```
% Define a function that creates an array of layers.
%The first block has 32 filters in the convolution layers.
% The number of filters doubles in each successive block.
unetBlock = @(block) [
    convolution2dLayer(3,2^(5+block),'Padding','Same')
    batchNormalizationLayer
    reluLayer
    convolution2dLayer(3,2^(5+block),'Padding','Same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,"Stride",2)];
%% Create a network that consists of four repeating blocks of layers.
% Add the prefix "encoder_" to all layer names in the network.
net = blockedNetwork(unetBlock,4,"NamePrefix","encoder_");
% Initialize network weights for input of size [224 224 3].
% net = initialize(net,dlarray(zeros(224,224,3),"SSC"));
% analyzeNetwork(net)
% save('bklayer','net')
layersTransfer=net.Layers(1:end);

net1=[
    % Input Layers
    imageInputLayer([227 227 3],"Name","data")
    convolution2dLayer([11 11],96,"Name","conv1","BiasLearnRateFactor",2,"Stride",[4 4])
    reluLayer("Name","relu1")
    crossChannelNormalizationLayer(5,"Name","norm1","K",1)
    maxPooling2dLayer([3 3],"Name","pool1","Stride",[2 2])

    % Conv for features extraction
    layersTransfer
    % FCN & Output Layers
    fullyConnectedLayer(4096,"Name","fc6","BiasLearnRateFactor",2)
    reluLayer("Name","relu6")
    dropoutLayer(0.5,"Name","drop6")
    fullyConnectedLayer(4096,"Name","fc7","BiasLearnRateFactor",2)
    reluLayer("Name","relu7")
    dropoutLayer(0.5,"Name","drop7")
    fullyConnectedLayer(1000,"Name","fc8")
    softmaxLayer("Name","prob")
    classificationLayer("Name","output")

 ];

lgraph=layerGraph(net1);

net2=dlnetwork(lgraph_1);


```

```
plot(lgraph)
analyzeNetwork(net1)
save('bklayer','net1')
```

2.5-1 SeriesNetwork

Load Pretrained AlexNet Convolutional Neural Network

加載預訓練的 AlexNet 卷積神經網絡並檢查層和類。

使用 alexnet 加載預訓練的 AlexNet 網絡。 輸出網絡是一個 SeriesNetwork 對象.

```
net = alexnet
```

使用 Layers 屬性，查看網絡架構。 該網絡由 25 層組成。 有 8 個具有可學習權重的層：5 個卷積層和 3 個全連接層.

```
net.Layers
```

```
ans = 
  25x1 Layer array with layers:

     1   'data'     Image Input                   227x227x3 images with 'zerocenter' normalization
     2   'conv1'    Convolution                   96 11x11x3 convolutions with stride [4  4] and padding [0  0  0  0]
     3   'relu1'    ReLU                          ReLU
     4   'norm1'    Cross Channel Normalization   cross channel normalization with 5 channels per element
     5   'pool1'    Max Pooling                   3x3 max pooling with stride [2  2] and padding [0  0  0  0]
     6   'conv2'    Grouped Convolution           2 groups of 128 5x5x48 convolutions with stride [1  1] and padding [2  2  2  2]
     7   'relu2'    ReLU                          ReLU
```

You can view the names of the classes learned by the network by viewing the `Classes` property of the classification output layer (the final layer). View the first 10 classes by selecting the first 10 elements.

```
net.Layers(end).Classes(1:10)
```

Define the convolutional neural network architecture.

```
layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer
```

```
    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];
```

2.5-2 DAG Network

創建用於深度學習的簡單有向無環圖 (DAG) 網絡。 訓練網絡對數字圖像進行分類。 本例中的簡單網絡包括：

- 創建層按順序連接的主分支。
- 包含單個 1×1 卷積層的快捷連接。 (快捷連接使參數梯度更容易從輸出層流向網絡的較早層。)
- 將網絡的主要分支創建為層數組。 添加層按元素對多個輸入求和。

  (指定附加層的輸入數量以求和。 所有層都必須有名稱，並且所有名稱都必須是唯一的。)

Create Simple DAG Network

Create a simple directed acyclic graph (DAG) network for deep learning. Train the network to classify images of digits. The simple network in this example consists of:

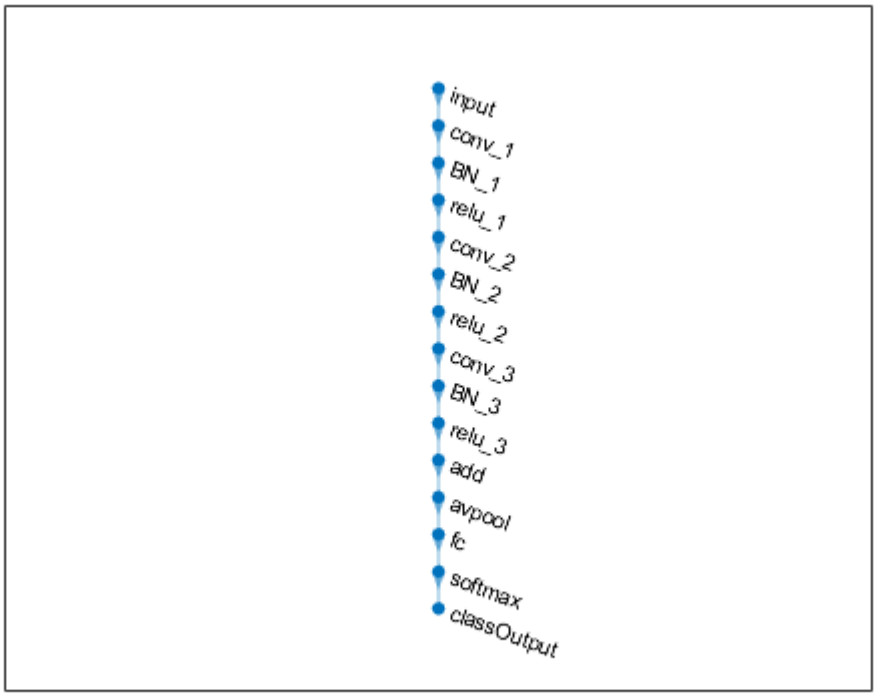
- A main branch with layers connected sequentially.
- A *shortcut connection* containing a single 1-by-1 convolutional layer. Shortcut connections enable the parameter gradients to flow more easily from the output layer to the earlier layers of the network.

Create the main branch of the network as a layer array. The addition layer sums multiple inputs element-wise. Specify the number of inputs for the addition layer to sum. All layers must have names and all names must be unique.

```
layers = [
    imageInputLayer([28 28 1],'Name','input')

    convolution2dLayer(5,16,'Padding','same','Name','conv_1')
    batchNormalizationLayer('Name','BN_1')
    reluLayer('Name','relu_1')

    convolution2dLayer(3,32,'Padding','same','Stride',2,'Name','conv_2')
    batchNormalizationLayer('Name','BN_2')
    reluLayer('Name','relu_2')
    convolution2dLayer(3,32,'Padding','same','Name','conv_3')
    batchNormalizationLayer('Name','BN_3')
    reluLayer('Name','relu_3')

    additionLayer(2,'Name','add')

    averagePooling2dLayer(2,'Stride',2,'Name','avpool')
    fullyConnectedLayer(10,'Name','fc')
    softmaxLayer('Name','softmax')
    classificationLayer('Name','classOutput')];
```

