Semantic image segmentation using deep learning

Goal: Build and evaluate semantic image segmentation solutions using deep learning architectures.

Learning objectives:

- Learn how to implement an image segmentation workflow in MATLAB
- Learn how to implement and evaluate contemporary (deep-learning-based) semantic image segmentation techniques in MATLAB
- Get acquainted with representative datasets and problems in image segmentation

Table of Contents

Semantic image segmentation using a pretrained network

Main steps:

1. Get the labeled data (CamVid dataset).
2. Explore, understand, and prepare the data.
3. Create network.
4. Train network.
5. Evaluate results visually (displaying a test image and overlaying predicted labels).
6. Evaluate results quantitatively using different metrics (class accuracy, IoU).
7. (OPTIONAL) Repeat steps 3 through 6 using different pretrained networks, training options, data augmentation options, and/or metrics.

Refs:

https://www.mathworks.com/help/vision/ug/semantic-segmentation-with-deep-learning.html#mw_44be2a2e-ec6b-4a03-9470-ea945f74515e

https://www.mathworks.com/help/vision/examples/semantic-segmentation-using-deep-learning.html

Example code

Step 1: Get the labeled data (CamVid dataset).

Download dataset (CamVid)

Download the CamVid image data (images and labels) from the URL provided via Canvas.

Organize your working folder by creating a `CamVid` folder, which should contain two subfolders (where the files downloaded in the previous step should be placed), called `images` and `labels`, and setting the `outputFolder` variable accordingly, as shown below.

```
outputFolder = fullfile('./CamVid/');
imageDir = fullfile(outputFolder,'images');
labelDir = fullfile(outputFolder,'labels');
```

Load CamVid data

The CamVid data set encodes the pixel labels as RGB images, where each class is represented by an RGB color. Here are the classes the dataset defines along with their RGB encodings.

```
classNames = [ ...
    "Animal", ...
    "Archway", ...
    "Bicyclist", ...
    "Bridge", ...
    "Building", ...
    "Car", ...
    "CartLuggagePram", ...
    "Child", ...
    "Column_Pole", ...
    "Fence", ...
    "LaneMkgsDriv", ...
    "LaneMkgsNonDriv", ...
    "Misc_Text", ...
    "MotorcycleScooter", ...
    "OtherMoving", ...
    "ParkingBlock", ...
    "Pedestrian", ...
    "Road", ...
    "RoadShoulder", ...
    "Sidewalk", ...
    "SignSymbol", ...
    "Sky", ...
    "SUVPickupTruck", ...
    "TrafficCone", ...
    "TrafficLight", ...
    "Train", ...
    "Tree", ...
    "Truck_Bus", ...
    "Tunnel", ...
    "VegetationMisc", ...
    "Wall"];
```

Define the mapping between label indices and class names such that `classNames(k)` corresponds to `labelIDs(k,:)`.

```
labelIDs = [ ...
    064 128 064; ... % "Animal"
    192 000 128; ... % "Archway"
    000 128 192; ... % "Bicyclist"
    000 128 064; ... % "Bridge"
    128 000 000; ... % "Building"
    064 000 128; ... % "Car"
    064 000 192; ... % "CartLuggagePram"
    192 128 064; ... % "Child"
    192 192 128; ... % "Column_Pole"
    064 064 128; ... % "Fence"
    128 000 192; ... % "LaneMkgsDriv"
    192 000 064; ... % "LaneMkgsNonDriv"
    128 128 064; ... % "Misc_Text"
    192 000 192; ... % "MotorcycleScooter"
    128 064 064; ... % "OtherMoving"
    064 192 128; ... % "ParkingBlock"
    064 064 000; ... % "Pedestrian"
    128 064 128; ... % "Road"
    128 128 192; ... % "RoadShoulder"
    000 000 192; ... % "Sidewalk"
    192 128 128; ... % "SignSymbol"
    128 128 128; ... % "Sky"
    064 128 192; ... % "SUVPickupTruck"
    000 000 064; ... % "TrafficCone"
    000 064 064; ... % "TrafficLight"
    192 064 128; ... % "Train"
    128 128 000; ... % "Tree"
    192 128 192; ... % "Truck_Bus"
    064 000 064; ... % "Tunnel"
    192 192 000; ... % "VegetationMisc"
    064 192 000];    % "Wall"
```

Create an `imageDatastore` to load the CamVid images.

```
imds = imageDatastore(fullfile(imageDir));
```

Create a `pixelLabelDatastore` to load the CamVid pixel labels.

```
pxds = pixelLabelDatastore(labelDir,classNames,labelIDs);
```

Step 2: Explore, understand, and prepare the data.

Explore images and pixel labels

Read the Mth image and corresponding pixel label image.

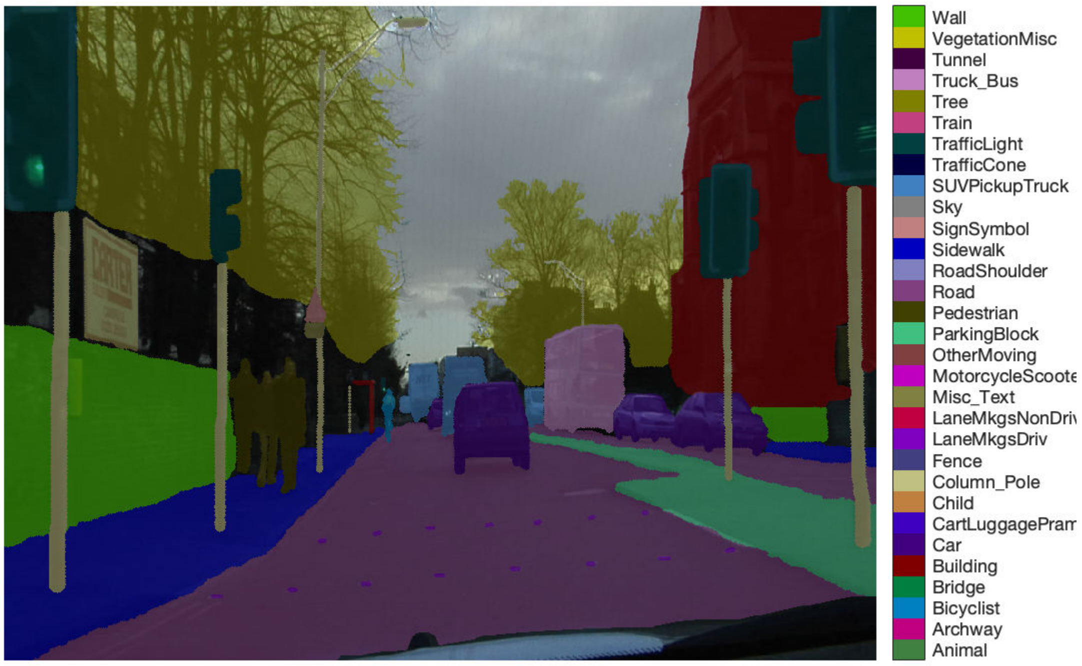
```
M = 15;
```

```
I = readimage(imds,M);
C = readimage(pxds,M);
```

