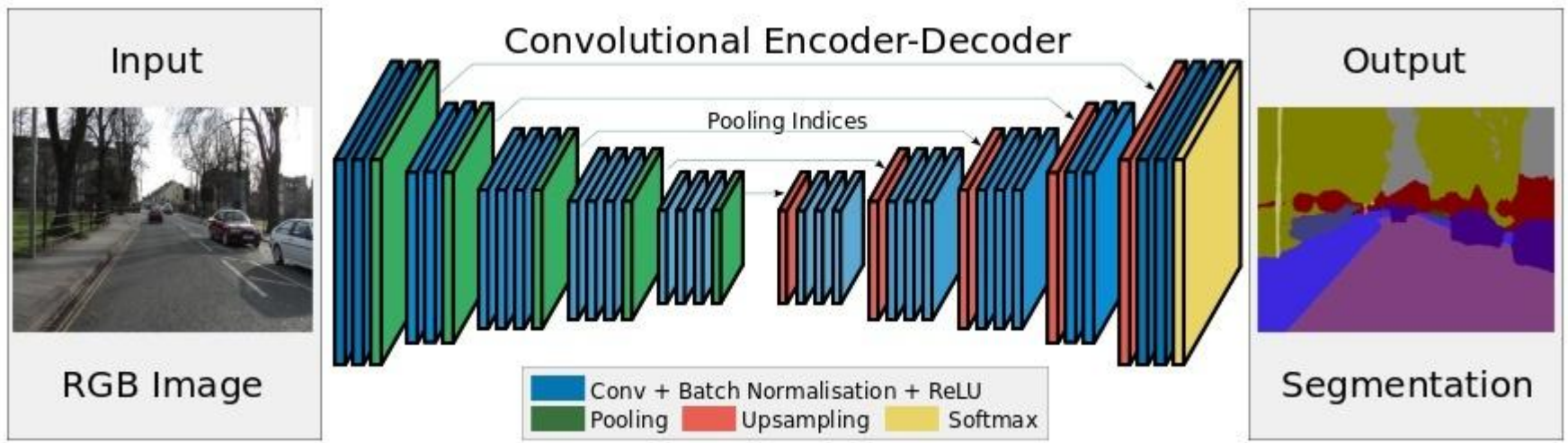
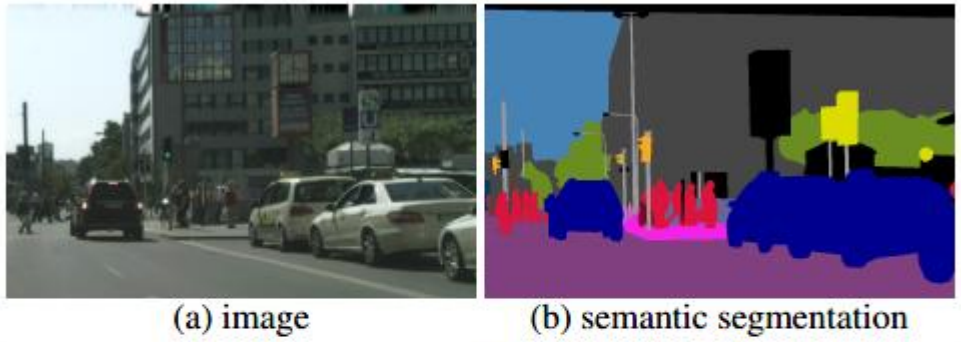
Semantic Segmentation with CNN



图像分割

图像分割主要包括语义分割（Semantic Segmentation）和实例分割（Instance Segmentation）。**语义分割**是对图像中的每个像素都划分出对应的类别，即实现像素级别的分类；而类的具体对象，即为实例，那么实例分割不但要进行像素级别的分类，还需在具体的类别基础上区别开不同的个体。例如，图像有多个人甲、乙、丙，那边他们的语义分割结果都是人，而实例分割结果却是不同的对象。另外，为了同时实现实例分割与不可数类别的语义分割，相关研究又提出了全景分割（Panoptic Segmentation）的概念。语义分割、实例分割和全景分割具体如图所示。



(a) image (b) semantic segmentation

(c) instance segmentation (d) panoptic segmentation

输入和输出

语义分割的输入是一张原始的RGB图像或者单通道图像，但是输出不再是简单的分类类别或者目标定位，而是带有各个像素类别标签的与输入同分辨率的分割图像。简单来说，我们的输入输出都是图像，而且是同样大小的图像。所以，语义分割的任务就是输入图像经过深度学习算法处理得到带有语义标签的同样尺寸的输出图像。

segmented

1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

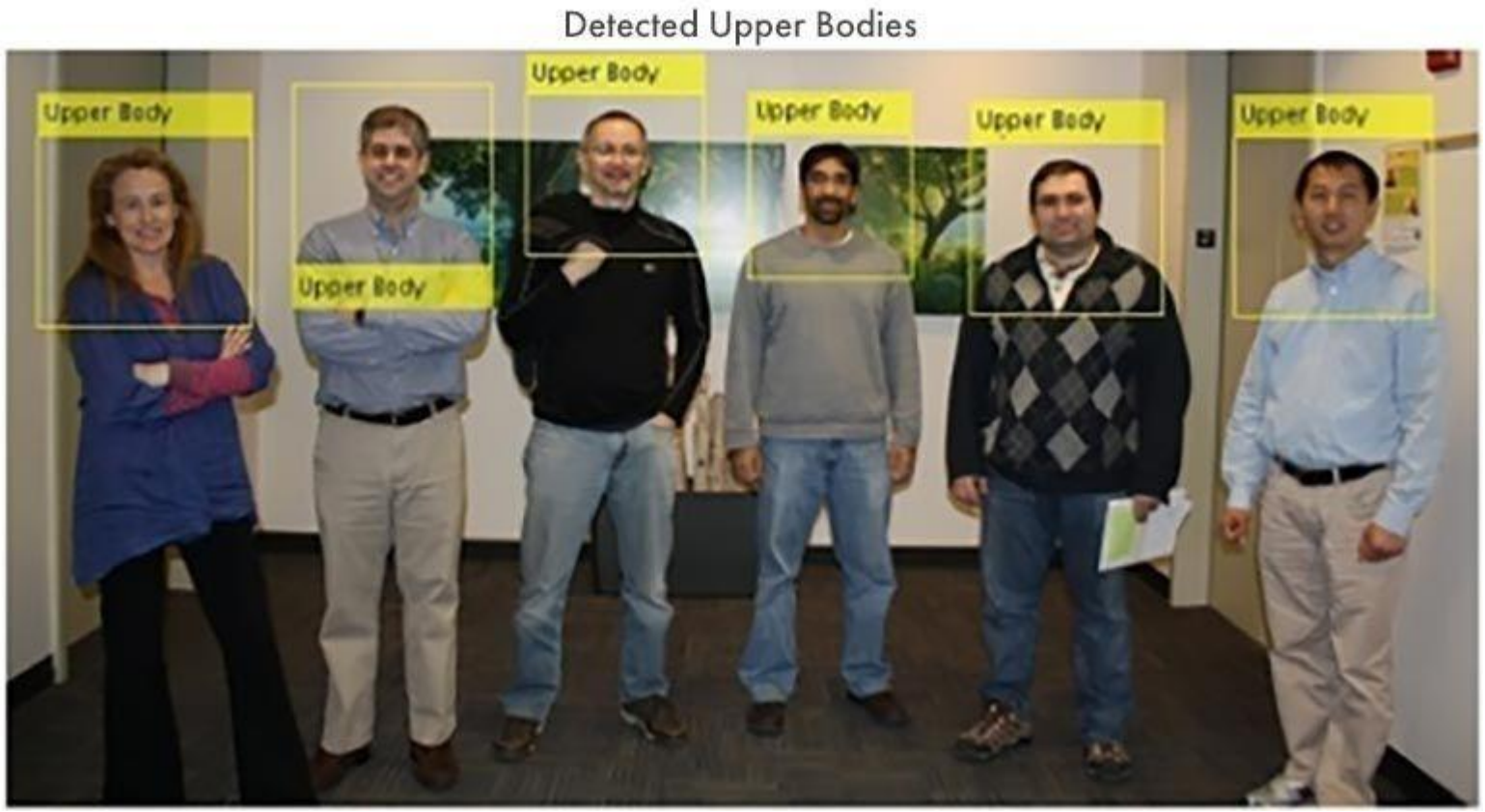
```
3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5
3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5
3 3 3 3 3 3 1 1 3 3 3 3 5 5 5 5 5 5
3 3 3 3 3 1 1 1 1 3 3 3 5 5 5 5 5 5
3 3 3 3 3 3 1 1 3 3 3 5 5 5 5 5 5 5
5 5 3 3 3 3 1 1 3 3 5 5 5 5 5 5 5 5
4 4 3 4 1 1 1 1 1 1 4 4 4 5 5 5 5 5
4 4 3 4 1 1 1 1 1 1 4 4 4 4 4 5 5 5
4 4 4 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4
3 3 3 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4
3 3 3 1 2 2 1 1 1 1 1 4 4 4 4 4 4 4
3 3 3 1 2 2 1 1 1 1 1 4 4 4 4 4 4 4
```

Input

Semantic Labels

语义分割 vs 对象检测

语义分割可以作为对象检测的一种有用替代方法，因为它允许感兴趣对象在像素级别上跨越图像中的多个区域。这种技术可以清楚地检测到形态不规则的对象，相比之下，对象检测要求对象必须位于有边界的方框内。



语义分割应用

因为语义分割会给图像中的像素加上标签，所以精确性高于其他形式的对象检测。这使得语义分割适用于各种需要准确图像映射的行业应用，比如：

自动驾驶 — 通过区分道路与障碍物，比如行人、人行道、电线杆和其他汽车，让汽车识别可行驶的路径
工业检测 — 用于检测材料中的缺陷，如晶圆检验
卫星影像 — 用于识别高山、河流、沙漠和其他地形
医学成像 — 用于分析和检测细胞中的癌变
机器人视觉 — 用于识别物体和地形并进行导航

语义分割步骤

训练语义分割网络进行图像分类的过程遵循以下步骤：

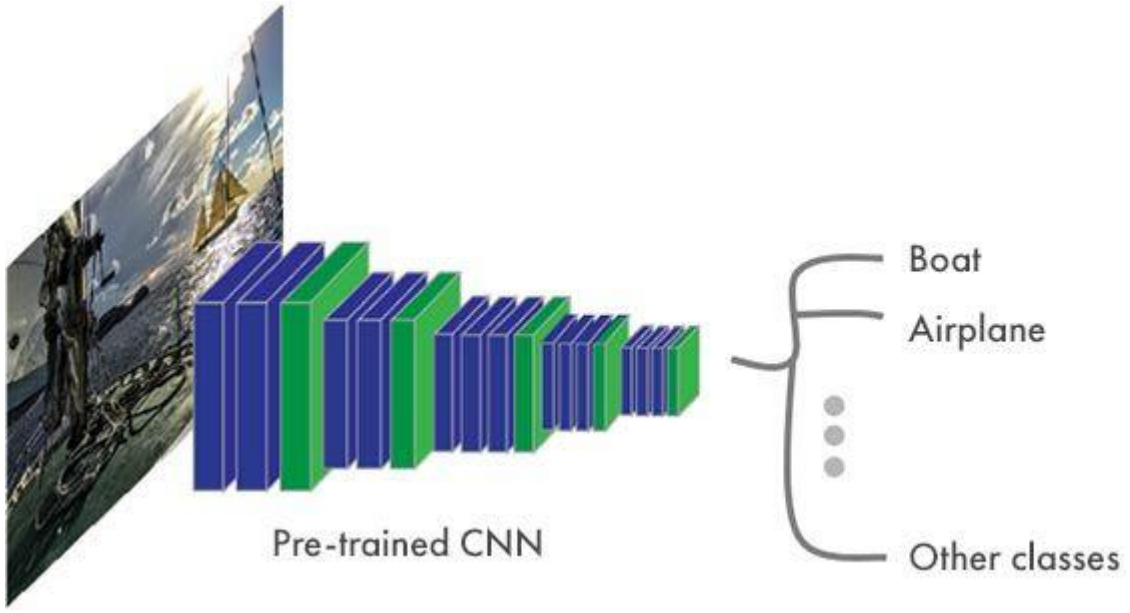
1. 分析一组带像素标签的图像。
2. 创建一个语义分割网络。
3. 训练该网络将图像划分为不同像素类别。
4. 评估网络的准确性。

