Case studies

Classifying the CIFAR-10 dataset using Convolutional Neural Networks

```
%% Classifying the CIFAR-10 dataset using Convolutional Neural Networks
% This example shows how to train a Convolutional Neural Network (CNN) from
% scratch using the dataset CIFAR10.
%
% Data Credit: Krizhevsky, A., & Hinton, G. (2009). Learning multiple
% layers of features from tiny images.

% Copyright 2016 The MathWorks, Inc.

%% Download the CIFAR-10 dataset
if ~exist('cifar-10-batches-mat','dir')
    cifar10Dataset = 'cifar-10-matlab';
    disp('Downloading 174MB CIFAR-10 dataset...');
    websave([cifar10Dataset,'.tar.gz'],...
        ['https://www.cs.toronto.edu/~kriz/',cifar10Dataset,'.tar.gz']);
    gunzip([cifar10Dataset,'.tar.gz'])
    delete([cifar10Dataset,'.tar.gz'])
    untar([cifar10Dataset,'.tar'])
    delete([cifar10Dataset,'.tar'])
end

%% Prepare the CIFAR-10 dataset
if ~exist('cifar10Train','dir')
    disp('Saving the Images in folders. This might take some time...');
    saveCIFAR10AsFolderOfImages('cifar-10-batches-mat', pwd, true);
end

%% Load image CIFAR-10 Training dataset (50000 32x32 colour images in 10 classes)
imsetTrain = imageSet('cifar10Train','recursive');

%% Display Sampling of Image Data
numClasses = size(imsetTrain,2);
imagesPerClass = 10;
imagesInMontage = cell(imagesPerClass,numClasses);
for i = 1:size(imagesInMontage,2)
    imagesInMontage(:,i) = ...
        imsetTrain(i).ImageLocation(randi(imsetTrain(i).Count, 1, ...
        imagesPerClass));
end
```

Step 1
Download
Matlab example

Step 2
Turn off use of gpuArray at
lines 66, 70 and 73

```
63    %% Define a CNN architecture
64    conv1 = convolution2dLayer(5,32,'Padding',2,...
65                               'BiasLearnRateFactor',2);
66    % conv1.Weights = gpuArray(single(randn([5 5 3 32])*0.0001));
67    conv1.Weights = single(randn([5 5 3 32])*0.0001);
68
69    fc1 = fullyConnectedLayer(64,'BiasLearnRateFactor',2);
70    % fc1.Weights = gpuArray(single(randn([64 576])*0.1));
71    fc1.Weights = single(randn([64 576])*0.1);
72    fc2 = fullyConnectedLayer(10,'BiasLearnRateFactor',2);
73    % fc2.Weights = gpuArray(single(randn([10 64])*0.1));
74    fc2.Weights = single(randn([10 64])*0.1);
75
```

Step 3
Insert lines 67, 71 and 74

Step 4
Dlowload

```
function saveCIFAR10AsFolderOfImages(inputPath, outputPath, varargin)
% saveCIFAR10AsFolderOfImages   Save the CIFAR-10 dataset as a folder of images
%   saveCIFAR10AsFolderOfImages(inputPath, outputPath) takes the CIFAR-10
%   dataset located at inputPath and saves it as a folder of images to the
%   directory outputPath. If inputPath or outputPath is an empty string, it
%   is assumed that the current folder should be used.
%
%   saveCIFAR10AsFolderOfImages(..., labelDirectories) will save the
%   CIFAR-10 data so that instances with the same label will be saved to
%   sub-directories with the name of that label.

% Check input directories are valid
if(~isempty(inputPath))
    assert(exist(inputPath,'dir') == 7);
end
if(~isempty(outputPath))
    assert(exist(outputPath,'dir') == 7);
end

% Check if we want to save each set with the same labels to its own
% directory.
if(isempty(varargin))
    labelDirectories = false;
else
    assert(nargin == 3);
    labelDirectories = varargin{1};
end

% Set names for directories
trainDirectoryName = 'cifar10Train';
testDirectoryName = 'cifar10Test';
```

Editor - /Users/apple/Documents/MATLAB/Examples/R2019a/nnet/TrainResidualNetworkOnCIFAR10Example/matlab_cifar_10.m

+9 | cnn_mnist.m | cnn_train.m | vl_nnconv.cpp | vl_nnconv.cu | TrainResidualNetworkOnCIFAR10Example.mlx | matlab_cifar_10.m

```
137     YTest = classify(net, XTest);
138
139     % Alternative way using imageDataStore
140     % imdsTest = imageDatastore(fullfile(pwd, 'cifar10Test'),...
141     %     'IncludeSubfolders',true,'LabelSource','foldernames');
142     % YTest = classify(net, imdsTest);
143
144     % Calculate the accuracy.
145 -   accuracy = sum(YTest == TTest)/numel(TTest)
```

Command Window

```
115 title('First Layer Weights');
Elapsed time is 12.206364 seconds.

accuracy =

    0.7302
```

Accuracy for testing

AlexNet

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

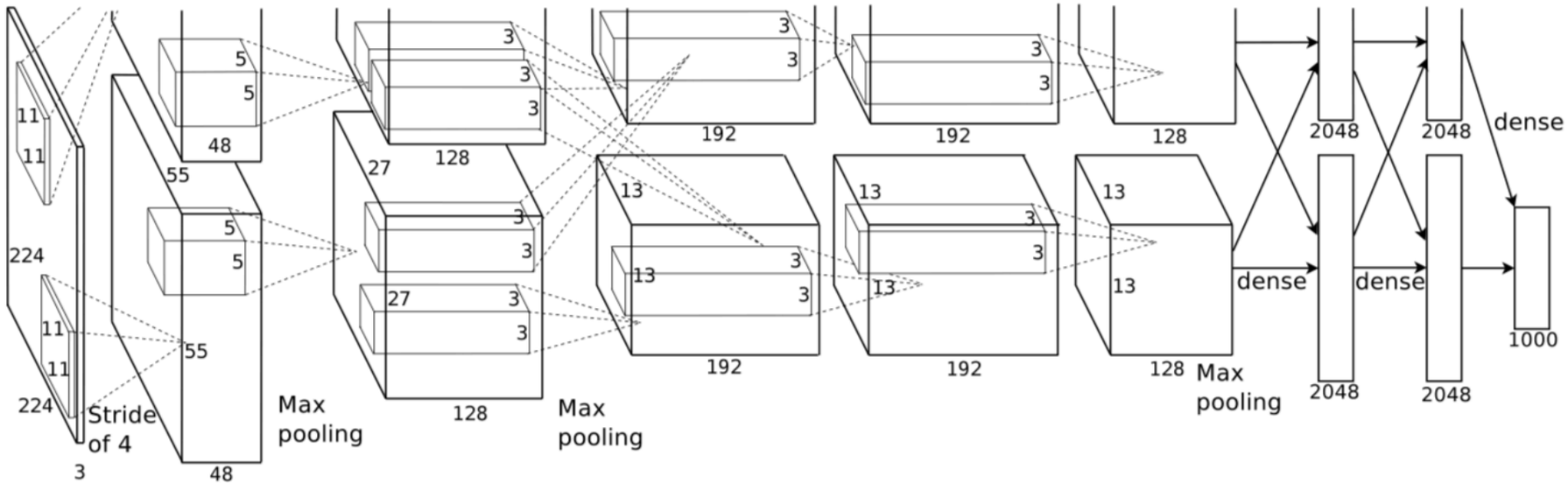


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Step 1

Command Window

```
>> net = alexnet

net =

  SeriesNetwork with properties:

    Layers: [25×1 nnet.cnn.layer.Layer]

fx >>
```

