Convolutional Neural Networks in Matlab

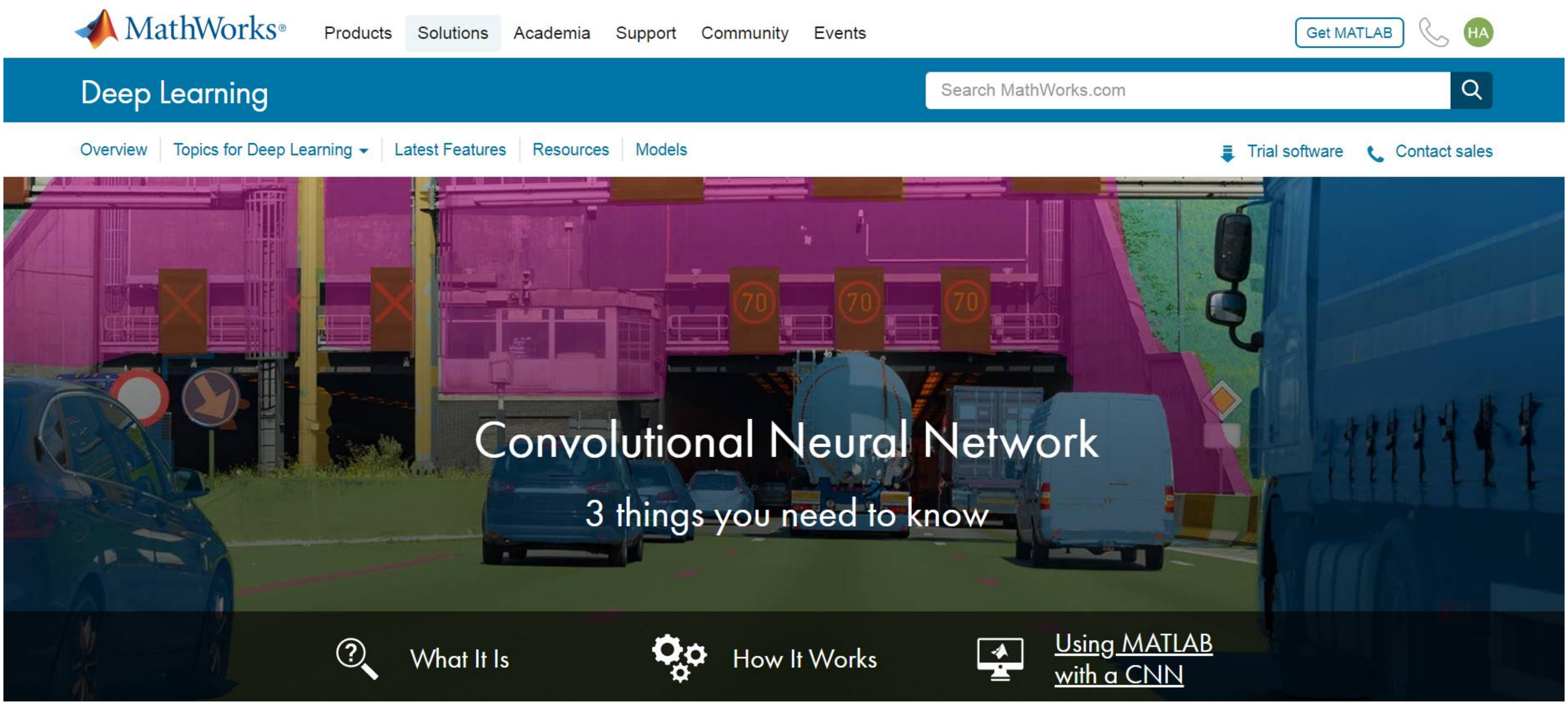
Dr. Héctor Gabriel Acosta Mesa

Instituto de Investigaciones en Inteligencia Artificial
heacosta@uv.mx
www.uv.mx/personal/heacosta