语义分割示例

下图显示了用于自动驾驶的语义分割的真实示例。道路的图像自动从其他车辆中分割出来。

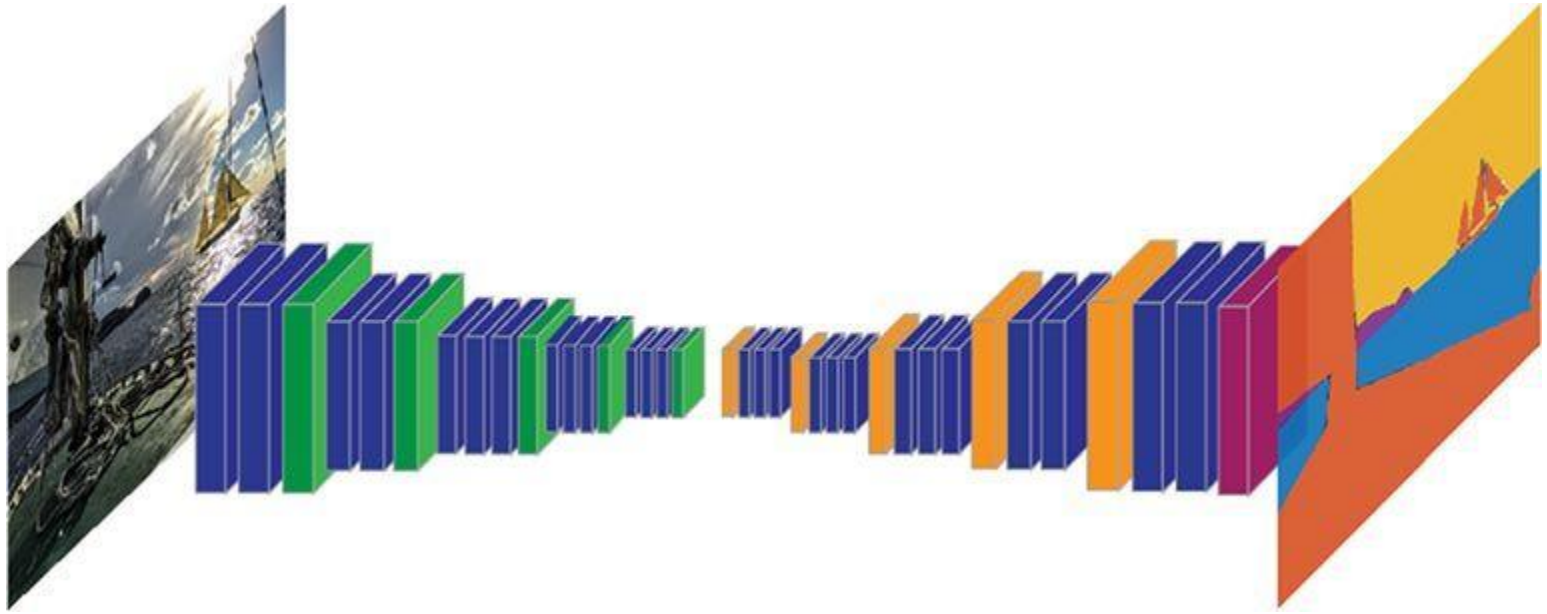
语义分割架构

语义分割的常用架构基于卷积神经网络 (CNN) 。典型的 CNN 架构如图 所示。此 CNN 将整个图像划分为许多预定义类别中的一个。



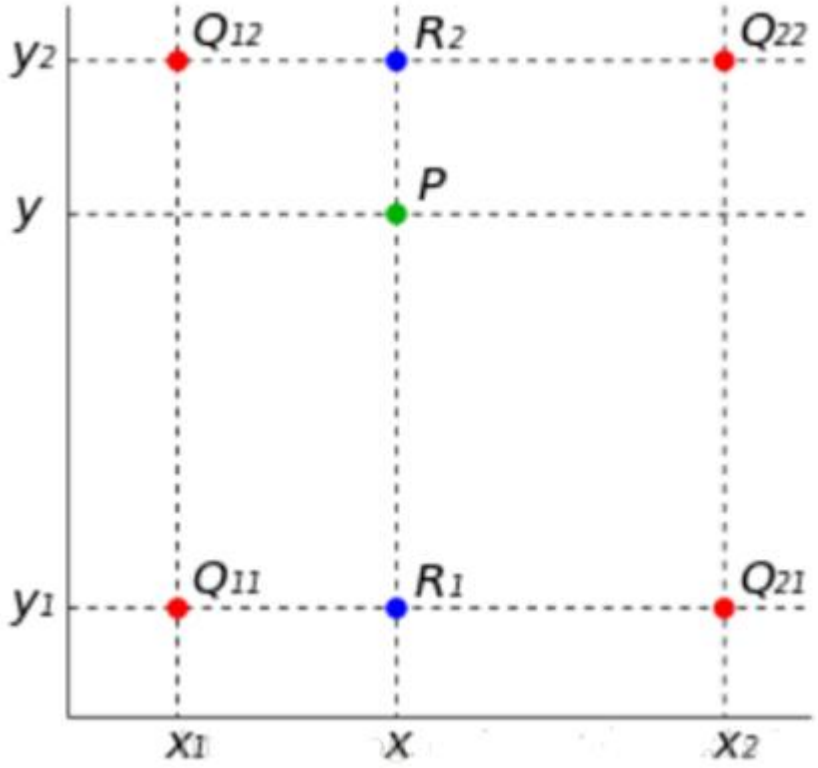
语义分割架构

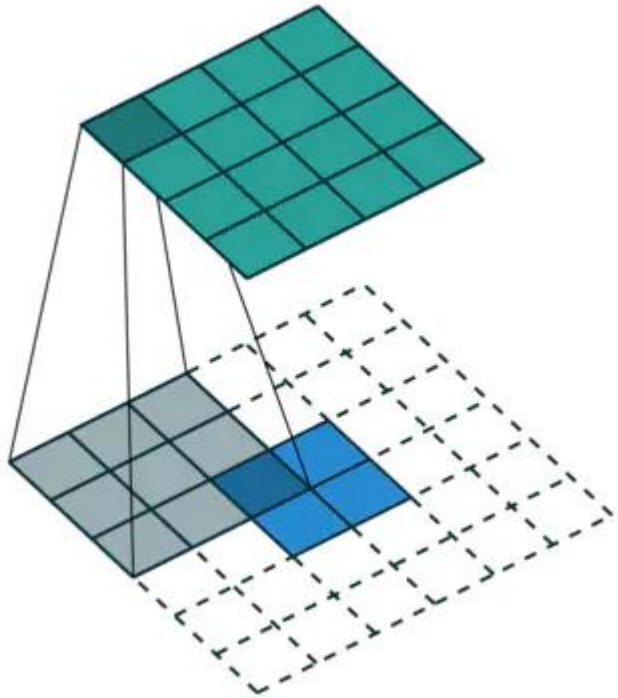
要在像素级别上分类，而不是对整个图像分类，可以追加一个 CNN 的逆向实现。上采样过程的执行次数与下采样过程相同，以确保最终图像的大小与输入图像相同。最后使用一个像素分类输出层，将每个像素映射到一个特定类。这就形成了一个编码器-解码器架构，从而实现语义分割。



上采样方法

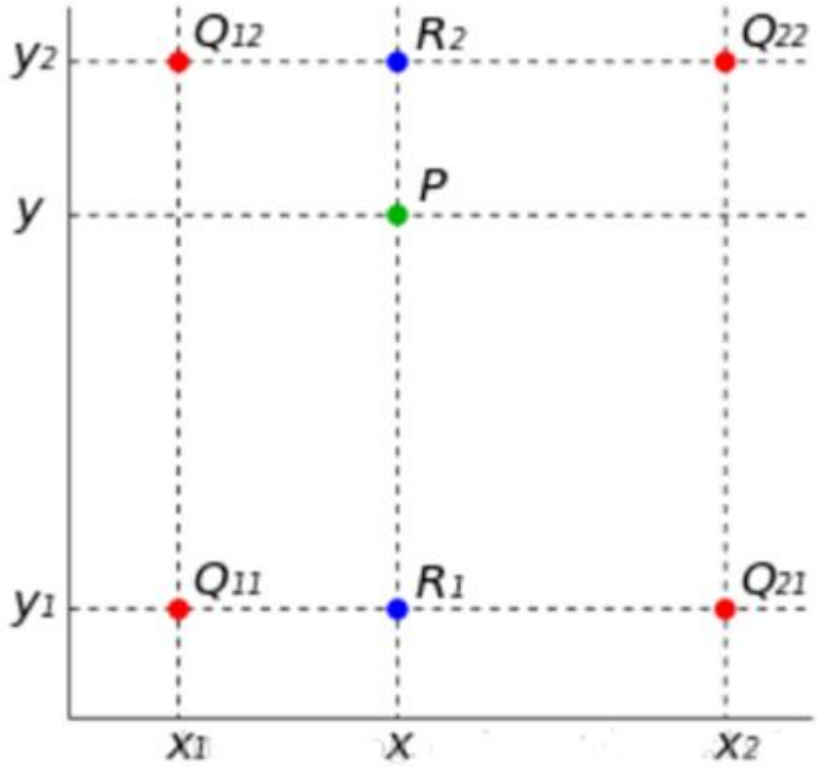
图像分割时，在卷积提取到抽象特征后需要通过上采样将feature map还原到原图大小。常见的上采样方法有双线性插值、转置卷积、上采样（unsampling）和上池化（unpooling）。

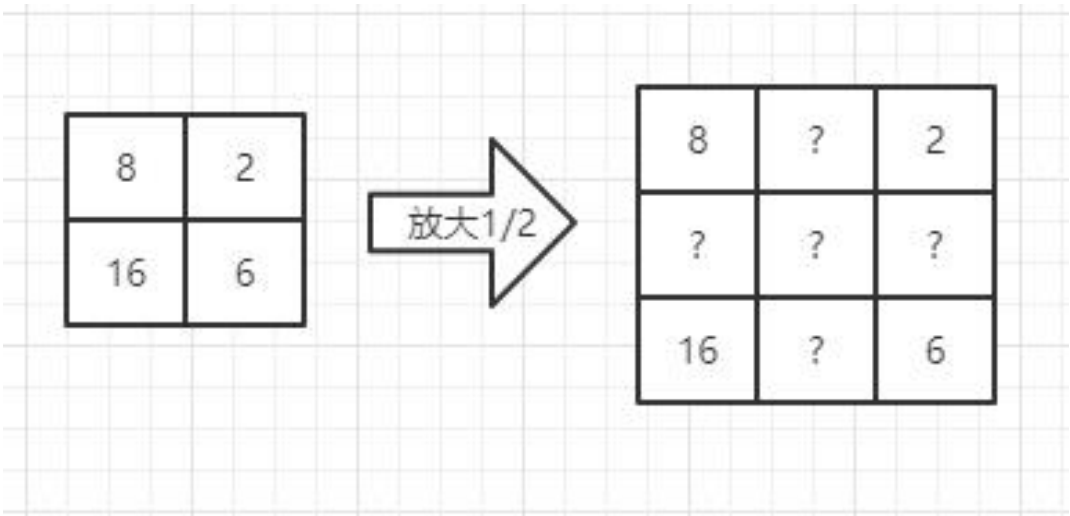




双线性插值

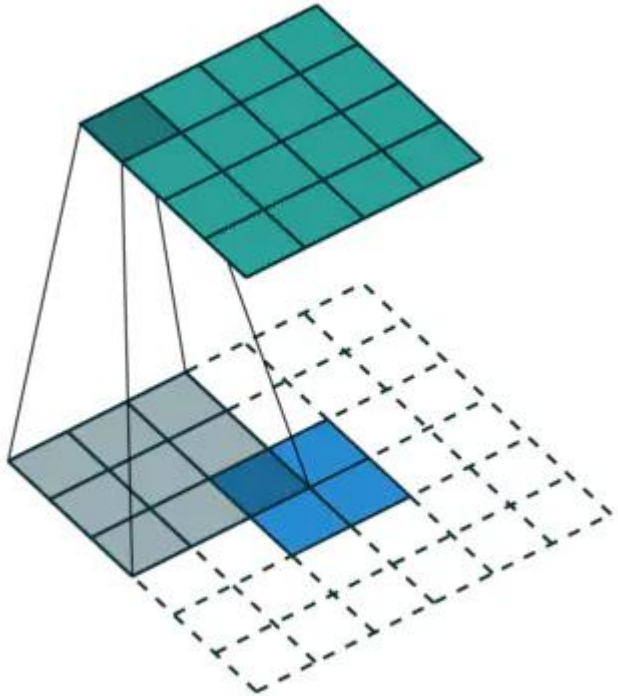
双线性插值是对线性插值在二维直角网格上的扩展，用于对双变量函数（例如 x 和 y）进行插值。其核心思想是在两个方向分别进行一次线性插值。双线性插值方法中不需要学习任何参数。





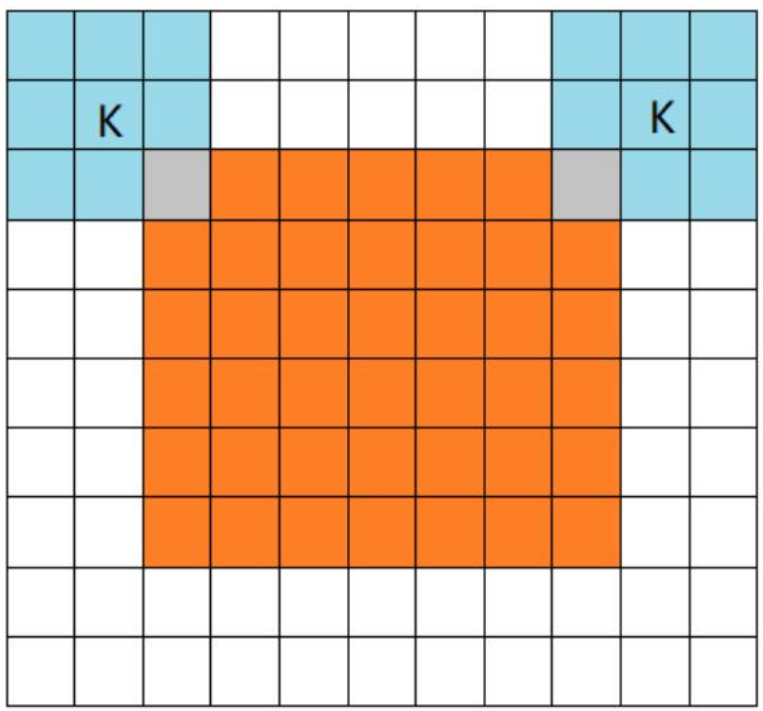
转置卷积

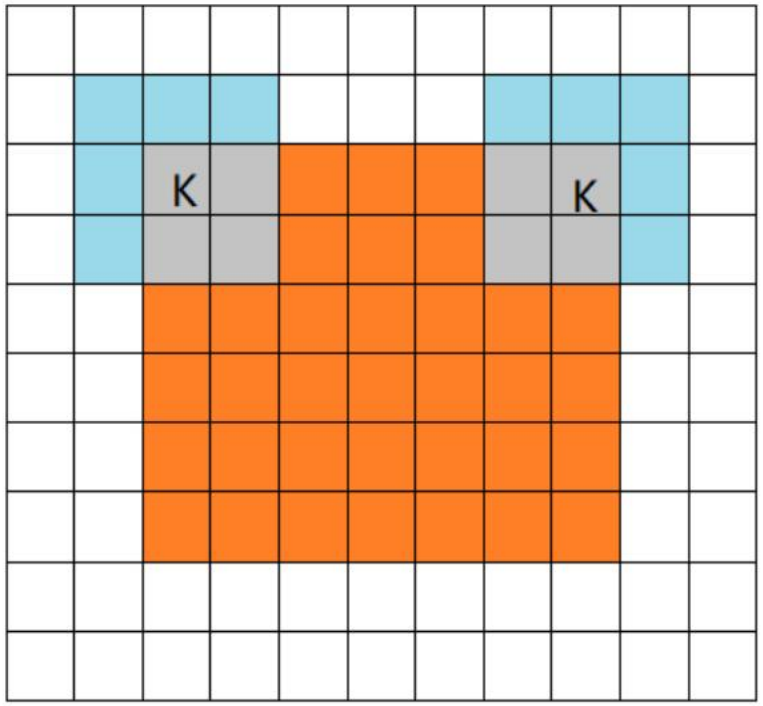
转置卷积(Transposed Convolution)也称之为反卷积(Deconvolution)和分数步长卷积(Fractionally-strided Convolution)

