DEEP NEURAL NETWORKS -> MATLAB

1. NN SETUP

☐ Create Simple Deep Learning Network for Classification- MATLAB & Simulink Example- MathWorks Australia

1. Load and explore image data.
2. Define the network architecture.
3. Specify training options.
4. Train the network.
5. Predict the labels of new data and calculate the classification accuracy

1. Load and explore image data.
   Note: imageDatastore automatically labels the images based on folder names and stores the data as an ImageDatastore object.
   Therefore best to import whole folder with subfolder (e.g. folder with top 5, 10 characters etc.)

   To load:

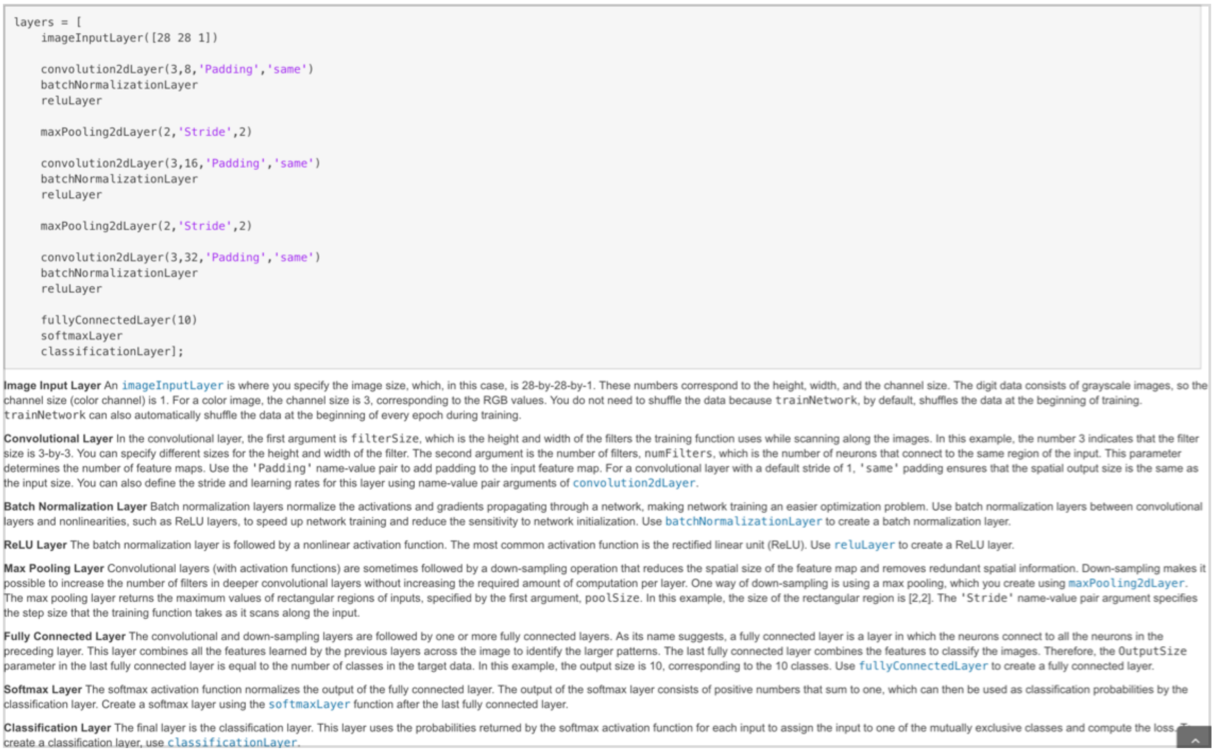
   ```
   imds = imageDatastore(digitDatasetPath, ...
       'IncludeSubfolders',true,'LabelSource','foldernames');
   ```

   OR
   Load in via Image Browser (Image Processing Toolbox)

   Display some images in the Datastore:

   ```
   figure;
   perm = randperm(10000,20);
   for i = 1:20
       subplot(4,5,i);
       imshow(imds.Files{perm(i)});
   end
   ```

2. **CNN elements Example**



```
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];
```

Image Input Layer An `imageInputLayer` is where you specify the image size, which, in this case, is 28-by-28-by-1. These numbers correspond to the height, width, and the channel size. The digit data consists of grayscale images, so the channel size (color channel) is 1. For a color image, the channel size is 3, corresponding to the RGB values. You do not need to shuffle the data because `trainNetwork`, by default, shuffles the data at the beginning of training. `trainNetwork` can also automatically shuffle the data at the beginning of every epoch during training.

