Visualize Features of a Convolutional Neural Network

This example shows how to visualize the features learned by convolutional neural networks.

Convolutional neural networks use *features* to classify images. The network learns these features itself during the training process. What the network learns during training is sometimes unclear. However, you can use the `deepDreamImage` function to visualize the features learned.

The *convolutional* layers output a 3D activation volume, where slices along the third dimension correspond to a single filter applied to the layer input. The channels output by *fully connected* layers at the end of the network correspond to high-level combinations of the features learned by earlier layers.

You can visualize what the learned features look like by using `deepDreamImage` to generate images that strongly activate a particular channel of the network layers.

The example requires Deep Learning Toolbox™ and Deep Learning Toolbox Model *for GoogLeNet Network* support package.

Load Pretrained Network

Load a pretrained GoogLeNet network.

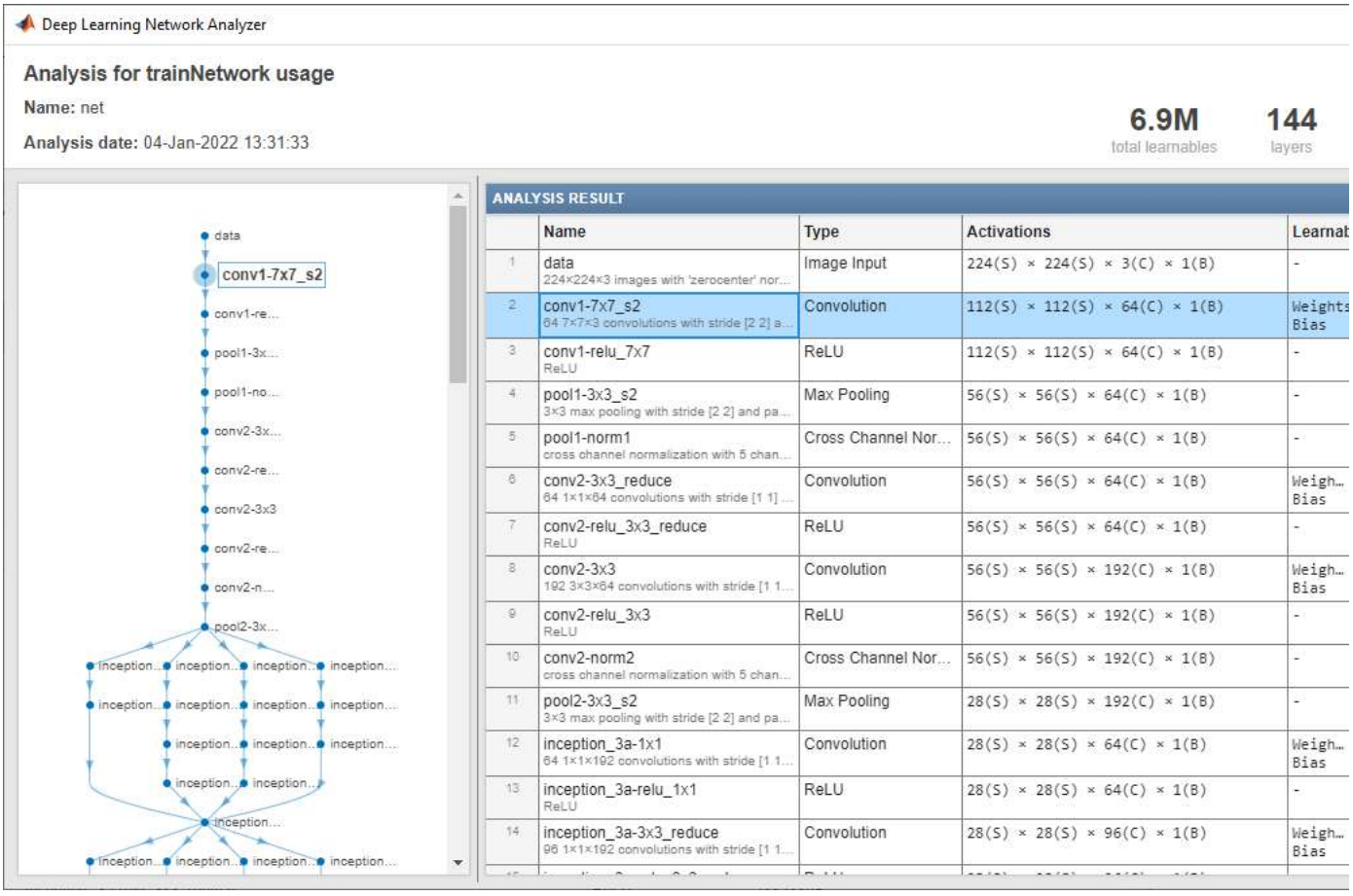
```
net = googlenet;
```

Visualize Early Convolutional Layers

There are multiple convolutional layers in the GoogLeNet network. The convolutional layers towards the beginning of the network have a small receptive field size and learn small, low-level features. The layers towards the end of the network have larger receptive field sizes and learn larger features.

Using `analyzeNetwork`, view the network architecture and locate the convolutional layers.

```
analyzeNetwork(net)
```



Features on Convolutional Layer 1

Set `layer` to be the first convolutional layer. This layer is the second layer in the network and is named `'conv1-7x7_s2'`.

```
layer = 2;
name = net.Layers(layer).Name
```

```
name = 'conv1-7x7_s2'
```