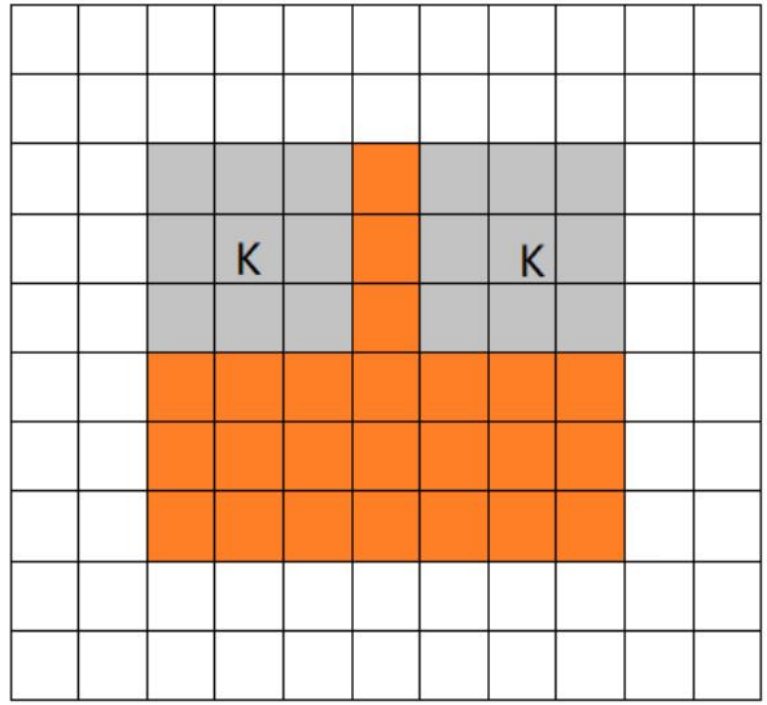


卷积的三种模式

卷积的三种模式:full, same, valid

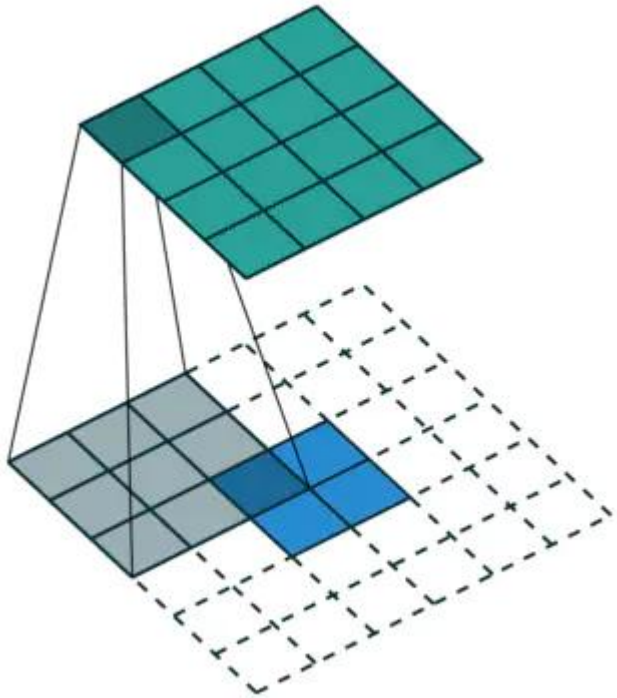




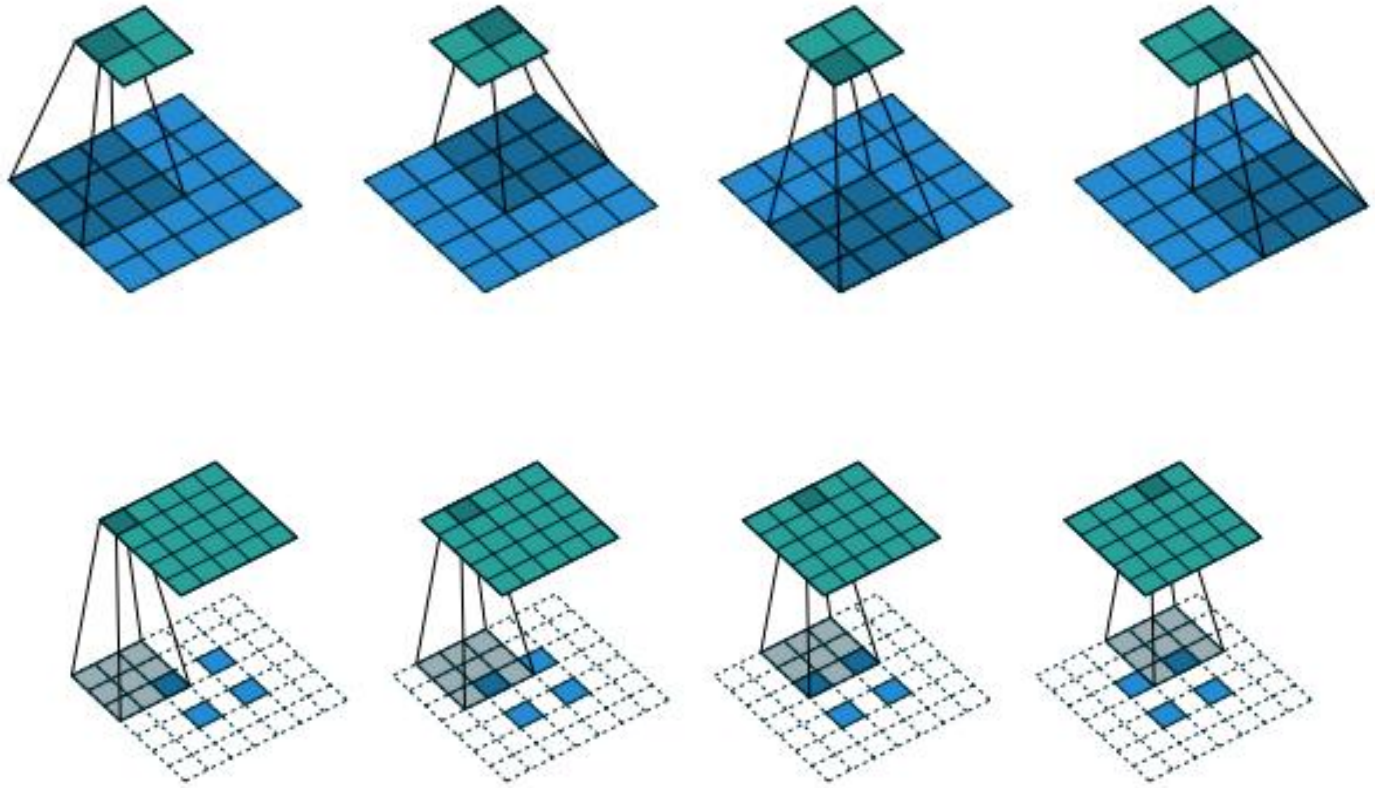


转置卷积

全卷积 vs 转置卷积



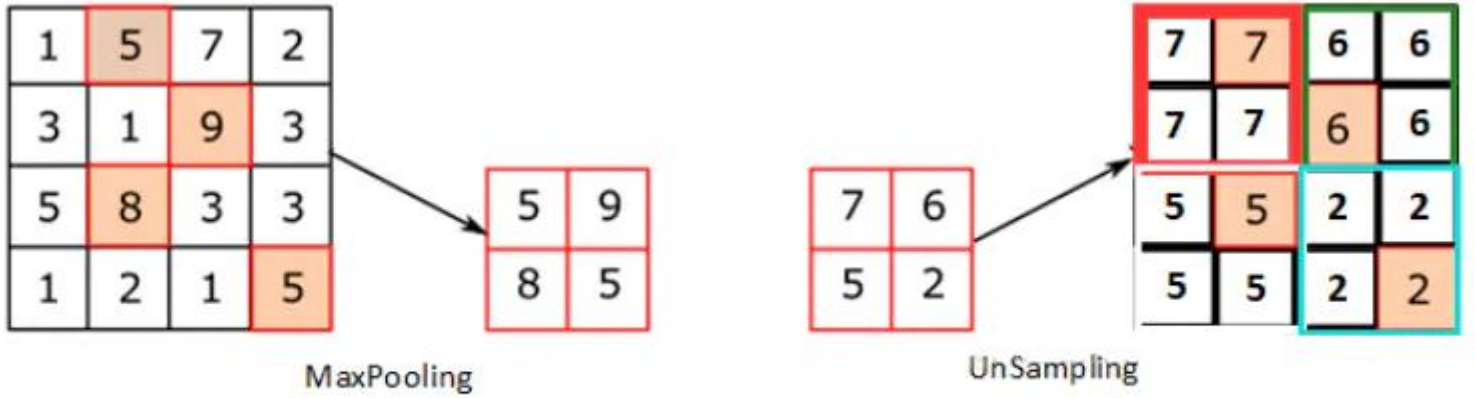
i=5，k=3，s=2，p=0



i=2，k=3，s=1，p=2

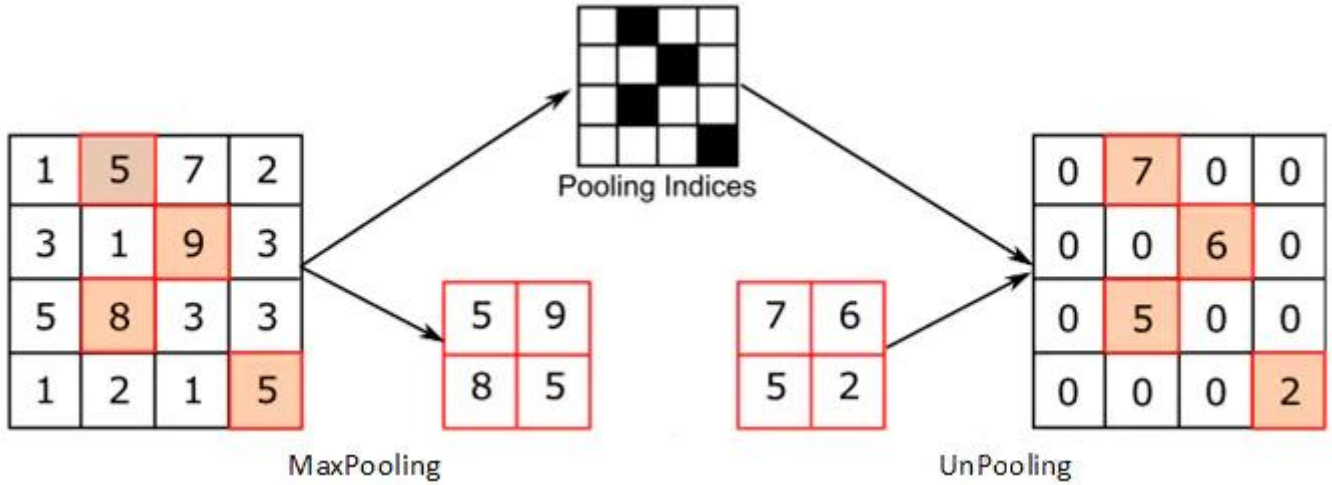
上采样(unsampling)

unsampling就是将输入feature map中的某个值映射填充到输出上采样的feature map的某片对应区域中，而且是全部填充的一样的值。



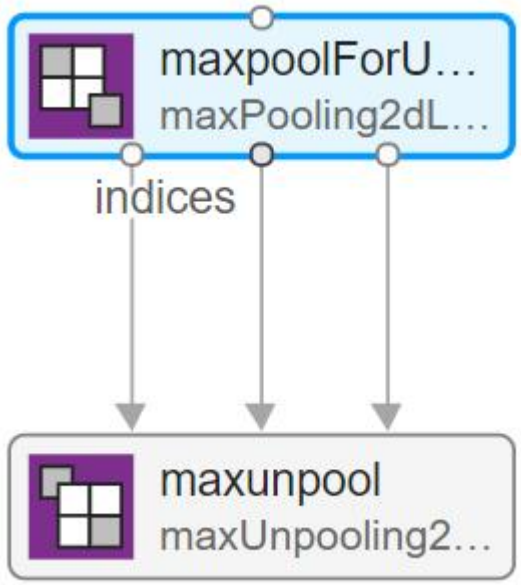
上池化(unpooling)

unpooling记录了原来pooling是取样的位置，在unpooling的时候将输入feature map中的值填充到原来记录的位置上，而其他位置则以0来进行填充。



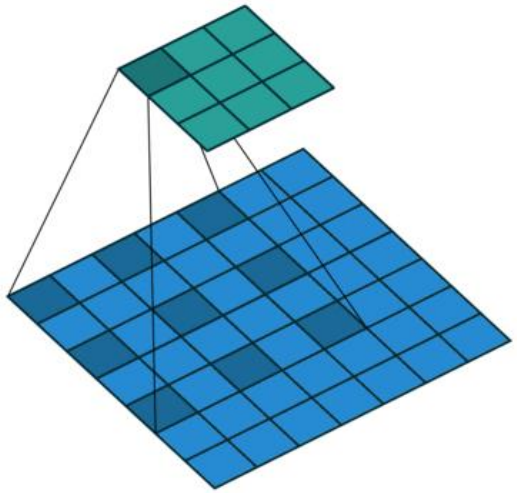
上池化(unpooling)

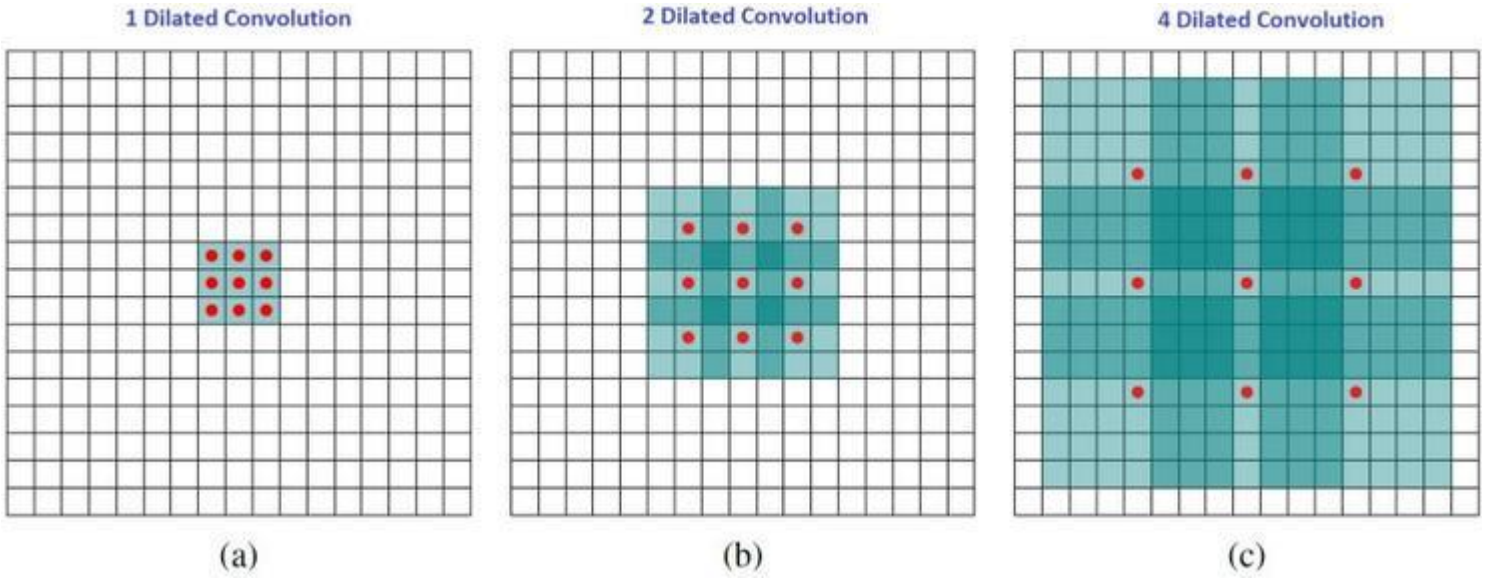
Matlab 提供了maxpoolForUnpool模块，其记录了indices和size信息，在unpooling的时候将这两路信息和输出数据一起连接到maxunpool模块。