APPS EDITOR PUBLISH VIEW

leNet

FAVORITES

Curve Fitting
Optimization
PID Tuner
Modbus Explorer
System Identification
Signal Analyzer
Wireless Waveform G...
Image Acquisition
Instrument Control
SimBiology
MATLAB Coder
Application Compiler
Image Region Analyzer

Step 2

MACHINE LEARNING AND DEEP LEARNING

Classification Learner
Deep Network Designer
Neural Net Clustering
Neural Net Fitting
Neural Net Pattern Rec...
Neural Net Time Series
Regression Learner

MATH, STATISTICS AND OPTIMIZATION

Curve Fitting
Distribution Fitter
Optimization
PDE Modeler

CONTROL SYSTEM DESIGN AND ANALYSIS

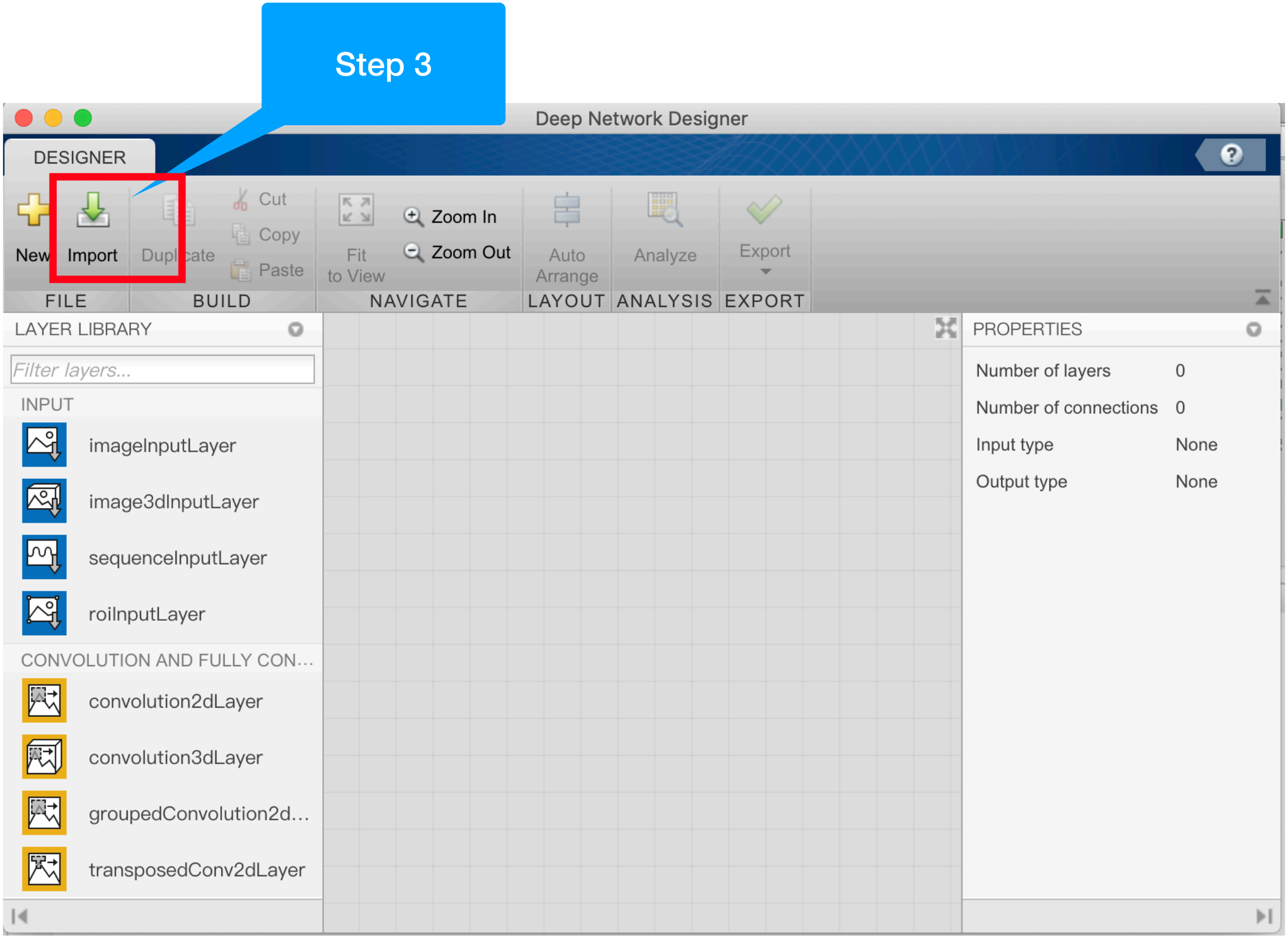
Control System Desi...
Control System Tuner
Diagnostic Feature Des...
Fuzzy Logic Designer
Linear System Analyzer
Model Reducer
MPC Designer
Neuro-Fuzzy Designer
PID Tuner
SLAM Map Builder
System Identification

