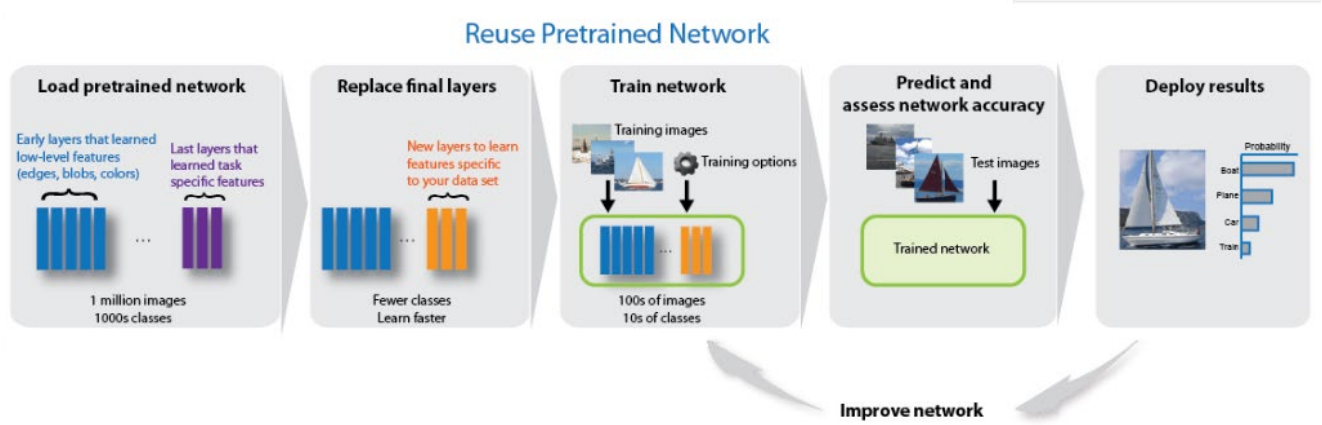
2.6 Transfer Learning Using Pretrained Network

此示例說明如何微調預訓練的 GoogLeNet 卷積神經網絡以對新的圖像集合執行分類。

GoogLeNet 已經在超過一百萬張圖像上進行了訓練，可以將圖像分類為 1000 個對像類別（例如鍵盤、咖啡杯、鉛筆和許多動物）。 該網絡已經為各種圖像學習了豐富的特徵表示。 該網絡將圖像作為輸入並輸出圖像中對象的標籤以及每個對像類別的概率。

遷移學習通常用於深度學習應用程序。 您可以採用預訓練網絡並將其用作學習新任務的起點。 使用遷移學習微調網絡通常比從頭開始訓練具有隨機初始化權重的網絡更快更容易。 您可以使用較少數量的訓練圖像快速將學習到的特徵轉移到新任務中。



Load Data

Unzip and load the new images as an image datastore. `imageDatastore` automatically labels the images based on folder names and stores the data as an `ImageDatastore` object. An image datastore enables you to store large image data, including data that does not fit in memory, and efficiently read batches of images during training of a convolutional neural network.

```
unzip('MerchData.zip');
imds = imageDatastore('MerchData', ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');
```

Divide the data into training and validation data sets. Use 70% of the images for training and 30% for validation. `splitEachLabel` splits the image datastore into two new datastores.

```
[imdsTrain,imdsValidation] = splitEachLabel(imds,0.7,'randomized');
```

This very small data set now contains 55 training images and 20 validation images. Display some sample images.

```
numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsTrain,idx(i));
    imshow(I)
end
```

Load Pretrained Network

Load the pretrained GoogLeNet neural network. If Deep Learning Toolbox™ Model *for GoogLeNet Network* is not installed, then the software provides a download link.