膨胀卷积

语义分割网络广泛使用膨胀卷积（也称为扩张卷积或空洞卷积），因为它们可以增加层的感受野（层可以看到的输入区域），而不增加参数或计算量。





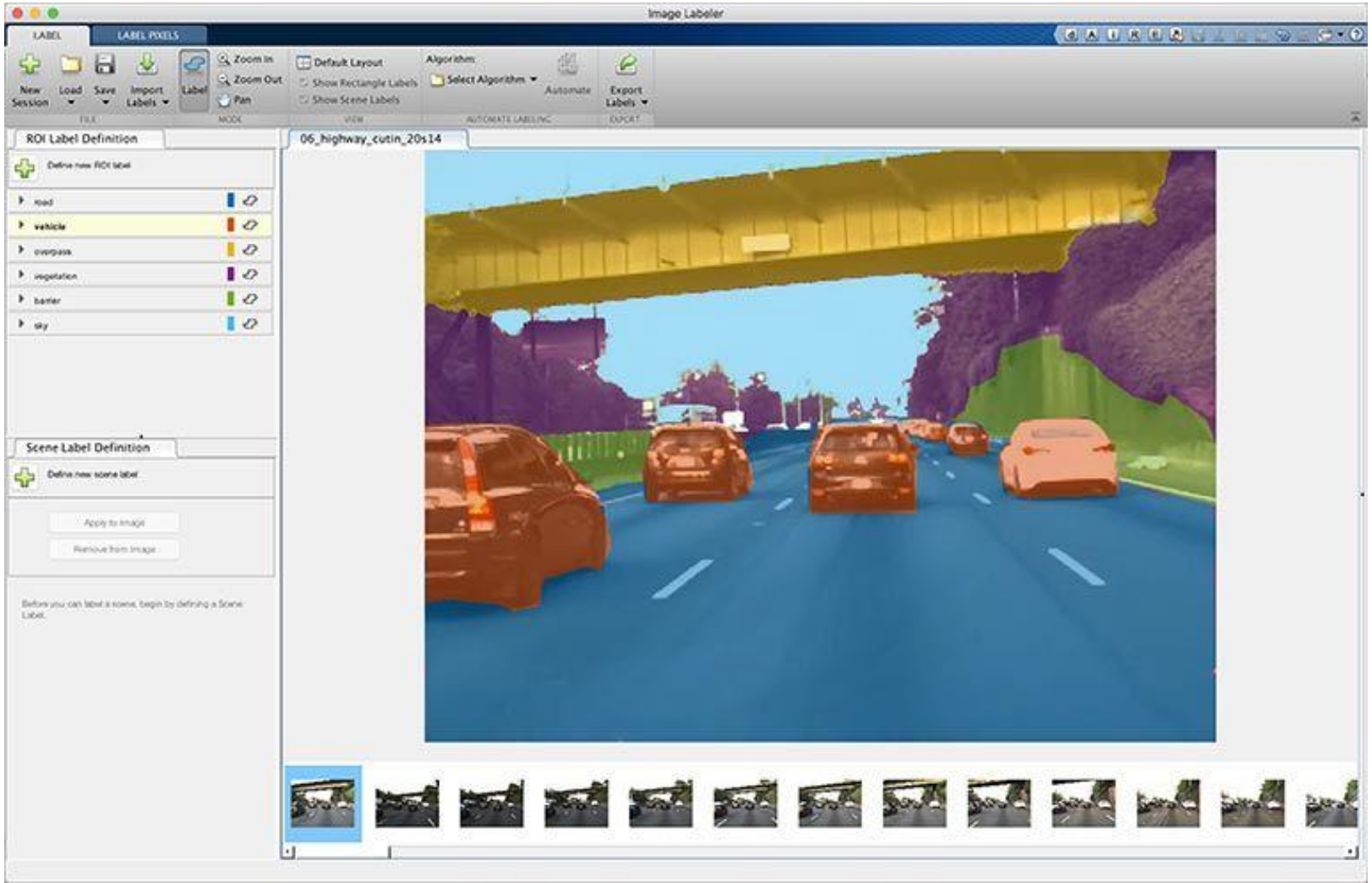
MATLAB 语义分割

在 MATLAB 中，执行语义分割的工作流程包括以下五个步骤：

1. 给数据加标签或获取带标签的数据。
2. 为原始图像和带标签的图像创建数据存储。
3. 分割数据存储。
4. 导入一个 CNN 并将其修改为 SegNet。
5. 训练和评估网络。

获取带标签的数据

深度学习模型建立于大量数据之上，语义分割也不例外。其中一个选择是在互联网上查找带标签的数据。如果要处理自己的数据集，则可以使用 MATLAB 中的 Image Labeler 应用程序创建标签。

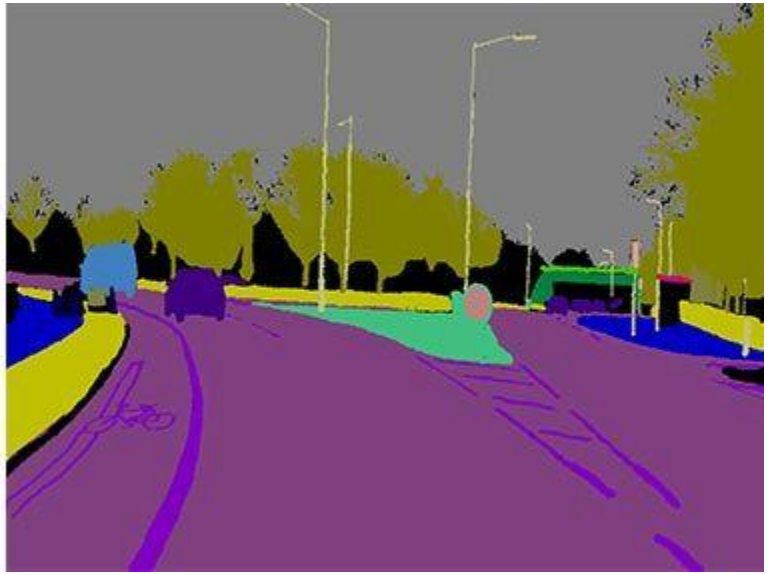


创建数据存储

在处理大量数据时，通常不可能将所有信息加载到内存中。要管理大型数据集，可以使用数据存储 DataStore。数据存储包含想要访问的文件的位置，只有在需要操作这些文件时，才将它们读入内存。

要创建语义分割网络，需要两个数据存储：

ImageDatastore，它包含原始图像
PixelLabelDatastore，它包含带标签的图像



分割数据存储

在创建语义分割网络之前，必须将数据存储分割为两部分：

训练集，用来训练语义分割网络
测试集，用来评估网络的准确性

创建语义分割网络

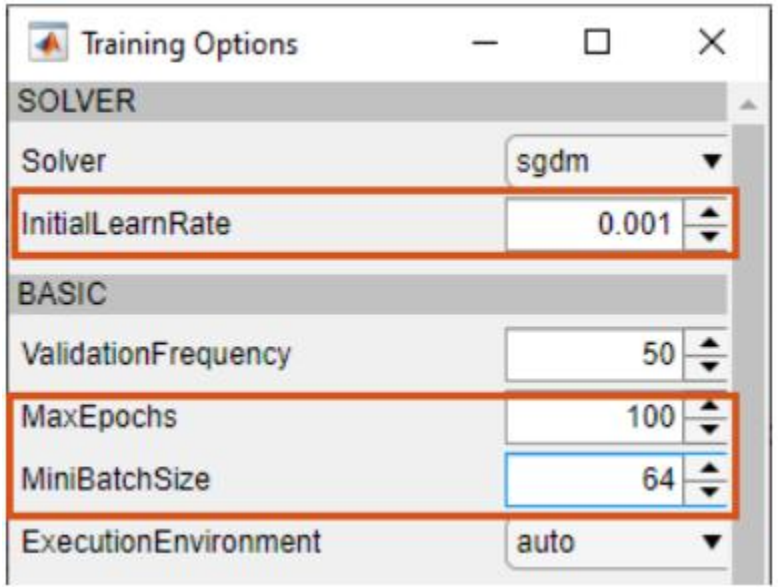
加载预训练的网络（如 VGG16），并创建像素级标注所需的编码器-解码器架构。

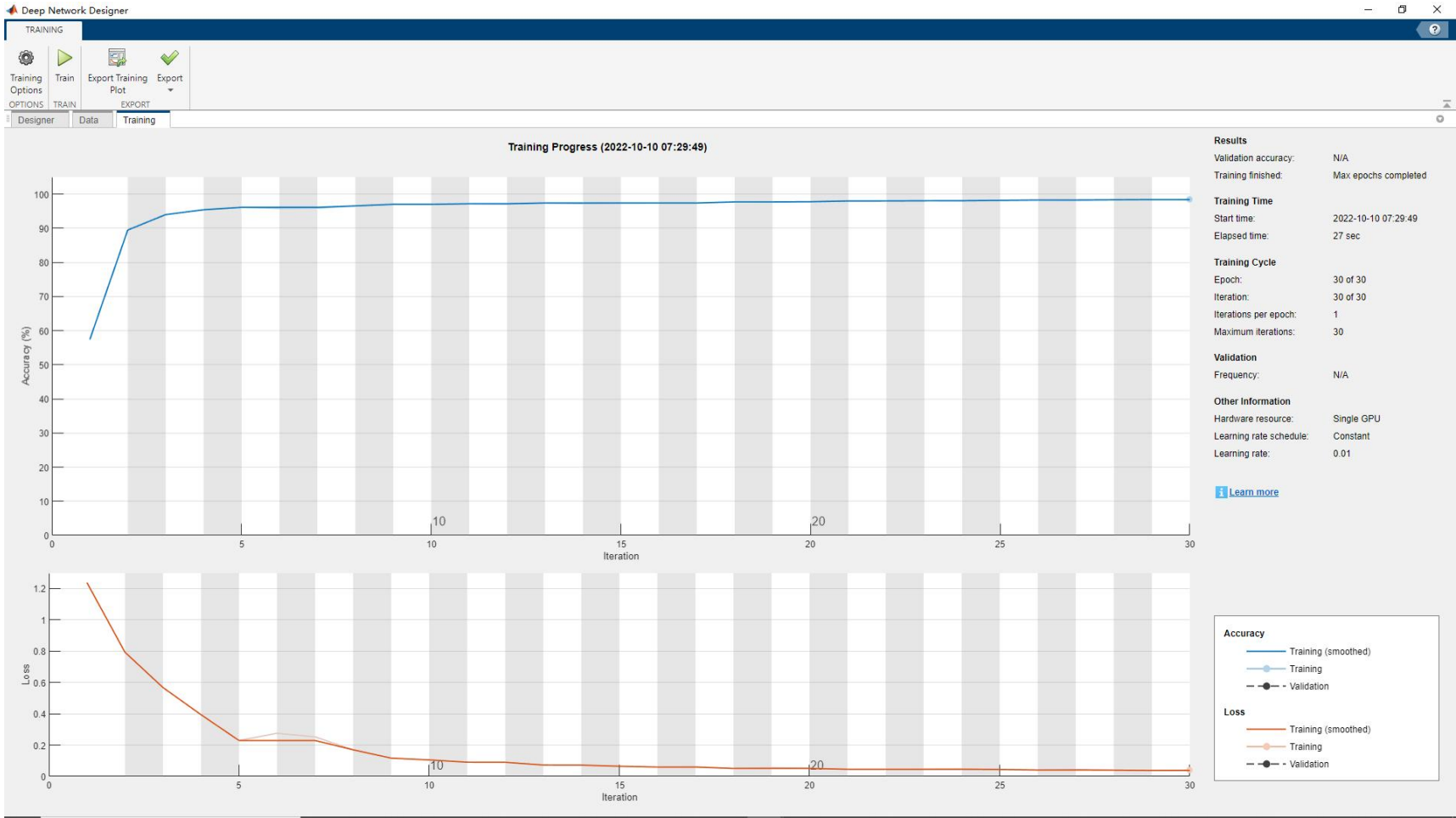
```
% pass the image size, number of predicted classes,
% and the pretrained CNN
% to create the complete SegNet architecture.

segnetLayers(imageSize,numClasses,model);
```

训练和评估网络

最后一步，设置网络的超参数并训练网络。





The steps for training a semantic segmentation network are as follows:

1. Analyze Training Data for Semantic Segmentation

2. Import Pixel Labeled Dataset For Semantic Segmentation

3. Create a Semantic Segmentation Network

4. Train A Semantic Segmentation Network

5. Evaluate and Inspect the Results of Semantic Segmentation

1. Analyze Training Data for Semantic Segmentation

To train a semantic segmentation network you need a collection of images and its corresponding collection of pixel labeled images. A pixel labeled image is an image where every pixel value represents the categorical label of that pixel.

The following code loads a small set of images and their corresponding pixel labeled images:

```
dataDir = fullfile(toolboxdir('vision'),'visiondata');
imDir = fullfile(dataDir,'building');
pxDir = fullfile(dataDir,'buildingPixelLabels');
```

1. Analyze Training Data for Semantic Segmentation

Load the image data using an imageDatastore. An image datastore can efficiently represent a large collection of images because images are only read into memory when needed.

```
imds = imageDatastore(imDir);
```

Read and display the first image.

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

1. Analyze Training Data for Semantic Segmentation

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 = ["sky" "grass" "building" "sidewalk"];
```

Define the label ID for each class name.

```
pixelLabelID = [1 2 3 4];
```

Create a pixelLabelDatastore.

```
pxds = pixelLabelDatastore(pxDir,classNames,pixelLabelID);
```

1. Analyze Training Data for Semantic Segmentation

Read the first pixel label image.