Visualize the first 36 features learned by this layer using `deepDreamImage` by setting `channels` to be the vector of indices `1:36`. Set `'PyramidLevels'` to 1 so that the images are not scaled. To display the images together, you can use `imtile`.

`deepDreamImage` uses a compatible GPU, by default, if available. Otherwise it uses the CPU. Using a GPU requires Parallel Computing Toolbox™ and a supported GPU device. For information on supported devices, see GPU Support by Release.

```
channels = 1:36;
I = deepDreamImage(net,name,channels, ...
    'PyramidLevels',1);
```

```
|=============================================|
|  Iteration  |  Activation  |  Pyramid Level  |
|             |   Strength   |                 |
|=============================================|
|           1 |         0.34 |               1 |
|           2 |         6.91 |               1 |
|           3 |        14.16 |               1 |
|           4 |        21.42 |               1 |
|           5 |        28.67 |               1 |
|           6 |        35.92 |               1 |
|           7 |        43.17 |               1 |
|           8 |        50.42 |               1 |
|           9 |        57.67 |               1 |
|          10 |        64.93 |               1 |
|=============================================|
```

```
figure
I = imtile(I,'ThumbnailSize',[64 64]);
imshow(I)
title(['Layer ',name,' Features'],'Interpreter','none')
```

These images mostly contain edges and colors, which indicates that the filters at layer `'conv1-7x7_s2'` are edge detectors and color filters.

Features on Convolutional Layer 2

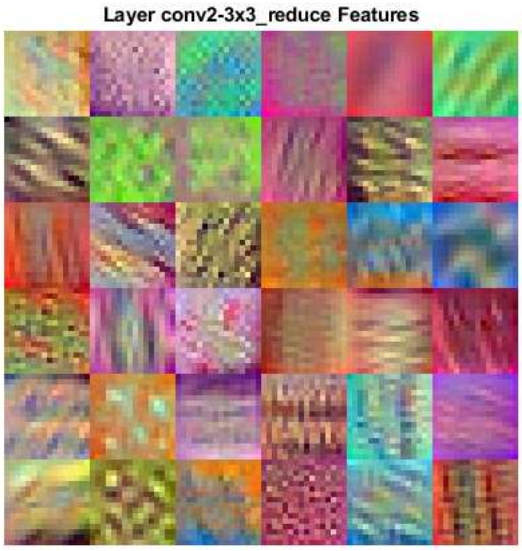
The second convolutional layer is named `'conv2-3x3_reduce'`, which corresponds to layer 6. Visualize the first 36 features learned by this layer by setting `channels` to be the vector of indices `1:36`.