Convolutional Layer In the convolutional layer, the first argument is `filterSize`, which is the height and width of the filters the training function uses while scanning along the images. In this example, the number 3 indicates that the filter size is 3-by-3. You can specify different sizes for the height and width of the filter. The second argument is the number of filters, `numFilters`, which is the number of neurons that connect to the same region of the input. This parameter determines the number of feature maps. Use the `'Padding'` name-value pair to add padding to the input feature map. For a convolutional layer with a default stride of 1, `'same'` padding ensures that the spatial output size is the same as the input size. You can also define the stride and learning rates for this layer using name-value pair arguments of `convolution2dLayer`.

Batch Normalization Layer Batch normalization layers normalize the activations and gradients propagating through a network, making network training an easier optimization problem. Use batch normalization layers between convolutional layers and nonlinearities, such as ReLU layers, to speed up network training and reduce the sensitivity to network initialization. Use `batchNormalizationLayer` to create a batch normalization layer.

ReLU Layer The batch normalization layer is followed by a nonlinear activation function. The most common activation function is the rectified linear unit (ReLU). Use `reluLayer` to create a ReLU layer.

Max Pooling Layer Convolutional layers (with activation functions) are sometimes followed by a down-sampling operation that reduces the spatial size of the feature map and removes redundant spatial information. Down-sampling makes it possible to increase the number of filters in deeper convolutional layers without increasing the required amount of computation per layer. One way of down-sampling is using a max pooling, which you create using `maxPooling2dLayer`. The max pooling layer returns the maximum values of rectangular regions of inputs, specified by the first argument, `poolSize`. In this example, the size of the rectangular region is [2,2]. The `'Stride'` name-value pair argument specifies the step size that the training function takes as it scans along the input.

Fully Connected Layer The convolutional and down-sampling layers are followed by one or more fully connected layers. As its name suggests, a fully connected layer is a layer in which the neurons connect to all the neurons in the preceding layer. This layer combines all the features learned by the previous layers across the image to identify the larger patterns. The last fully connected layer combines the features to classify the images. Therefore, the `OutputSize` parameter in the last fully connected layer is equal to the number of classes in the target data. In this example, the output size is 10, corresponding to the 10 classes. Use `fullyConnectedLayer` to create a fully connected layer.

Softmax Layer The softmax activation function normalizes the output of the fully connected layer. The output of the softmax layer consists of positive numbers that sum to one, which can then be used as classification probabilities by the classification layer. Create a softmax layer using the `softmaxLayer` function after the last fully connected layer.