The pixel label image is returned as a categorical array where `C(i,j)` is the categorical label assigned to pixel `I(i,j)`. Display the pixel label image on top of the image.

```
B = labeloverlay(I,C,'Colormap',labelIDs./255);
figure
imshow(B)

% Add a colorbar.
N = numel(classNames);
ticks = 1/(N*2):1/N:1;
colorbar('TickLabels',cellstr(classNames),'Ticks',ticks,'TickLength',0,'TickLabelInter
colormap(labelIDs./255)
```



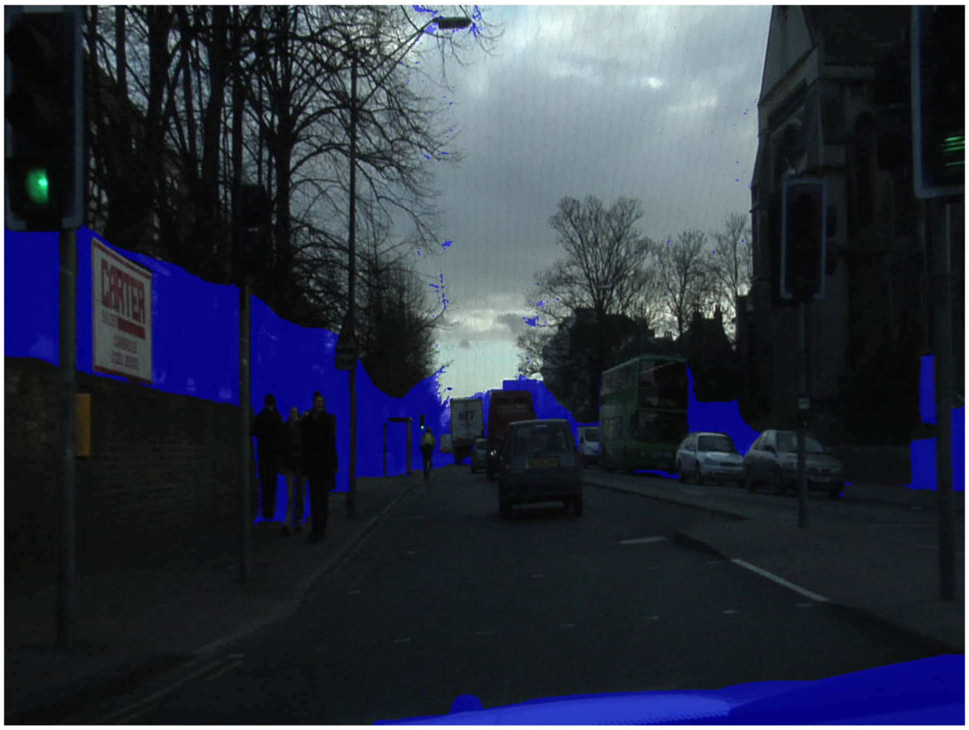
Undefined or Void Labels

It is common for pixel labeled datasets to include "undefined" or "void" labels. These are used to designate pixels that were not labeled. For example, in CamVid, the label ID [0 0 0] is used to designate the "void" class. Training algorithms and evaluation algorithms are not expected to include these labels in any computations.

The "void" class need not be explicitly named when using `pixelLabelDatastore`. Any label ID that is not mapped to a class name is automatically labeled "undefined" and is excluded from computations. To see the undefined pixels, use `isundefined` to create a mask and then display it on top of the image.

```
undefinedPixels = isundefined(C);
B = labeloverlay(I,undefinedPixels);
```

```
figure
imshow(B)
title('Undefined Pixel Labels')
```



Clean up data (e.g., consolidating labels into fewer categories/classes)

To make training easier, we will group the 32 original classes in CamVid to 11 classes, as follows:

```
classes = [
    "Sky"
    "Building"
    "Pole"
    "Road"
    "Pavement"
    "Tree"
    "SignSymbol"
    "Fence"
    "Car"
    "Pedestrian"
    "Bicyclist"
    ];
```

To reduce 32 classes into 11, multiple classes from the original dataset are grouped together. For example, "Car" is a combination of "Car", "SUVPickupTruck", "Truck_Bus", "Train", and "OtherMoving".