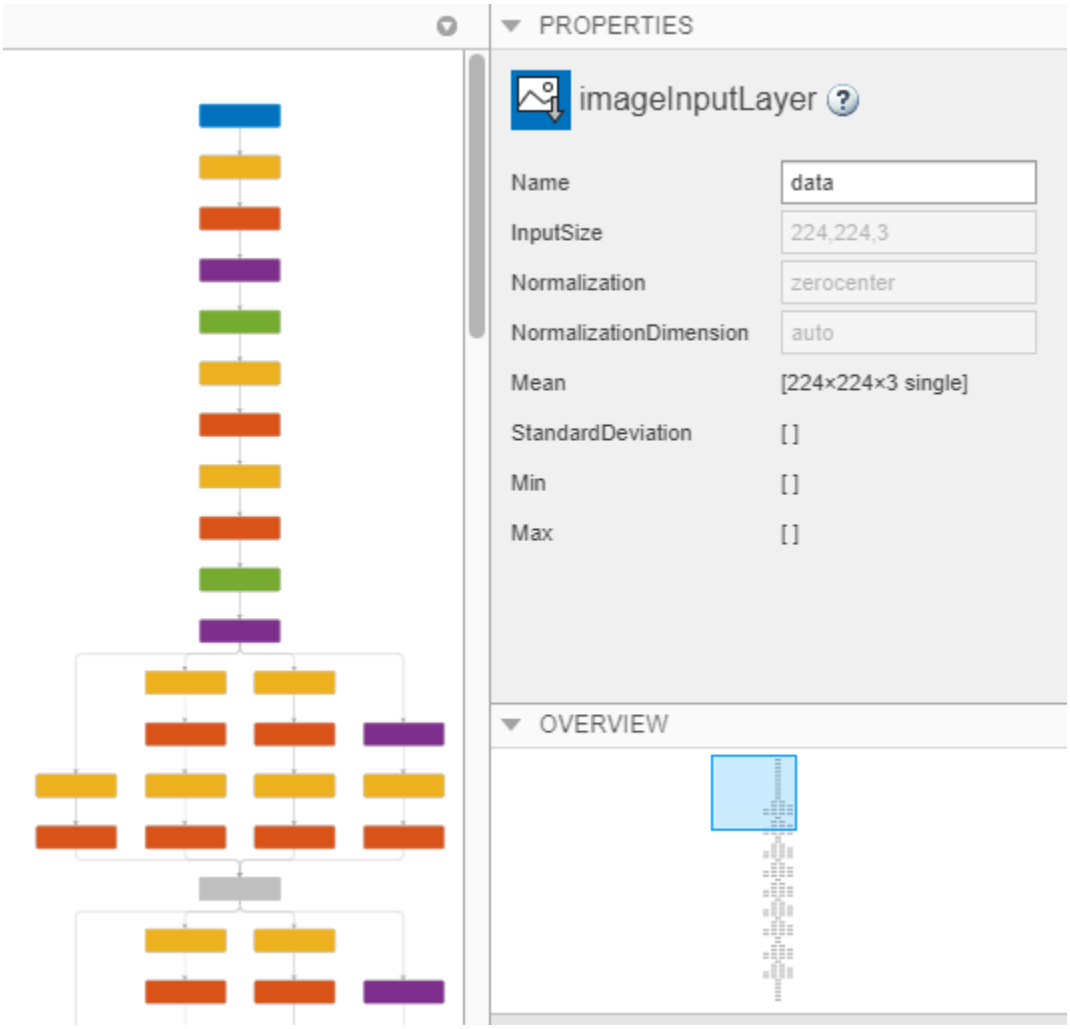
```
net = googlenet;
```

Use `deepNetworkDesigner` to display an interactive visualization of the network architecture and detailed information about the network layers.

deepNetworkDesigner **(video)**

https://ww2.mathworks.cn/videos/interactively-modify-a-deep-learning-network-for-transfer-learning-1547157074175.html

```
deepNetworkDesigner(net)
```



The first layer, which is the image input layer, requires input images of size 224-by-224-by-3, where 3 is the number of color channels.

```
inputSize = net.Layers(1).InputSize
```

Replace Final Layers

預訓練網絡 net 的全連接層和分類層配置為 1000 個類。 GoogLeNet 中的 loss3-classifier 和 output 這兩個層包含有關如何將網絡提取的特徵組合成類別概率、損失值和預測標籤的信息。 要重新訓練預訓練網絡以對新圖像進行分類，請將這兩層替換為適應新數據集的新層。

從訓練好的網絡中提取層圖。

```
numClasses = numel(categories(imdsTrain.Labels))
newLearnableLayer = fullyConnectedLayer(numClasses, ...
    'Name','new_fc', ...
    'WeightLearnRateFactor',10, ...
    'BiasLearnRateFactor',10);
```

```
    lgraph = replaceLayer(lgraph,'loss3-classifier',newLearnableLayer);
```

The classification layer specifies the output classes of the network. Replace the classification layer with a new one without class labels. `trainNetwork` automatically sets the output classes of the layer at training time.

```
newClassLayer = classificationLayer('Name','new_classoutput');
lgraph = replaceLayer(lgraph,'output',newClassLayer);
```