To suppress detailed output on the optimization process, set `'Verbose'` to `'false'` in the call to `deepDreamImage`.

```
layer = 6;
name = net.Layers(layer).Name
```

```
name = 'conv2-3x3_reduce'
```

```
channels = 1:36;
I = deepDreamImage(net,name,channels, ...
    'Verbose',false, ...
    'PyramidLevels',1);
figure
I = imtile(I,'ThumbnailSize',[64 64]);
imshow(I)
name = net.Layers(layer).Name;
title(['Layer ',name,' Features'],'Interpreter','none')
```



Filters for this layer detect more complex patterns than the first convolutional layer.

Visualize Deeper Convolutional Layers

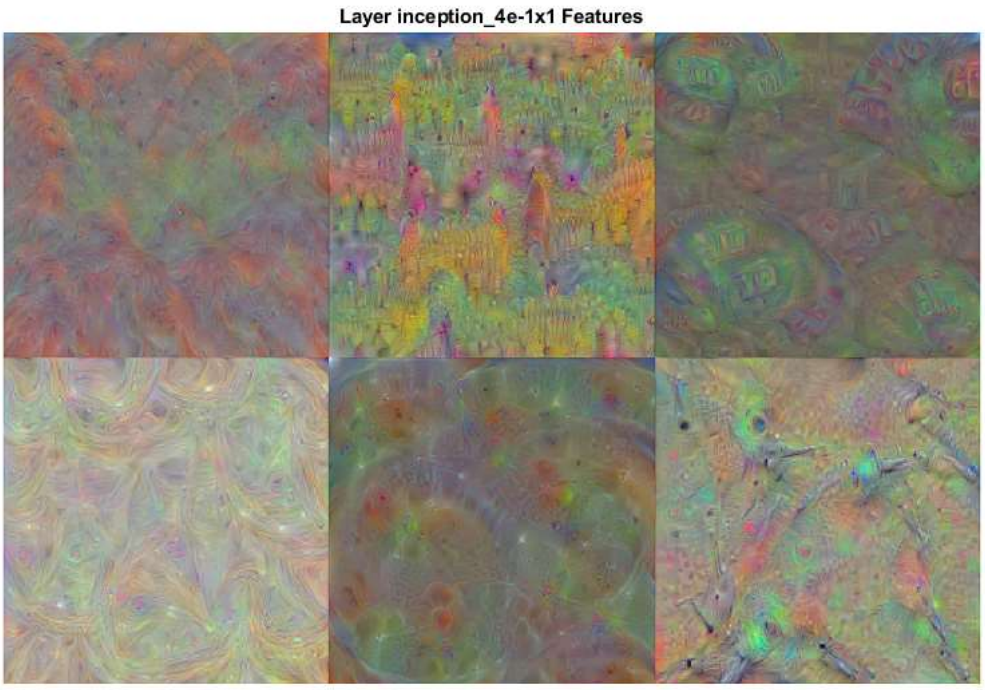
The deeper layers learn high-level combinations of features learned by the earlier layers.

Increasing the number of pyramid levels and iterations per pyramid level can produce more detailed images at the expense of additional computation. You can increase the number of iterations using the `'NumIterations'` option and increase the number of pyramid levels using the `'PyramidLevels'` option.

```
layer = 97;
name = net.Layers(layer).Name
```

```
name = 'inception_4e-1x1'
```

```
channels = 1:6;
I = deepDreamImage(net,name,channels, ...
    'Verbose',false, ...
    "NumIterations",20, ...
    'PyramidLevels',2);
figure
I = imtile(I,'ThumbnailSize',[250 250]);
imshow(I)
name = net.Layers(layer).Name;
title(['Layer ',name,' Features'],'Interpreter','none')
```



Notice that the layers which are deeper into the network yield more detailed filters which have learned complex patterns and textures.

Visualize Fully Connected Layer

To produce images that resemble each class the most closely, select the fully connected layer, and set `channels` to be the indices of the classes.