Return the grouped label IDs by using the supporting function `camvidPixelLabelIDs`, which is listed at the end of this example.

```
labelIDs = camvidPixelLabelIDs();
```

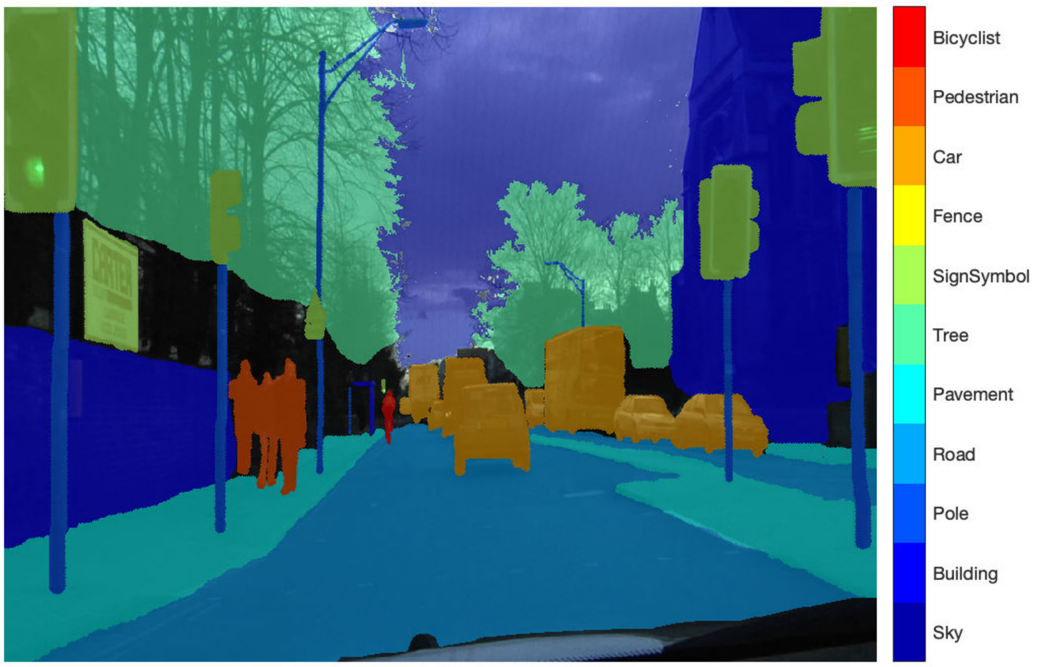
Use the classes and label IDs to create the `pixelLabelDatastore`.

```
labelDir = fullfile(outputFolder,'labels');
pxds = pixelLabelDatastore(labelDir,classes,labelIDs);
```

Read the Mth pixel label image and display it on top of the image.

```
C = readimage(pxds,M);
cmap = jet(numel(classes));
B = labeloverlay(I,C,'Colormap',cmap);
figure
imshow(B)

% add colorbar
N = numel(classes);
ticks = 1/(N*2):1/N:1;
colorbar('TickLabels',cellstr(classes),'Ticks',ticks,'TickLength',0,'TickLabelInterpre
colormap(cmap)
```



The `pixelLabelDatastore` with the new class names can now be used to train a network for the 11 classes without having to modify the original CamVid pixel labels.

Explore dataset properties/statistics

To see the distribution of class labels in the CamVid dataset, use `countEachLabel`. This function counts the number of pixels by class label.

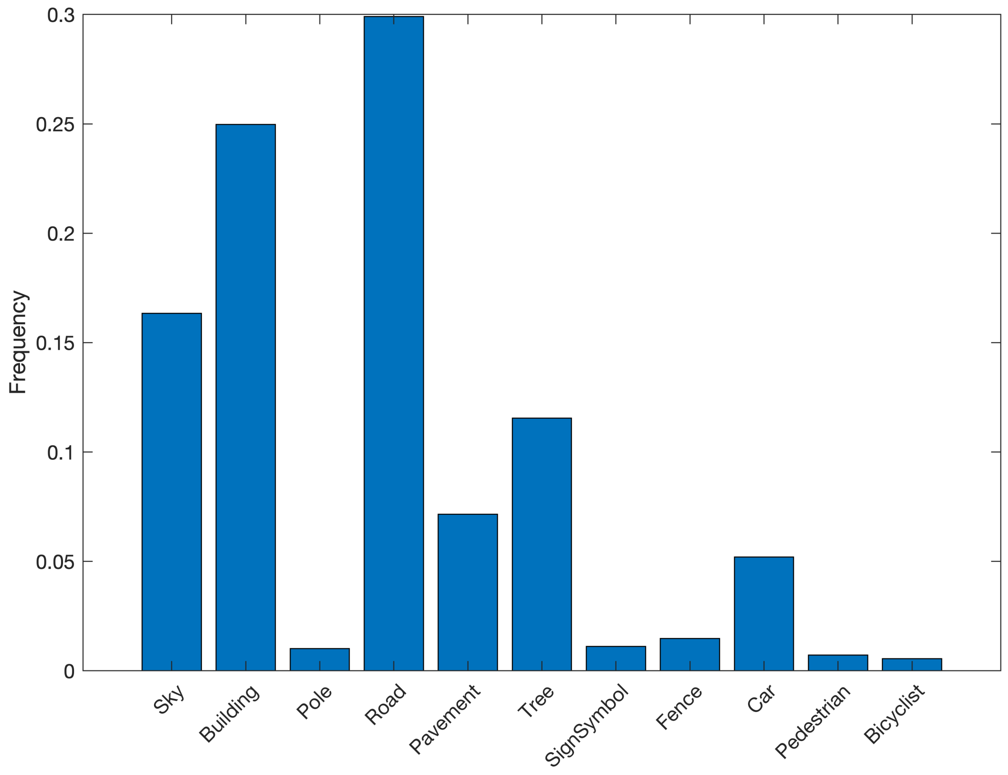
```
tbl = countEachLabel(pxds)
```

tbl = 11×3 table

	Name	PixelCount	ImagePixelCount
1	'Sky'	76801167	483148800
2	'Building'	117373718	483148800
3	'Pole'	4798742	483148800
4	'Road'	140535728	484531200
5	'Pavement'	33614414	472089600
6	'Tree'	54258673	447897600
7	'SignSymbol'	5224247	468633600
8	'Fence'	6921061	251596800
9	'Car'	24436957	483148800
10	'Pedestrian'	3402909	444441600
11	'Bicyclist'	2591222	261964800

Visualize the pixel counts by class.

```
frequency = tbl.PixelCount/sum(tbl.PixelCount);
bar(1:numel(classes),frequency)
xticks(1:numel(classes))
xticklabels(tbl.Name)
xtickangle(45)
ylabel('Frequency')
```



Prepare training, validation, and test sets

Deeplab v3+ is trained using 60% of the images from the dataset. The rest of the images are split evenly in 20% and 20% for validation and testing respectively. The following code randomly splits the image and pixel label data into a training, validation and test set.

```
[imdsTrain, imdsVal, imdsTest, pxdsTrain, pxdsVal, pxdsTest] = partitionCamVidData(imd
```

The 60/20/20 split results in the following number of training, validation and test images:

```
numTrainingImages = numel(imdsTrain.Files)
```

```
numTrainingImages = 421
```

```
numValImages = numel(imdsVal.Files)
```

```
numValImages = 140
```

```
numTestingImages = numel(imdsTest.Files)
```

```
numTestingImages = 140
```

Step 3: Create network.