Convolutional Neural Networks

Cuerpo Académico de Investigación y Aplicaciones de la Inteligencia Artificial

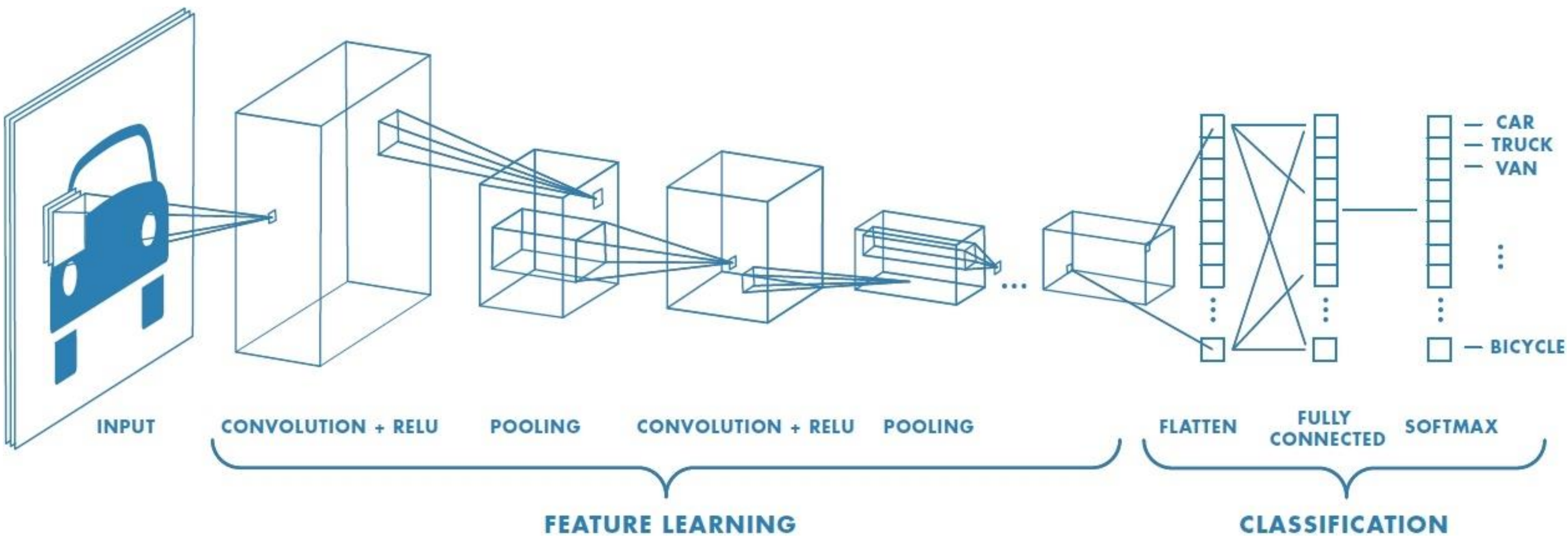
CNN in Matlab



Material taken from MathWorks

https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html

Convolutional Neural Networks



MATLAB Commands
`>> newnet = trainNetwork(data,layers,options)`

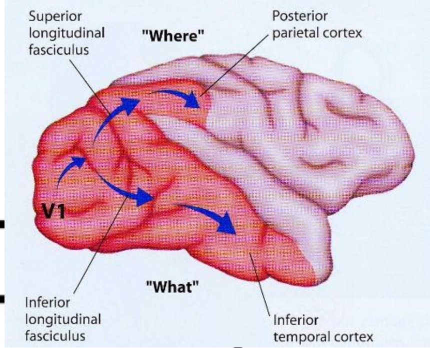
MATLAB Variables
`layers`
`data`
`options`