```
C = readimage(pxds,1);
```

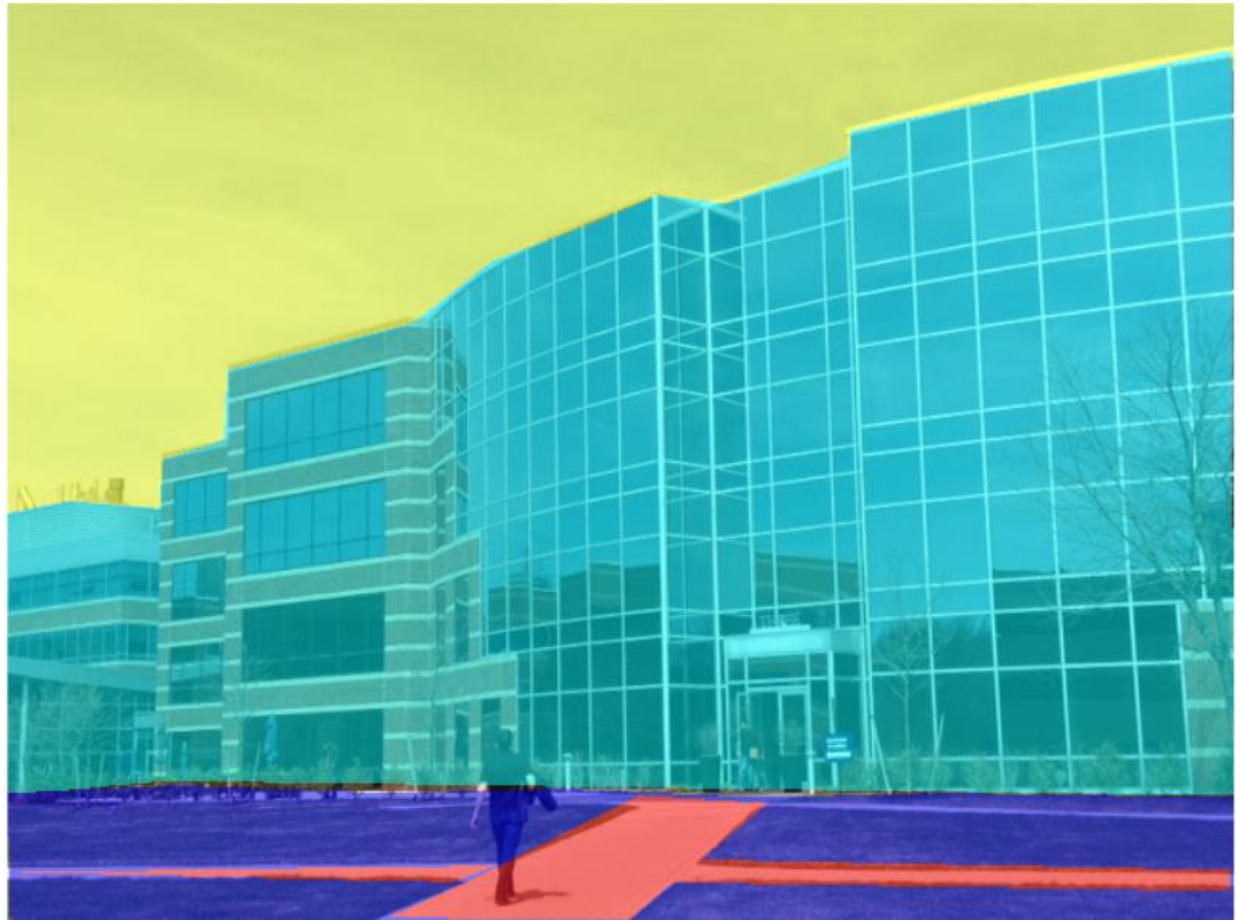
The output C is a categorical matrix where C(i,j) is the categorical label of pixel I(i,j).

```
size(C)
ans = 1×2
   480   640
C(50,50)
ans =
     sky
C(350,350)
ans =
     building
```

1. Analyze Training Data for Semantic Segmentation

Overlay the pixel labels on the image to see how different parts of the image are labeled.

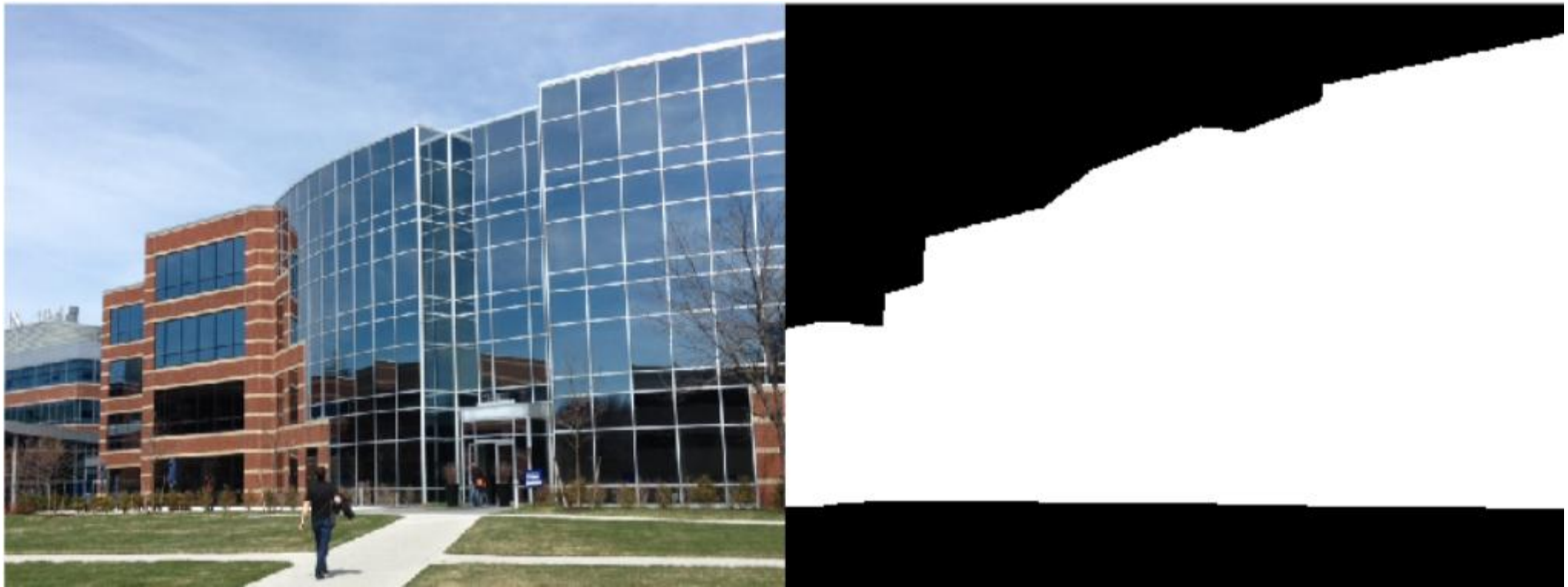
```
B = labeloverlay(I,C);
figure
imshow(B)
```



1. Analyze Training Data for Semantic Segmentation

The categorical output format simplifies tasks that require doing things by class names. For instance, you can create a binary mask of just the building:

```
buildingMask = C == 'building';
figure
imshowpair(I, buildingMask,'montage')
```



2. Import Pixel Labeled Dataset For Semantic Segmentation

A pixel labeled dataset is a collection of images and a corresponding set of ground truth pixel labels used for training semantic segmentation networks. There are many public datasets that provide annotated images with per-pixel labels. To illustrate the steps for importing these types of datasets, the example uses the **CamVid** dataset from the University of Cambridge.

The CamVid dataset is a collection of images containing street level views obtained while driving. The dataset provides pixel-level labels for 32 semantic classes including car, pedestrian, and road. The steps shown to import CamVid can be used to import other pixel labeled datasets.

2. Import Pixel Labeled Dataset For Semantic Segmentation

CamVid Pixel Labels

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"];
```

2. Import Pixel Labeled Dataset For Semantic Segmentation

CamVid Pixel Labels

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"
```

Note that other datasets have different formats of encoding data. For example, the **PASCAL VOC** dataset uses numeric label IDs between 0 and 21 to encode their class labels.

2. Import Pixel Labeled Dataset For Semantic Segmentation

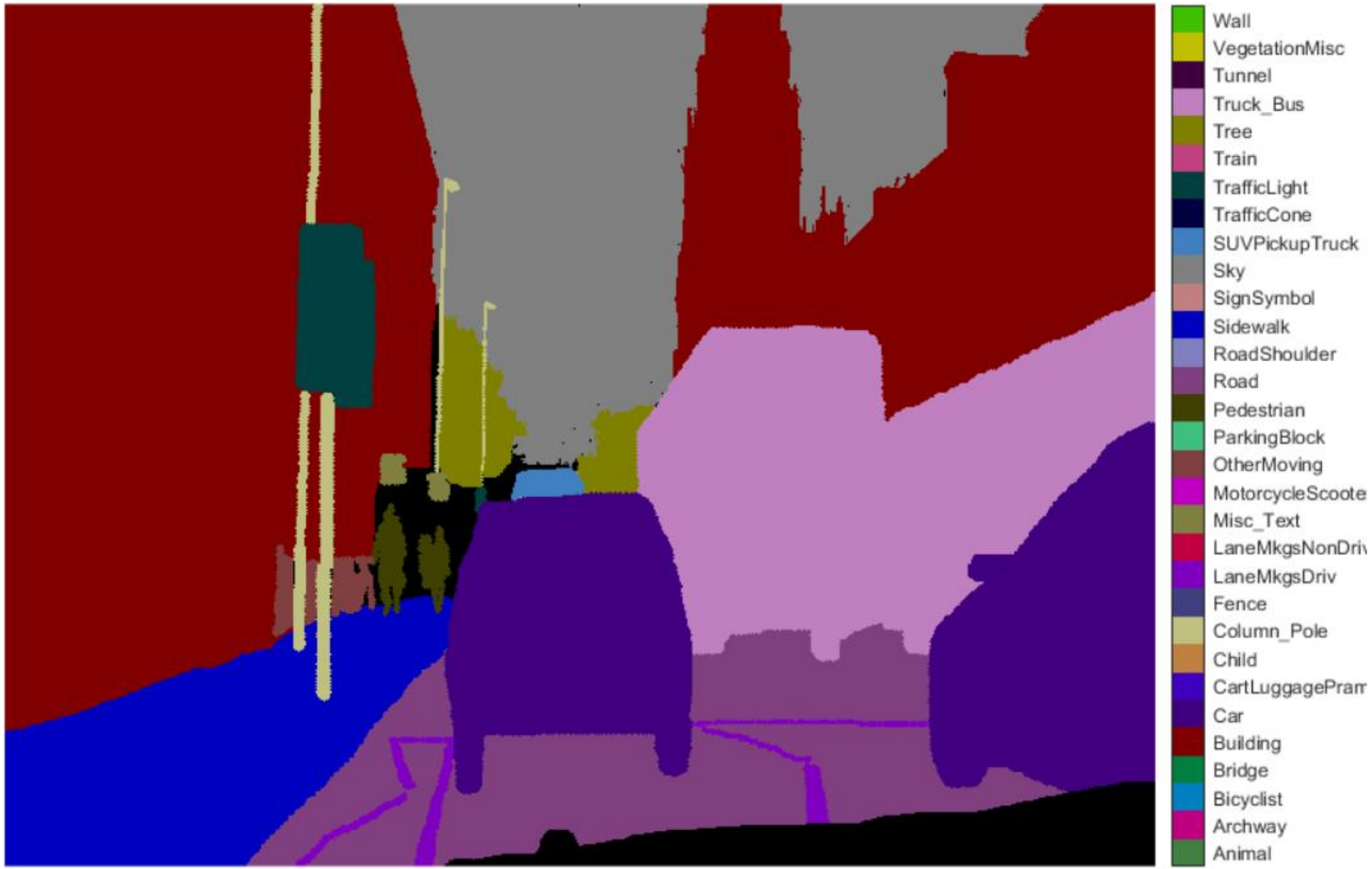
Visualize the pixel labels for one of the CamVid images.

```
labels = imread(fullfile(labelDir,'0001TP_006690_L.png'));
figure
imshow(labels)
```

% Add colorbar to show class to color mapping.

```
N = numel(classNames);
ticks = 1/(N*2):1/N:1;
colorbar('TickLabels',cellstr(classNames),'Ticks',ticks,'TickLength',0,'TickLabel
Interpreter','none');
colormap(labelIDs./255)
```

2. Import Pixel Labeled Dataset For Semantic Segmentation



2. Import Pixel Labeled Dataset For Semantic Segmentation

Load CamVid Data

A pixel labeled dataset can be loaded using an imageDatastore and a pixelLabelDatastore.

Create an imageDatastore to load the CamVid images.

```
imds = imageDatastore(fullfile(imageDir,'701_StillsRaw_full'));
```

Create a pixelLabelDatastore to load the CamVid pixel labels.

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

Read the 10th image and corresponding pixel label image.

```
I = readimage(imds,10);
C = readimage(pxds,10);
```

2. Import Pixel Labeled Dataset For Semantic Segmentation

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,'TickLabel
Interpreter','none');
colormap(labelIDs./255)
```

2. Import Pixel Labeled Dataset For Semantic Segmentation

2. Import Pixel Labeled Dataset For Semantic Segmentation

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')
```

2. Import Pixel Labeled Dataset For Semantic Segmentation

2. Import Pixel Labeled Dataset For Semantic Segmentation

Combine Classes

When working with public datasets, you may need to combine some of the classes to better suit your application. For example, you may want to train a semantic segmentation network that segments a scene into 4 classes: road, sky, vehicle, pedestrian, and background. To do this with the CamVid dataset, group the label IDs defined above to fit the new classes. First, define the new class names.

```
newClassNames = ["road","sky","vehicle","pedestrian","background"];
```

2. Import Pixel Labeled Dataset For Semantic Segmentation

Next, group label IDs using a cell array of M-by-3 matrices.