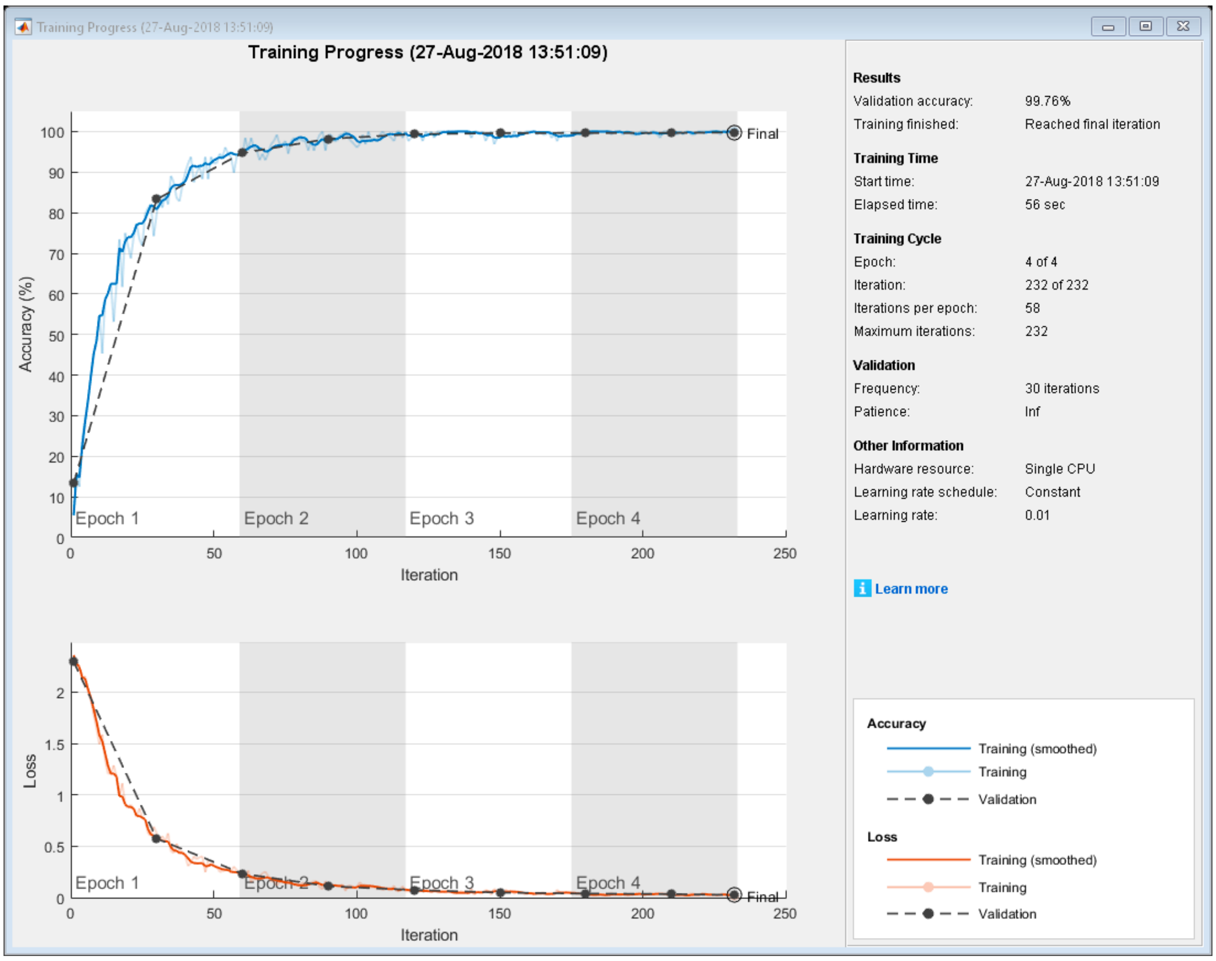
Classification Layer The final layer is the classification layer. This layer uses the probabilities returned by the softmax activation function for each input to assign the input to one of the mutually exclusive classes and compute the loss. To create a classification layer, use `classificationLayer`.

3. **Training options**

```
    options = trainingOptions('sgdm', ...
  'InitialLearnRate',0.01, ...
  'MaxEpochs',4, ...
  'Shuffle','every-epoch', ...
  'ValidationData',imdsValidation, ...
  'ValidationFrequency',30, ...
  'Verbose',false, ...
  'Plots','training-progress');
```

4. **Train Network**

```
Training net = trainNetwork(imdsTrain,layers,options);
```



Results

Validation accuracy:	99.76%
Training finished:	Reached final iteration

Training Time

Start time:	27-Aug-2018 13:51:09
Elapsed time:	56 sec

Training Cycle

Epoch:	4 of 4
Iteration:	232 of 232
Iterations per epoch:	58
Maximum iterations:	232