Convolutional Neural Networks

Classification

CNN - Deep Learning Toolbox™

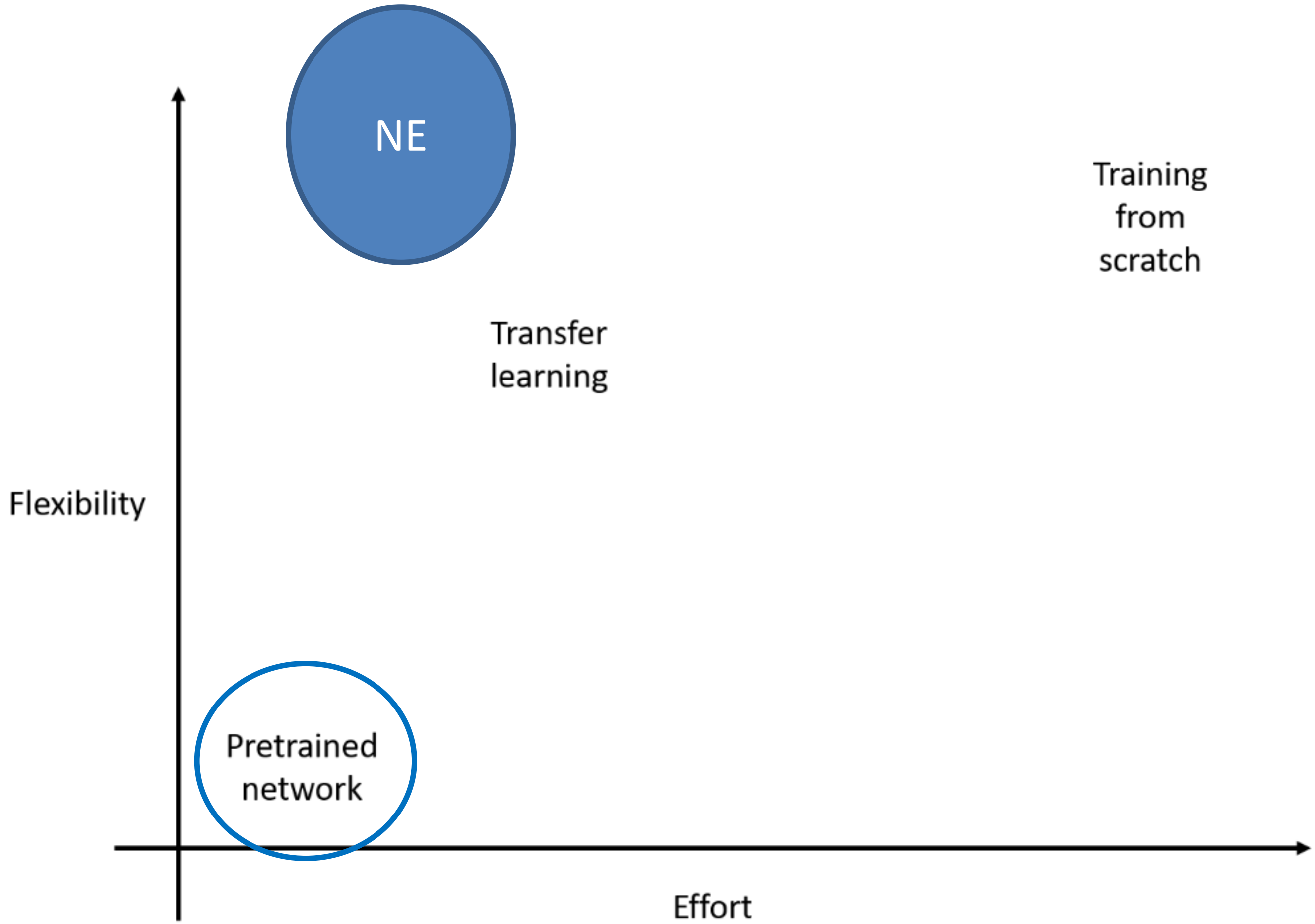
Regression



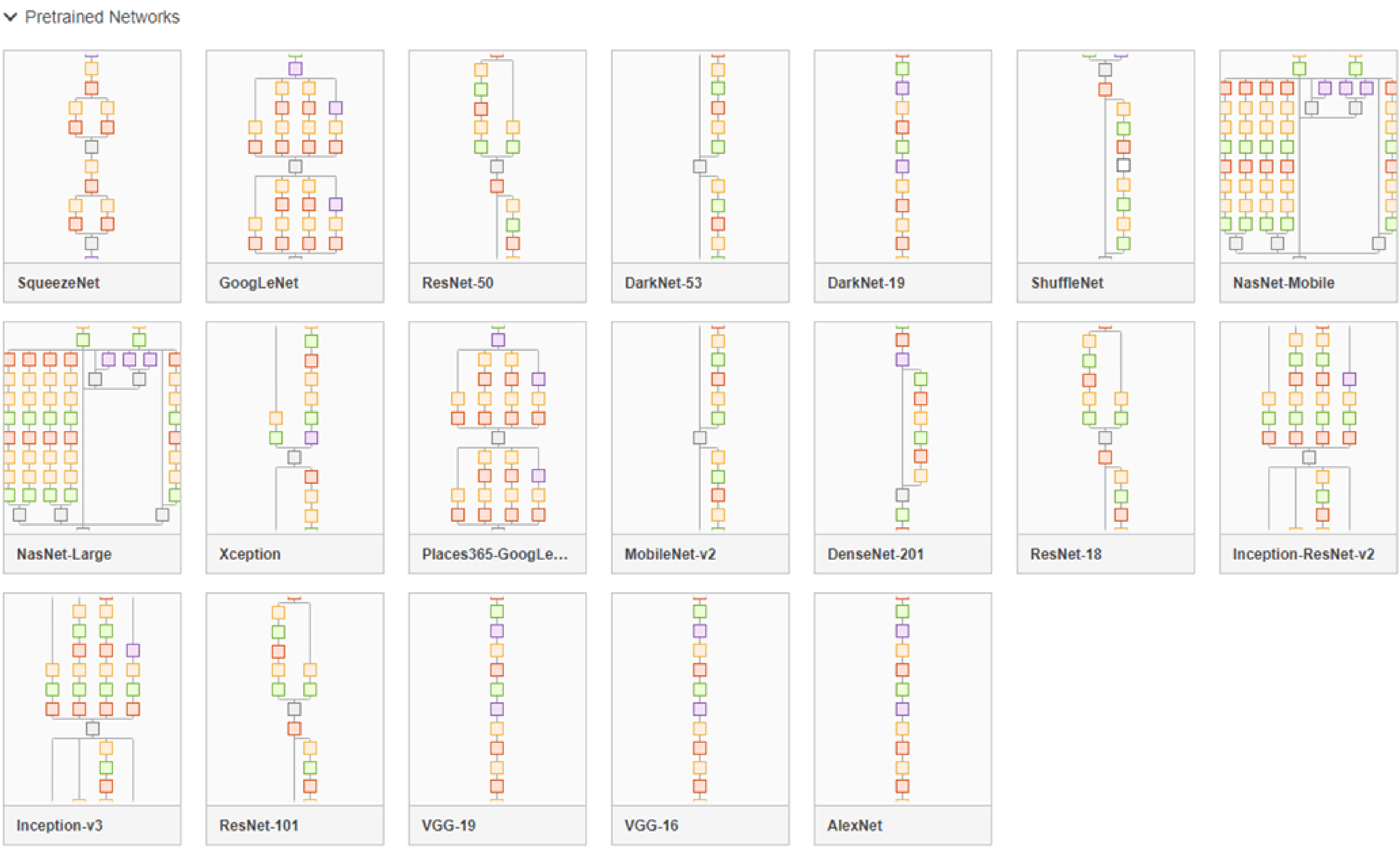
Object detection

R-CNN - Computer Vision Toolbox™

Convolutional Neural Networks