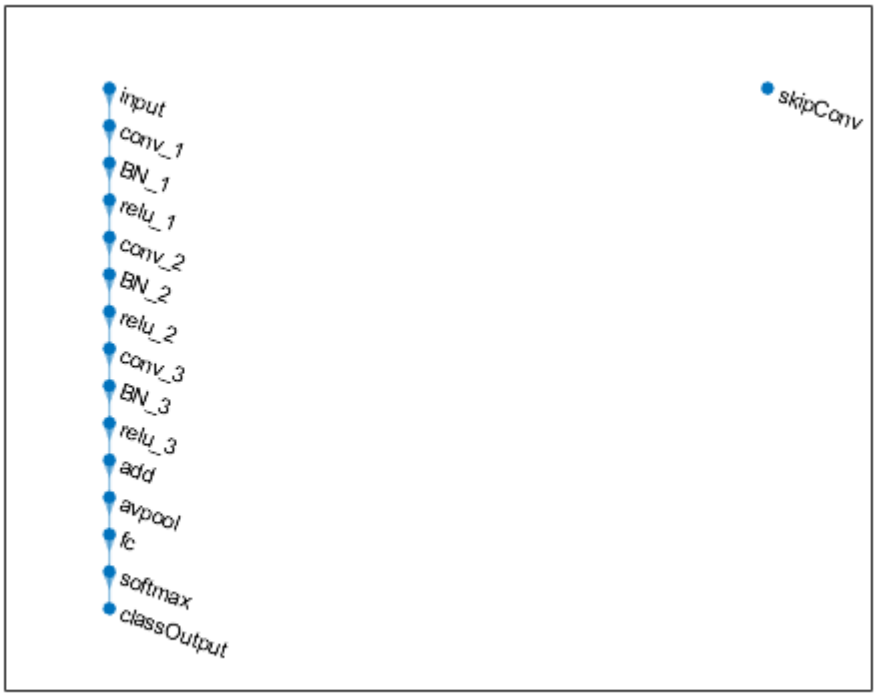
Create a layer graph from the layer array. `layerGraph` connects all the layers in `layers` sequentially. Plot the layer graph.

```
lgraph = layerGraph(layers);
figure
plot(lgraph)
```



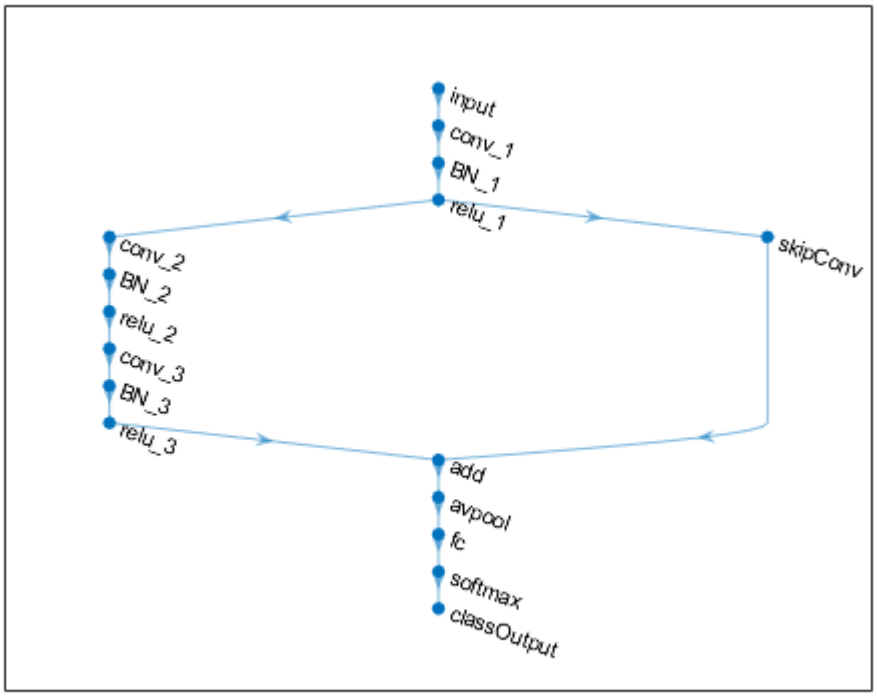
Create the 1-by-1 convolutional layer and add it to the layer graph. Specify the number of convolutional filters and the stride so that the activation size matches the activation size of the `'relu_3'` layer. This arrangement enables the addition layer to add the outputs of the `'skipConv'` and `'relu_3'` layers. To check that the layer is in the graph, plot the layer graph.

```
skipConv = convolution2dLayer(1,32,'Stride',2,'Name','skipConv');
lgraph = addLayers(lgraph,skipConv);
figure
plot(lgraph)
```



Create the shortcut connection from the `'relu_1'` layer to the `'add'` layer. Because you specified two as the number of inputs to the addition layer when you created it, the layer has two inputs named `'in1'` and `'in2'`. The `'relu_3'` layer is already connected to the `'in1'` input. Connect the `'relu_1'` layer to the `'skipConv'` layer and the `'skipConv'` layer to the `'in2'` input of the `'add'` layer. The addition layer now sums the outputs of the `'relu_3'` and `'skipConv'` layers. To check that the layers are connected correctly, plot the layer graph.

```
lgraph = connectLayers(lgraph,'relu_1','skipConv');
lgraph = connectLayers(lgraph,'skipConv','add/in2');
figure
plot(lgraph);
```



Load the training and validation data, which consists of 28-by-28 grayscale images of digits.

```
[XTrain,YTrain] = digitTrain4DArrayData;
[XValidation,YValidation] = digitTest4DArrayData;
```

Specify training options and train the network. `trainNetwork` validates the network using the validation data every `ValidationFrequency` iterations.

```
options = trainingOptions('sgdm', ...
    'MaxEpochs',8, ...
    'Shuffle','every-epoch', ...
    'ValidationData',{XValidation,YValidation}, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
net = trainNetwork(XTrain,YTrain,lgraph,options);
```

Display the properties of the trained network. The network is a `DAGNetwork` object.

```
net
```

Classify the validation images and calculate the accuracy. The network is very accurate.

```
YPredicted = classify(net,XValidation);
```

Create Deep Learning Network Architecture

Script for creating the layers for a deep learning network with the following properties:

```
Number of layers: 71
Number of connections: 78
```

Create Layer Graph

Create the layer graph variable to contain the network layers.

```
lgraph = layerGraph();
```

