Train U-Net for semantic segmentation

Carlos Enrique Lopez Jimenez A01283855

Genaro Gallardo Bórquez A01382459

Claudia Esmeralda González Castillo A01411506

Jesús Eduardo Martínez Herrera A01283785

Mario Veccio Castro Berrones A00826824

```matlab
% Load training images and pixel labels
dataSetDir = fullfile(toolboxdir('vision'),'visiondata','triangleImages');
imageDir = fullfile(dataSetDir,'trainingImages');
labelDir = fullfile(dataSetDir,'trainingLabels');

% Create an imageDatastore object to store the training images.
imds = imageDatastore(imageDir);

% Define the class names and their associated label IDs.
classNames = ["triangle","background"];
labelIDs   = [255 0];

% Create a pixelLabelDatastore object to store the ground truth pixel
% labels for the training images.
pxds = pixelLabelDatastore(labelDir,classNames,labelIDs);

% Crating the U-Net network
imageSize = [32 32];
numClasses = 2;
lgraph = unetLayers(imageSize, numClasses)
```

```
lgraph = 
  LayerGraph with properties:

         Layers: [58×1 nnet.cnn.layer.Layer]
    Connections: [61×2 table]
     InputNames: {'ImageInputLayer'}
    OutputNames: {'Segmentation-Layer'}
```

```matlab

ds = combine(imds,pxds); % Datastore for training the network.

% Training options
options = trainingOptions('sgdm', ...
    'InitialLearnRate',1e-2, ...
    'MaxEpochs',30, ...
    'VerboseFrequency',10);

% Train the network
net = trainNetwork(ds,lgraph,options)
```

```
Training on single CPU.
Initializing input data normalization.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:57 |       87.60% |       1.0842 |          0.0100 |
|      10 |          10 |       00:02:40 |       94.75% |       0.6044 |          0.0100 |
|      20 |          20 |       00:03:58 |       94.74% |       0.1054 |          0.0100 |
|      30 |          30 |       00:05:21 |       96.13% |       0.0825 |          0.0100 |
|=========================================================================================|
Training finished: Max epochs completed.
net = 
  DAGNetwork with properties:

         Layers: [58×1 nnet.cnn.layer.Layer]
    Connections: [61×2 table]
     InputNames: {'ImageInputLayer'}
    OutputNames: {'Segmentation-Layer'}
```

Test your trained U-Net

Specify test images and labels

```
testImagesDir = fullfile(dataSetDir,'testImages');
testimds = imageDatastore(testImagesDir);
testLabelsDir = fullfile(dataSetDir,'testLabels');

% Ground truth pixel labels for the test images
pxdsTruth = pixelLabelDatastore(testLabelsDir,classNames,labelIDs);

% Prediction
pxdsResults = semanticseg(testimds,net,"WriteLocation",tempdir);
```

```
Running semantic segmentation network
-------------------------------------
* Processed 100 images.
```

```

% Evaluate the Quality of the Prediction
metrics = evaluateSemanticSegmentation(pxdsResults,pxdsTruth);
```

```
Evaluating semantic segmentation results
----------------------------------------
* Selected metrics: global accuracy, class accuracy, IoU, weighted IoU, BF score.
* Processed 100 images.
* Finalizing... Done.
* Data set metrics:

    GlobalAccuracy    MeanAccuracy    MeanIoU    WeightedIoU    MeanBFScore
    ______________    ____________    _______    ___________    ___________

        0.9669          0.79153       0.71051      0.94236        0.63329
```

```

% Inspect class metrcis
metrics.ClassMetrics
```

```
ans = 2×3 table
```