Train Network

The network requires input images of size 224-by-224-by-3, but the images in the image datastores have different sizes. Use an augmented image datastore to automatically resize the training images. Specify additional augmentation operations to perform on the training images: randomly flip the training images along the vertical axis, and randomly translate them up to 30 pixels horizontally and vertically. Data augmentation helps prevent the network from overfitting and memorizing the exact details of the training images.

```
pixelRange = [-30 30];
imageAugmenter = imageDataAugmenter( ...
    'RandXReflection',true, ...
    'RandXTranslation',pixelRange, ...
    'RandYTranslation',pixelRange);
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain, ...
    'DataAugmentation',imageAugmenter);
```

To automatically resize the validation images without performing further data augmentation, use an augmented image datastore without specifying any additional preprocessing operations.

```
augimdsValidation = augmentedImageDatastore(inputSize(1:2),imdsValidation);
```

指定訓練選項。 對於遷移學習，保留預訓練網絡早期層的特徵（遷移層權重）。 要減慢遷移層的學習速度，請將初始學習率設置為較小的值。 在上一步中，您增加了全連接層的學習率因子以加快新的最終層的學習速度。 這種學習率設置的組合導致僅在新層中快速學習而在其他層中學習較慢。 執行遷移學習時，您不需要訓練那麼多 epoch。 一個時期是整個訓練數據集的完整訓練週期。 指定小批量大小和驗證數據。 該軟件在訓練期間每次 ValidationFrequency 迭代時驗證網絡

```
options = trainingOptions('sgdm', ...
    'MiniBatchSize',10, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',1e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',3, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

訓練由傳輸層和新層組成的網絡。 默認情況下，trainNetwork 使用可用的 GPU。 這需要 Parallel Computing Toolbox™ 和支持的 GPU 設備。 有關受支持設備的信息，請參閱各版本的 GPU 支持。 否則，它使用 CPU。 您還可以使用 trainingOptions 的 “ExecutionEnvironment” 名稱-值對參數指定執行環境

```
netTransfer = trainNetwork(augimdsTrain,lgraph,options);
```

Classify Validation Images

Classify the validation images using the fine-tuned network.

```
[YPred,scores] = classify(netTransfer,augimdsValidation);
```

Display four sample validation images with their predicted labels.

```
idx = randperm(numel(imdsValidation.Files),4);
figure
for i = 1:4
    subplot(2,2,i)
    I = readimage(imdsValidation,idx(i));
    imshow(I)
    label = YPred(idx(i));
    title(string(label));
end
```

Calculate the classification accuracy on the validation set. Accuracy is the fraction of labels that the network predicts correctly.

```
YValidation = imdsValidation.Labels;
accuracy = mean(YPred == YValidation)
```

https://www.youtube.com/watch?v=v0ZwOiQhpi4

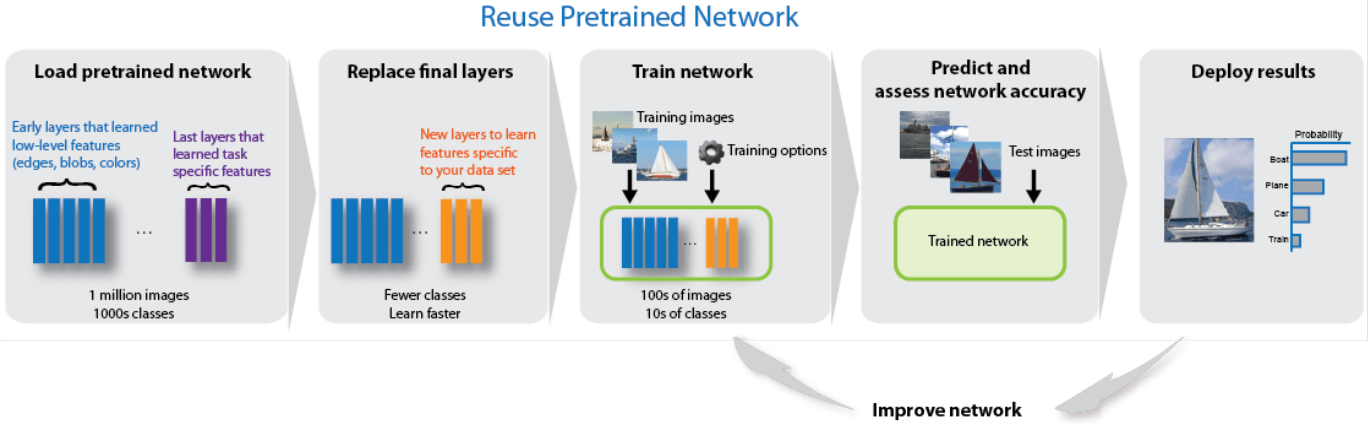
(video)

Train Deep Learning Network to Classify New Images

This example shows how to use transfer learning to retrain a convolutional neural network to classify a new set of images.

Pretrained image classification networks have been trained on over a million images and can classify images into 1000 object categories, such as keyboard, coffee mug, pencil, and many animals. The networks have learned rich feature representations for a wide range of images. The network takes an image as input, and then outputs a label for the object in the image together with the probabilities for each of the object categories.

Transfer learning is commonly used in deep learning applications. You can take a pretrained network and use it as a starting point to learn a new task. Fine-tuning a network with transfer learning is usually much faster and easier than training a network from scratch with randomly initialized weights. You can quickly transfer learned features to a new task using a smaller number of training images.



Load Data

Unzip and load the new images as an image datastore. This very small data set contains only 75 images. Divide the data into training and validation data sets. Use 70% of the images for training and 30% for validation.