Download a pre-trained version of Deeplab v3+ network with weights initialized from a pre-trained Resnet-18 network

```
pretrainedURL = 'https://www.mathworks.com/supportfiles/vision/data/deeplabv3plusResne
```

```
pretrainedFolder = fullfile(tempdir,'pretrainedNetwork');
pretrainedNetwork = fullfile(pretrainedFolder,'deeplabv3plusResnet18CamVid.mat');
if ~exist(pretrainedNetwork,'file')
    mkdir(pretrainedFolder);
    disp('Downloading pretrained network (58 MB)...');
    websave(pretrainedNetwork,pretrainedURL);
end
```

```
Warning: Directory already exists.
Downloading pretrained network (58 MB)...
```

Create network

```
% Specify the network image size. This is typically the same as the traing image sizes
imageSize = [720 960 3];

% Specify the number of classes.
numClasses = numel(classes);

% Create DeepLab v3+.
lgraph = deeplabv3plusLayers(imageSize, numClasses, "resnet18");
```

Balance classes using class weighting

```
imageFreq = tbl.PixelCount ./ tbl.ImagePixelCount;
classWeights = median(imageFreq) ./ imageFreq
```

```
classWeights = 11×1
    0.3182
    0.2082
    5.0924
    0.1744
    0.7103
    0.4175
    4.5371
    1.8386
    1.0000
    6.6059
      ⋮
```

Specify the class weights using a `pixelClassificationLayer`.

```
pxLayer = pixelClassificationLayer('Name','labels','Classes',tbl.Name,'ClassWeights',c
lgraph = replaceLayer(lgraph,"classification",pxLayer);
```

Select training options

```
% Define validation data.
pximdsVal = pixelLabelImageDatastore(imdsVal,pxdsVal);

% Define training options.
options = trainingOptions('sgdm', ...
    'LearnRateSchedule','piecewise',...
    'LearnRateDropPeriod',10,...
    'LearnRateDropFactor',0.3,...
    'Momentum',0.9, ...
    'InitialLearnRate',1e-3, ...
    'L2Regularization',0.005, ...
    'ValidationData',pximdsVal,...
    'MaxEpochs',30, ...
    'MiniBatchSize',8, ...
    'Shuffle','every-epoch', ...
    'CheckpointPath', tempdir, ...
    'VerboseFrequency',2,...
    'Plots','training-progress',...
    'ValidationPatience', 4);
```

Configure `imageDataAugmenter` object to perform data augmentation

```
augmenter = imageDataAugmenter('RandXReflection',true,...
    'RandXTranslation',[-10 10],'RandYTranslation',[-10 10]);
```

Step 4: Train network

```
pximds = pixelLabelImageDatastore(imdsTrain,pxdsTrain, ...
    'DataAugmentation',augmenter);
```

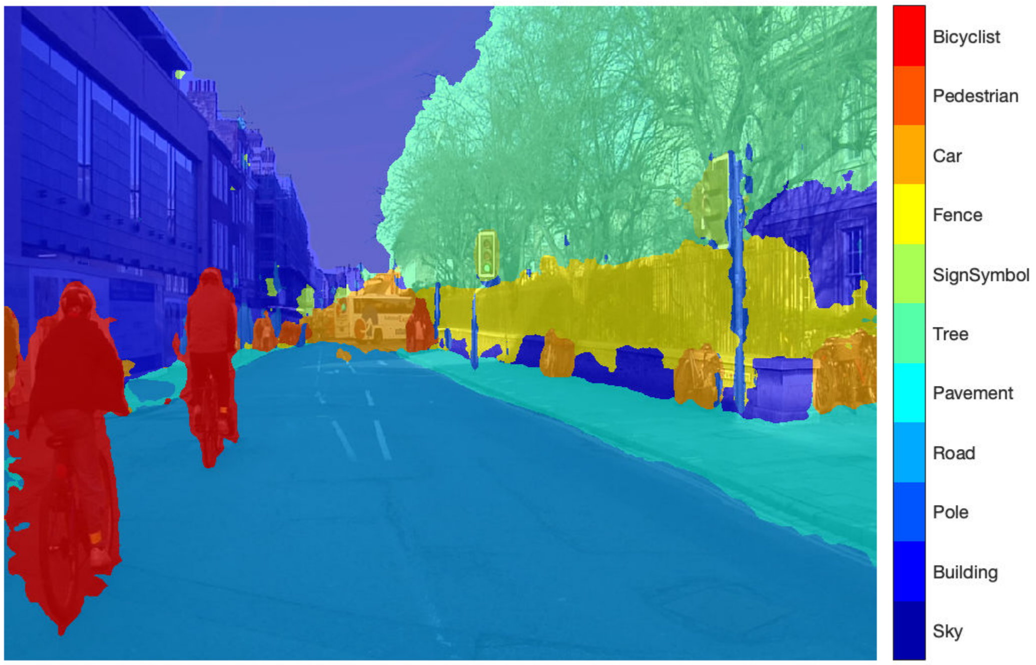
Start training using `trainNetwork` if the `doTraining` flag is true. Otherwise, load a pretrained network.

```
doTraining = false;
if doTraining
    [net, info] = trainNetwork(pximds,lgraph,options);
else
    data = load(pretrainedNetwork);
    net = data.net;
end
```