Validation

Frequency:	30 iterations
Patience:	Inf

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.01

5. **Assess Accuracy: (uses accuracy here not AUC)**

```
YPred = classify(net,imdsValidation);
YValidation = imdsValidation.Labels;
accuracy = sum(YPred == YValidation)/numel(YValidation)
```

2. Deep Network Designer

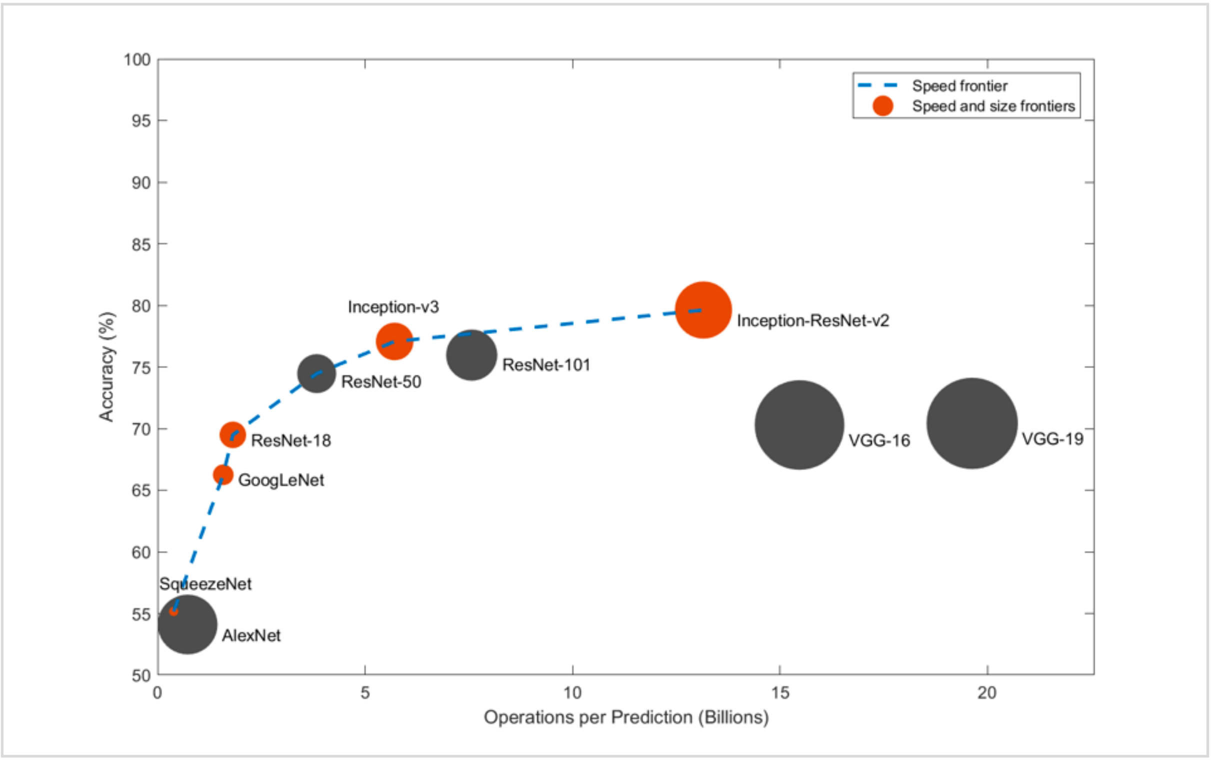
- [ ] Edit and build deep learning networks - MATLAB- MathWorks Australia