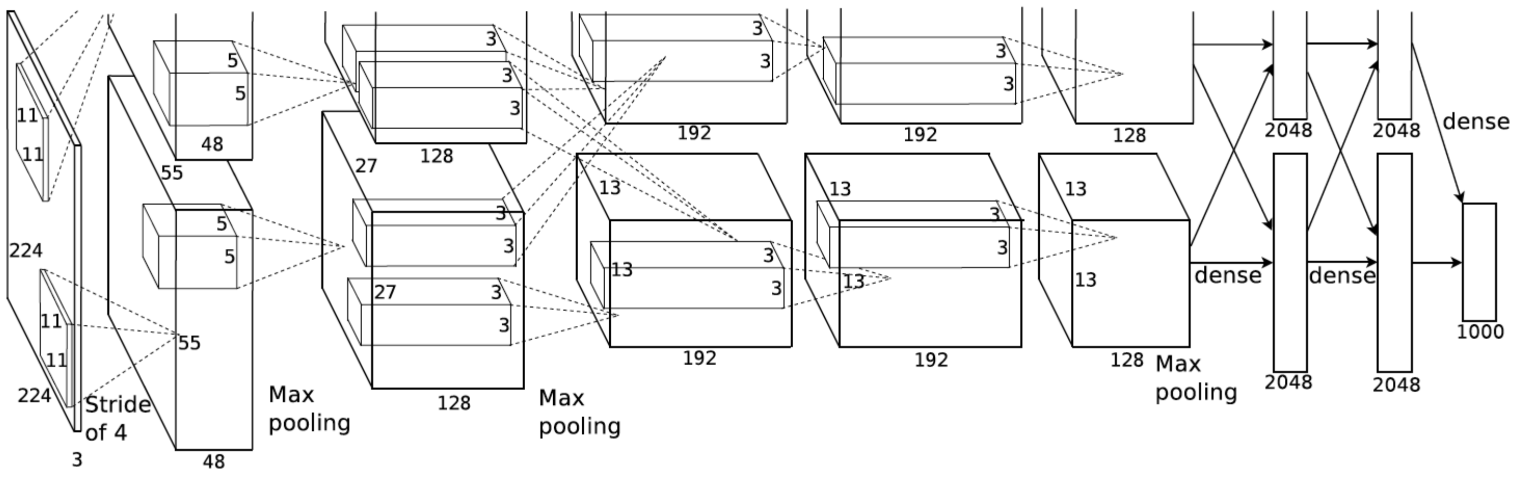


Networks


Pretrained Networks
SqueezeNet
GoogLeNet
ResNet-50
DarkNet-53
DarkNet-19
ShuffleNet
NasNet-Mobile
NasNet-Large
Xception
Places365-GoogLe...
MobileNet-v2
DenseNet-201
ResNet-18
Inception-ResNet-v2
Inception-v3
ResNet-101
VGG-19
VGG-16
AlexNet