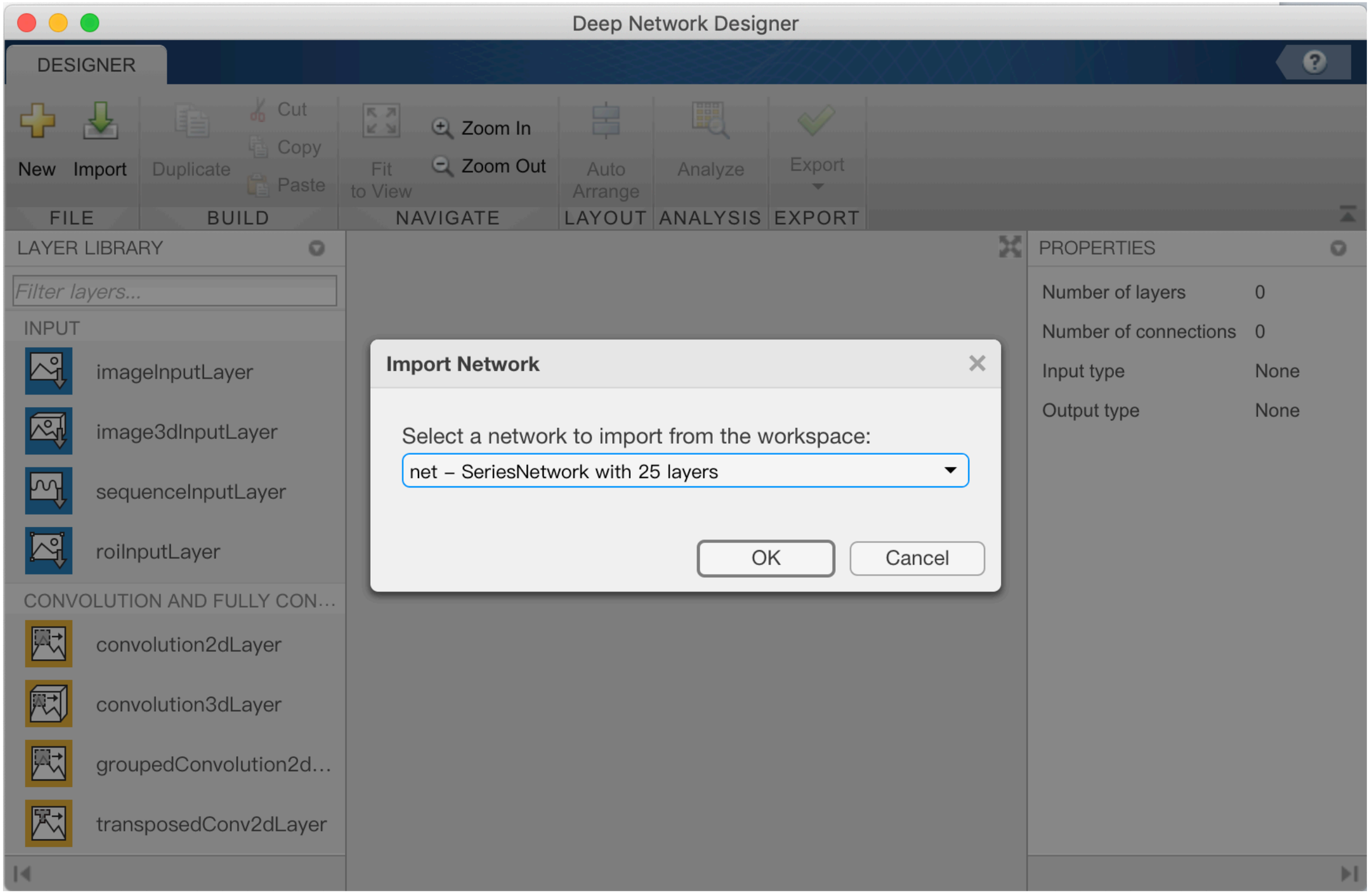
AUTOMOTIVE

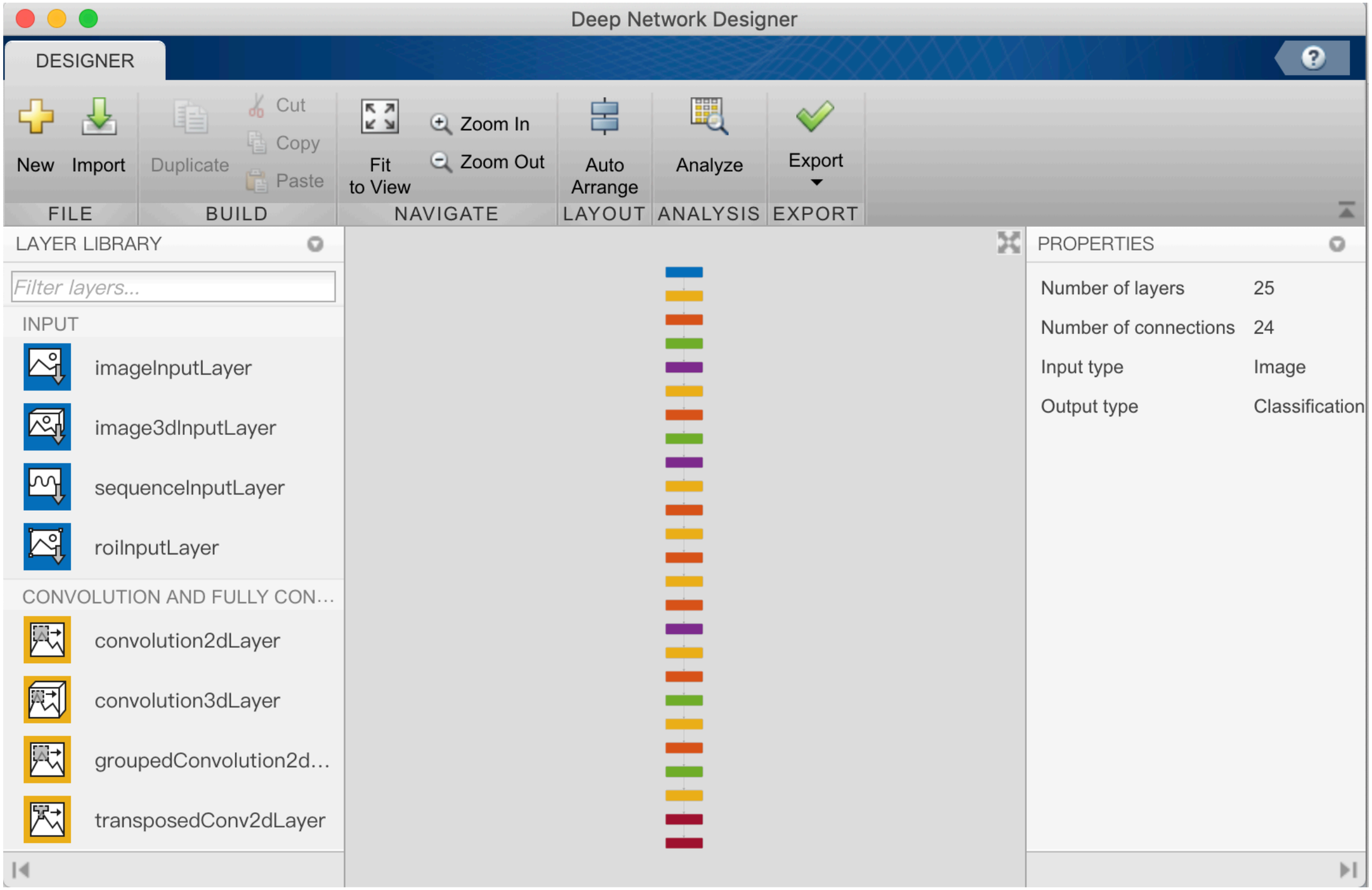
Driving Scenario De...
Ground Truth Labeler