```
unzip('MerchData.zip');
imds = imageDatastore('MerchData', ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');
[imdsTrain,imdsValidation] = splitEachLabel(imds,0.7);
```

Load Pretrained Network

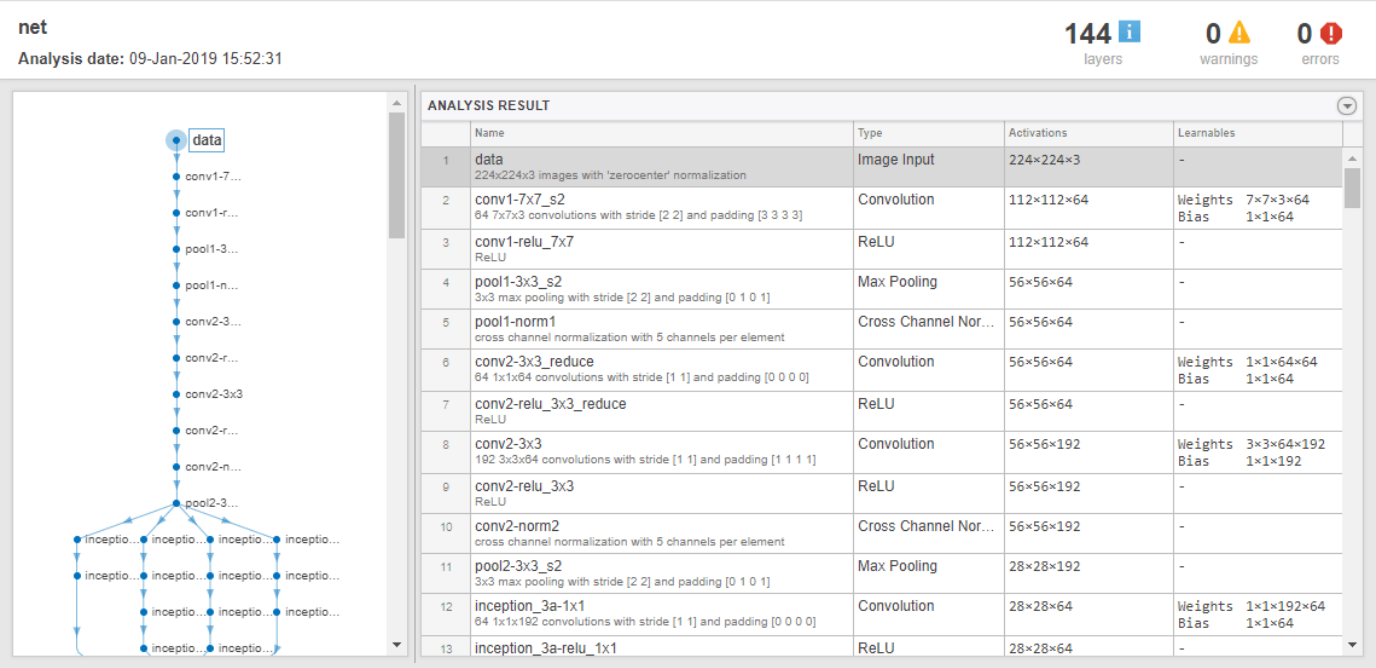
Load a pretrained GoogLeNet network. If the Deep Learning Toolbox™ Model *for GoogLeNet Network* support package is not installed, then the software provides a download link.

To try a different pretrained network, open this example in MATLAB® and select a different network. For example, you can try `squeezenet`, a network that is even faster than `googlenet`. You can run this example with other pretrained networks. For a list of all available networks, see “Load Pretrained Networks” on page 1-14.