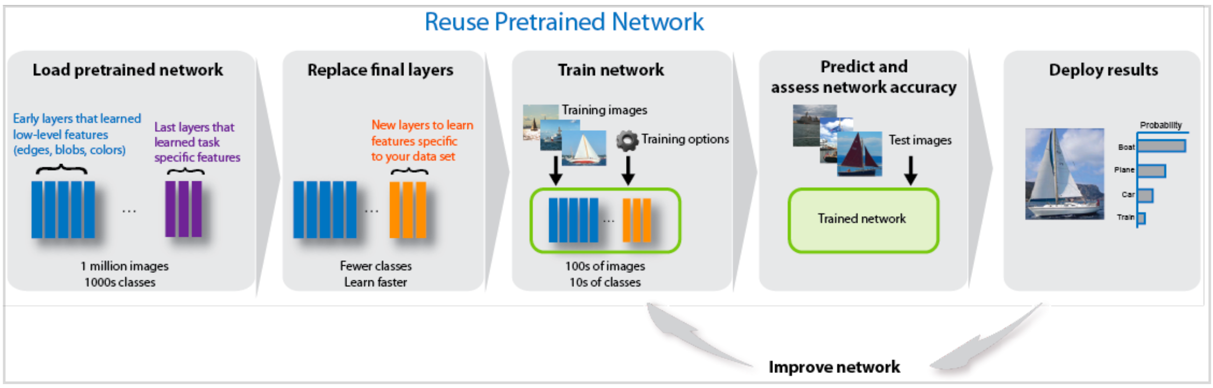
REFERENCE Pretrained Convolutional Neural Networks- MATLAB & Simulink- MathWorks Australia

Transfer Learning Using AlexNet- MATLAB & Simulink- MathWorks Australia

Classify Image Using GoogLeNet- MATLAB & Simulink- MathWorks Australia

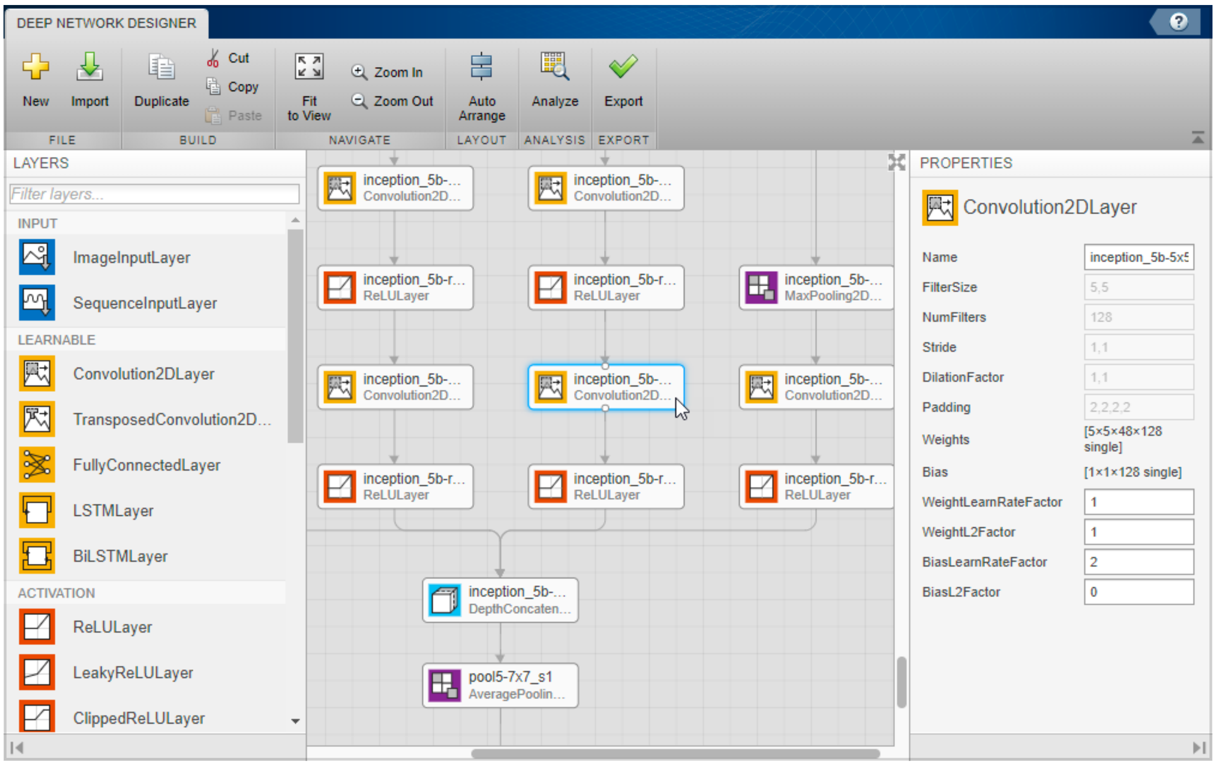
MATLAB RECOMMENDS SQUEEZENET or **ALEXNET** for transfer learning initially due to speed of training vs. accuracy trade-off