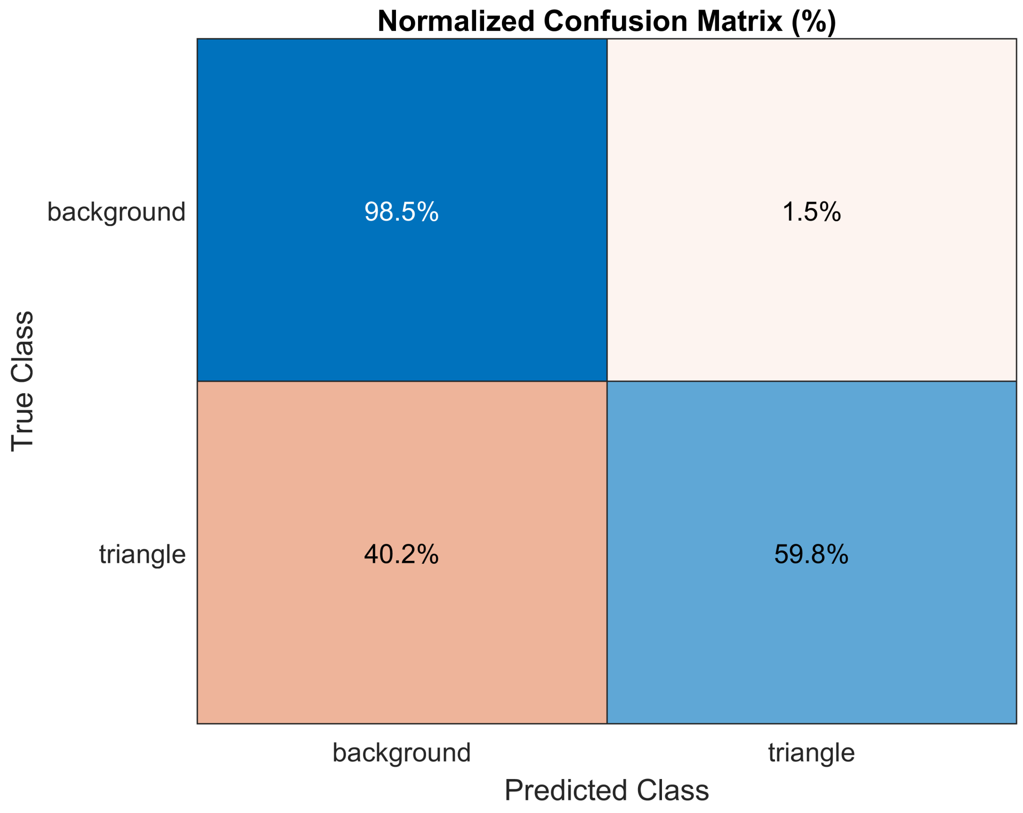
	Accuracy	IoU	MeanBFScore
1 triangle	0.5983	0.4551	0.4167
2 background	0.9848	0.9660	0.8499

```
% Display confusion matrix
metrics.ConfusionMatrix
```

```
ans = 2×2 table
```

	triangle	background
1 triangle	2830	1900
2 background	1489	96181

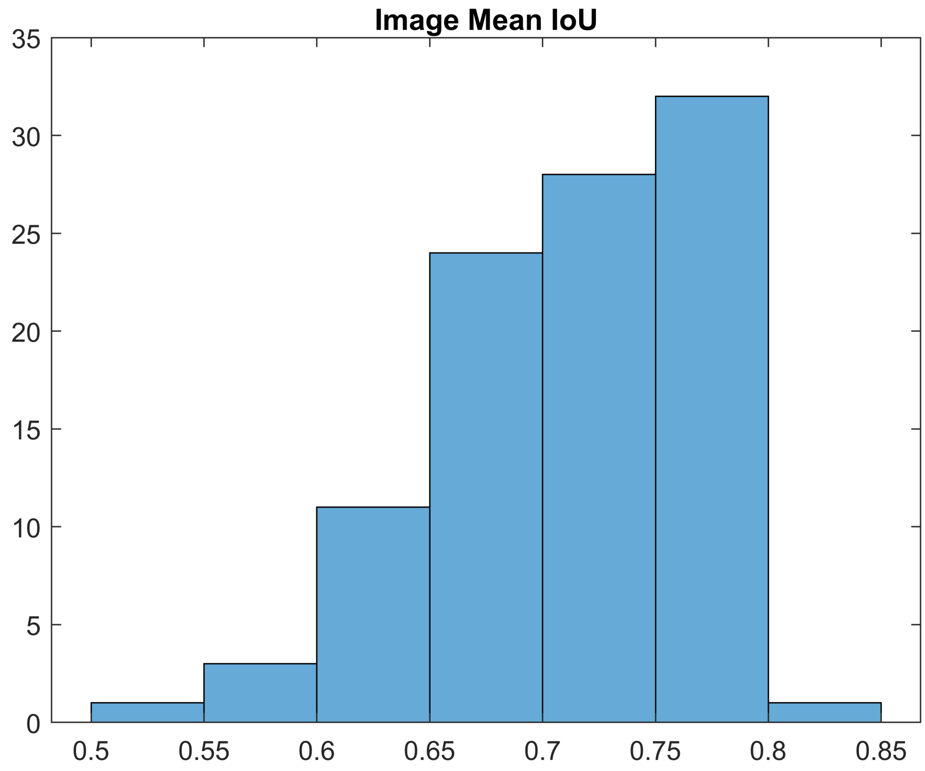
```
% Visualize the normalized confusion matrix as a confusion chart in a figure window.
figure (2)
cm = confusionchart(metrics.ConfusionMatrix.Variables, ...
  classNames, Normalization='row-normalized');
cm.Title = 'Normalized Confusion Matrix (%)';
```



```
% Inspect an Image Metric
imageIoU = metrics.ImageMetrics.MeanIoU;

figure (3)
histogram(imageIoU)
```

```
title('Image Mean IoU')
```



Test image with the lowest IoU.