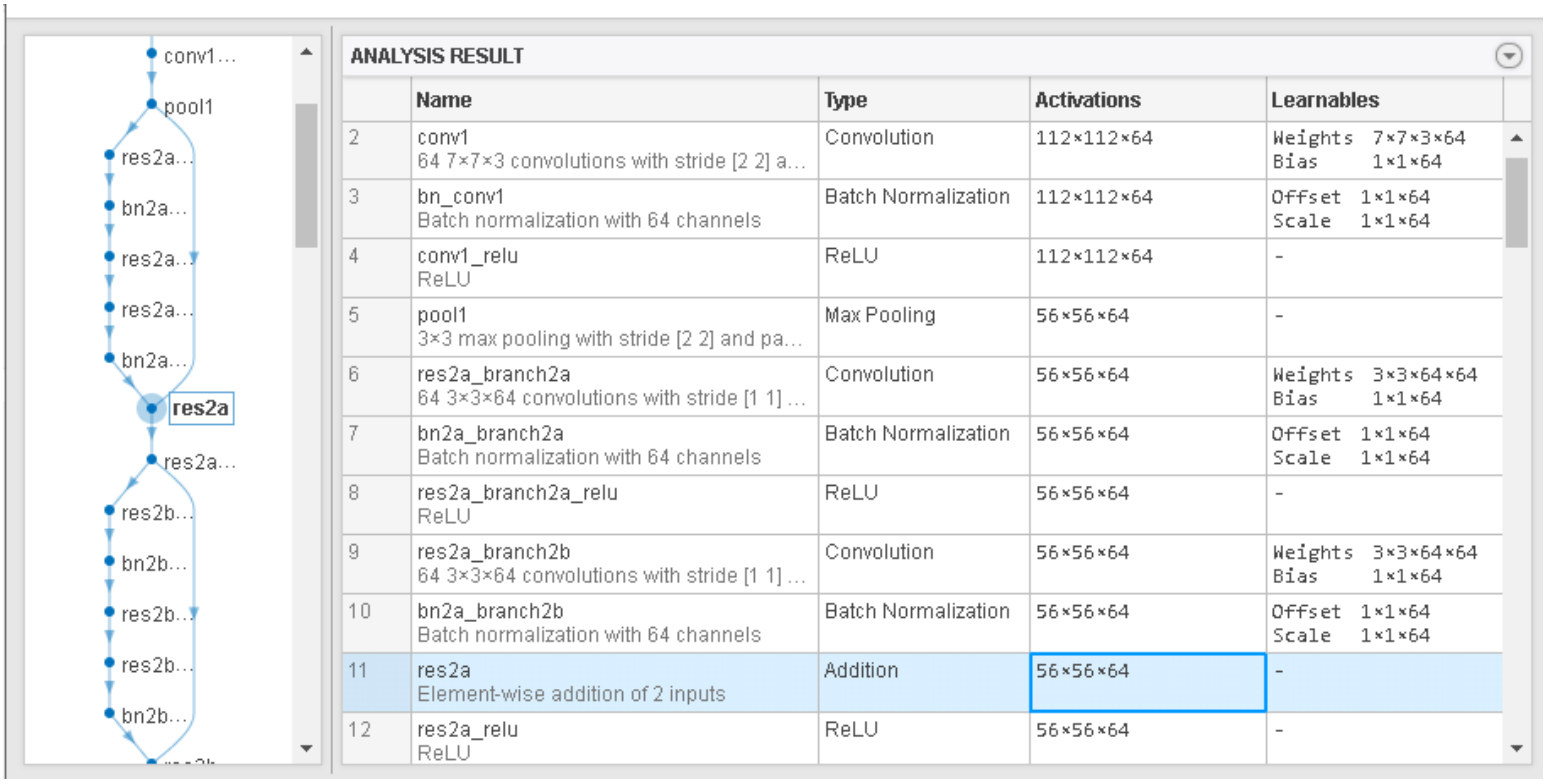
Add Layer Branches

Add the branches of the network to the layer graph. Each branch is a linear array of layers.

```
tempLayers = [
    imageInputLayer([224 224 3],"Name","data","Normalization","zscore")
    convolution2dLayer([7
7],64,"Name","conv1","BiasLearnRateFactor",0,"Padding",[3 3 3 3],"Stride",[2 2])
    batchNormalizationLayer("Name","bn_conv1")
    reluLayer("Name","conv1_relu")
    maxPooling2dLayer([3 3],"Name","pool1","Padding",[1 1 1 1],"Stride",[2
2])];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3
3],64,"Name","res2a_branch2a","BiasLearnRateFactor",0,"Padding",[1 1 1 1])
    batchNormalizationLayer("Name","bn2a_branch2a")
    reluLayer("Name","res2a_branch2a_relu")
    convolution2dLayer([3
3],64,"Name","res2a_branch2b","BiasLearnRateFactor",0,"Padding",[1 1 1 1])
    batchNormalizationLayer("Name","bn2a_branch2b")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res2a")
    reluLayer("Name","res2a_relu")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3
3],64,"Name","res2b_branch2a","BiasLearnRateFactor",0,"Padding",[1 1 1 1])
    batchNormalizationLayer("Name","bn2b_branch2a")
    reluLayer("Name","res2b_branch2a_relu")
```

```
    convolution2dLayer([3
3],64,"Name","res2b_branch2b","BiasLearnRateFactor",0,"Padding",[1 1 1 1])
    batchNormalizationLayer("Name","bn2b_branch2b")];
 lgraph = addLayers(lgraph,tempLayers);

 tempLayers = [
    additionLayer(2,"Name","res2b")
    reluLayer("Name","res2b_relu")];
 lgraph = addLayers(lgraph,tempLayers);
```

```
lgraph = connectLayers(lgraph,"pool1","res2a_branch2a");
lgraph = connectLayers(lgraph,"pool1","res2a/in2");
lgraph = connectLayers(lgraph,"bn2a_branch2b","res2a/in1");
```



	Name	Type	Activations	Learnables
2	conv1 64 7×7×3 convolutions with stride [2 2] a...	Convolution	112×112×64	Weights 7×7×3×64 Bias 1×1×64
3	bn_conv1 Batch normalization with 64 channels	Batch Normalization	112×112×64	Offset 1×1×64 Scale 1×1×64
4	conv1_relu ReLU	ReLU	112×112×64	-
5	pool1 3×3 max pooling with stride [2 2] and pa...	Max Pooling	56×56×64	-
6	res2a_branch2a 64 3×3×64 convolutions with stride [1 1] ...	Convolution	56×56×64	Weights 3×3×64×64 Bias 1×1×64
7	bn2a_branch2a Batch normalization with 64 channels	Batch Normalization	56×56×64	Offset 1×1×64 Scale 1×1×64
8	res2a_branch2a_relu ReLU	ReLU	56×56×64	-
9	res2a_branch2b 64 3×3×64 convolutions with stride [1 1] ...	Convolution	56×56×64	Weights 3×3×64×64 Bias 1×1×64
10	bn2a_branch2b Batch normalization with 64 channels	Batch Normalization	56×56×64	Offset 1×1×64 Scale 1×1×64
11	res2a Element-wise addition of 2 inputs	Addition	56×56×64	-
12	res2a_relu ReLU	ReLU	56×56×64	-

Plot Layers

```
plot(lgraph);
```

Create Deep Learning Network Architecture

Script for creating the layers for a deep learning network with the following properties:

```
Number of layers: 71
Number of connections: 78
```

Run the script to create the layers in the workspace variable `lgraph`.

To learn more, see Generate MATLAB Code From Deep Network Designer.

Auto-generated by MATLAB on 08-Apr-2023 13:15:30

Create Layer Graph

Create the layer graph variable to contain the network layers.

```
lgraph = layerGraph();
```