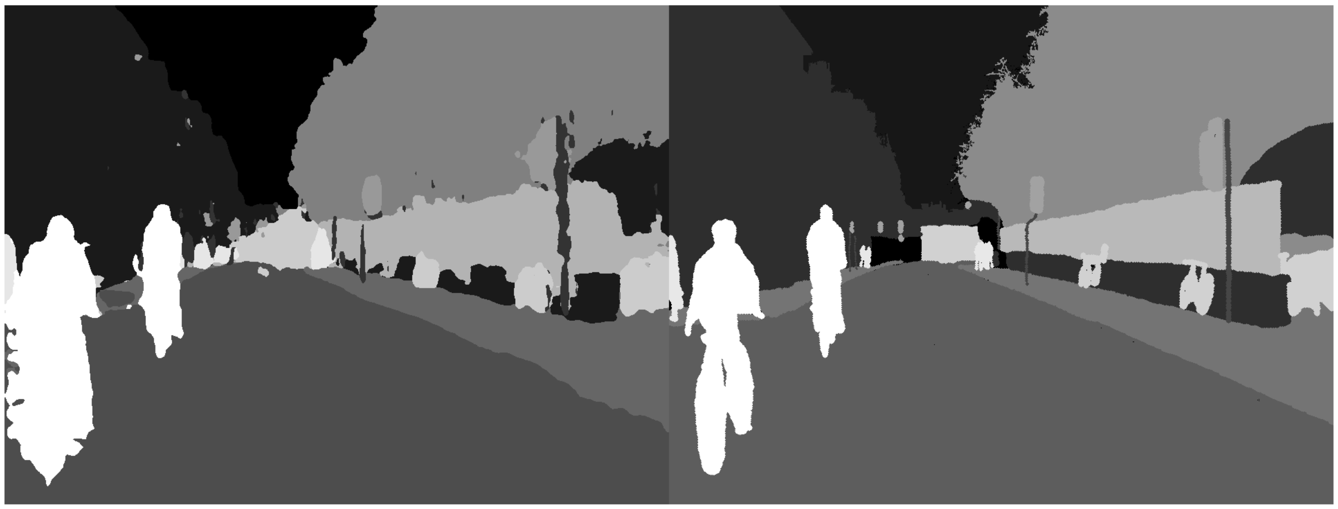
Step 5: Evaluate results visually (displaying a test image and overlaying predicted labels)

```
I = readimage(imdsTest,35);
C = semanticseg(I, net);

B = labeloverlay(I,C,'Colormap',cmap,'Transparency',0.4);
imshow(B)
pixelLabelColorbar(cmap, classes);
```



```
expectedResult = readimage(pxdsTest,35);
actual = uint8(C);
expected = uint8(expectedResult);
imshowpair(actual, expected, 'montage')
```



```
iou = jaccard(C,expectedResult);
table(classes,iou)
```

```
ans = 11×2 table
```

	classes	iou
1	"Sky"	0.9184
2	"Building"	0.8448
3	"Pole"	0.3120
4	"Road"	0.9370
5	"Pavement"	0.8284
6	"Tree"	0.8964
7	"SignSymbol"	0.5764
8	"Fence"	0.7105
9	"Car"	0.6669
10	"Pedestrian"	0.4842
11	"Bicyclist"	0.6843

Step 6: Evaluate results quantitatively using different metrics (class accuracy, IoU)

```
pxdsResults = semanticseg(imdsTest,net, ...
    'MiniBatchSize',4, ...
    'WriteLocation',tempdir, ...
    'Verbose',true);
```

```
Running semantic segmentation network
-------------------------------------
* Processed 140 images.
```

```
metrics = evaluateSemanticSegmentation(pxdsResults,pxdsTest,'Verbose',true);
```

```
Evaluating semantic segmentation results
----------------------------------------
* Selected metrics: global accuracy, class accuracy, IoU, weighted IoU, BF score.
* Processed 140 images.
* Finalizing... Done.
* Data set metrics:

    GlobalAccuracy    MeanAccuracy    MeanIoU    WeightedIoU    MeanBFScore
    ______________    ____________    _______    ___________    ___________

       0.87695          0.85392       0.6302       0.80851        0.65051
```

```
metrics.NormalizedConfusionMatrix
```

ans = 11×11 table

...

	Sky	Building	Pole	Road	Pavement	Tree	SignSymbol
1 Sky	0.9311	0.0058	0.0137	0	0	0.0466	0.0023
2 Building	0.0071	0.7845	0.0753	0.0002	0.0113	0.0289	0.0448

	Sky	Building	Pole	Road	Pavement	Tree	SignSymbol
3 Pole	0.0114	0.0661	0.7159	0.0004	0.0165	0.0634	0.0477
4 Road	0	0.0002	0.0014	0.9302	0.0496	0.0004	0.0001
5 Pavement	0	0.0078	0.0153	0.0642	0.8847	0.0015	0.0008
6 Tree	0.0289	0.0226	0.0328	0.0003	0.0013	0.8738	0.0112
7 SignSymbol	0.0006	0.0725	0.0575	0.0013	0.0032	0.0351	0.7936
8 Fence	0	0.0460	0.0470	0.0003	0.0116	0.0330	0.0044
9 Car	0.0002	0.0176	0.0158	0.0059	0.0026	0.0046	0.0070
10 Pedestrian	0	0.0251	0.0292	0.0009	0.0086	0.0047	0.0019
11 Bicyclist	0	0.0061	0.0060	0.0025	0.0041	0.0090	0.0003

```
metrics.DataSetMetrics
```

```
ans = 1×5 table
```

	GlobalAccuracy	MeanAccuracy	MeanIoU	WeightedIoU	MeanBFScore
1	0.8770	0.8539	0.6302	0.8085	0.6505

```
metrics.ClassMetrics
```

```
ans = 11×3 table
```

