Train A Semantic Segmentation Network

Load the training data.

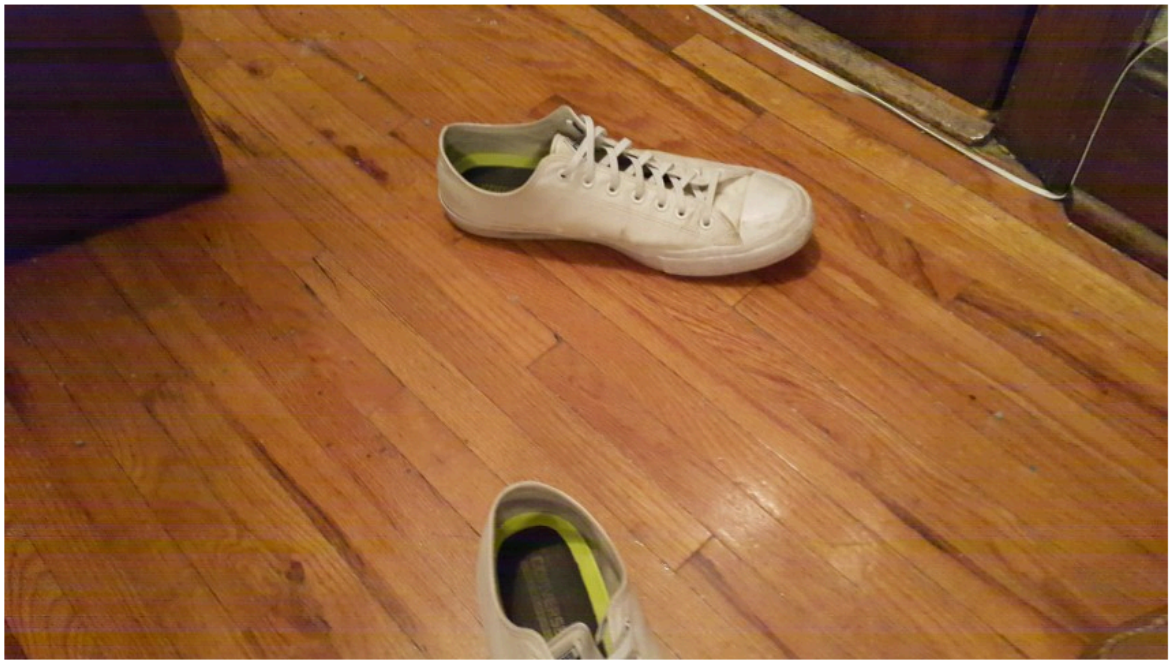
```
imageDir = 'C:\Users\grm\Google Drive\aStThomas\7ArtificalIntelligence\Assignments\5 Object Det
labelDir = 'C:\Users\grm\Google Drive\aStThomas\7ArtificalIntelligence\Assignments\5 Object Det
```

Create an image datastore for the images.

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

Read and display the first image.

```
I = readimage(imds,1);
figure
imshow(I)
```



Create a `pixelLabelDatastore` for the ground truth pixel labels.

Load the pixel label images using a `pixelLabelDatastore` to define the mapping between label IDs and categorical names. In the dataset used here, the labels are "sky", "grass", "building", and "sidewalk". The label IDs for these classes are 1, 2, 3, 4, respectively.

Define the class names.

```
classNames = ["Cats" "Boots"];
```

Define the label ID for each class name.

```
labelIDs = [1 2];
```

Create a `pixelLabelDatastore`.

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

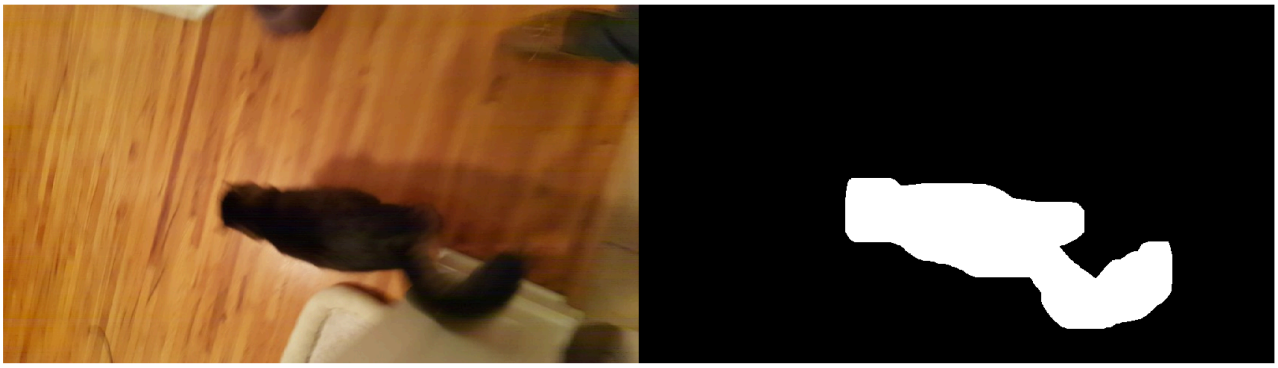
Read the first pixel label image.

```
D = readimage(pxds,2);
```