Add Layer Branches

Add the branches of the network to the layer graph. Each branch is a linear array of layers.

```
tempLayers = [
    imageInputLayer([224 224 3],"Name","data","Normalization","zscore")
    convolution2dLayer([7 7],64,"Name","conv1","BiasLearnRateFactor",0,"Padding",[3 3 3 3],"Str
    batchNormalizationLayer("Name","bn_conv1")
    reluLayer("Name","conv1_relu")
    maxPooling2dLayer([3 3],"Name","pool1","Padding",[1 1 1 1],"Stride",[2 2])];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3 3],64,"Name","res2a_branch2a","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn2a_branch2a")
    reluLayer("Name","res2a_branch2a_relu")
    convolution2dLayer([3 3],64,"Name","res2a_branch2b","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn2a_branch2b")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res2a")
    reluLayer("Name","res2a_relu")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3 3],64,"Name","res2b_branch2a","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn2b_branch2a")
    reluLayer("Name","res2b_branch2a_relu")
    convolution2dLayer([3 3],64,"Name","res2b_branch2b","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn2b_branch2b")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res2b")
    reluLayer("Name","res2b_relu")];
```

```
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3 3],128,"Name","res3a_branch2a","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn3a_branch2a")
    reluLayer("Name","res3a_branch2a_relu")
    convolution2dLayer([3 3],128,"Name","res3a_branch2b","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn3a_branch2b")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],128,"Name","res3a_branch1","BiasLearnRateFactor",0,"Stride",[2 2])
    batchNormalizationLayer("Name","bn3a_branch1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res3a")
    reluLayer("Name","res3a_relu")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3 3],128,"Name","res3b_branch2a","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn3b_branch2a")
    reluLayer("Name","res3b_branch2a_relu")
    convolution2dLayer([3 3],128,"Name","res3b_branch2b","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn3b_branch2b")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res3b")
    reluLayer("Name","res3b_relu")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3 3],256,"Name","res4a_branch2a","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn4a_branch2a")
    reluLayer("Name","res4a_branch2a_relu")
    convolution2dLayer([3 3],256,"Name","res4a_branch2b","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn4a_branch2b")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],256,"Name","res4a_branch1","BiasLearnRateFactor",0,"Stride",[2 2])
    batchNormalizationLayer("Name","bn4a_branch1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res4a")
    reluLayer("Name","res4a_relu")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3 3],256,"Name","res4b_branch2a","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn4b_branch2a")
```

```
    reluLayer("Name","res4b_branch2a_relu")
    convolution2dLayer([3 3],256,"Name","res4b_branch2b","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn4b_branch2b")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res4b")
    reluLayer("Name","res4b_relu")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3 3],512,"Name","res5a_branch2a","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn5a_branch2a")
    reluLayer("Name","res5a_branch2a_relu")
    convolution2dLayer([3 3],512,"Name","res5a_branch2b","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn5a_branch2b")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],512,"Name","res5a_branch1","BiasLearnRateFactor",0,"Stride",[2 2])
    batchNormalizationLayer("Name","bn5a_branch1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res5a")
    reluLayer("Name","res5a_relu")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3 3],512,"Name","res5b_branch2a","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn5b_branch2a")
    reluLayer("Name","res5b_branch2a_relu")
    convolution2dLayer([3 3],512,"Name","res5b_branch2b","BiasLearnRateFactor",0,"Padding",[1 1
    batchNormalizationLayer("Name","bn5b_branch2b")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res5b")
    reluLayer("Name","res5b_relu")
    globalAveragePooling2dLayer("Name","pool5")
    fullyConnectedLayer(1000,"Name","fc1000")
    softmaxLayer("Name","prob")
    classificationLayer("Name","ClassificationLayer_predictions")];
lgraph = addLayers(lgraph,tempLayers);

% clean up helper variable
clear tempLayers;
```

Connect Layer Branches

Connect all the branches of the network to create the network graph.

```
lgraph = connectLayers(lgraph,"pool1","res2a_branch2a");
lgraph = connectLayers(lgraph,"pool1","res2a/in2");
```

```
lgraph = connectLayers(lgraph,"bn2a_branch2b","res2a/in1");
lgraph = connectLayers(lgraph,"res2a_relu","res2b_branch2a");
lgraph = connectLayers(lgraph,"res2a_relu","res2b/in2");
lgraph = connectLayers(lgraph,"bn2b_branch2b","res2b/in1");
lgraph = connectLayers(lgraph,"res2b_relu","res3a_branch2a");
lgraph = connectLayers(lgraph,"res2b_relu","res3a_branch1");
lgraph = connectLayers(lgraph,"bn3a_branch1","res3a/in2");
lgraph = connectLayers(lgraph,"bn3a_branch2b","res3a/in1");
lgraph = connectLayers(lgraph,"res3a_relu","res3b_branch2a");
lgraph = connectLayers(lgraph,"res3a_relu","res3b/in2");
lgraph = connectLayers(lgraph,"bn3b_branch2b","res3b/in1");
lgraph = connectLayers(lgraph,"res3b_relu","res4a_branch2a");
lgraph = connectLayers(lgraph,"res3b_relu","res4a_branch1");
lgraph = connectLayers(lgraph,"bn4a_branch2b","res4a/in1");
lgraph = connectLayers(lgraph,"bn4a_branch1","res4a/in2");
lgraph = connectLayers(lgraph,"res4a_relu","res4b_branch2a");
lgraph = connectLayers(lgraph,"res4a_relu","res4b/in2");
lgraph = connectLayers(lgraph,"bn4b_branch2b","res4b/in1");
lgraph = connectLayers(lgraph,"res4b_relu","res5a_branch2a");
lgraph = connectLayers(lgraph,"res4b_relu","res5a_branch1");
lgraph = connectLayers(lgraph,"bn5a_branch1","res5a/in2");
lgraph = connectLayers(lgraph,"bn5a_branch2b","res5a/in1");
lgraph = connectLayers(lgraph,"res5a_relu","res5b_branch2a");
lgraph = connectLayers(lgraph,"res5a_relu","res5b/in2");
lgraph = connectLayers(lgraph,"bn5b_branch2b","res5b/in1");
```

Plot Layers

```
plot(lgraph);
```

Create Deep Learning Network Architecture

Script for creating the layers for a deep learning network with the following properties:

```
Number of layers: 144
Number of connections: 170
```

Run the script to create the layers in the workspace variable `lgraph`.

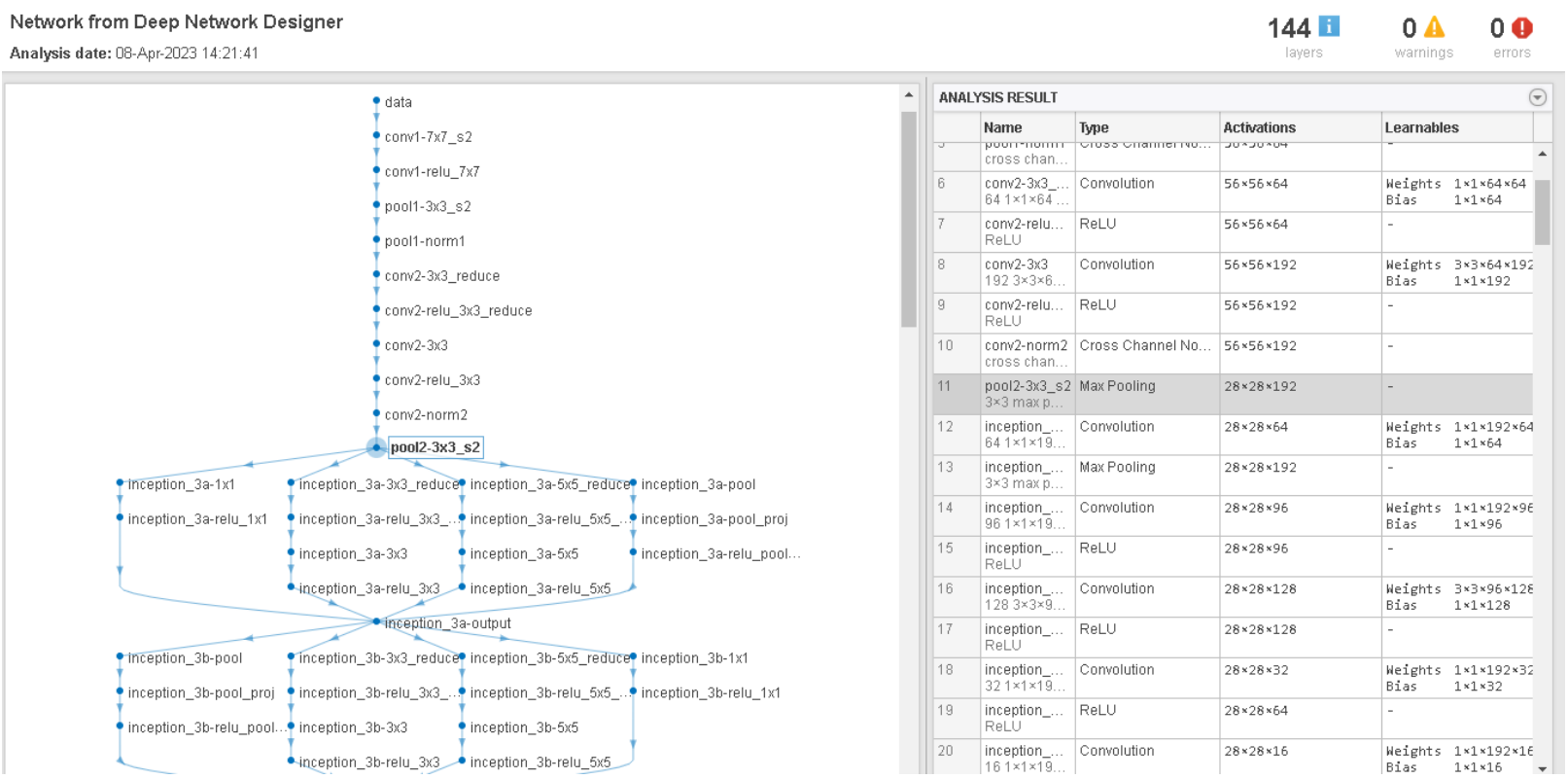
To learn more, see Generate MATLAB Code From Deep Network Designer.