```
net = googlenet;
```

Use `analyzeNetwork` to display an interactive visualization of the network architecture and detailed information about the network layers.

```
analyzeNetwork(net)
```



The first element of the `Layers` property of the network is the image input layer. For a GoogLeNet network, this layer requires input images of size 224-by-224-by-3, where 3 is the number of color channels. Other networks can require input images with different sizes. For example, the Xception network requires images of size 299-by-299-by-3.

```
net.Layers(1)
```

```
ans =
  ImageInputLayer with properties:

                     Name: 'data'
                InputSize: [224 224 3]

   Hyperparameters
          DataAugmentation: 'none'
             Normalization: 'zerocenter'
    NormalizationDimension: 'auto'
                     Mean: [224×224×3 single]
```

```
inputSize = net.Layers(1).InputSize;
```

Replace Final Layers

The convolutional layers of the network extract image features that the last learnable layer and the final classification layer use to classify the input image. These two layers, `'loss3-classifier'` and `'output'` in GoogLeNet, contain information on how to combine the features that the network extracts into class probabilities, a loss value, and predicted labels. To retrain a pretrained network to classify new images, replace these two layers with new layers adapted to the new data set.

Extract the layer graph from the trained network. If the network is a `SeriesNetwork` object, such as AlexNet, VGG-16, or VGG-19, then convert the list of layers in `net.Layers` to a layer graph.

```
if isa(net,'SeriesNetwork')
  lgraph = layerGraph(net.Layers);
else
  lgraph = layerGraph(net);
end
```

Find the names of the two layers to replace. You can do this manually or you can use the supporting function findLayersToReplace to find these layers automatically.

```
[learnableLayer,classLayer] = findLayersToReplace(lgraph);
[learnableLayer,classLayer]
```

```
ans =
  1×2 Layer array with layers:

     1   'loss3-classifier'   Fully Connected         1000 fully connected layer
     2   'output'             Classification Output   crossentropyex with 'tench' and 999 other c
```

In most networks, the last layer with learnable weights is a fully connected layer. Replace this fully connected layer with a new fully connected layer with the number of outputs equal to the number of classes in the new data set (5, in this example). In some networks, such as SqueezeNet, the last learnable layer is a 1-by-1 convolutional layer instead. In this case, replace the convolutional layer with a new convolutional layer with the number of filters equal to the number of classes. To learn faster in the new layer than in the transferred layers, increase the learning rate factors of the layer.

```
numClasses = numel(categories(imdsTrain.Labels));

if isa(learnableLayer,'nnet.cnn.layer.FullyConnectedLayer')
    newLearnableLayer = fullyConnectedLayer(numClasses, ...
        'Name','new_fc', ...
        'WeightLearnRateFactor',10, ...
        'BiasLearnRateFactor',10);

elseif isa(learnableLayer,'nnet.cnn.layer.Convolution2DLayer')
    newLearnableLayer = convolution2dLayer(1,numClasses, ...
        'Name','new_conv', ...
        'WeightLearnRateFactor',10, ...
        'BiasLearnRateFactor',10);
end

lgraph = replaceLayer(lgraph,learnableLayer.Name,newLearnableLayer);
```

The classification layer specifies the output classes of the network. Replace the classification layer with a new one without class labels. `trainNetwork` automatically sets the output classes of the layer at training time.

```
newClassLayer = classificationLayer('Name','new_classoutput');
lgraph = replaceLayer(lgraph,classLayer.Name,newClassLayer);
```

To check that the new layers are connected correctly, plot the new layer graph and zoom in on the last layers of the network.

```
figure('Units','normalized','Position',[0.3 0.3 0.4 0.4]);
plot(lgraph)
ylim([0,10])
```

輸入預訓練網路：**AlexNet**

```
net = alexnet;
analyzeNetwork(net) %查看網路詳情
inputSize = net.Layers(1).InputSize %網路的輸入大小
```

```
inputSize = 1×3
   227   227     3
```

更改預訓練網路內容

```
layersTransfer = net.Layers(1:end-3); %抓出 AlexNet 內的卷積層部分
numClasses = numel(categories(imdsTrain.Labels)) %數據集的類別數
```

```
numClasses = 5
```

```
layers = [
    layersTransfer

fullyConnectedLayer(numClasses,'WeightLearnRateFactor',20,'BiasLearnRat
eFactor',20)
    softmaxLayer
    classificationLayer];
```

凍結的層

net=alexnet;

% Extract the layer graph from the trained network. If the network is a SeriesNetwork object, such as
% AlexNet, VGG-16, or VGG-19, then convert the list of layers in net.Layers to a layer graph.