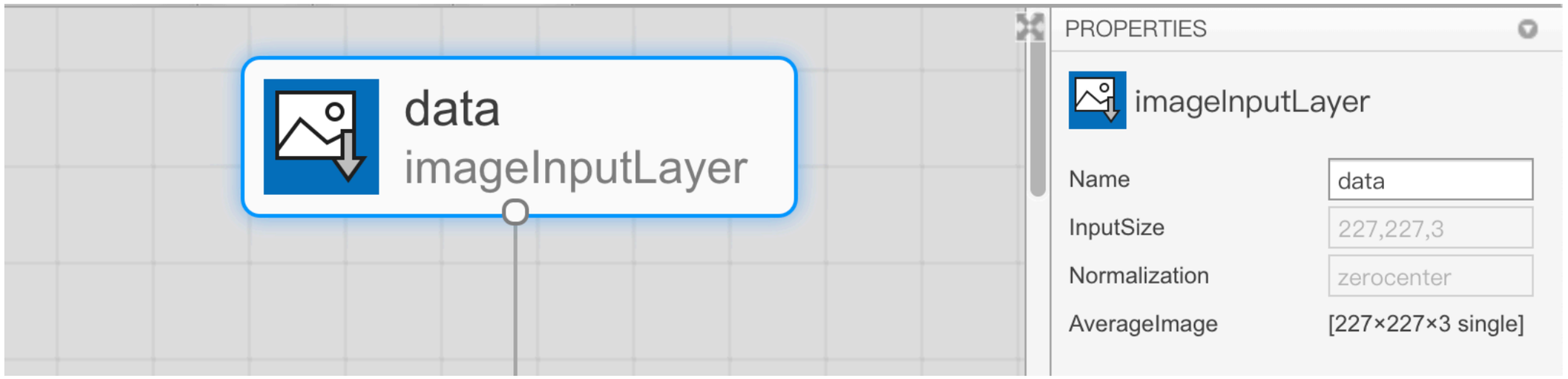
SIGNAL PROCESSING AND COMMUNICATIONS

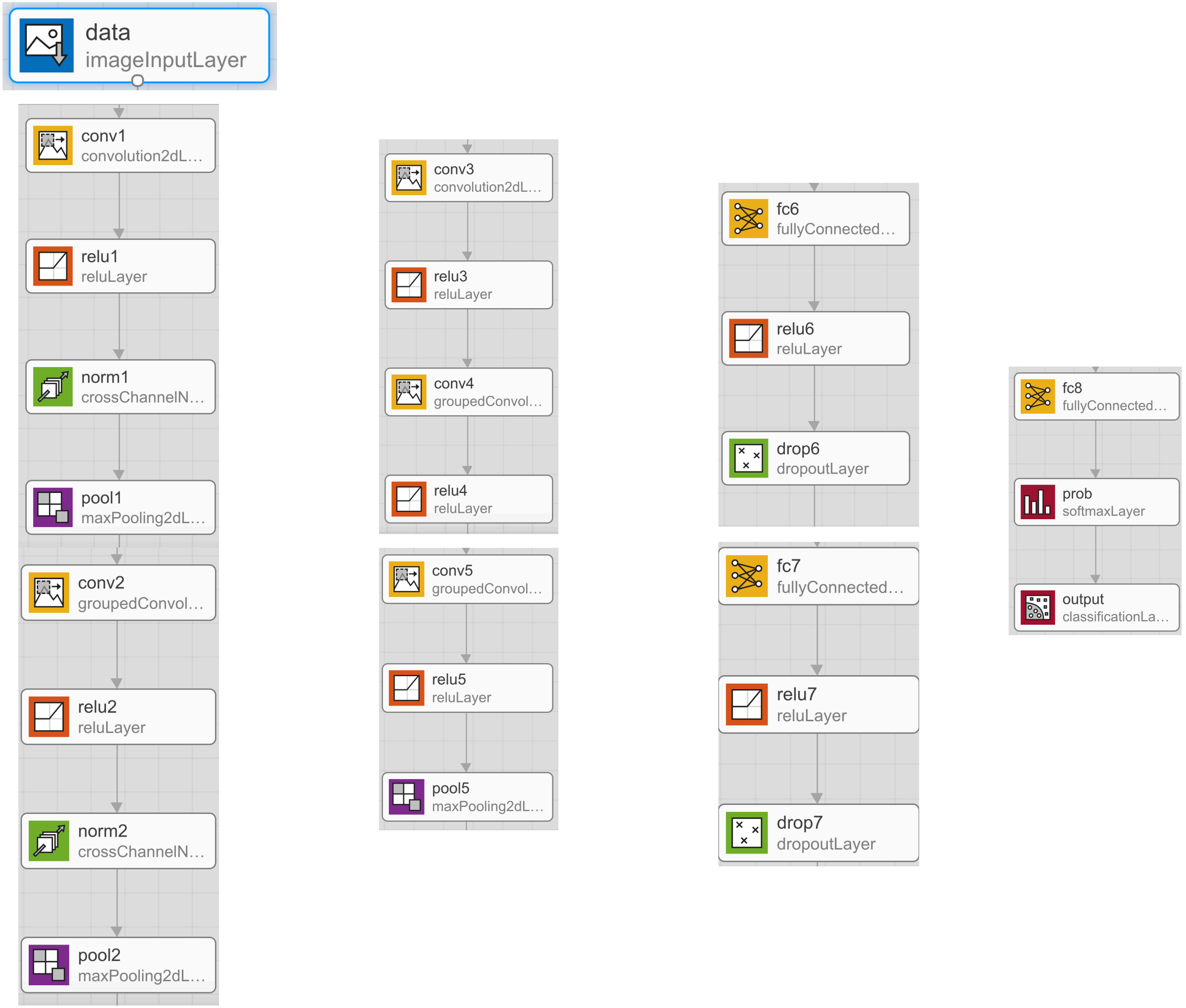
Antenna Designer
Audio Labeler
Bit Error Rate Analysis
Eye Diagram Scope
Filter Builder
Filter Designer
Impulse Response M...
LTE Throughpu...
LTE Waveform Generator
Radar Equation Ca...
Radar Waveform A...
RF Budget Analyzer


Step 3
Deep Network Designer
DESIGNER
New
Import
Duplicate
Cut
Copy
Paste
Fit to View
Zoom In
Zoom Out
Auto Arrange
Analyze
Export
FILE
BUILD
NAVIGATE
LAYOUT
ANALYSIS
EXPORT
LAYER LIBRARY
Filter layers...
INPUT
imageInputLayer
image3dInputLayer
sequenceInputLayer
roiInputLayer
CONVOLUTION AND FULLY CON...
convolution2dLayer
convolution3dLayer
groupedConvolution2d...
transposedConv2dLayer
PROPERTIES
Number of layers 0
Number of connections 0
Input type None
Output type None


Deep Network Designer
DESIGNER
New
Import
Duplicate
Cut
Copy
Paste
Fit to View
Zoom In
Zoom Out
Auto Arrange
Analyze
Export
FILE
BUILD
NAVIGATE
LAYOUT
ANALYSIS
EXPORT
LAYER LIBRARY
Filter layers...
INPUT
imageInputLayer
image3dInputLayer
sequenceInputLayer
roiInputLayer
CONVOLUTION AND FULLY CON...
convolution2dLayer
convolution3dLayer
groupedConvolution2d...
transposedConv2dLayer
Import Network
Select a network to import from the workspace:
net – SeriesNetwork with 25 layers
OK
Cancel
PROPERTIES
Number of layers 0
Number of connections 0
Input type None
Output type None