Visualize training images and ground truth pixel labels.

```
I = read(imds);
C = read(pxds);

I = imresize(I, 3);
L = imresize(uint8(C), 3);
imshowpair(I, L, 'montage')
```



```
Warning: Image is too big to fit on screen; displaying at 33%
```

Check output of all loaded files.

```
N = numel(imds.Files);
for i = 1:N
    fn = imds.Files{i};
    im = readimage(imds, i);
    fprintf('Filename: %s\nSize %d-by-%d-by-%d\n', fn, size(im,1), size(im,2), size(im,3));
end
```

```
Filename: C:\Users\grm\Google Drive\aStThomas\7ArtificalIntelligence\Assignments\5 Object Detection\catsboots\imgSu
Size 480-by-853-by-3
```

```
Filename: C:\Users\grm\Google Drive\aStThomas\7ArtificalIntelligence\Assignments\5 Object Detection\catsboots\imgSu
Size 480-by-853-by-3
Filename: C:\Users\grm\Google Drive\aStThomas\7ArtificalIntelligence\Assignments\5 Object Detection\catsboots\imgSu
Size 480-by-853-by-3
Filename: C:\Users\grm\Google Drive\aStThomas\7ArtificalIntelligence\Assignments\5 Object Detection\catsboots\imgSu
Size 480-by-853-by-3
```

Create a semantic segmentation network. This network uses a simple semantic segmentation network based on a downsampling and upsampling design.

```
imageSize = [480 853 3];

layers = [
    imageInputLayer(imageSize)
    convolution2dLayer(3, 8, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    fullyConnectedLayer(2) % this should be equal to class count
    softmaxLayer
    classificationLayer];
```

Setup training options.

```
opts = trainingOptions('sgdm', ...
    'InitialLearnRate', 1e-3, ...
    'MaxEpochs', 100, ...
    'MiniBatchSize', 2);
```

Create a pixel label image datastore that contains training data.

```
trainingData = pixelLabelImageDatastore(imds, pxds);
```

Train the network.

```
net = trainNetwork(trainingData, layers, opts);
```

```
Error using trainNetwork (line 150)
Invalid training data. The output size ([1 1 2]) of the last layer does not match the response size ([480
853 2]).
```

Read and display a test image.

```
testImage = imread('20181105_172802 (Small).jpg');
imshow(testImage)
```

Segment the test image and display the results.

```
C = semanticseg(testImage, net);
B = labeloverlay(testImage, C);
imshow(B)
```

Improve the results

The network failed to segment the triangles and classified every pixel as "background". The training appeared to be going well with training accuracies greater than 90%. However, the network only learned to classify the background class. To understand why this happened, you can count the occurrence of each pixel label across the dataset.

```
tbl = countEachLabel(trainingData)
```

The majority of pixel labels are for the background. The poor results are due to the class imbalance. Class imbalance biases the learning process in favor of the dominant class. That's why every pixel is classified as "background". To fix this, use class weighting to balance the classes. There are several methods for computing class weights. One common method is inverse frequency weighting where the class weights are the inverse of the class frequencies. This increases weight given to under-represented classes.

```
totalNumberOfPixels = sum(tbl.PixelCount);
frequency = tbl.PixelCount / totalNumberOfPixels;
classWeights = 1./frequency
```

Class weights can be specified using the `pixelClassificationLayer`. Update the last layer to use a `pixelClassificationLayer` with inverse class weights.

```
layers(end) = pixelClassificationLayer('Classes',tbl.Name,'ClassWeights',classWeights);
```

Train network again.

```
net = trainNetwork(trainingData,layers,opts);
```

Try to segment the test image again.

```
C = semanticseg(testImage,net);
B = labeloverlay(testImage,C);
imshow(B)
```

Using class weighting to balance the classes produced a better segmentation result. Additional steps to improve the results include increasing the number of epochs used for training, adding more training data, or modifying the network.