The Deep Network Designer app lets you build, visualize, and edit deep learning networks. Using this app, you can:

- Import pretrained networks and edit them for transfer learning.
- Import and edit networks, and build new networks.
- Drag and drop to add new layers and create new connections.
- View and edit layer properties.
- Analyze the network to ensure you define the architecture correctly, and detect problems before training.
- After you finish designing a network, you can export it to the workspace where you can save or train the network.


DEEP NETWORK DESIGNER
New
Import
Duplicate
Cut
Copy
Paste
Fit to View
Zoom In
Zoom Out
Auto Arrange
Analyze
Export
FILE
BUILD
NAVIGATE
LAYOUT
ANALYSIS
EXPORT
LAYERS
Filter layers...
INPUT
ImageInputLayer
SequenceInputLayer
LEARNABLE
Convolution2DLayer
TransposedConvolution2D...
FullyConnectedLayer
LSTMLayer
BiLSTMLayer
ACTIVATION
ReLULayer
LeakyReLULayer
ClippedReLULayer
inception_5b-... Convolution2D...
inception_5b-... Convolution2D...
inception_5b-r... ReLULayer
inception_5b-r... ReLULayer
inception_5b-... MaxPooling2D...
inception_5b-... Convolution2D...
inception_5b-... Convolution2D...
inception_5b-... Convolution2D...
inception_5b-r... ReLULayer
inception_5b-r... ReLULayer
inception_5b-r... ReLULayer
inception_5b-... DepthConcaten...
pool5-7x7_s1 AveragePoolin...
PROPERTIES
Convolution2DLayer

Name	inception_5b-5x5
FilterSize	5,5
NumFilters	128
Stride	1,1
DilationFactor	1,1
Padding	2,2,2,2
Weights	[5×5×48×128 single]
Bias	[1×1×128 single]
WeightLearnRateFactor	1
WeightL2Factor	1
BiasLearnRateFactor	2
BiasL2Factor	0

To Import an existing DNN:

net = squeezenet

OR

Open the app.

deepNetworkDesigner

In the File section, click Import and choose the network to load from the workspace.

Use the plot to explore and visualize the network.

To EDIT the network for Transfer Learning :

1. In the File section, click Import and choose the network to load from the workspace. Use the plot to explore and visualize the network.

2. Edit the network to specify a new number of classes in your data. Drag a new fully connected layer onto the canvas and edit OutputSize property to a new number of classes. Delete the last fully connected layer and connect up your new layer instead.

3. Delete the classification output layer. Then, drag a new classification output layer onto the canvas and connect it instead. The output layer auto settings will learn the number of classes during training.

4. To check that the network is ready for training, click Analyze in the Analysis section.

5. Return to the Deep Network Designer. To export the network to the workspace for training, in the Export section, click Export.

Analyze Network

Analyze deep learning network architecture - MATLAB analyzeNetwork- MathWorks Australia

Use the network analyzer to visualize and understand the network architecture, check that you have defined the architecture correctly, and detect problems before training. Problems that analyzeNetwork detects include missing or disconnected layers, mismatching or incorrect sizes of layer inputs, incorrect number of layer inputs, and invalid graph structures.

Transfer Learning with Deep Learning Network Designer

Transfer Learning with Deep Network Designer- MATLAB & Simulink- MathWorks Australia

1. Choose a pretrained network and import it into the app.
2. Replace the final layers with new layers adapted to the new data set:
3. Specify the new number of classes in your training images.
4. Set learning rates to learn faster in the new layers than in the transferred layers.
5. Export the network for training at the command line.