Find the test image with the lowest IoU.

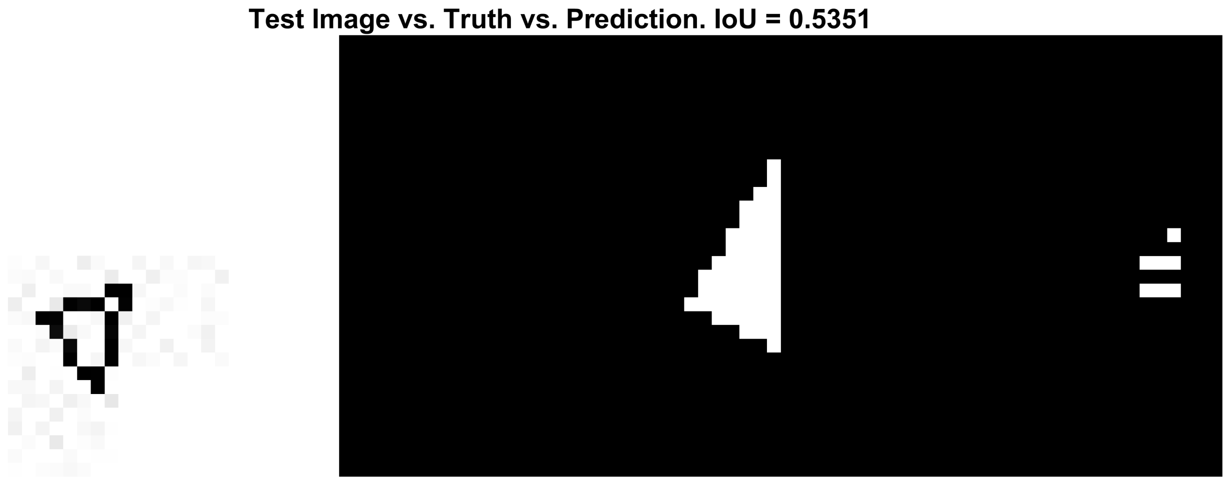
```
[minIoU, worstImageIndex] = min(imageIoU);
minIoU = minIoU(1);
worstImageIndex = worstImageIndex(1);

% Read the test image with the worst IoU, its ground truth labels, and its predicted labels for
worstTestImage = readimage(imds,worstImageIndex);
worstTrueLabels = readimage(pxdsTruth,worstImageIndex);
worstPredictedLabels = readimage(pxdsResults,worstImageIndex);

% Convert the label images to images that can be displayed in a figure window.
worstTrueLabelImage = im2uint8(worstTrueLabels == classNames(1));
worstPredictedLabelImage = im2uint8(worstPredictedLabels == classNames(1));

% Display the worst test image, the ground truth, and the prediction.
worstMontage = cat(4,worstTestImage,worstTrueLabelImage,worstPredictedLabelImage);
worstMontage = imresize(worstMontage,4,"nearest");

figure (4)
montage(worstMontage,'Size',[1 3])
title(['Test Image vs. Truth vs. Prediction. IoU = ' num2str(minIoU)])
```



Test Image vs. Truth vs. Prediction. IoU = 0.5351

Test image with the highest IoU.

Codigo completado

```
[maxIoU, bestImageIndex] = max(imageIoU);
maxIoU = maxIoU(1);
bestImageIndex = bestImageIndex(1);

bestTestImage = readimage(imds,bestImageIndex);
bestTrueLabels = readimage(pxdsTruth,bestImageIndex);
bestPredictedLabels = readimage(pxdsResults,bestImageIndex);

bestTrueLabelImage = im2uint8(bestTrueLabels == classNames(1));
bestPredictedLabelImage = im2uint8(bestPredictedLabels == classNames(1));

bestMontage = cat(4,bestTestImage,bestTrueLabelImage,bestPredictedLabelImage);
bestMontage = imresize(bestMontage,4,"nearest");

figure (5)
montage(bestMontage,'Size',[1 3])
title(['Test Image vs. Truth vs. Prediction. IoU = ' num2str(maxIoU)])
```

Test Image vs. Truth vs. Prediction. IoU = 0.82258