Auto-generated by MATLAB on 08-Apr-2023 14:20:21

Create Layer Graph

Create the layer graph variable to contain the network layers.

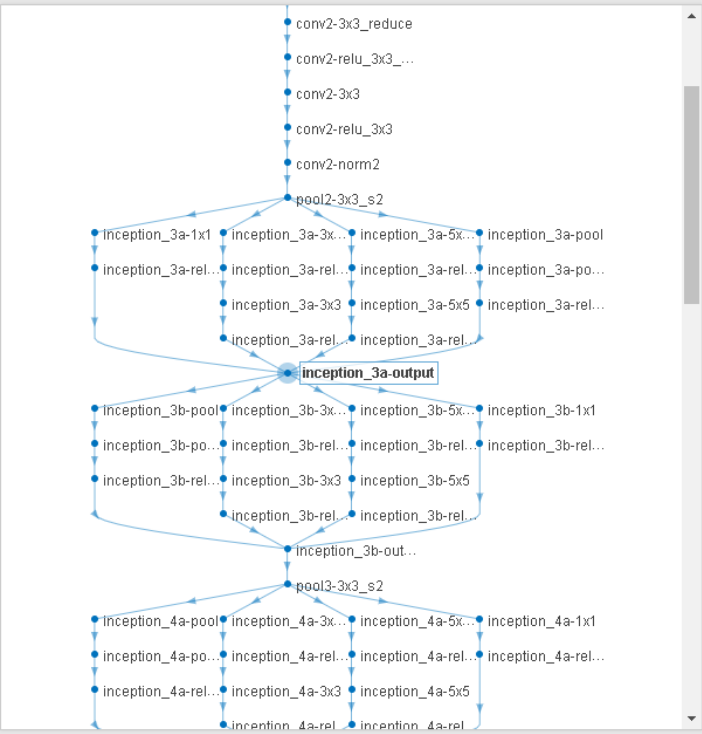
```
lgraph = layerGraph();
```



Network from Deep Network Designer

Analysis date: 08-Apr-2023 14:21:41

144 layers | 0 warnings | 0 errors



ANALYSIS RESULT

	Name	Type	Activations	Learnables
	ReLU			
16	inception_3a-3x3 128 3×3×96 convolutions with stride [1 1] an...	Convolution	28×28×128	Weights 3×3×96×128 Bias 1×1×128
17	inception_3a-relu_3x3 ReLU	ReLU	28×28×128	-
18	inception_3a-pool_proj 32 1×1×192 convolutions with stride [1 1] an...	Convolution	28×28×32	Weights 1×1×192×32 Bias 1×1×32
19	inception_3a-relu_1x1 ReLU	ReLU	28×28×64	-
20	inception_3a-5x5_reduce 16 1×1×192 convolutions with stride [1 1] an...	Convolution	28×28×16	Weights 1×1×192×16 Bias 1×1×16
21	inception_3a-relu_5x5_reduce ReLU	ReLU	28×28×16	-
22	inception_3a-5x5 32 5×5×16 convolutions with stride [1 1] and ...	Convolution	28×28×32	Weights 5×5×16×32 Bias 1×1×32
23	inception_3a-relu_5x5 ReLU	ReLU	28×28×32	-
24	inception_3a-relu_pool_proj ReLU	ReLU	28×28×32	-
25	inception_3a-output Depth concatenation of 4 inputs	Depth concatenation	28×28×256	-
26	inception_3b-pool 3×3 max pooling with stride [1 1] and paddin...	Max Pooling	28×28×256	-
27	inception_3b-pool_proj 64 1×1×256 convolutions with stride [1 1] an...	Convolution	28×28×64	Weights 1×1×256×64 Bias 1×1×64
28	inception_3b-3x3_reduce 128 1×1×256 convolutions with stride [1 1] a...	Convolution	28×28×128	Weigh... 1×1×256×1... Bias 1×1×128
29	inception_3b-relu_3x3_reduce ReLU	ReLU	28×28×128	-
30	inception_3b-1x1 128 1×1×256 convolutions with stride [1 1] a...	Convolution	28×28×128	Weigh... 1×1×256×1... Bias 1×1×128
31	inception_3b-relu_1x1 ReLU	ReLU	28×28×128	-

```
lgraph = connectLayers(lgraph,"pool2-3x3_s2","inception_3a-1x1");
lgraph = connectLayers(lgraph,"pool2-3x3_s2","inception_3a-pool");
lgraph = connectLayers(lgraph,"pool2-3x3_s2","inception_3a-3x3_reduce");
lgraph = connectLayers(lgraph,"pool2-3x3_s2","inception_3a-5x5_reduce");
lgraph = connectLayers(lgraph,"inception_3a-relu_3x3","inception_3a-
output/in2");
lgraph = connectLayers(lgraph,"inception_3a-relu_1x1","inception_3a-
output/in1");
lgraph = connectLayers(lgraph,"inception_3a-relu_5x5","inception_3a-
output/in3");
lgraph = connectLayers(lgraph,"inception_3a-relu_pool_proj","inception_3a-
output/in4");
lgraph = connectLayers(lgraph,"inception_3a-output","inception_3b-pool");
lgraph = connectLayers(lgraph,"inception_3a-output","inception_3b-3x3_reduce");
lgraph = connectLayers(lgraph,"inception_3a-output","inception_3b-1x1");
lgraph = connectLayers(lgraph,"inception_3a-output","inception_3b-5x5_reduce");
lgraph = connectLayers(lgraph,"inception_3b-relu_1x1","inception_3b-
output/in1");
lgraph = connectLayers(lgraph,"inception_3b-relu_3x3","inception_3b-
output/in2");
lgraph = connectLayers(lgraph,"inception_3b-relu_pool_proj","inception_3b-
output/in4");
```

```
lgraph = connectLayers(lgraph,"inception_3b-relu_5x5","inception_3b-output/in3");
```

lgraph = connectLayers(lgraph,"inception_3b-relu_5x5","inception_3b-output/in3");