```
groupedLabelIDs = {
    % road
    [
    128 064 128; ... % "Road"
    128 000 192; ... % "LaneMkgsDriv"
    192 000 064; ... % "LaneMkgsNonDriv"
    000 000 192; ... % "Sidewalk"
    064 192 128; ... % "ParkingBlock"
    128 128 192; ... % "RoadShoulder"
    ]

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

    % "vehicle"
    [
    064 000 128; ... % "Car"
    064 128 192; ... % "SUVPickupTruck"
    192 128 192; ... % "Truck_Bus"
    192 064 128; ... % "Train"
    000 128 192; ... % "Bicyclist"
    192 000 192; ... % "MotorcycleScooter"
    128 064 064; ... % "OtherMoving"
    ]
    % "pedestrian"
    [
    064 064 000; ... % "Pedestrian"
    192 128 064; ... % "Child"
    064 000 192; ... % "CartLuggagePram"
    064 128 064; ... % "Animal"
    ]

    % "background"
    [
    128 128 000; ... % "Tree"
    192 192 000; ... % "VegetationMisc"
    192 128 128; ... % "SignSymbol"
    128 128 064; ... % "Misc_Text"
    000 064 064; ... % "TrafficLight"
    064 064 128; ... % "Fence"
    192 192 128; ... % "Column_Pole"
    000 000 064; ... % "TrafficCone"
    000 128 064; ... % "Bridge"
    128 000 000; ... % "Building"
    064 192 000; ... % "Wall"
    064 000 064; ... % "Tunnel"
    192 000 128; ... % "Archway"
    ]
    };
```

2. Import Pixel Labeled Dataset For Semantic Segmentation

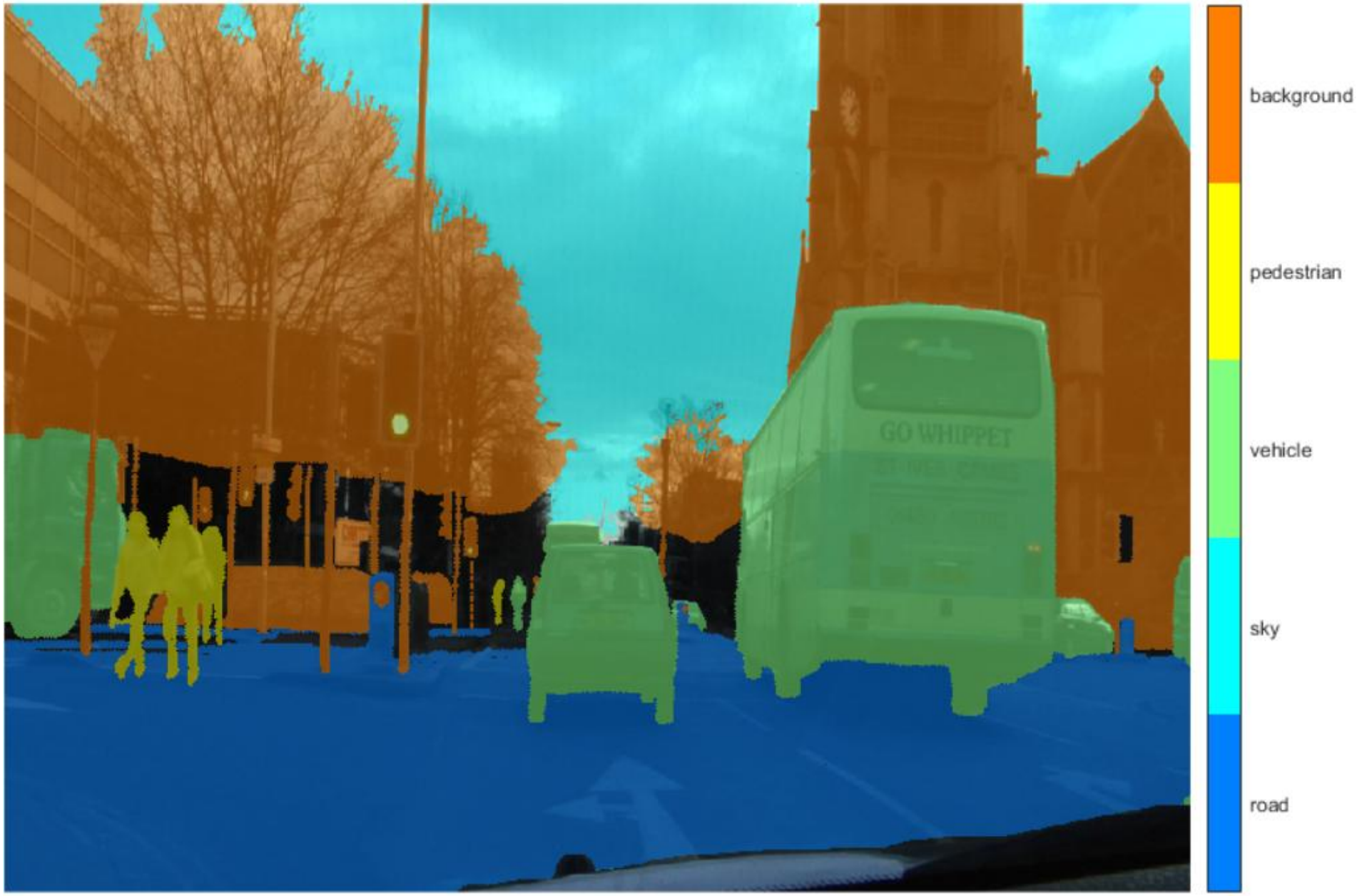
Create a pixelLabelDatastore using the new class and label IDs.

```
pxds = pixelLabelDatastore(labelDir,newClassNames,groupedLabelIDs);
```

Read the 10th pixel label image and display it on top of the image.

```
C = readimage(pxds,10);
cmap = jet(numel(newClassNames));
B = labeloverlay(I,C,'Colormap',cmap);
figure
imshow(B)
% add colorbar
N = numel(newClassNames);
ticks = 1/(N*2):1/N:1;
colorbar('TickLabels',cellstr(newClassNames),'Ticks',ticks,'TickLength',0,'TickLa
belInterpreter','none');
colormap(cmap)
```

2. Import Pixel Labeled Dataset For Semantic Segmentation



3. Create a Semantic Segmentation Network

Create a simple semantic segmentation network and learn about common layers found in many semantic segmentation networks. A common pattern in semantic segmentation networks requires the downsampling of an image between convolutional and ReLU layers, and then upsample the output to match the input size.

This operation is analogous to the standard scale-space analysis using image pyramids. During this process however, a network performs the operations using non-linear filters optimized for a specific set of classes that you want to segment.

3. Create a Semantic Segmentation Network

Create An Image Input Layer

A semantic segmentation network starts with an imageInputLayer, which defines the smallest image size the network can process. Most semantic segmentation networks are fully convolutional, which means they can process images that are larger than the specified input size. Here, an image size of [32 32 3] is used for the network to process 64x64 RGB images.

```
inputSize = [32 32 3];
imgLayer = imageInputLayer(inputSize)
```

3. Create a Semantic Segmentation Network

Create Downsampling Network

Start with the convolution and ReLU layers. The convolution layer padding is selected such that the output size of the convolution layer is the same as the input size. This makes it easier to construct a network because the input and output sizes between most layers remain the same as you progress through the network.

```
filterSize = 3;
numFilters = 32;
conv = convolution2dLayer(filterSize,numFilters,'Padding',1);
relu = reluLayer();
```

The downsampling is performed using a max pooling layer. Create a max pooling layer to downsample the input by a factor of 2 by setting the 'Stride' parameter to 2.

```
poolSize = 2;
maxPoolDownsample2x = maxPooling2dLayer(poolSize,'Stride',2);
```

3. Create a Semantic Segmentation Network

Create Downsampling Network

Stack the convolution, ReLU, and max pooling layers to create a network that downsamples its input by a factor of 4.

```
downsamplingLayers = [
    conv
    relu
    maxPoolDownsample2x
    conv
    relu
    maxPoolDownsample2x
    ]
```

3. Create a Semantic Segmentation Network

Create Upsampling Network

The upsampling is done using the tranposed convolution layer (also commonly referred to as "deconv" or "deconvolution" layer). When a transposed convolution is used for upsampling, it performs the upsampling and the filtering at the same time.

Create a transposed convolution layer to upsample by 2.

```
filterSize = 4;
transposedConvUpsample2x = transposedConv2dLayer(4,numFilters,'Stride',2,'Cropping',1);
```

The 'Cropping' parameter is set to 1 to make the output size equal twice the input size.

Stack the transposed convolution and relu layers. An input to this set of layers is upsampled by 4.

```
upsamplingLayers = [
    transposedConvUpsample2x
    relu
    transposedConvUpsample2x
    relu
    ]
```

3. Create a Semantic Segmentation Network

Create A Pixel Classification Layer

The final set of layers are responsible for making pixel classifications. These final layers process an input that has the same spatial dimensions (height and width) as the input image. However, the number of channels (third dimension) is larger and is equal to number of filters in the last transposed convolution layer. This third dimension needs to be squeezed down to the number of classes we wish to segment. This can be done using a 1-by-1 convolution layer whose number of filters equal the number of classes.

```
numClasses = 3;
conv1x1 = convolution2dLayer(1,numClasses);
```

Following this 1-by-1 convolution layer are the softmax and pixel classification layers. These two layers combine to predict the categorical label for each image pixel.

```
finalLayers = [
    conv1x1
    softmaxLayer()
    pixelClassificationLayer()
    ]
```

3. Create a Semantic Segmentation Network

Stack All Layers

Stack all the layers to complete the semantic segmentation network.

```
net = [
    imgLayer
    downsamplingLayers
    upsamplingLayers
    finalLayers
    ]
```

```
net =
  14×1 Layer array with layers:

     1   ''   Image Input                 32×32×3 images with 'zerocenter' normalization
     2   ''   Convolution                 32 3×3 convolutions with stride [1  1] and padding [1  1  1  1]
     3   ''   ReLU                        ReLU
     4   ''   Max Pooling                 2×2 max pooling with stride [2  2] and padding [0  0  0  0]
     5   ''   Convolution                 32 3×3 convolutions with stride [1  1] and padding [1  1  1  1]
     6   ''   ReLU                        ReLU
     7   ''   Max Pooling                 2×2 max pooling with stride [2  2] and padding [0  0  0  0]
     8   ''   Transposed Convolution      32 4×4 transposed convolutions with stride [2  2] and cropping [1  1  1  1]
     9   ''   ReLU                        ReLU
    10   ''   Transposed Convolution      32 4×4 transposed convolutions with stride [2  2] and cropping [1  1  1  1]
    11   ''   ReLU                        ReLU
    12   ''   Convolution                 3 1×1 convolutions with stride [1  1] and padding [0  0  0  0]
    13   ''   Softmax                     softmax
    14   ''   Pixel Classification Layer  Cross-entropy loss
```

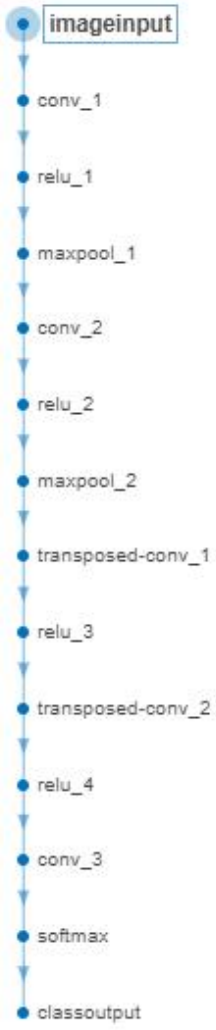
Deep Learning Network Analyzer

Analysis for trainNetwork usage

Name: net

Analysis date: 2022-10-10 11:11:37

43k	14	0	0
total learnables	layers	warnings	errors



ANALYSIS RESULT

	Name	Type	Activations	Learnable Prope...
1	imageinput 32×32×3 images with 'zerocenter' norm...	Image Input	32(S) × 32(S) × 3(C) × 1(B)	-
2	conv_1 32 3×3 convolutions with stride [1 1] and...	Convolution	32(S) × 32(S) × 32(C) × 1(B)	Weig… 3 × 3 × 3 … Bias 1 × 1 × 32
3	relu_1 ReLU	ReLU	32(S) × 32(S) × 32(C) × 1(B)	-
4	maxpool_1 2×2 max pooling with stride [2 2] and pa...	Max Pooling	16(S) × 16(S) × 32(C) × 1(B)	-
5	conv_2 32 3×3 convolutions with stride [1 1] and...	Convolution	16(S) × 16(S) × 32(C) × 1(B)	Weig… 3 × 3 × 32… Bias 1 × 1 × 32
6	relu_2 ReLU	ReLU	16(S) × 16(S) × 32(C) × 1(B)	-
7	maxpool_2 2×2 max pooling with stride [2 2] and pa...	Max Pooling	8(S) × 8(S) × 32(C) × 1(B)	-
8	transposed-conv_1 32 4×4 transposed convolutions with stri...	Transposed Convol...	16(S) × 16(S) × 32(C) × 1(B)	Weig… 4 × 4 × 32… Bias 1 × 1 × 32
9	relu_3 ReLU	ReLU	16(S) × 16(S) × 32(C) × 1(B)	-
10	transposed-conv_2 32 4×4 transposed convolutions with stri...	Transposed Convol...	32(S) × 32(S) × 32(C) × 1(B)	Weig… 4 × 4 × 32… Bias 1 × 1 × 32
11	relu_4 ReLU	ReLU	32(S) × 32(S) × 32(C) × 1(B)	-
12	conv_3 3 1×1 convolutions with stride [1 1] and ...	Convolution	32(S) × 32(S) × 3(C) × 1(B)	Weig… 1 × 1 × 32… Bias 1 × 1 × 3
13	softmax softmax	Softmax	32(S) × 32(S) × 3(C) × 1(B)	-
14	classoutput Cross-entropy loss	Pixel Classification ...	32(S) × 32(S) × 3(C) × 1(B)	-

4. Train A Semantic Segmentation Network

Load the training data.

```
dataSetDir = fullfile(toolboxdir('vision'),'visiondata','triangleImages');
imageDir = fullfile(dataSetDir,'trainingImages');
labelDir = fullfile(dataSetDir,'trainingLabels');
Create an image datastore for the images.
imds = imageDatastore(imageDir);
```

Create a pixelLabelDatastore for the ground truth pixel labels.

```
classNames = ["triangle","background"];
labelIDs   = [255 0];
pxds = pixelLabelDatastore(labelDir,classNames,labelIDs);
```

4. Train A Semantic Segmentation Network

Visualize training images and ground truth pixel labels.

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

I = imresize(I,5);
L = imresize(uint8(C{1}),5);
imshowpair(I,L,'montage')
```

4. Train A Semantic Segmentation Network

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

```
numFilters = 64;
filterSize = 3;
numClasses = 2;
layers = [
    imageInputLayer([32 32 1])
    convolution2dLayer(filterSize,numFilters,'Padding',1)
    reluLayer()
    maxPooling2dLayer(2,'Stride',2)
    convolution2dLayer(filterSize,numFilters,'Padding',1)
    reluLayer()
    transposedConv2dLayer(4,numFilters,'Stride',2,'Cropping',1);
    convolution2dLayer(1,numClasses);
    softmaxLayer()
    pixelClassificationLayer()
    ];
```

4. Train A Semantic Segmentation Network

Setup training options.

```
opts = trainingOptions('sgdm', ...
    'InitialLearnRate',1e-3, ...
    'MaxEpochs',100, ...
    'MiniBatchSize',64);
```

Combine the image and pixel label datastore for training.

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

4. Train A Semantic Segmentation Network

Train the network.