```
if isa(net,'SeriesNetwork')
lgraph = layerGraph(net.Layers);
else
lgraph = layerGraph(net);
end
```

網絡現在已準備好在新的圖像集上進行重新訓練。 或者，您可以“凍結”通過將這些層中的學習率設置為零來調整網絡中較早層的權重。 期間

訓練時，trainNetwork 不更新凍結層的參數。 因為梯度 不需要計算凍結層的數量，凍結許多初始層的權重可以顯著加快網絡訓練。 如果新的數據集很小，那麼凍結較早的網絡圖層還可以防止這些圖層過度擬合新數據集。

提取層圖的層和連接並選擇要凍結的層。 在 GoogLeNet，前 10 層構成了網絡的初始“主幹”。 使用輔助功能 freezeWeights 將前 10 層的學習率設置為零。 使用輔助功能 createLgraphUsingConnections 以原始順序重新連接所有層。 新的圖層圖包含相同的層，但較早層的學習率設置為零。

```
layers = lgraph.Layers;
connections = lgraph.Connections;
layers(1:10) = freezeWeights(layers(1:10));
lgraph = createLgraphUsingConnections(layers,connections);
```

Create Deep Learning Network Architecture

deepNetworkDesigner to build alexnet and export to code

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

```
Number of layers: 25
Number of connections: 24
```

Create Array of Layers

```
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])
    groupedConvolution2dLayer([5
5],128,2,"Name","conv2","BiasLearnRateFactor",2,"Padding",[2 2 2 2])
    reluLayer("Name","relu2")
    crossChannelNormalizationLayer(5,"Name","norm2","K",1)
    maxPooling2dLayer([3 3],"Name","pool2","Stride",[2 2])
    convolution2dLayer([3
3],384,"Name","conv3","BiasLearnRateFactor",2,"Padding",[1 1 1 1])
    reluLayer("Name","relu3")
    groupedConvolution2dLayer([3
3],192,2,"Name","conv4","BiasLearnRateFactor",2,"Padding",[1 1 1 1])
    reluLayer("Name","relu4")
    groupedConvolution2dLayer([3
3],128,2,"Name","conv5","BiasLearnRateFactor",2,"Padding",[1 1 1 1])
```

```
    reluLayer("Name","relu5")
    maxPooling2dLayer([3 3],"Name","pool5","Stride",[2 2])
    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","BiasLearnRateFactor",2)
    softmaxLayer("Name","prob")
    classificationLayer("Name","output")];
```

Plot Layers

```
for i=1:length(layers)
    fprintf('The %d layer with name: %s. \n',i,layers(i).Name);
end

lgraph = layerGraph(layers);
layers = lgraph.Layers;
connections = lgraph.Connections;
layers(1:10) = freezeWeights(layers(1:10));
lgraph = createLgraphUsingConnections(layers,connections);
plot(lgraph);
```

Network	Depth	Size	Parameters (Millions)	Image Input Size
squeezenet	18	5.2 MB	1.24	227-by-227
googlenet	22	27 MB	7.0	224-by-224
inceptionv3	48	89 MB	23.9	299-by-299
densenet201	201	77 MB	20.0	224-by-224
mobilenetv2	53	13 MB	3.5	224-by-224
resnet18	18	44 MB	11.7	224-by-224
resnet50	50	96 MB	25.6	224-by-224
resnet101	101	167 MB	44.6	224-by-224
xception	71	85 MB	22.9	299-by-299
inceptionresnetv2	164	209 MB	55.9	299-by-299
shufflenet	50	5.4 MB	1.4	224-by-224
nasnetmobile	*	20 MB	5.3	224-by-224
nasnetlarge	*	332 MB	88.9	331-by-331
darknet19	19	78 MB	20.8	256-by-256
darknet53	53	155 MB	41.6	256-by-256
efficientnetb0	82	20 MB	5.3	224-by-224
alexnet	8	227 MB	61.0	227-by-227
vgg16	16	515 MB	138	224-by-224
vgg19	19	535 MB	144	224-by-224