Select the fully connected layer (layer 142).

```
layer = 142;
name = net.Layers(layer).Name
```

```
name = 'loss3-classifier'
```

Select the classes you want to visualize by setting `channels` to be the indices of those class names.

```
channels = [123 985 690 470 580 620];
```

The classes are stored in the `Classes` property of the output layer (the last layer). You can view the names of the selected classes by selecting the entries in `channels`.

```
net.Layers(end).Classes(channels)
```

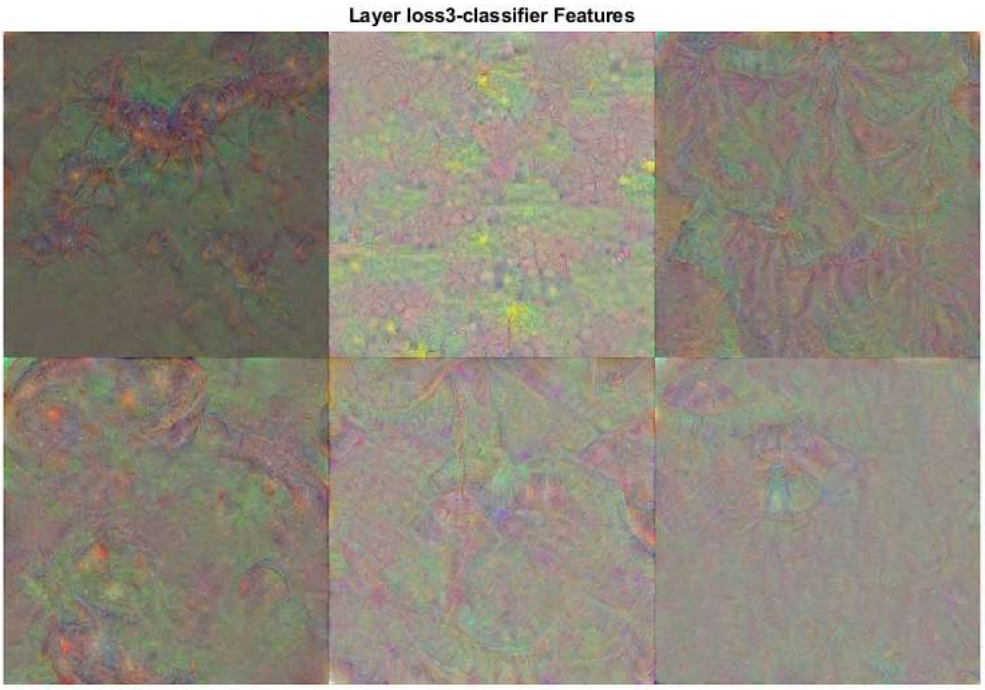
```
ans = 6×1 categorical
American lobster
rapeseed
overskirt
caldron
grand piano
lampshade
```

Generate detailed images that strongly activate these classes. Set `'NumIterations'` to 100 in the call to `deepDreamImage` to produce more detailed images. The images generated from the fully connected layer correspond to the image classes.

```
I = deepDreamImage(net,name,channels, ...
    'Verbose',true, ...
    'NumIterations',10, ...
    'PyramidLevels',2);
```

```
|=============================================|
|  Iteration  |  Activation  |  Pyramid Level  |
|             |   Strength   |                 |
|=============================================|
|           1 |         0.58 |               1 |
|           2 |         0.81 |               1 |
|           3 |         3.93 |               1 |
|           4 |         6.84 |               1 |
|           5 |        12.19 |               1 |
|           6 |        15.48 |               1 |
|           7 |        17.89 |               1 |
|           8 |        23.05 |               1 |
|           9 |        26.60 |               1 |
|          10 |        26.64 |               1 |
|           1 |         9.47 |               2 |
|           2 |        14.61 |               2 |
|           3 |        19.82 |               2 |
|           4 |        18.26 |               2 |
|           5 |        24.02 |               2 |
|           6 |        26.69 |               2 |
|           7 |        26.47 |               2 |
|           8 |        31.49 |               2 |
```

```
figure
I = imtile(I,'ThumbnailSize',[250 250]);
imshow(I)
name = net.Layers(layer).Name;
title(['Layer ',name,' Features'])
```



The images generated strongly activate the selected classes. The image generated for the ‘zebra’ class contain distinct zebra stripes, whilst the image generated for the ‘castle’ class contains turrets and windows.