Deep Network Designer
DESIGNER
New
Import
Duplicate
Cut
Copy
Paste
Fit to View
Zoom In
Zoom Out
Auto Arrange
Analyze
Export
FILE
BUILD
NAVIGATE
LAYOUT
ANALYSIS
EXPORT
LAYER LIBRARY
Filter layers...
INPUT
imageInputLayer
image3dInputLayer
sequenceInputLayer
roiInputLayer
CONVOLUTION AND FULLY CON...
convolution2dLayer
convolution3dLayer
groupedConvolution2d...
transposedConv2dLayer
PROPERTIES
Number of layers 25
Number of connections 24
Input type Image
Output type Classification


data
imageInputLayer
PROPERTIES
imageInputLayer
Name
data
InputSize
227,227,3
Normalization
zerocenter
AverageImage
[227×227×3 single]


data
imageInputLayer
conv1
convolution2dL…
relu1
reluLayer
norm1
crossChannelN…
pool1
maxPooling2dL…
conv2
groupedConvol…
relu2
reluLayer
norm2
crossChannelN…
pool2
maxPooling2dL…
conv3
convolution2dL…
relu3
reluLayer
conv4
groupedConvol…
relu4
reluLayer
conv5
groupedConvol…
relu5
reluLayer
pool5
maxPooling2dL…
fc6
fullyConnected…
relu6
reluLayer
drop6
dropoutLayer
fc7
fullyConnected…
relu7
reluLayer
drop7
dropoutLayer
fc8
fullyConnected…
prob
softmaxLayer
output
classificationLa…

LeNet

Editor - /Users/apple/Desktop/Jiann-Ming Wu/code2019/MatConvNet/matconvnet-1.0-beta25/examples/mni...

+5 | cnn_mnist_experiments.m | vl_simplenn.m | cnn_mnist_init.m | matlab_cifar_10.m | +

```
 9
10 -    f=1/100 ;
11 -    net.layers = {} ;
12 -    net.layers{end+1} = struct('type', 'conv', ...
13                               'weights', {{f*randn(5,5,1
14                               'stride', 1, ...
15                               'pad', 0) ;
16 -    net.layers{end+1} = struct('type', 'pool'
```

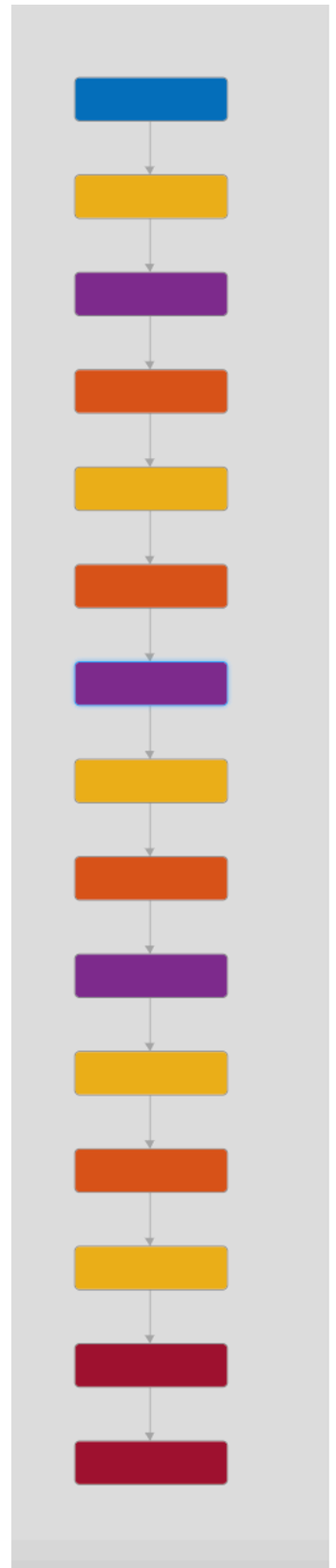
Command Window

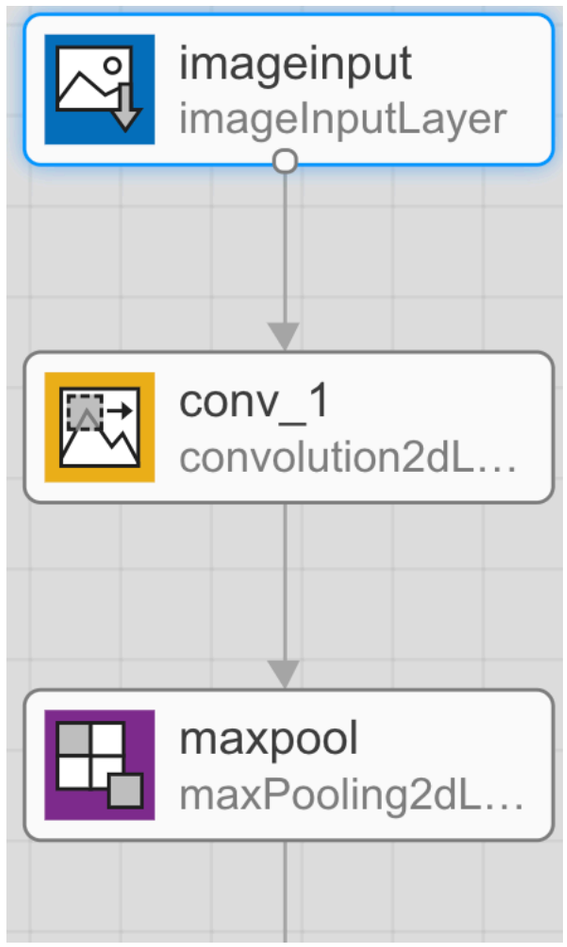
```
>> load('layers_3.mat')
>>
```

Step 1

Workspace

Name	Value
layers_3	15x1 Layer





PROPERTIES

imageInputLayer

Name	imageinput
InputSize	32,32,3
Normalization	zerocenter
AverageImage	[ ]

