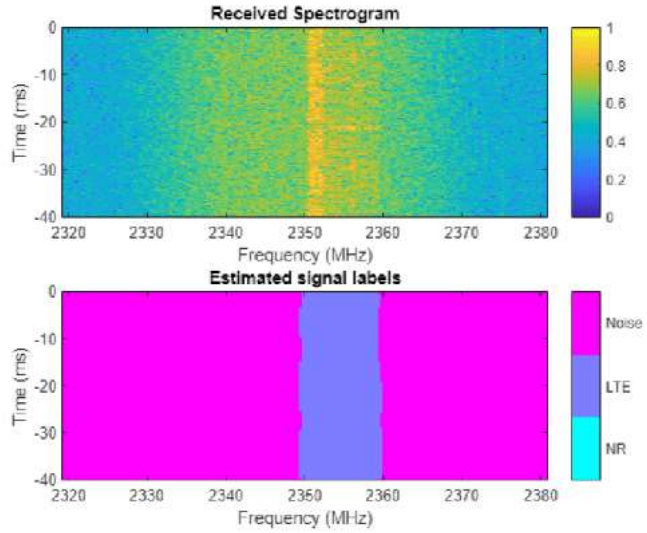
```
runSDRSection = false;
if helperIsPlutoSDRInstalled()
  radios = findPlutoRadio();
  if length(radios) >= 1
```

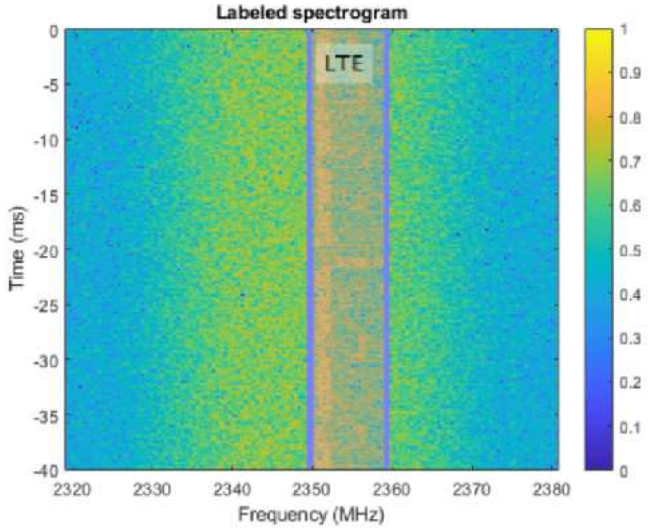
```
    runSDRSection = true;
  else
    disp("At least one ADALM-PLUTO radios is needed. Skipping SDR test.")
  end
else
    disp("Communications Toolbox Support Package for Analog Devices ADALM-PLUTO Radi
    disp("Click Add-Ons in the Home tab of the MATLAB toolstrip to install the suppc
    disp("Skipping SDR test.")
end
```

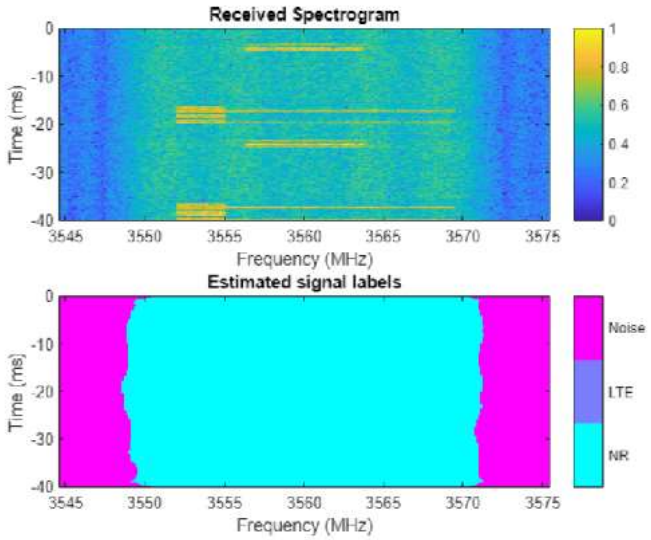
```
Communications Toolbox Support Package for Analog Devices ADALM-PLUTO Radio not found.
Click Add-Ons in the Home tab of the MATLAB toolstrip to install the support package.
Skipping SDR test.
```