	Accuracy	IoU	MeanBFScore
1 Sky	0.9311	0.9021	0.8952
2 Building	0.7845	0.7610	0.5851
3 Pole	0.7159	0.2148	0.5144
4 Road	0.9302	0.9146	0.7670
5 Pavement	0.8847	0.7057	0.7092
6 Tree	0.8738	0.7632	0.7088
7 SignSymbol	0.7936	0.3931	0.4830
8 Fence	0.8151	0.4648	0.4856
9 Car	0.9096	0.7680	0.6923
10 Pedestrian	0.8763	0.4366	0.6079
11 Bicyclist	0.8784	0.6083	0.5509

```
ans = sortrows(ans,'MeanBFScore','descend')
```

```
ans = 11×3 table
```

	Accuracy	IoU	MeanBFScore
1 Sky	0.9311	0.9021	0.8952
2 Road	0.9302	0.9146	0.7670

	Accuracy	IoU	MeanBFScore
3 Pavement	0.8847	0.7057	0.7092
4 Tree	0.8738	0.7632	0.7088
5 Car	0.9096	0.7680	0.6923
6 Pedestrian	0.8763	0.4366	0.6079
7 Building	0.7845	0.7610	0.5851
8 Bicyclist	0.8784	0.6083	0.5509
9 Pole	0.7159	0.2148	0.5144
10 Fence	0.8151	0.4648	0.4856
11 SignSymbol	0.7936	0.3931	0.4830

```
ans = sortrows(ans,'Accuracy','descend')
```

```
ans = 11×3 table
```

	Accuracy	IoU	MeanBFScore
1 Sky	0.9311	0.9021	0.8952
2 Road	0.9302	0.9146	0.7670
3 Car	0.9096	0.7680	0.6923
4 Pavement	0.8847	0.7057	0.7092
5 Bicyclist	0.8784	0.6083	0.5509
6 Pedestrian	0.8763	0.4366	0.6079
7 Tree	0.8738	0.7632	0.7088
8 Fence	0.8151	0.4648	0.4856
9 SignSymbol	0.7936	0.3931	0.4830
10 Building	0.7845	0.7610	0.5851
11 Pole	0.7159	0.2148	0.5144

Step 7: (OPTIONAL) Repeat steps 3 through 6 using different pretrained networks, training options, data augmentation options, and/or metrics.

Hint: See example at https://www.mathworks.com/help/vision/ref/deeplabv3pluslayers.html

We will use an VGG 16 network pretrained for CamVid. It is about twice as large as the ResNet18 network used above.

```
% From https://blogs.mathworks.com/deep-learning/2018/06/08/semantic-segmentation-usin
pretrainedURL = 'https://www.mathworks.com/supportfiles/vision/data/segnetVGG16CamVid.
pretrainedFolder = fullfile(tempdir,'pretrainedNetwork');
pretrainedNetwork = fullfile(pretrainedFolder,'segnetVGG16CamVid.mat');
```

```
if ~exist(pretrainedNetwork,'file')
    mkdir(pretrainedFolder);
    disp('Downloading pretrained network (107 MB)...');
    websave(pretrainedNetwork,pretrainedURL);
end
```

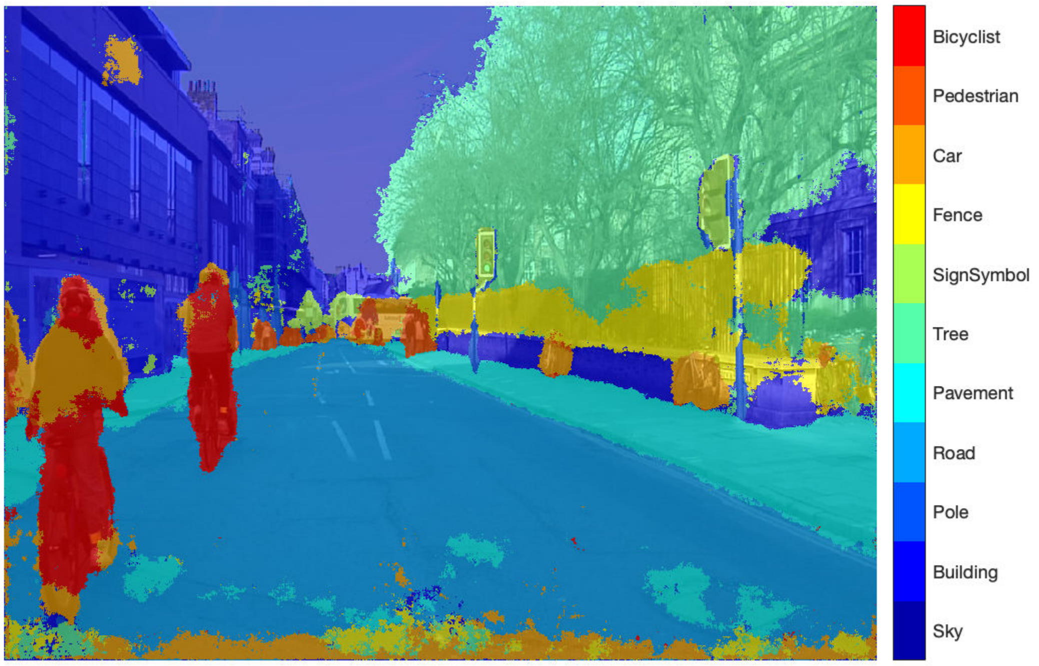
```
Warning: Directory already exists.
Downloading pretrained network (107 MB)...
```

```
data = load(pretrainedNetwork);
net = data.net;
```

Evaluate the new network (store in the "x" variables, so we can compare with the previous network later).