PROPERTIES

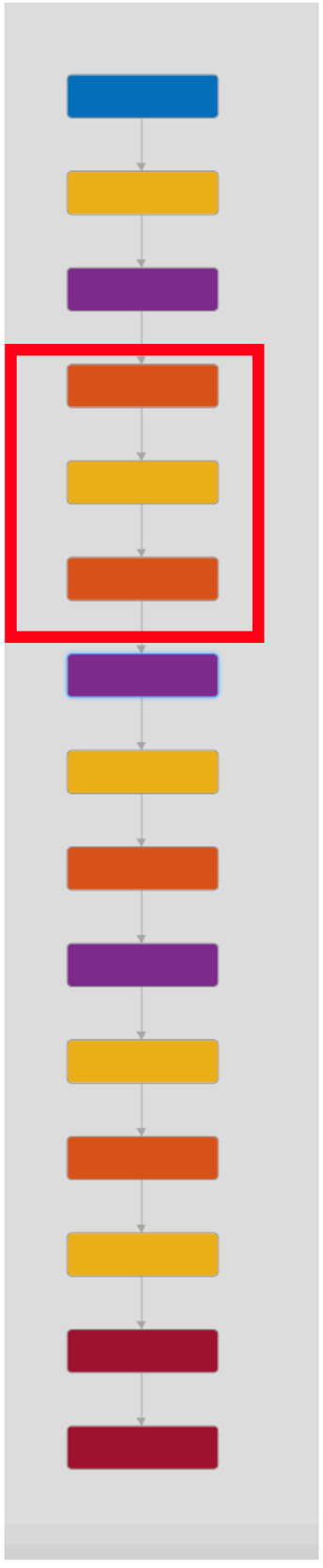
maxPooling2dLayer

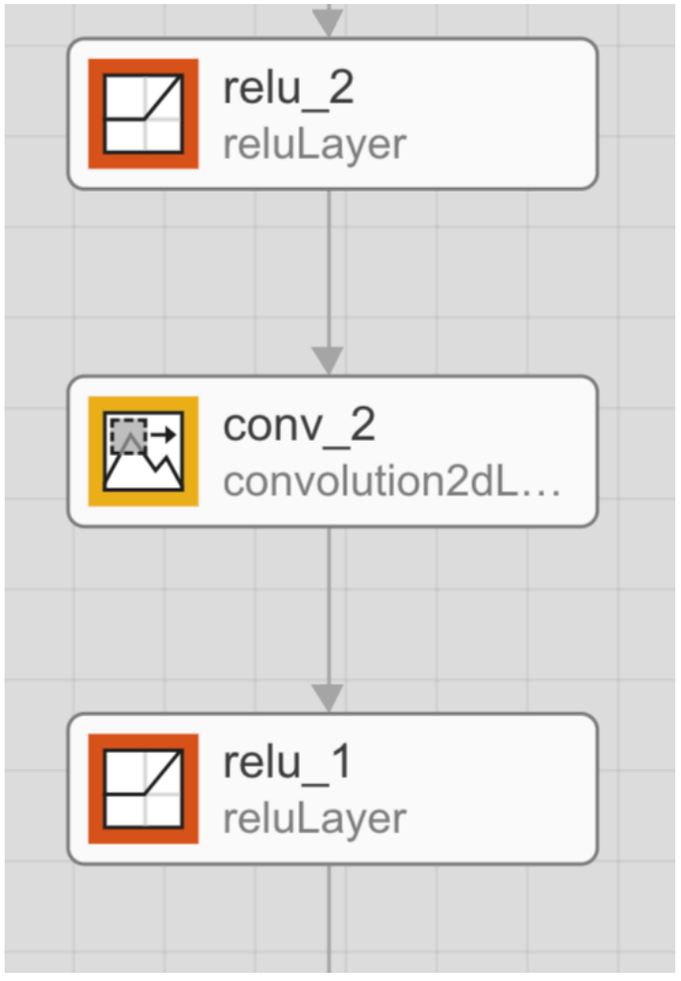
Name	maxpool
PoolSize	3,3
Stride	2,2
Padding	0,1,0,1

PROPERTIES

convolution2dLayer

Name	conv_1
FilterSize	5,5
NumFilters	32
Stride	1,1
DilationFactor	1,1
Padding	same
Weights	[ ]
Bias	[ ]
WeightLearnRateFactor	1
WeightL2Factor	1
BiasLearnRateFactor	1
BiasL2Factor	0
WeightsInitializer	glorot
BiasInitializer	zeros





PROPERTIES

reluLayer

Name	relu_2

PROPERTIES

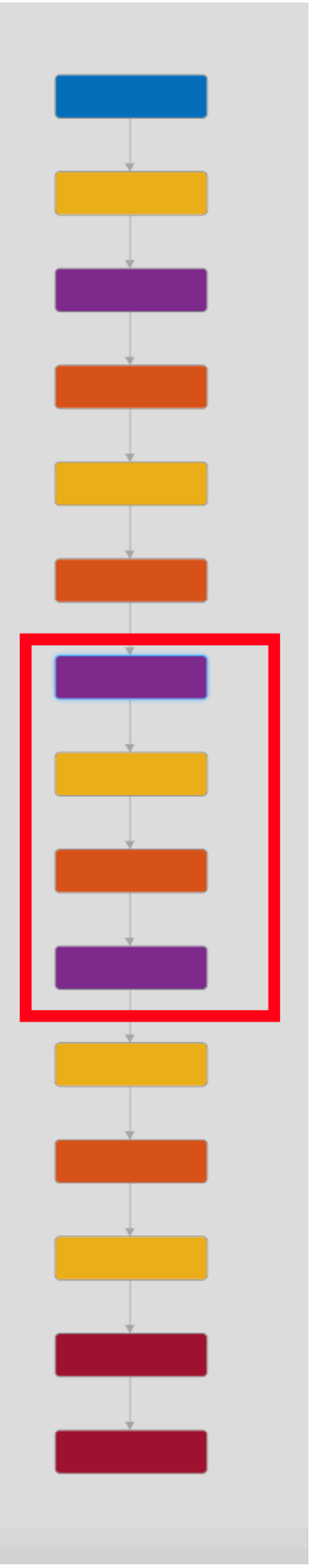
convolution2dLayer

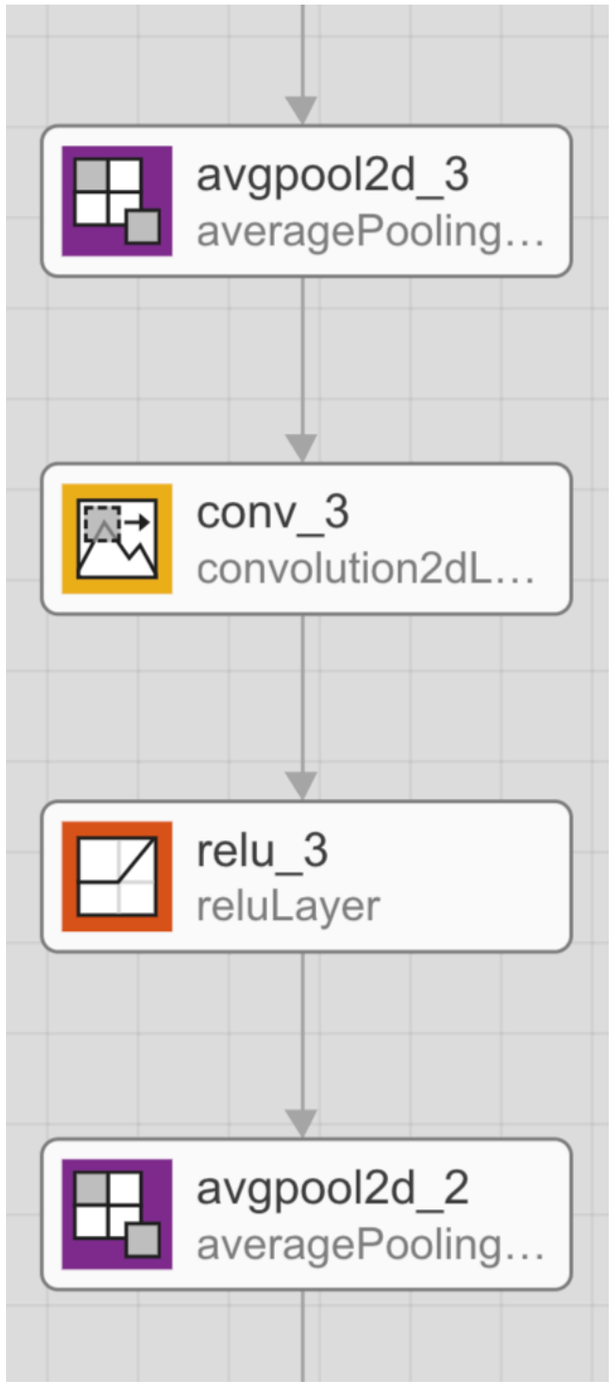
Name	conv_2
FilterSize	5,5
NumFilters	32
Stride	1,1
DilationFactor	1,1
Padding	same
Weights	[ ]
Bias	[ ]
WeightLearnRateFactor	1
WeightL2Factor	1
BiasLearnRateFactor	1
BiasL2Factor	0
WeightsInitializer	glorot
BiasInitializer	zeros