Alexnet

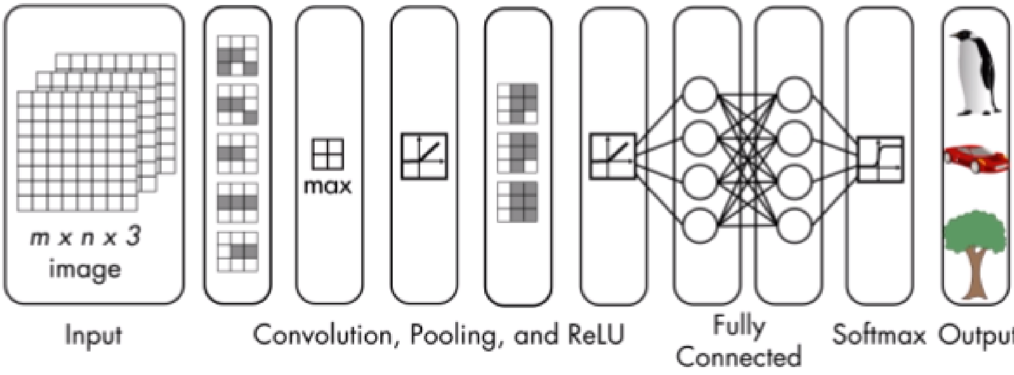
AlexNet is a pretrained Convolutional Neural Network (CNN) that has been trained on approximately 1.2 million images from the ImageNet Dataset (http://image-net.org/index). The model has 23 layers and can classify images into 1000 object categories (e.g. keyboard, mouse, coffee mug, pencil).



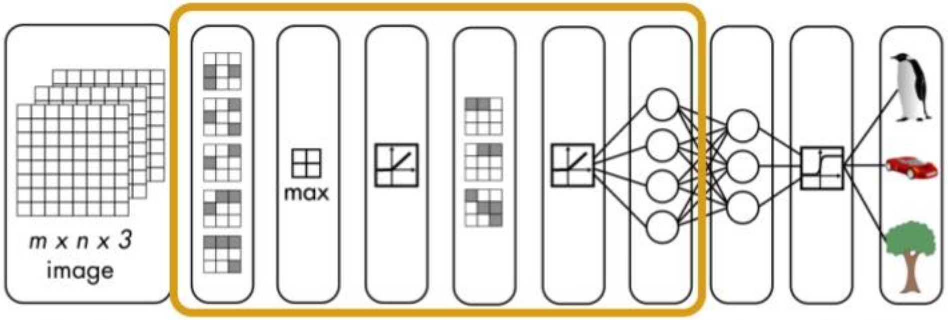
Alexnet in Matlab

Install: **Deep Learning Toolbox Model for AlexNet**
Network support package for the pretrained weights.

```
%Get AlexNet

net = alexnet;

layers = net.Layers;
```

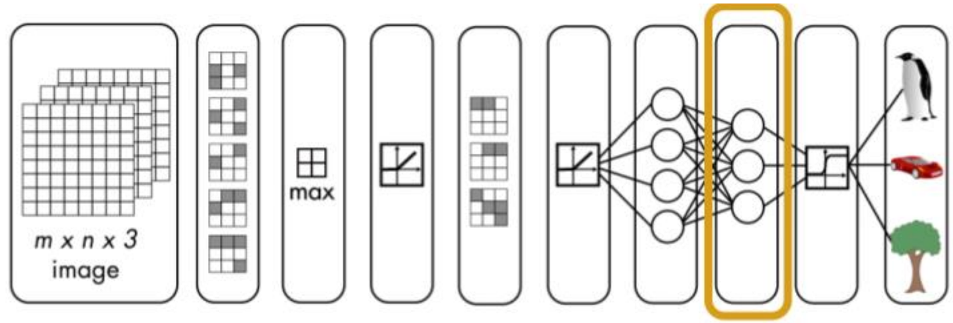


Structure



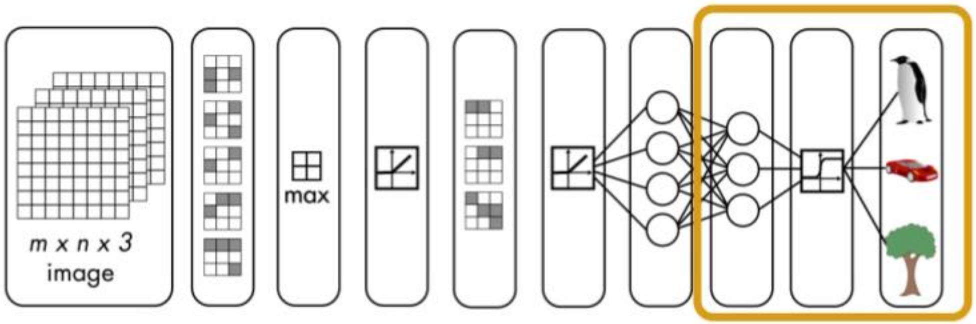
1	'data'	Image Input	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
6	'conv2'	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
11	'relu3'	ReLU	ReLU
12	'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc6'	Fully Connected	4096 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	'fc8'	Fully Connected	1000 fully connected layer
24	'prob'	Softmax	softmax
25	'output'	Classification Output	crossentropyex with 'tench', 'goldfish', and 998 other classes