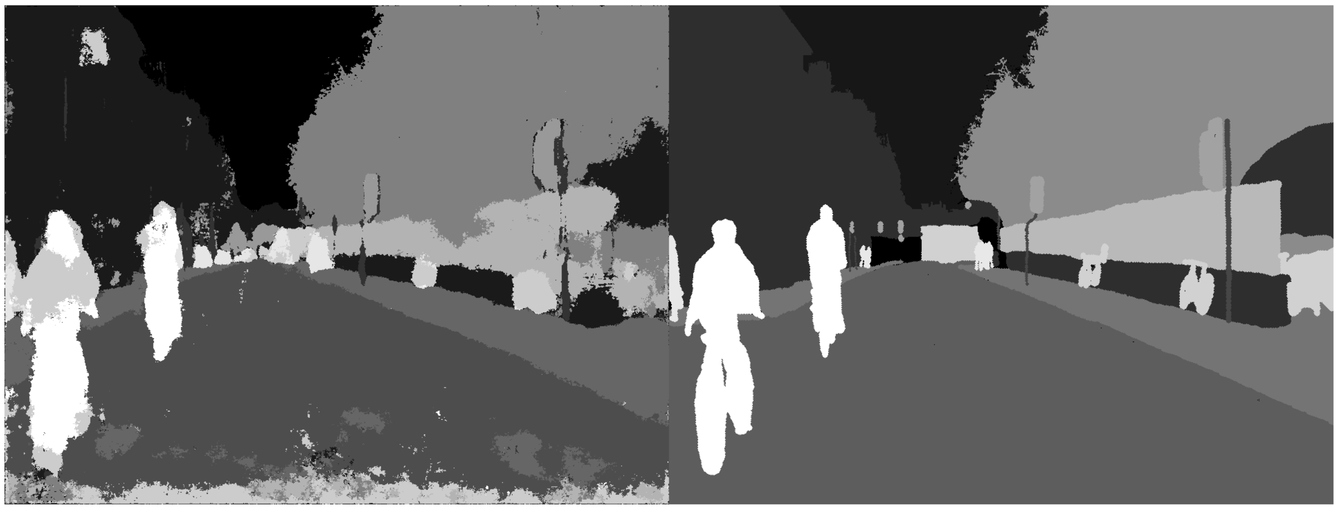
```
Ix = readimage(imdsTest,35);
Cx = semanticseg(Ix, net);

Bx = labeloverlay(Ix,Cx,'Colormap',cmap,'Transparency',0.4);
imshow(Bx)
pixelLabelColorbar(cmap, classes);
```



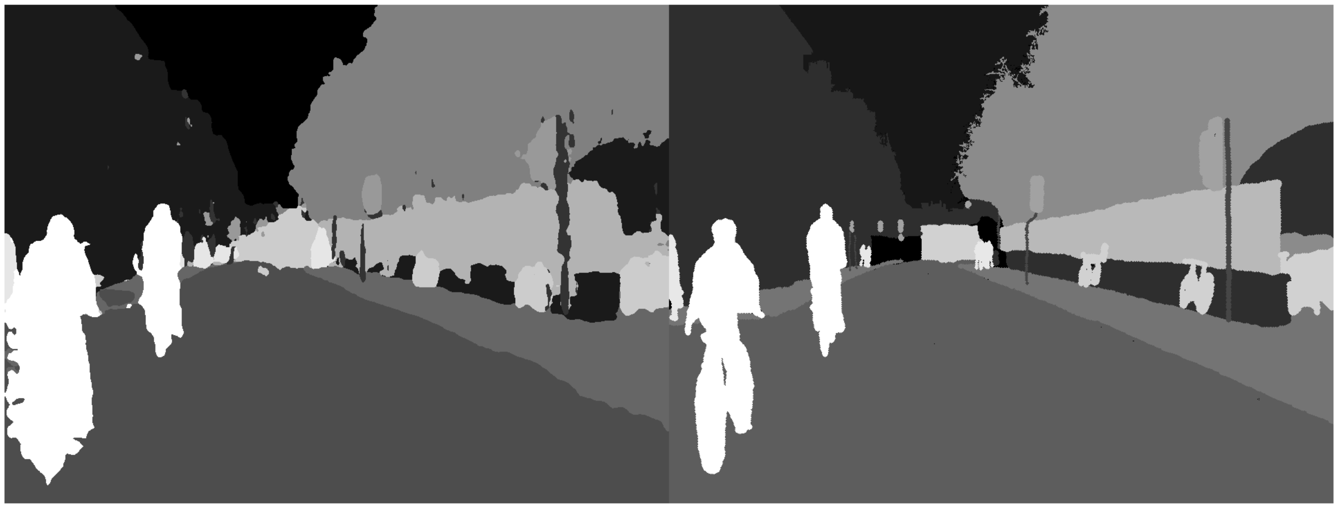
```
expectedResultx = readimage(pxdsTest,35);
actualx = uint8(Cx);
expectedx = uint8(expectedResultx);
```

```
imshowpair(actualx, expectedx, 'montage')
```



For comparison, show the result of the previous network.

```
imshowpair(actual, expected, 'montage')
```



Show metrics.

```
ioux = jaccard(Cx,expectedResultx);
table(classes,ioux)
```

```
ans = 11×2 table
```

	classes	ioux
1	"Sky"	0.9284
2	"Building"	0.8018
3	"Pole"	0.2044

	classes	ioux
4	"Road"	0.7873
5	"Pavement"	0.6642
6	"Tree"	0.8343
7	"SignSymbol"	0.4856
8	"Fence"	0.4793
9	"Car"	0.1168
10	"Pedestrian"	0.3473
11	"Bicyclist"	0.6206

```
pxdsResultsx = semanticseg(imdsTest,net, ...
    'MiniBatchSize',4, ...
    'WriteLocation',tempdir, ...
    'Verbose',true);
```

```
Running semantic segmentation network
-------------------------------------
* Processed 140 images.
```

```
metricsx = evaluateSemanticSegmentation(pxdsResultsx,pxdsTest,'Verbose',true);
```

```
Evaluating semantic segmentation results
----------------------------------------
* Selected metrics: global accuracy, class accuracy, IoU, weighted IoU, BF score.
* Processed 140 images.
* Finalizing... Done.
* Data set metrics:

    GlobalAccuracy    MeanAccuracy    MeanIoU    WeightedIoU    MeanBFScore
    ______________    ____________    _______    ___________    ___________

        0.8225          0.73203       0.53004      0.72418        0.52491
```

```
metricsx.ClassMetrics
```

ans = 11×3 table

	Accuracy	IoU	MeanBFScore
1 Sky	0.9521	0.8990	0.8768
2 Building	0.7907	0.7342	0.5589
3 Pole	0.5894	0.2633	0.5578
4 Road	0.8067	0.7728	0.4604
5 Pavement	0.6950	0.5505	0.5174
6 Tree	0.8866	0.7058	0.5466
7 SignSymbol	0.5710	0.3447	0.4603
8 Fence	0.6580	0.2522	0.3127