Add Layer Branches

Add the branches of the network to the layer graph. Each branch is a linear array of layers.

```
 tempLayers = [
     imageInputLayer([224 224 3],"Name","data")
     convolution2dLayer([7 7],64,"Name","conv1-7x7_s2","BiasLearnRateFactor",2,"Padding",[3 3 3 3],"Stride",[2 2])
     reluLayer("Name","conv1-relu_7x7")
     maxPooling2dLayer([3 3],"Name","pool1-3x3_s2","Padding",[0 1 0 1],"Stride",[2 2])
     crossChannelNormalizationLayer(5,"Name","pool1-norm1","K",1)
     convolution2dLayer([1 1],64,"Name","conv2-3x3_reduce","BiasLearnRateFactor",2)
     reluLayer("Name","conv2-relu_3x3_reduce")
     convolution2dLayer([3 3],192,"Name","conv2-3x3","BiasLearnRateFactor",2,"Padding",[1 1 1 1])
     reluLayer("Name","conv2-relu_3x3")
     crossChannelNormalizationLayer(5,"Name","conv2-norm2","K",1)
     maxPooling2dLayer([3 3],"Name","pool2-3x3_s2","Padding",[0 1 0 1],"Stride",[2 2])];
 lgraph = addLayers(lgraph,tempLayers);

 tempLayers = [
     convolution2dLayer([1 1],64,"Name","inception_3a-1x1","BiasLearnRateFactor",2)
     reluLayer("Name","inception_3a-relu_1x1")];
 lgraph = addLayers(lgraph,tempLayers);

 tempLayers = [
     maxPooling2dLayer([3 3],"Name","inception_3a-pool","Padding",[1 1 1 1])
     convolution2dLayer([1 1],32,"Name","inception_3a-pool_proj","BiasLearnRateFactor",2)
     reluLayer("Name","inception_3a-relu_pool_proj")];
 lgraph = addLayers(lgraph,tempLayers);

 tempLayers = [
     convolution2dLayer([1 1],96,"Name","inception_3a-3x3_reduce","BiasLearnRateFactor",2)
     reluLayer("Name","inception_3a-relu_3x3_reduce")
```

```
    convolution2dLayer([3 3],128,"Name","inception_3a-
3x3","BiasLearnRateFactor",2,"Padding",[1 1 1 1])
    reluLayer("Name","inception_3a-relu_3x3")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],16,"Name","inception_3a-
5x5_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_3a-relu_5x5_reduce")
    convolution2dLayer([5 5],32,"Name","inception_3a-
5x5","BiasLearnRateFactor",2,"Padding",[2 2 2 2])
    reluLayer("Name","inception_3a-relu_5x5")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = depthConcatenationLayer(4,"Name","inception_3a-output");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    maxPooling2dLayer([3 3],"Name","inception_3b-pool","Padding",[1 1 1 1])
    convolution2dLayer([1 1],64,"Name","inception_3b-
pool_proj","BiasLearnRateFactor",2)
    reluLayer("Name","inception_3b-relu_pool_proj")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],128,"Name","inception_3b-
3x3_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_3b-relu_3x3_reduce")
    convolution2dLayer([3 3],192,"Name","inception_3b-
3x3","BiasLearnRateFactor",2,"Padding",[1 1 1 1])
    reluLayer("Name","inception_3b-relu_3x3")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],128,"Name","inception_3b-
1x1","BiasLearnRateFactor",2)
    reluLayer("Name","inception_3b-relu_1x1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],32,"Name","inception_3b-
5x5_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_3b-relu_5x5_reduce")
    convolution2dLayer([5 5],96,"Name","inception_3b-
5x5","BiasLearnRateFactor",2,"Padding",[2 2 2 2])
```

```
    reluLayer("Name","inception_3b-relu_5x5")];
 lgraph = addLayers(lgraph,tempLayers);

 tempLayers = [
    depthConcatenationLayer(4,"Name","inception_3b-output")
    maxPooling2dLayer([3 3],"Name","pool3-3x3_s2","Padding",[0 1 0
1],"Stride",[2 2])];
 lgraph = addLayers(lgraph,tempLayers);
```

Plot Layers

```
 plot(lgraph);
```

Create Deep Learning Network Architecture

Script for creating the layers for a deep learning network with the following properties:

```
Number of layers: 144
Number of connections: 170
```

Run the script to create the layers in the workspace variable `lgraph`.

To learn more, see Generate MATLAB Code From Deep Network Designer.

Auto-generated by MATLAB on 08-Apr-2023 13:08:20

Create Layer Graph

Create the layer graph variable to contain the network layers.

```
lgraph = layerGraph();
```