PROPERTIES

reluLayer

Name	relu_1





PROPERTIES

averagePooling2dLayer

Name	avgpool2d_3
PoolSize	3,3
Stride	2,2
Padding	0,1,0,1

PROPERTIES

convolution2dLayer

Name	conv_3
FilterSize	5,5
NumFilters	32
Stride	1,1
DilationFactor	1,1
Padding	same
Weights	[ ]
Bias	[ ]
WeightLearnRateFactor	1
WeightL2Factor	1
BiasLearnRateFactor	1
BiasL2Factor	0
WeightsInitializer	glorot
BiasInitializer	zeros

PROPERTIES

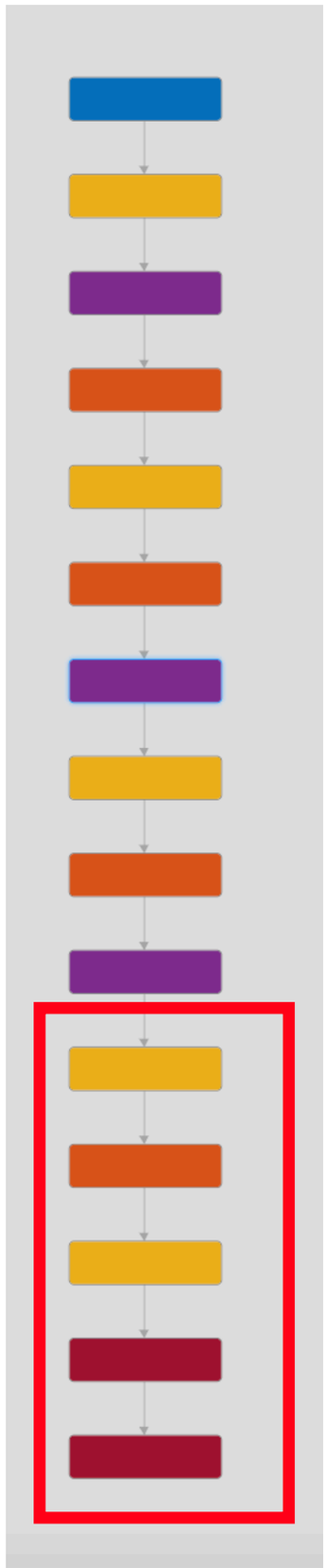
reluLayer

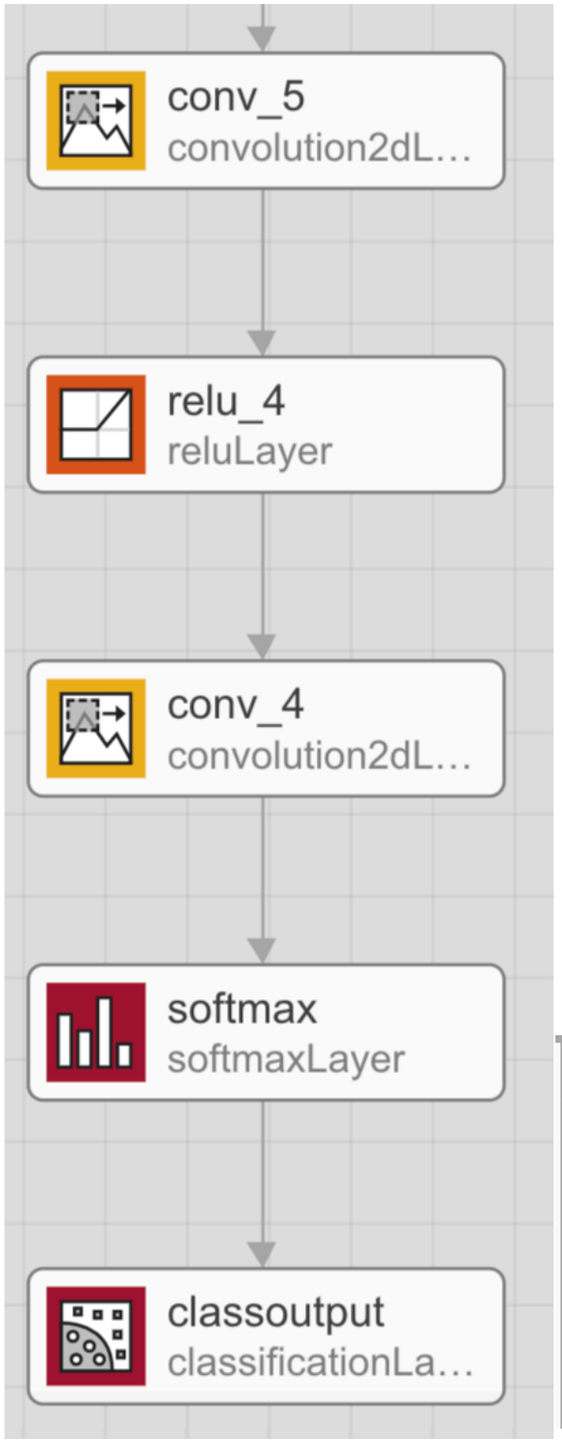
Name	relu_3

PROPERTIES

averagePooling2dLayer

Name	avgpool2d_2
PoolSize	3,3
Stride	2,2
Padding	0,1,0,1





PROPERTIES

convolution2dLayer

Name	conv_5
FilterSize	4,4
NumFilters	64
Stride	1,1
DilationFactor	1,1
Padding	0,0,0,0
Weights	[ ]
Bias	[ ]
WeightLearnRateFactor	1
WeightL2Factor	1
BiasLearnRateFactor	1
BiasL2Factor	0
WeightsInitializer	glorot
BiasInitializer	zeros