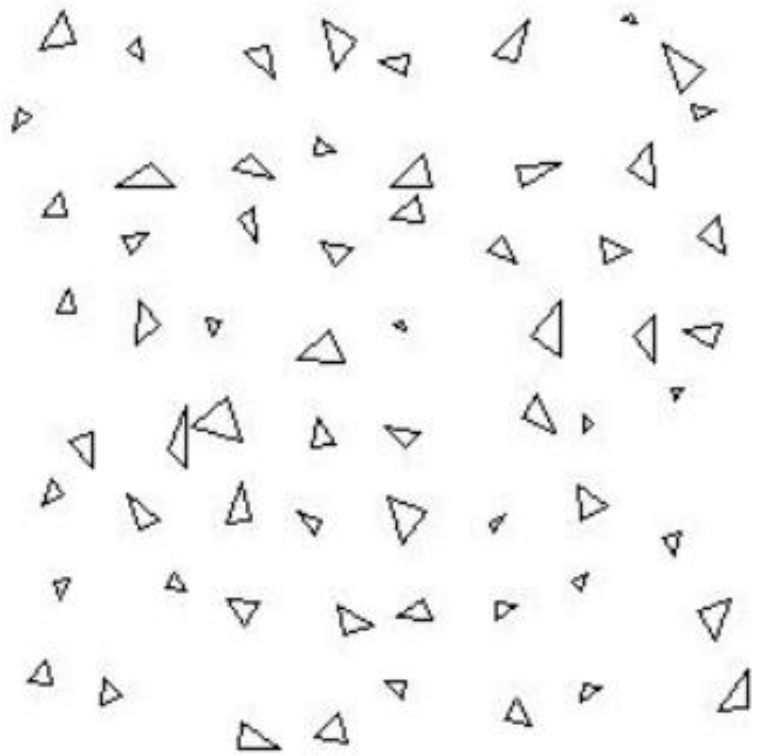
```
net = trainNetwork(trainingData,layers,opts);
```

```
Training on single GPU.
Initializing input data normalization.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:01 |       43.14% |       1.5346 |          0.0010 |
|      17 |          50 |       00:00:14 |       97.25% |       0.0926 |          0.0010 |
|      34 |         100 |       00:00:26 |       98.12% |       0.0553 |          0.0010 |
|      50 |         150 |       00:00:39 |       98.46% |       0.0457 |          0.0010 |
|      67 |         200 |       00:00:52 |       98.46% |       0.0429 |          0.0010 |
|      84 |         250 |       00:01:05 |       98.73% |       0.0354 |          0.0010 |
|     100 |         300 |       00:01:18 |       98.77% |       0.0340 |          0.0010 |
|=========================================================================================|
Training finished: Max epochs completed.
```

4. Train A Semantic Segmentation Network

Read and display a test image.

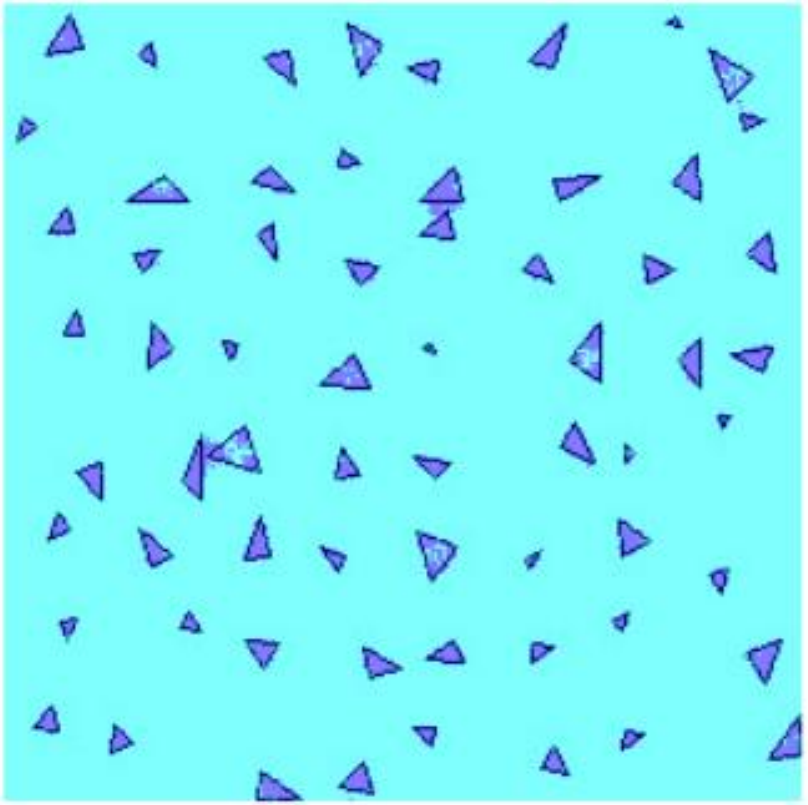
```
testImage = imread('triangleTest.jpg');
imshow(testImage)
```



4. Train A Semantic Segmentation Network

Segment the test image and display the results.

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



5. Evaluate and Inspect the Results of Semantic Segmentation

Import a test data set, run a pretrained semantic segmentation network, and evaluate and inspect semantic segmentation quality metrics for the predicted results.

Import a Data Set

The triangleImages data set has 100 test images with ground truth labels. Define the location of the data set.

```
dataSetDir = fullfile(toolboxdir('vision'),'visiondata','triangleImages');
```

Define the location of the test images.

```
testImagesDir = fullfile(dataSetDir,'testImages');
Create an imageDatastore object holding the test images.
imds = imageDatastore(testImagesDir);
```

5. Evaluate and Inspect the Results of Semantic Segmentation

Define the location of the ground truth labels.

```
testLabelsDir = fullfile(dataSetDir,'testLabels');
```

Define the class names and their associated label IDs. The label IDs are the pixel values used in the image files to represent each class.

```
classNames = ["triangle" "background"];
labelIDs = [255 0];
```

Create a pixelLabelDatastore object holding the ground truth pixel labels for the test images.

```
pxdsTruth = pixelLabelDatastore(testLabelsDir,classNames,labelIDs);
```

5. Evaluate and Inspect the Results of Semantic Segmentation

Run a Semantic Segmentation Classifier

Load a semantic segmentation network that has been trained on the training images of triangleImages.

```
net = load('triangleSegmentationNetwork.mat');
net = net.net;
```

Run the network on the test images. Predicted labels are written to disk in a temporary directory and returned as a pixelLabelDatastore object.

```
pxdsResults = semanticseg(imds,net,"WriteLocation",tempdir);
Running semantic segmentation network
-------------------------------------
* Processed 100 images.
```

net.net.Layers

	1
1	*1x1 ImageInputLayer*
2	*1x1 Convolution2DLayer*
3	*1x1 ReLULayer*
4	*1x1 MaxPooling2DLayer*
5	*1x1 Convolution2DLayer*
6	*1x1 ReLULayer*
7	*1x1 TransposedConvolution2DLayer*
8	*1x1 Convolution2DLayer*
9	*1x1 SoftmaxLayer*
10	*1x1 PixelClassificationLayer*

5. Evaluate and Inspect the Results of Semantic Segmentation

Evaluate the Quality of the Prediction

The predicted labels are compared to the ground truth labels. While the semantic segmentation metrics are being computed, progress is printed to the Command Window.

```
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
    ______________    ____________    _______    ___________    ___________

       0.90624           0.95085       0.61588      0.87529        0.40652
```

5. Evaluate and Inspect the Results of Semantic Segmentation

Inspect Class Metrics

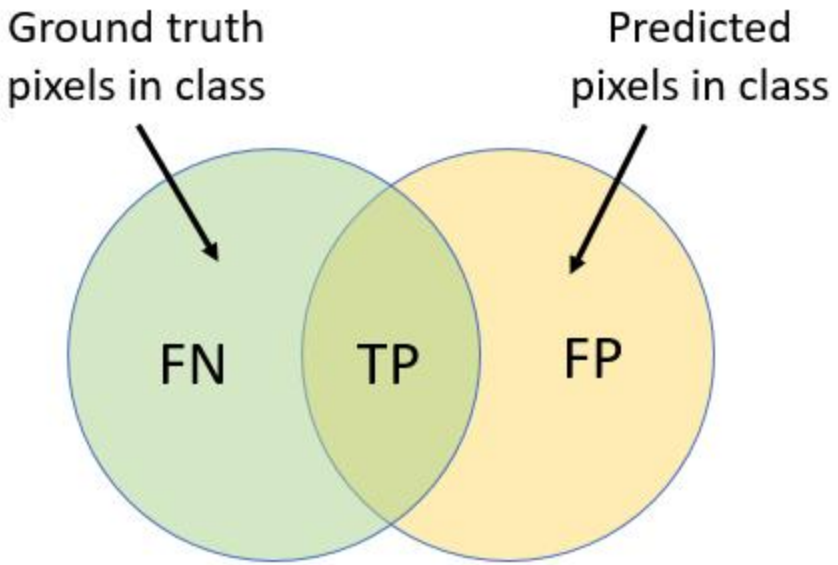
Display the classification accuracy, the intersection over union (IoU), and the boundary F-1 score for each class in the data set.

`metrics.ClassMetrics`

```
ans = 2×3 table
```

		Accuracy	IoU	MeanBFScore
1	triangle	1	0.3301	0.0287
2	background	0.9017	0.9017	0.7844

IoU score = TP / (TP + FP + FN)



5. Evaluate and Inspect the Results of Semantic Segmentation

Display the Confusion Matrix

`metrics.ConfusionMatrix`

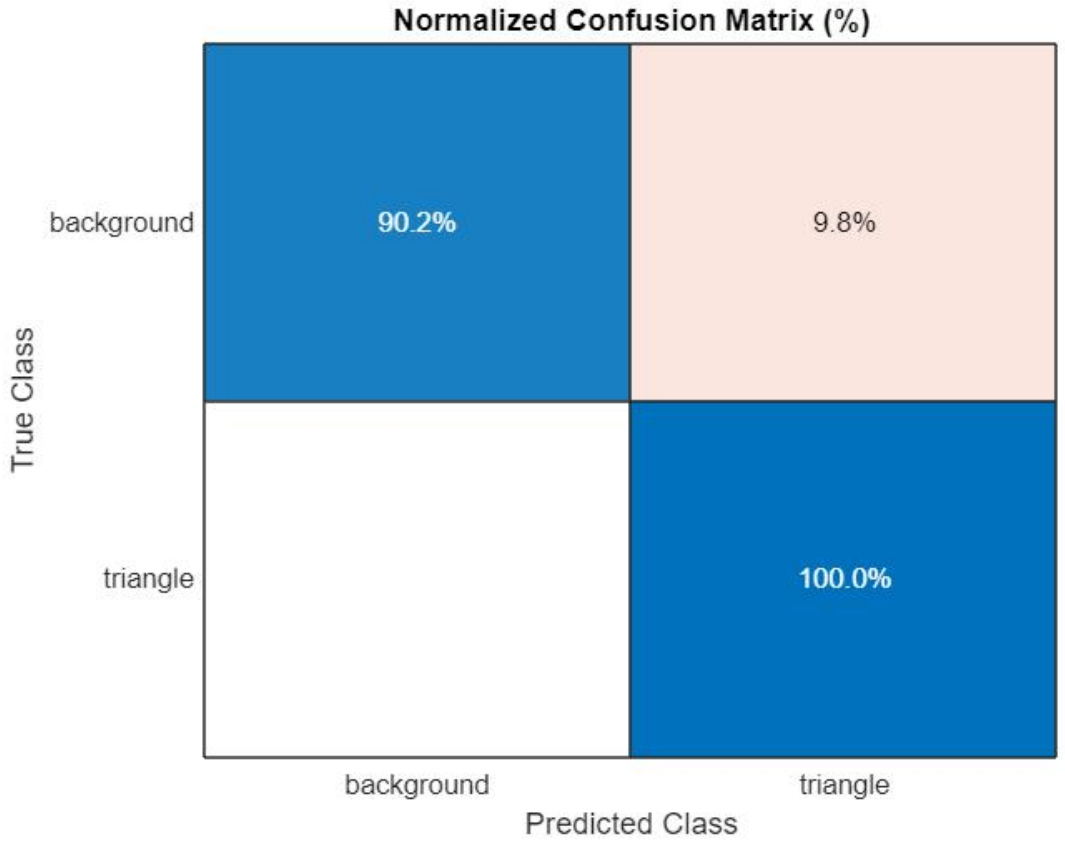
```
ans = 2×2 table
```

		triangle	background
1	triangle	4730	0
2	background	9601	88069

5. Evaluate and Inspect the Results of Semantic Segmentation

Visualize the normalized confusion matrix as a confusion chart in a figure window.

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

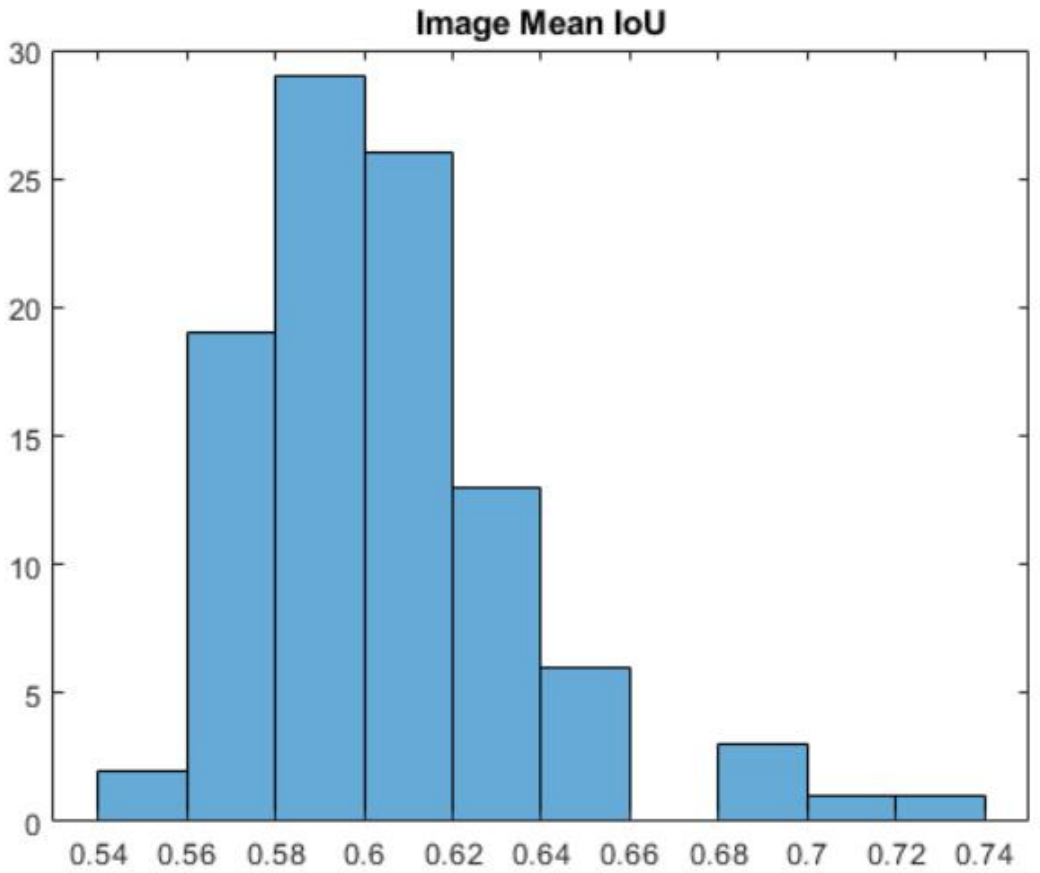


5. Evaluate and Inspect the Results of Semantic Segmentation

Inspect an Image Metric

Visualize the histogram of the per-image intersection over union (IoU).

```
imageIoU = metrics.ImageMetrics.MeanIoU;
figure
histogram(imageIoU)
title('Image Mean IoU')
```



5. Evaluate and Inspect the Results of Semantic Segmentation

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 comparison.

```
worstTestImage = readimage(imds,worstImageIndex);
worstTrueLabels = readimage(pxdsTruth,worstImageIndex);
worstPredictedLabels = readimage(pxdsResults,worstImageIndex);
```

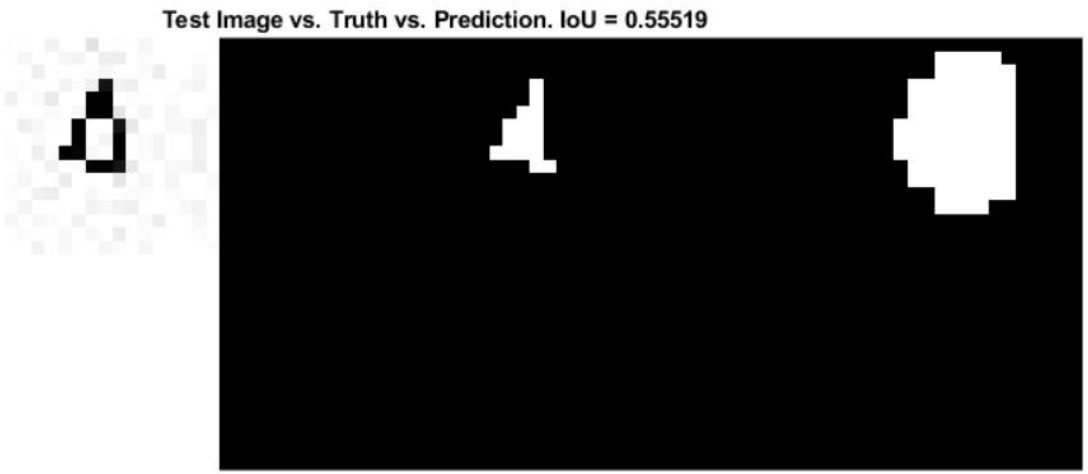
5. Evaluate and Inspect the Results of Semantic Segmentation

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



5. Evaluate and Inspect the Results of Semantic Segmentation

Similarly, find the test image with the highest IoU.