Add Layer Branches

Add the branches of the network to the layer graph. Each branch is a linear array of layers.

```
tempLayers = [
    imageInputLayer([224 224 3],"Name","data")
    convolution2dLayer([7 7],64,"Name","conv1-7x7_s2","BiasLearnRateFactor",2,"Padding",[3 3 3
    reluLayer("Name","conv1-relu_7x7")
    maxPooling2dLayer([3 3],"Name","pool1-3x3_s2","Padding",[0 1 0 1],"Stride",[2 2])
    crossChannelNormalizationLayer(5,"Name","pool1-norm1","K",1)
    convolution2dLayer([1 1],64,"Name","conv2-3x3_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","conv2-relu_3x3_reduce")
    convolution2dLayer([3 3],192,"Name","conv2-3x3","BiasLearnRateFactor",2,"Padding",[1 1 1 1
    reluLayer("Name","conv2-relu_3x3")
    crossChannelNormalizationLayer(5,"Name","conv2-norm2","K",1)
    maxPooling2dLayer([3 3],"Name","pool2-3x3_s2","Padding",[0 1 0 1],"Stride",[2 2])];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    maxPooling2dLayer([3 3],"Name","inception_3a-pool","Padding",[1 1 1 1])
    convolution2dLayer([1 1],32,"Name","inception_3a-pool_proj","BiasLearnRateFactor",2)
    reluLayer("Name","inception_3a-relu_pool_proj")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],16,"Name","inception_3a-5x5_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_3a-relu_5x5_reduce")
    convolution2dLayer([5 5],32,"Name","inception_3a-5x5","BiasLearnRateFactor",2,"Padding",[2
    reluLayer("Name","inception_3a-relu_5x5")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],96,"Name","inception_3a-3x3_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_3a-relu_3x3_reduce")
    convolution2dLayer([3 3],128,"Name","inception_3a-3x3","BiasLearnRateFactor",2,"Padding",[1
    reluLayer("Name","inception_3a-relu_3x3")];
```

```
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],64,"Name","inception_3a-1x1","BiasLearnRateFactor",2)
    reluLayer("Name","inception_3a-relu_1x1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = depthConcatenationLayer(4,"Name","inception_3a-output");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],128,"Name","inception_3b-1x1","BiasLearnRateFactor",2)
    reluLayer("Name","inception_3b-relu_1x1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],32,"Name","inception_3b-5x5_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_3b-relu_5x5_reduce")
    convolution2dLayer([5 5],96,"Name","inception_3b-5x5","BiasLearnRateFactor",2,"Padding",[2
    reluLayer("Name","inception_3b-relu_5x5")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],128,"Name","inception_3b-3x3_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_3b-relu_3x3_reduce")
    convolution2dLayer([3 3],192,"Name","inception_3b-3x3","BiasLearnRateFactor",2,"Padding",[1
    reluLayer("Name","inception_3b-relu_3x3")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    maxPooling2dLayer([3 3],"Name","inception_3b-pool","Padding",[1 1 1 1])
    convolution2dLayer([1 1],64,"Name","inception_3b-pool_proj","BiasLearnRateFactor",2)
    reluLayer("Name","inception_3b-relu_pool_proj")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    depthConcatenationLayer(4,"Name","inception_3b-output")
    maxPooling2dLayer([3 3],"Name","pool3-3x3_s2","Padding",[0 1 0 1],"Stride",[2 2])];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],96,"Name","inception_4a-3x3_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4a-relu_3x3_reduce")
    convolution2dLayer([3 3],208,"Name","inception_4a-3x3","BiasLearnRateFactor",2,"Padding",[1
    reluLayer("Name","inception_4a-relu_3x3")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    maxPooling2dLayer([3 3],"Name","inception_4a-pool","Padding",[1 1 1 1])
    convolution2dLayer([1 1],64,"Name","inception_4a-pool_proj","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4a-relu_pool_proj")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
```

```
    convolution2dLayer([1 1],16,"Name","inception_4a-5x5_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4a-relu_5x5_reduce")
    convolution2dLayer([5 5],48,"Name","inception_4a-5x5","BiasLearnRateFactor",2,"Padding",[2
    reluLayer("Name","inception_4a-relu_5x5")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],192,"Name","inception_4a-1x1","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4a-relu_1x1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = depthConcatenationLayer(4,"Name","inception_4a-output");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],160,"Name","inception_4b-1x1","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4b-relu_1x1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],24,"Name","inception_4b-5x5_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4b-relu_5x5_reduce")
    convolution2dLayer([5 5],64,"Name","inception_4b-5x5","BiasLearnRateFactor",2,"Padding",[2
    reluLayer("Name","inception_4b-relu_5x5")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    maxPooling2dLayer([3 3],"Name","inception_4b-pool","Padding",[1 1 1 1])
    convolution2dLayer([1 1],64,"Name","inception_4b-pool_proj","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4b-relu_pool_proj")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],112,"Name","inception_4b-3x3_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4b-relu_3x3_reduce")
    convolution2dLayer([3 3],224,"Name","inception_4b-3x3","BiasLearnRateFactor",2,"Padding",[1
    reluLayer("Name","inception_4b-relu_3x3")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = depthConcatenationLayer(4,"Name","inception_4b-output");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    maxPooling2dLayer([3 3],"Name","inception_4c-pool","Padding",[1 1 1 1])
    convolution2dLayer([1 1],64,"Name","inception_4c-pool_proj","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4c-relu_pool_proj")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],128,"Name","inception_4c-1x1","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4c-relu_1x1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
```

```
    convolution2dLayer([1 1],128,"Name","inception_4c-3x3_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4c-relu_3x3_reduce")
    convolution2dLayer([3 3],256,"Name","inception_4c-3x3","BiasLearnRateFactor",2,"Padding",[1
    reluLayer("Name","inception_4c-relu_3x3")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],24,"Name","inception_4c-5x5_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4c-relu_5x5_reduce")
    convolution2dLayer([5 5],64,"Name","inception_4c-5x5","BiasLearnRateFactor",2,"Padding",[2
    reluLayer("Name","inception_4c-relu_5x5")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = depthConcatenationLayer(4,"Name","inception_4c-output");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    maxPooling2dLayer([3 3],"Name","inception_4d-pool","Padding",[1 1 1 1])
    convolution2dLayer([1 1],64,"Name","inception_4d-pool_proj","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4d-relu_pool_proj")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],32,"Name","inception_4d-5x5_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4d-relu_5x5_reduce")
    convolution2dLayer([5 5],64,"Name","inception_4d-5x5","BiasLearnRateFactor",2,"Padding",[2
    reluLayer("Name","inception_4d-relu_5x5")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],112,"Name","inception_4d-1x1","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4d-relu_1x1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],144,"Name","inception_4d-3x3_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4d-relu_3x3_reduce")
    convolution2dLayer([3 3],288,"Name","inception_4d-3x3","BiasLearnRateFactor",2,"Padding",[1
    reluLayer("Name","inception_4d-relu_3x3")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = depthConcatenationLayer(4,"Name","inception_4d-output");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    maxPooling2dLayer([3 3],"Name","inception_4e-pool","Padding",[1 1 1 1])
    convolution2dLayer([1 1],128,"Name","inception_4e-pool_proj","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4e-relu_pool_proj")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],160,"Name","inception_4e-3x3_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4e-relu_3x3_reduce")
    convolution2dLayer([3 3],320,"Name","inception_4e-3x3","BiasLearnRateFactor",2,"Padding",[1
```

```
    reluLayer("Name","inception_4e-relu_3x3")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],256,"Name","inception_4e-1x1","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4e-relu_1x1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],32,"Name","inception_4e-5x5_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_4e-relu_5x5_reduce")
    convolution2dLayer([5 5],128,"Name","inception_4e-5x5","BiasLearnRateFactor",2,"Padding",[2
    reluLayer("Name","inception_4e-relu_5x5")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    depthConcatenationLayer(4,"Name","inception_4e-output")
    maxPooling2dLayer([3 3],"Name","pool4-3x3_s2","Padding",[0 1 0 1],"Stride",[2 2])];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],256,"Name","inception_5a-1x1","BiasLearnRateFactor",2)
    reluLayer("Name","inception_5a-relu_1x1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    maxPooling2dLayer([3 3],"Name","inception_5a-pool","Padding",[1 1 1 1])
    convolution2dLayer([1 1],128,"Name","inception_5a-pool_proj","BiasLearnRateFactor",2)
    reluLayer("Name","inception_5a-relu_pool_proj")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],160,"Name","inception_5a-3x3_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_5a-relu_3x3_reduce")
    convolution2dLayer([3 3],320,"Name","inception_5a-3x3","BiasLearnRateFactor",2,"Padding",[1
    reluLayer("Name","inception_5a-relu_3x3")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],32,"Name","inception_5a-5x5_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_5a-relu_5x5_reduce")
    convolution2dLayer([5 5],128,"Name","inception_5a-5x5","BiasLearnRateFactor",2,"Padding",[2
    reluLayer("Name","inception_5a-relu_5x5")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = depthConcatenationLayer(4,"Name","inception_5a-output");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    maxPooling2dLayer([3 3],"Name","inception_5b-pool","Padding",[1 1 1 1])
    convolution2dLayer([1 1],128,"Name","inception_5b-pool_proj","BiasLearnRateFactor",2)
    reluLayer("Name","inception_5b-relu_pool_proj")];
lgraph = addLayers(lgraph,tempLayers);
```

```
tempLayers = [
    convolution2dLayer([1 1],192,"Name","inception_5b-3x3_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_5b-relu_3x3_reduce")
    convolution2dLayer([3 3],384,"Name","inception_5b-3x3","BiasLearnRateFactor",2,"Padding",[1
    reluLayer("Name","inception_5b-relu_3x3")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],384,"Name","inception_5b-1x1","BiasLearnRateFactor",2)
    reluLayer("Name","inception_5b-relu_1x1")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1 1],48,"Name","inception_5b-5x5_reduce","BiasLearnRateFactor",2)
    reluLayer("Name","inception_5b-relu_5x5_reduce")
    convolution2dLayer([5 5],128,"Name","inception_5b-5x5","BiasLearnRateFactor",2,"Padding",[2
    reluLayer("Name","inception_5b-relu_5x5")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    depthConcatenationLayer(4,"Name","inception_5b-output")
    globalAveragePooling2dLayer("Name","pool5-7x7_s1")
    dropoutLayer(0.4,"Name","pool5-drop_7x7_s1")
    fullyConnectedLayer(1000,"Name","loss3-classifier","BiasLearnRateFactor",2)
    softmaxLayer("Name","prob")
    classificationLayer("Name","output")];
lgraph = addLayers(lgraph,tempLayers);

% clean up helper variable
clear tempLayers;
```