Structure



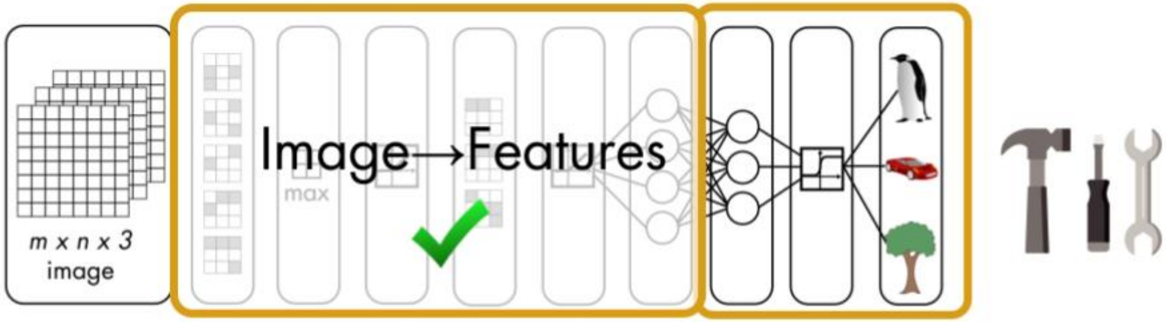
1	'data'	Image Input	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
6	'conv2'	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
11	'relu3'	ReLU	ReLU
12	'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc6'	Fully Connected	4096 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	'fc8'	Fully Connected	1000 fully connected layer
24	'prob'	Softmax	softmax
25	'output'	Classification Output	crossentropyex with 'tench', 'goldfish', and 998 other classes

Structure



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'	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
11	'relu3'	ReLU	ReLU
12	'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc6'	Fully Connected	4096 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	'fc8'	Fully Connected	1000 fully connected layer
24	'prob'	Softmax	softmax
25	'output'	Classification Output	crossentropyex with 'tench', 'goldfish', and 998 other classes