	Accuracy	IoU	MeanBFScore
9 Car	0.8879	0.5126	0.2955
10 Pedestrian	0.6751	0.3992	0.6082
11 Bicyclist	0.5398	0.3962	0.4171

Go back to top

Supporting Functions

```
function labelIDs = camvidPixelLabelIDs()
% Return the label IDs corresponding to each class.
%
% The CamVid dataset has 32 classes. Group them into 11 classes following
% the original SegNet training methodology [1].
%
% The 11 classes are:
%   "Sky" "Building", "Pole", "Road", "Pavement", "Tree", "SignSymbol",
%   "Fence", "Car", "Pedestrian",  and "Bicyclist".
%
% CamVid pixel label IDs are provided as RGB color values. Group them into
% 11 classes and return them as a cell array of M-by-3 matrices. The
% original CamVid class names are listed alongside each RGB value. Note
% that the Other/Void class are excluded below.
labelIDs = { ...

    % "Sky"
    [
    128 128 128; ... % "Sky"
    ]

    % "Building"
    [
    000 128 064; ... % "Bridge"
    128 000 000; ... % "Building"
    064 192 000; ... % "Wall"
    064 000 064; ... % "Tunnel"
    192 000 128; ... % "Archway"
    ]

    % "Pole"
    [
    192 192 128; ... % "Column_Pole"
    000 000 064; ... % "TrafficCone"
    ]

    % Road
    [
    128 064 128; ... % "Road"
    128 000 192; ... % "LaneMkgsDriv"
    192 000 064; ... % "LaneMkgsNonDriv"
    ]

    % "Pavement"
    [
    000 000 192; ... % "Sidewalk"
    064 192 128; ... % "ParkingBlock"
    128 128 192; ... % "RoadShoulder"
    ]

    % "Tree"
    [
    128 128 000; ... % "Tree"
    192 192 000; ... % "VegetationMisc"
    ]

    % "SignSymbol"
    [
    192 128 128; ... % "SignSymbol"
    128 128 064; ... % "Misc_Text"
    000 064 064; ... % "TrafficLight"
    ]

    % "Fence"
    [
    064 064 128; ... % "Fence"
    ]

    % "Car"
    [
    064 000 128; ... % "Car"
    064 128 192; ... % "SUVPickupTruck"
    192 128 192; ... % "Truck_Bus"
    192 064 128; ... % "Train"
    128 064 064; ... % "OtherMoving"
    ]

    % "Pedestrian"
    [
    064 064 000; ... % "Pedestrian"
    192 128 064; ... % "Child"
    064 000 192; ... % "CartLuggagePram"
    064 128 064; ... % "Animal"
    ]

    % "Bicyclist"
    [
    000 128 192; ... % "Bicyclist"
    192 000 192; ... % "MotorcycleScooter"
    ]

    };
```

```
end
```

```
function pixelLabelColorbar(cmap, classNames)
% Add a colorbar to the current axis. The colorbar is formatted
% to display the class names with the color.

colormap(gca,cmap)

% Add colorbar to current figure.
c = colorbar('peer', gca);

% Use class names for tick marks.
c.TickLabels = classNames;
numClasses = size(cmap,1);

% Center tick labels.
c.Ticks = 1/(numClasses*2):1/numClasses:1;

% Remove tick mark.
c.TickLength = 0;
end
```

```
% function cmap = camvidColorMap()
% % Define the colormap used by CamVid dataset.
%
% cmap = [
%     128 128 128   % Sky
%     128 0 0       % Building
%     192 192 192   % Pole
%     128 64 128    % Road
%     60 40 222     % Pavement
%     128 128 0     % Tree
%     192 128 128   % SignSymbol
%     64 64 128     % Fence
%     64 0 128      % Car
%     64 64 0       % Pedestrian
%     0 128 192     % Bicyclist
%     ];
%
% % Normalize between [0 1].
% cmap = cmap ./ 255;
% end
```

```
function [imdsTrain, imdsVal, imdsTest, pxdsTrain, pxdsVal, pxdsTest] = partitionCamVi
% Partition CamVid data by randomly selecting 60% of the data for training. The
% rest is used for testing.
% Set initial random state for example reproducibility.
rng(0);
numFiles = numel(imds.Files);
shuffledIndices = randperm(numFiles);

% Use 60% of the images for training.
numTrain = round(0.60 * numFiles);
trainingIdx = shuffledIndices(1:numTrain);

% Use 20% of the images for validation
numVal = round(0.20 * numFiles);
valIdx = shuffledIndices(numTrain+1:numTrain+numVal);

% Use the rest for testing.
testIdx = shuffledIndices(numTrain+numVal+1:end);

% Create image datastores for training and test.
trainingImages = imds.Files(trainingIdx);
valImages = imds.Files(valIdx);
testImages = imds.Files(testIdx);

imdsTrain = imageDatastore(trainingImages);
imdsVal = imageDatastore(valImages);
imdsTest = imageDatastore(testImages);

% Extract class and label IDs info.
classes = pxds.ClassNames;
labelIDs = camvidPixelLabelIDs();

% Create pixel label datastores for training and test.
trainingLabels = pxds.Files(trainingIdx);
valLabels = pxds.Files(valIdx);
testLabels = pxds.Files(testIdx);

pxdsTrain = pixelLabelDatastore(trainingLabels, classes, labelIDs);
pxdsVal = pixelLabelDatastore(valLabels, classes, labelIDs);
pxdsTest = pixelLabelDatastore(testLabels, classes, labelIDs);
end
```