Connect Layer Branches

Connect all the branches of the network to create the network graph.

```
lgraph = connectLayers(lgraph,"pool2-3x3_s2","inception_3a-pool");
lgraph = connectLayers(lgraph,"pool2-3x3_s2","inception_3a-5x5_reduce");
lgraph = connectLayers(lgraph,"pool2-3x3_s2","inception_3a-3x3_reduce");
lgraph = connectLayers(lgraph,"pool2-3x3_s2","inception_3a-1x1");
lgraph = connectLayers(lgraph,"inception_3a-relu_pool_proj","inception_3a-output/in4");
lgraph = connectLayers(lgraph,"inception_3a-relu_5x5","inception_3a-output/in3");
lgraph = connectLayers(lgraph,"inception_3a-relu_3x3","inception_3a-output/in2");
lgraph = connectLayers(lgraph,"inception_3a-relu_1x1","inception_3a-output/in1");
lgraph = connectLayers(lgraph,"inception_3a-output","inception_3b-1x1");
lgraph = connectLayers(lgraph,"inception_3a-output","inception_3b-5x5_reduce");
lgraph = connectLayers(lgraph,"inception_3a-output","inception_3b-3x3_reduce");
lgraph = connectLayers(lgraph,"inception_3a-output","inception_3b-pool");
lgraph = connectLayers(lgraph,"inception_3b-relu_1x1","inception_3b-output/in1");
lgraph = connectLayers(lgraph,"inception_3b-relu_5x5","inception_3b-output/in3");
lgraph = connectLayers(lgraph,"inception_3b-relu_3x3","inception_3b-output/in2");
lgraph = connectLayers(lgraph,"inception_3b-relu_pool_proj","inception_3b-output/in4");
lgraph = connectLayers(lgraph,"pool3-3x3_s2","inception_4a-3x3_reduce");
lgraph = connectLayers(lgraph,"pool3-3x3_s2","inception_4a-pool");
lgraph = connectLayers(lgraph,"pool3-3x3_s2","inception_4a-5x5_reduce");
```

```
lgraph = connectLayers(lgraph,"pool3-3x3_s2","inception_4a-1x1");
lgraph = connectLayers(lgraph,"inception_4a-relu_5x5","inception_4a-output/in3");
lgraph = connectLayers(lgraph,"inception_4a-relu_pool_proj","inception_4a-output/in4");
lgraph = connectLayers(lgraph,"inception_4a-relu_3x3","inception_4a-output/in2");
lgraph = connectLayers(lgraph,"inception_4a-relu_1x1","inception_4a-output/in1");
lgraph = connectLayers(lgraph,"inception_4a-output","inception_4b-1x1");
lgraph = connectLayers(lgraph,"inception_4a-output","inception_4b-5x5_reduce");
lgraph = connectLayers(lgraph,"inception_4a-output","inception_4b-pool");
lgraph = connectLayers(lgraph,"inception_4a-output","inception_4b-3x3_reduce");
lgraph = connectLayers(lgraph,"inception_4b-relu_1x1","inception_4b-output/in1");
lgraph = connectLayers(lgraph,"inception_4b-relu_5x5","inception_4b-output/in3");
lgraph = connectLayers(lgraph,"inception_4b-relu_3x3","inception_4b-output/in2");
lgraph = connectLayers(lgraph,"inception_4b-relu_pool_proj","inception_4b-output/in4");
lgraph = connectLayers(lgraph,"inception_4b-output","inception_4c-pool");
lgraph = connectLayers(lgraph,"inception_4b-output","inception_4c-1x1");
lgraph = connectLayers(lgraph,"inception_4b-output","inception_4c-3x3_reduce");
lgraph = connectLayers(lgraph,"inception_4b-output","inception_4c-5x5_reduce");
lgraph = connectLayers(lgraph,"inception_4c-relu_1x1","inception_4c-output/in1");
lgraph = connectLayers(lgraph,"inception_4c-relu_pool_proj","inception_4c-output/in4");
lgraph = connectLayers(lgraph,"inception_4c-relu_5x5","inception_4c-output/in3");
lgraph = connectLayers(lgraph,"inception_4c-relu_3x3","inception_4c-output/in2");
lgraph = connectLayers(lgraph,"inception_4c-output","inception_4d-pool");
lgraph = connectLayers(lgraph,"inception_4c-output","inception_4d-5x5_reduce");
lgraph = connectLayers(lgraph,"inception_4c-output","inception_4d-1x1");
lgraph = connectLayers(lgraph,"inception_4c-output","inception_4d-3x3_reduce");
lgraph = connectLayers(lgraph,"inception_4d-relu_pool_proj","inception_4d-output/in4");
lgraph = connectLayers(lgraph,"inception_4d-relu_5x5","inception_4d-output/in3");
lgraph = connectLayers(lgraph,"inception_4d-relu_3x3","inception_4d-output/in2");
lgraph = connectLayers(lgraph,"inception_4d-relu_1x1","inception_4d-output/in1");
lgraph = connectLayers(lgraph,"inception_4d-output","inception_4e-pool");
lgraph = connectLayers(lgraph,"inception_4d-output","inception_4e-3x3_reduce");
lgraph = connectLayers(lgraph,"inception_4d-output","inception_4e-1x1");
lgraph = connectLayers(lgraph,"inception_4d-output","inception_4e-5x5_reduce");
lgraph = connectLayers(lgraph,"inception_4e-relu_3x3","inception_4e-output/in2");
lgraph = connectLayers(lgraph,"inception_4e-relu_1x1","inception_4e-output/in1");
lgraph = connectLayers(lgraph,"inception_4e-relu_5x5","inception_4e-output/in3");
lgraph = connectLayers(lgraph,"inception_4e-relu_pool_proj","inception_4e-output/in4");
lgraph = connectLayers(lgraph,"pool4-3x3_s2","inception_5a-1x1");
lgraph = connectLayers(lgraph,"pool4-3x3_s2","inception_5a-pool");
lgraph = connectLayers(lgraph,"pool4-3x3_s2","inception_5a-3x3_reduce");
lgraph = connectLayers(lgraph,"pool4-3x3_s2","inception_5a-5x5_reduce");
lgraph = connectLayers(lgraph,"inception_5a-relu_pool_proj","inception_5a-output/in4");
lgraph = connectLayers(lgraph,"inception_5a-relu_1x1","inception_5a-output/in1");
lgraph = connectLayers(lgraph,"inception_5a-relu_3x3","inception_5a-output/in2");
lgraph = connectLayers(lgraph,"inception_5a-relu_5x5","inception_5a-output/in3");
lgraph = connectLayers(lgraph,"inception_5a-output","inception_5b-pool");
lgraph = connectLayers(lgraph,"inception_5a-output","inception_5b-3x3_reduce");
lgraph = connectLayers(lgraph,"inception_5a-output","inception_5b-1x1");
lgraph = connectLayers(lgraph,"inception_5a-output","inception_5b-5x5_reduce");
lgraph = connectLayers(lgraph,"inception_5b-relu_1x1","inception_5b-output/in1");
lgraph = connectLayers(lgraph,"inception_5b-relu_pool_proj","inception_5b-output/in4");
lgraph = connectLayers(lgraph,"inception_5b-relu_5x5","inception_5b-output/in3");
lgraph = connectLayers(lgraph,"inception_5b-relu_3x3","inception_5b-output/in2");
```

Plot Layers

```
plot(lgraph);
```