Structure

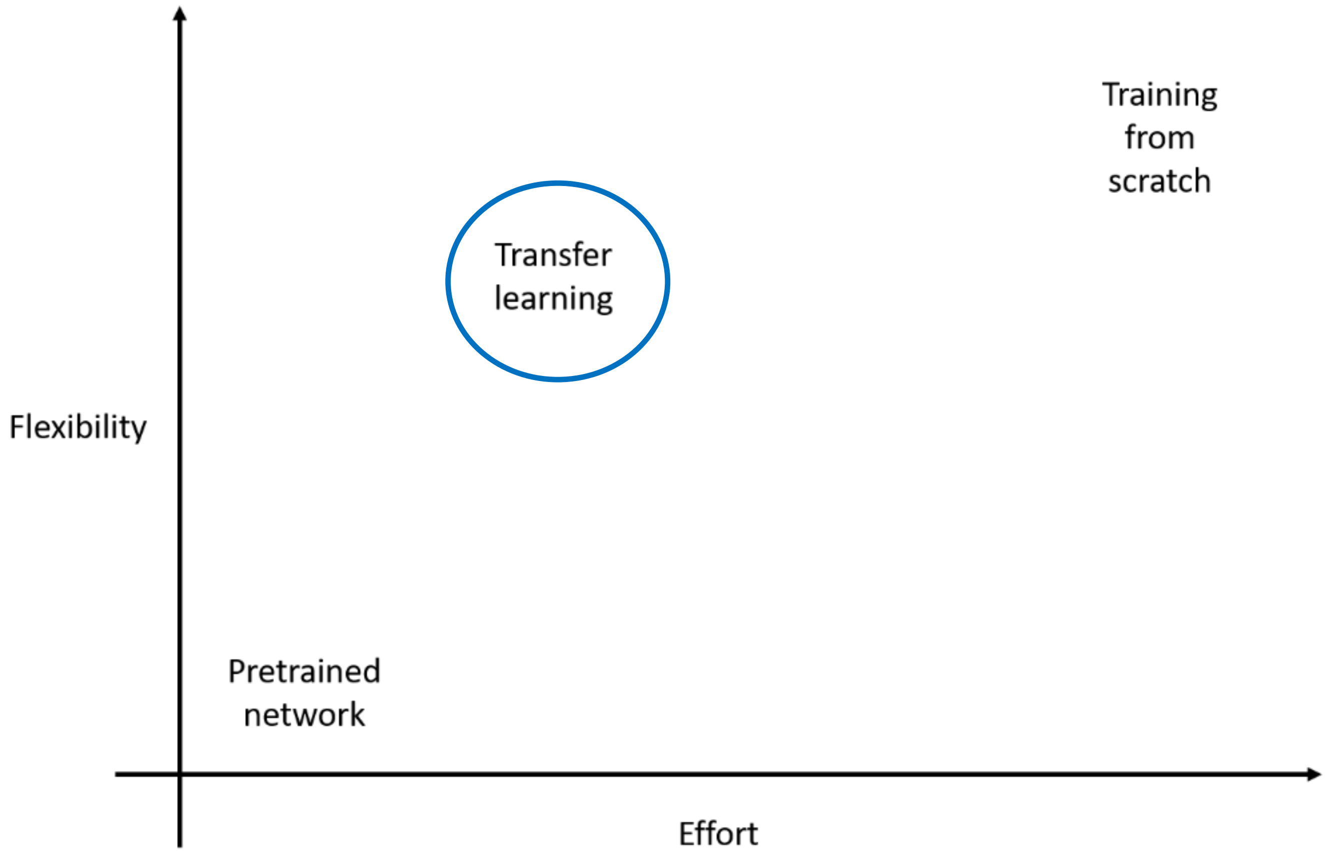


1	'data'	Image Input	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
6	'conv2'	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
11	'relu3'	ReLU	ReLU
12	'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc6'	Fully Connected	4096 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	'fc8'	Fully Connected	1000 fully connected layer
24	'prob'	Softmax	softmax
25	'output'	Classification Output	crossentropyex with 'tench', 'goldfish', and 998 other classes

Pretrained Network

```
% CNN in 10 lines
camera=webcam;

nnet=alexnet;

while true
    picture=camera.snapshot;
    picture=imresize(picture,[227,227]);

    label=classify(nnet,picture);

    image(picture)
    title(char(label))
    drawnow;
end
```

Transfer learning



Transfer learning

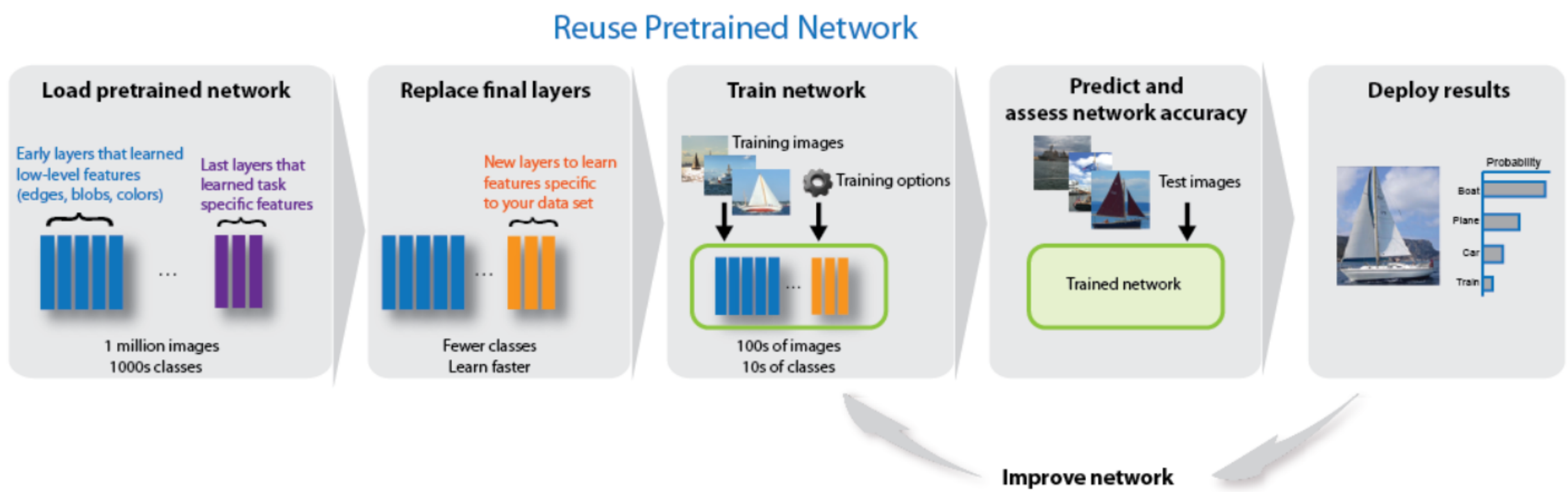
This example shows how to fine-tune a pretrained AlexNet convolutional neural network to perform classification on a new collection of images.