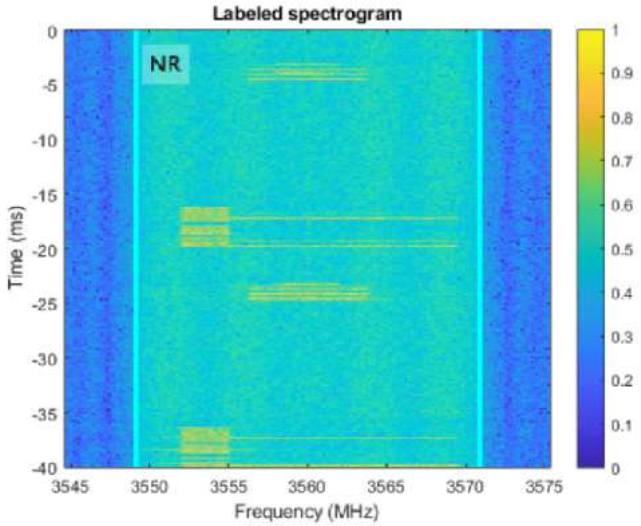
Get

```
if runSDRSection
  % Set up PlutoSDR receiver
  rx = sdrrx('Pluto');
  rx.CenterFrequency = 2.43e9;
  rx.BasebandSampleRate = sampleRate;
  rx.SamplesPerFrame = frameDuration*rx.BasebandSampleRate;
  rx.OutputDataType = 'single';
  rx.EnableBurstMode = true;
  rx.NumFramesInBurst = 1;
  Nfft = 4096;
  overlap = 10;

  meanAllScores = zeros([imageSize numel(classNames)]);
  segResults = zeros([imageSize 10]);
  for frameCnt=1:10
    rxWave = rx();
    rxSpectrogram = helperSpecSenseSpectrogramImage(rxWave,Nfft,sampleRate,imageSize

    [segResults(:,:,frameCnt),scores,allScores] = semanticseg(rxSpectrogram,net);
    meanAllScores = (meanAllScores*(frameCnt-1) + allScores) / frameCnt;
  end
  release(rx)

  [~,predictedLabels] = max(meanAllScores,[],3);
  figure
  helperSpecSenseDisplayResults(rxSpectrogram,[],predictedLabels,classNames,...
    sampleRate,rx.CenterFrequency,frameDuration)
  figure
  freqBand = helperSpecSenseDisplayIdentifiedSignals(rxSpectrogram,predictedLabels,.
    classNames,sampleRate,rx.CenterFrequency,frameDuration)
else
  figure
  imshow('lte_capture_result1.png',InitialMagnification='fit')
  figure
  imshow('lte_capture_result2.png',InitialMagnification='fit')
  figure
  imshow('nr_capture_result1.png',InitialMagnification='fit')
  figure
  imshow('nr_capture_result2.png',InitialMagnification='fit')
end
```


Received Spectrogram
Time (ms)
Frequency (MHz)
Estimated signal labels
Noise
LTE
NR


Labeled spectrogram
LTE
Time (ms)
Frequency (MHz)


Received Spectrogram
Time (ms)
Frequency (MHz)
Estimated signal labels
Noise
LTE
NR



Conclusions and Further Exploration

The trained network can distinguish 5G NR, LTE, and WLAN signals including two example captures from real base stations. The network may not be able to identify every captured signal correctly. In such cases, enhance the training data either by generating more representative synthetic signals or capturing over-the-air signals and including these in the training set. The results obtained after training can differ from the results mentioned here for different networks due to random initial conditions.

Different network structures result in different accuracy for detection. The table shows detection accuracy results for custom, ResNet-18, MobileNetv2, and ResNet-50, which have 1.4M, 20.6M, 43.9M and 46.9M learnables, respectively. Detection accuracy results are for overall test set, only synthetic signals, and only for captured signals. The overall and captured only tests include signals from the test set. The synthetic tests include generated signals in the LTE_NR_WLAN directory, which are not used in training. Increasing the network complexity results in increased accuracy.

	Custom(1.4M)			ResNet18 (20.6M)			MobileNetv2 (43.9M)			ResNet50 (46.9M)		
	Overall	Synthetic	Captured	Overall	Synthetic	Captured	Overall	Synthetic	Captured	Overall	Synthetic	Captured
LTE	98.9	95.7	98.9	99.9	91.4	99.9	99.9	98.3	99.9	100	97.2	100
NR	96.2	94.1	75.9	99.8	94.7	99.9	99.4	96.1	99.5	100	95.9	99.9
Noise	96.3	81.4	90.8	97.8	84.8	97.7	97.7	82.9	97.7	97.8	85.4	97.9
Unknown	99.3	N/A	91.9	100	N/A	100	100	N/A	100	100	N/A	100
WLAN	91	77.3	91	99	93.0	99.0	99	94.2	99	99.3	92.6	99.3

You can use the Identify LTE and NR Signals from Captured Data Using SDR and Deep Learning (Wireless Testbench) example to identify LTE and 5G NR signals using the trained networks.

If you need to monitor wider bands of spectrum, increase the `sampleRate`, regenerate the training data, capture signals using the Capture and Label NR and LTE Signals for AI Training (Wireless Testbench) example, and retrain the network.

Supporting Functions

```
function net = loadNetworkFromMATFile(baselineNetwork)
    switch baselineNetwork
      case "custom"
        net = load("specSenseTrainedNetCustom.mat",'net');
      case "resnet18"
        net = load("specSenseTrainedNetResnet18.mat",'net');
```

```
        case "resnet50"
            net = load("specSenseTrainedNetResnet50.mat",'net');
        case "mobilenetv2"
            net = load("specSenseTrainedNetMobileNetv2.mat",'net');
        otherwise
            error("Unknown baseline network: " + baselineNetwork)
    end
    net = net.net;
end

function loss = lossFunction(ypred,yactual,weights)
% Compute weighted cross-entropy loss.
cdim = find(dims(ypred) == 'C');
loss = crossentropy(ypred,yactual,weights,WeightsFormat="C",NormalizationFactor="nor
wn = shiftdim(weights(:)',-(cdim-2));
wnT = extractdata(yactual).*wn;
normFac = sum(wnT(:))+eps('single');
loss = loss/normFac;
end
```

See Also

`featureInputLayer` (Deep Learning Toolbox) | `fullyConnectedLayer` (Deep Learning Toolbox) | `reluLayer` (Deep Learning Toolbox) | `softmaxLayer` (Deep Learning Toolbox) | `pixelLabelDatastore` (Computer Vision Toolbox) | `countEachLabel` (Computer Vision Toolbox)

Topics

Deep Learning in MATLAB (Deep Learning Toolbox)