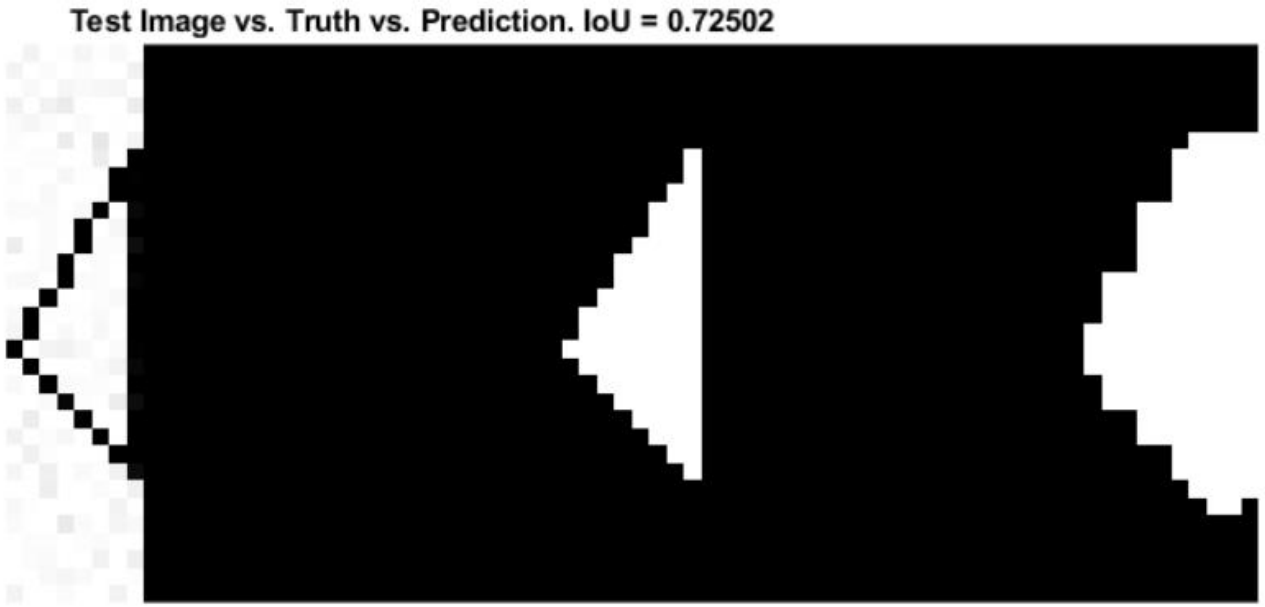
```
[maxIoU, bestImageIndex] = max(imageIoU);
maxIoU = maxIoU(1);
bestImageIndex = bestImageIndex(1);
```

Repeat the previous steps to read, convert, and display the test image with the best IoU with its ground truth and predicted labels.

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

5. Evaluate and Inspect the Results of Semantic Segmentation



5. Evaluate and Inspect the Results of Semantic Segmentation

Specify Metrics to Evaluate

Optionally, list the metric(s) you would like to evaluate using the 'Metrics' parameter.

Define the metrics to compute.

```
evaluationMetrics = ["accuracy" "iou"];
```

Compute these metrics for the triangleImages test data set.

```
metrics =
evaluateSemanticSegmentation(pxdsResults,pxdsTruth,"Metrics",evaluationMetrics);
```

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

MeanAccuracy	MeanIoU
0.95085	0.61588

5. Evaluate and Inspect the Results of Semantic Segmentation

Display the chosen metrics for each class.

```
metrics.ClassMetrics
```

```
ans = 2×2 table
```

		Accuracy	IoU
1	triangle	1	0.3301
2	background	0.9017	0.9017

使用扩张卷积进行语义分割

语义分割网络对图像中的每个像素进行分类，从而生成按类分割的图像。语义分割的应用包括自动驾驶中的道路分割以及医疗诊断中的癌细胞分割。语义分割网络广泛使用扩张卷积（也称为空洞卷积），因为它们可以增加层的感受野（层可以看到的输入区域），而不增加参数或计算量。

使用扩张卷积进行语义分割

加载训练数据

该示例使用 32×32 三角形图像的简单数据集进行说明。该数据集包括附带的像素标签真实值数据。使用 imageDatastore 和 pixelLabelDatastore 加载训练数据。

```
dataFolder = fullfile(toolboxdir('vision'),'visiondata','triangleImages');
imageFolderTrain = fullfile(dataFolder,'trainingImages');
labelFolderTrain = fullfile(dataFolder,'trainingLabels');
```

为图像创建一个 imageDatastore。

```
imdsTrain = imageDatastore(imageFolderTrain);
```

为真实值像素标签创建一个 pixelLabelDatastore。

```
classNames = ["triangle" "background"];
labels = [255 0];
pxdsTrain = pixelLabelDatastore(labelFolderTrain,classNames,labels)
```

使用扩张卷积进行语义分割

创建语义分割网络

此示例使用一个基于扩张卷积的简单语义分割网络。

创建一个用于训练数据的数据源，并获取每个标签的像素计数。

```
ds = combine(imdsTrain,pxdsTrain);
tbl = countEachLabel(pxdsTrain)
```

```
tbl=2×3 table
        Name         PixelCount    ImagePixelCount
    _____________    __________    _______________

    {'triangle'  }       10326           2.048e+05
    {'background'}    1.9447e+05         2.048e+05
```

使用扩张卷积进行语义分割

大多数像素标签用于背景。这种类不平衡使学习过程偏向主导类。要解决此问题，请使用类权重来平衡各类。您可以使用几种方法来计算类权重。一种常见的方法是逆频率加权，其中类权重是类频率的倒数。此方法会增加指定给表示不足的类的权重。使用逆频率加权计算类权重。

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

使用扩张卷积进行语义分割

指定对应于卷积层、批量归一化层和 ReLU 层的三个块。对于每个卷积层，指定 32 个具有递增扩张系数的 3×3 滤波器，并通过将 'Padding' 选项设置为 'same' 来填充输入以使输入的大小与输出相同。要对像素进行分类，请包括一个具有 K 个 1×1 卷积的卷积层，其后是softmax 层和具有逆类权重的 pixelClassificationLayer。

```
inputSize = [32 32 1];
filterSize = 3;
numFilters = 32;
numClasses = numel(classNames);
layers = [
    imageInputLayer(inputSize)
    convolution2dLayer(filterSize,numFilters,'DilationFactor',1,'Padding','same')
    batchNormalizationLayer
    reluLayer
    convolution2dLayer(filterSize,numFilters,'DilationFactor',2,'Padding','same')
    batchNormalizationLayer
    reluLayer
    convolution2dLayer(filterSize,numFilters,'DilationFactor',4,'Padding','same')
    batchNormalizationLayer
    reluLayer
    convolution2dLayer(1,numClasses)
    softmaxLayer
    pixelClassificationLayer('Classes',classNames,'ClassWeights',classWeights)];
```

使用扩张卷积进行语义分割

ANALYSIS RESULT

	Name	Type	Activations	Learnable Properties
1	imageinput 32×32×1 images with 'zerocenter' normalization	Image Input	32(S) × 32(S) × 1(C) × 1(B)	-
2	conv_1 32 3×3 convolutions with stride [1 1] and padding 'same'	Convolution	32(S) × 32(S) × 32(C) × 1(B)	Weights 3 × 3 × 1 × 32 Bias 1 × 1 × 32
3	batchnorm_1 Batch normalization	Batch Normalization	32(S) × 32(S) × 32(C) × 1(B)	Offset 1 × 1 × 32 Scale 1 × 1 × 32
4	relu_1 ReLU	ReLU	32(S) × 32(S) × 32(C) × 1(B)	-
5	conv_2 32 3×3 convolutions with stride [1 1], dilation factor [2 2], and padding 'same'	Convolution	32(S) × 32(S) × 32(C) × 1(B)	Weights 3 × 3 × 32 × 32 Bias 1 × 1 × 32
6	batchnorm_2 Batch normalization	Batch Normalization	32(S) × 32(S) × 32(C) × 1(B)	Offset 1 × 1 × 32 Scale 1 × 1 × 32
7	relu_2 ReLU	ReLU	32(S) × 32(S) × 32(C) × 1(B)	-
8	conv_3 32 3×3 convolutions with stride [1 1], dilation factor [4 4], and padding 'same'	Convolution	32(S) × 32(S) × 32(C) × 1(B)	Weights 3 × 3 × 32 × 32 Bias 1 × 1 × 32
9	batchnorm_3 Batch normalization	Batch Normalization	32(S) × 32(S) × 32(C) × 1(B)	Offset 1 × 1 × 32 Scale 1 × 1 × 32
10	relu_3 ReLU	ReLU	32(S) × 32(S) × 32(C) × 1(B)	-
11	conv_4 2 1×1 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	32(S) × 32(S) × 2(C) × 1(B)	Weights 1 × 1 × 32 × 2 Bias 1 × 1 × 2
12	softmax softmax	Softmax	32(S) × 32(S) × 2(C) × 1(B)	-
13	classoutput Class weighted cross-entropy loss with classes 'triangle' and 'background'	Pixel Classification ...	32(S) × 32(S) × 2(C) × 1(B)	-

使用扩张卷积进行语义分割

训练网络

指定训练选项。

```
options = trainingOptions('sgdm', ...
    'MaxEpochs', 100, ...
    'MiniBatchSize', 64, ...
    'InitialLearnRate', 1e-3);
```

使用 trainNetwork 训练网络。

```
net = trainNetwork(ds,layers,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 |       00:00:02 |       91.62% |       1.6825 |          0.0010 |
|      17 |          50 |       00:00:30 |       88.56% |       0.2393 |          0.0010 |
|      34 |         100 |       00:00:51 |       92.08% |       0.1672 |          0.0010 |
|      50 |         150 |       00:01:18 |       93.17% |       0.1472 |          0.0010 |
|      67 |         200 |       00:01:44 |       94.15% |       0.1313 |          0.0010 |
|      84 |         250 |       00:02:08 |       94.47% |       0.1167 |          0.0010 |
|     100 |         300 |       00:02:37 |       95.04% |       0.1100 |          0.0010 |
|=========================================================================================|
Training finished: Max epochs completed.
```

使用扩张卷积进行语义分割

测试网络

加载测试数据。为图像创建一个 imageDatastore。为真实值像素标签创建一个 pixelLabelDatastore。

```
imageFolderTest = fullfile(dataFolder,'testImages');
imdsTest = imageDatastore(imageFolderTest);
labelFolderTest = fullfile(dataFolder,'testLabels');
pxdsTest = pixelLabelDatastore(labelFolderTest,classNames,labels);
```

使用测试数据和经过训练的网络进行预测。

```
pxdsPred = semanticseg(imdsTest,net,'MiniBatchSize',32,'WriteLocation',tempdir);
```

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

使用扩张卷积进行语义分割

使用 evaluateSemanticSegmentation 评估预测准确度。

```
metrics = evaluateSemanticSegmentation(pxdsPred,pxdsTest);
```

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

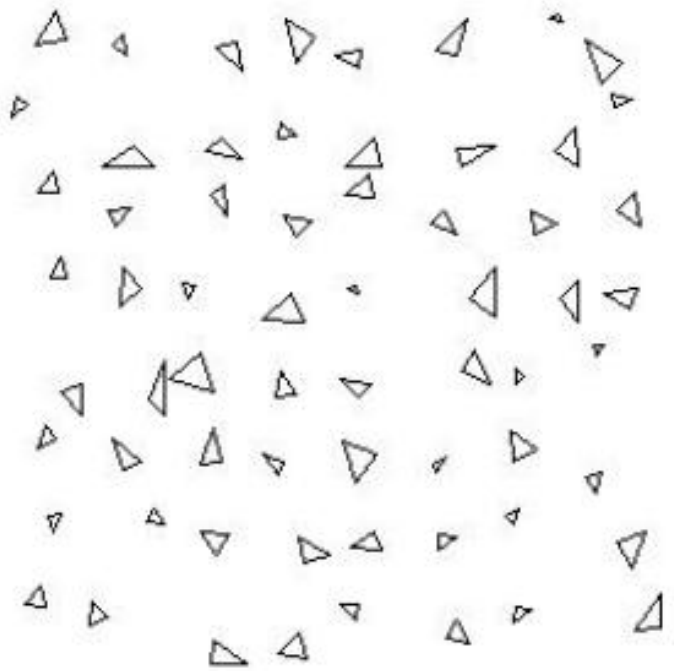
GlobalAccuracy	MeanAccuracy	MeanIoU	WeightedIoU	MeanBFScore
0.95237	0.97352	0.72081	0.92889	0.46416

使用扩张卷积进行语义分割

分割新图像

读取并显示测试图像 triangleTest.jpg。

```
imgTest = imread('triangleTest.jpg');
figure
imshow(imgTest)
```



使用 semanticseg 分割测试图像，并使用 labeloverlay 显示结果。

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