PROPERTIES

convolution2dLayer

Name	conv_4
FilterSize	1,1
NumFilters	10
Stride	1,1
DilationFactor	1,1
Padding	0,0,0,0
Weights	[ ]
Bias	[ ]
WeightLearnRateFactor	1
WeightL2Factor	1
BiasLearnRateFactor	1
BiasL2Factor	0
WeightsInitializer	glorot
BiasInitializer	zeros

PROPERTIES

reluLayer

Name	relu_4

PROPERTIES

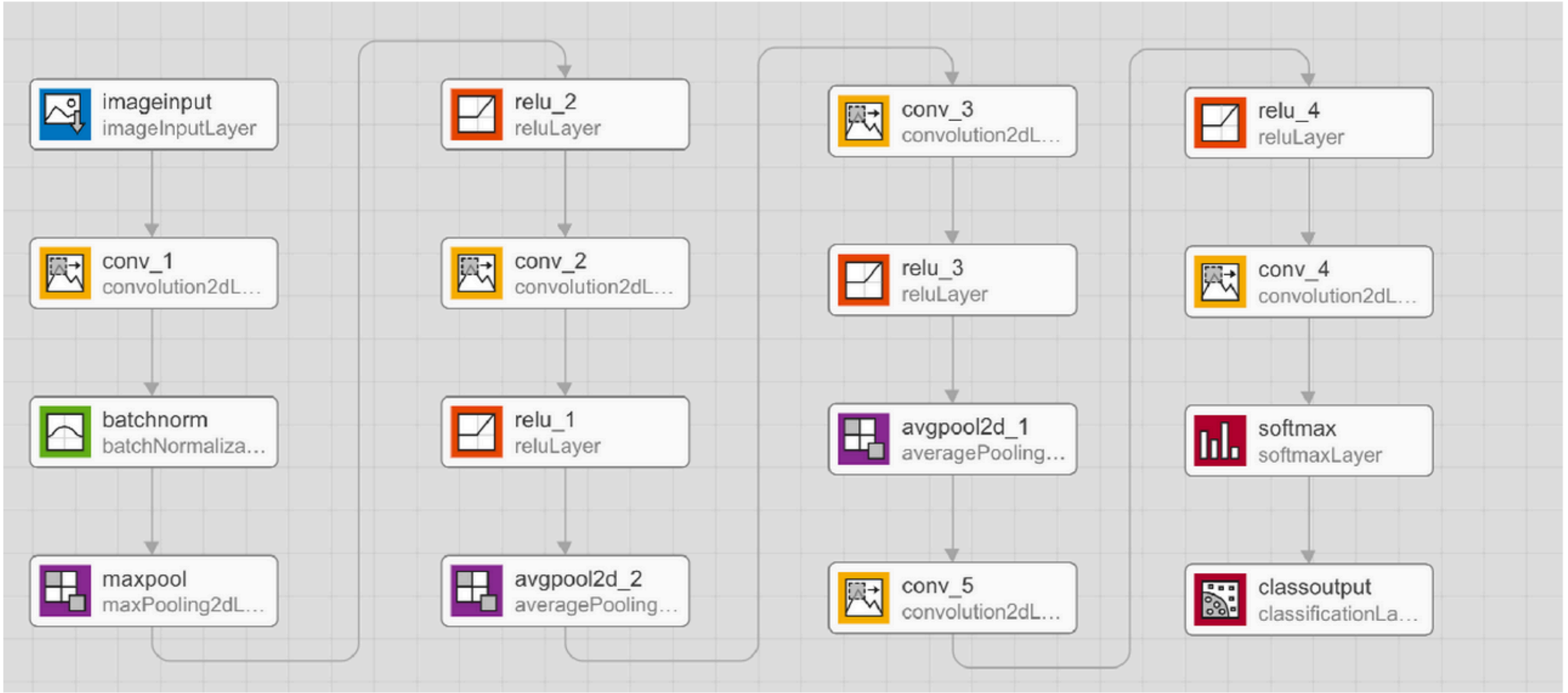
softmaxLayer

Name	softmax

PROPERTIES

classificationLayer

Name	classoutput
Classes	auto
OutputSize	auto
LossFunction	crossentropyex


imageinput
imageInputLayer
conv_1
convolution2dL…
batchnorm
batchNormaliza…
maxpool
maxPooling2dL…
relu_2
reluLayer
conv_2
convolution2dL…
relu_1
reluLayer
avgpool2d_2
averagePooling…
conv_3
convolution2dL…
relu_3
reluLayer
avgpool2d_1
averagePooling…
conv_5
convolution2dL…
relu_4
reluLayer
conv_4
convolution2dL…
softmax
softmaxLayer
classoutput
classificationLa…

```
load layers_3.mat
load data.mat
load labels.mat

XTrain=data;
YTrain=labels;
YTrain=categorical(YTrain');
idx = randperm(size(XTrain,4),10000);
XValidation = XTrain(:,:,:,idx(1:9000));
Xtest = XTrain(:,:,:,idx(9001:10000));
XTrain(:,:,:,idx) = [];
YValidation = YTrain(idx(1:9000));
Ytest = YTrain(idx(9001:10000));
YTrain(idx) = [];
```

TRAINING

```
imageSize = [32 32 3];
augimds = augmentedImageDatastore(imageSize,XTrain,YTrain);

options =  trainingOptions('sgdm', ...
    'MaxEpochs',15, ...
    'Shuffle','every-epoch', ...
    'Plots','training-progress', ...
    'Verbose',false, ...
    'ValidationData',{XValidation,YValidation}, ...
    'ExecutionEnvironment','cpu');

net = trainNetwork(augimds,layers_3,options);
 YPred = classify(net,Xtest);
 correct1 = sum(YPred==Ytest)/1000
```

```
load layers_3.mat
load data.mat
load labels.mat

XTrain=data;
YTrain=labels;
YTrain=categorical(YTrain');
idx = randperm(size(XTrain,4),10000);
XValidation = XTrain(:,:,:,idx(1:9000));
Xtest = XTrain(:,:,:,idx(9001:10000));
XTrain(:,:,:,idx) = [];
YValidation = YTrain(idx(1:9000));
Ytest = YTrain(idx(9001:10000));
YTrain(idx) = [];
```

```
imageSize = [32 32 3];
augimds = augmentedImageDatastore(imageSize,XTrain,YTrain);

options =  trainingOptions('sgdm', ...
    'MaxEpochs',15, ...
    'Shuffle','every-epoch', ...
    'Plots','training-progress', ...
    'Verbose',false, ...
    'ValidationData',{XValidation,YValidation}, ...
    'ExecutionEnvironment','cpu');

net = trainNetwork(augimds,layers_3,options);
 YPred = classify(net,Xtest);
 correct1 = sum(YPred==Ytest)/1000
```

TESTING