AlexNet has 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 network has learned rich feature representations for a wide range of images. The network takes an image as input and 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 with randomly initialized weights from scratch. You can quickly transfer learned features to a new task using a smaller number of training images.

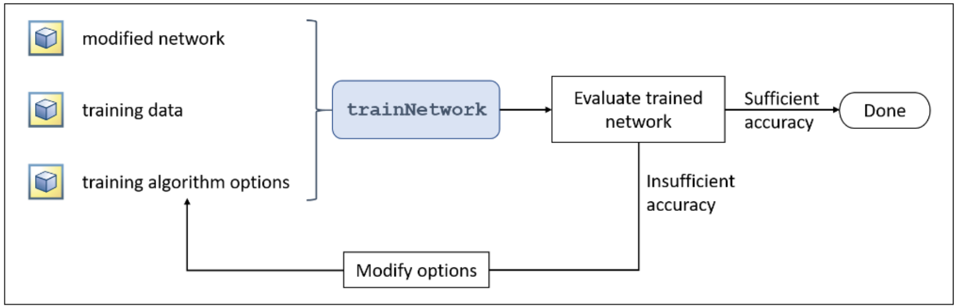
This example uses:
Deep Learning Toolbox
Deep Learning Toolbox Model for AlexNet Network

View MATLAB Command



Transfer learning

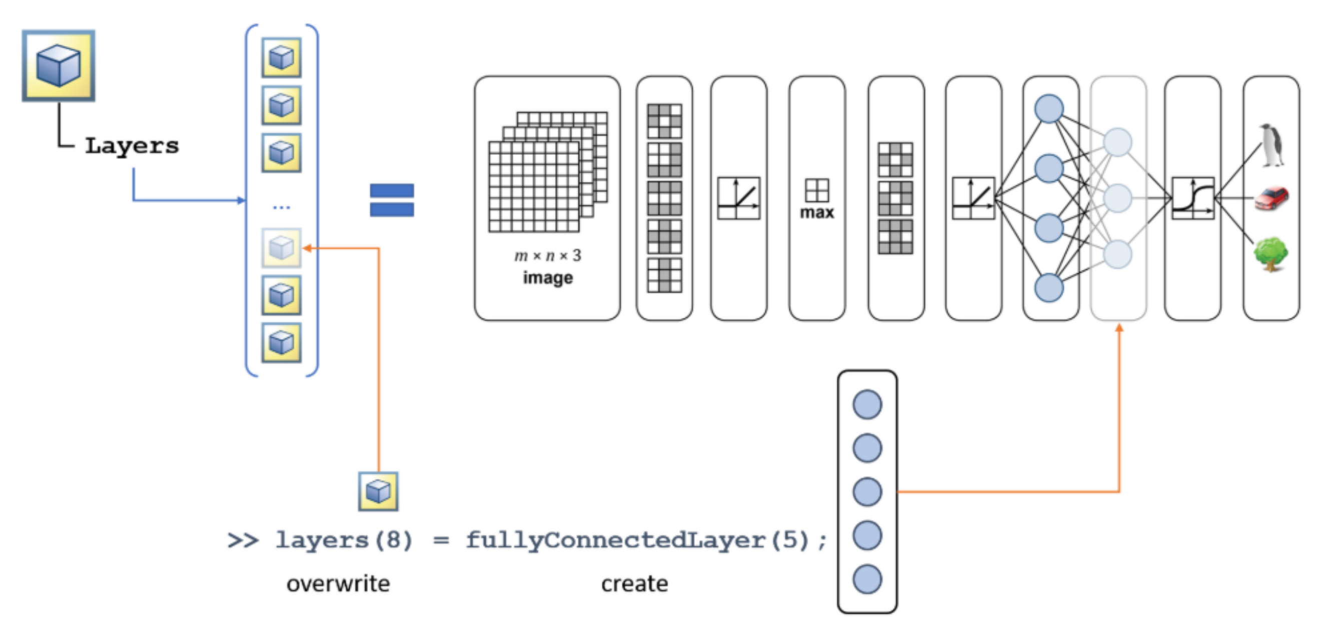
Typical workflow for transfer learning



To perform transfer learning, you need to create three components:

1. An array of layers representing the network architecture. For transfer learning, this is created by modifying a preexisting network such as AlexNet.
2. Images with known labels to be used as training data. This is typically provided as a datastore.

Transfer learning

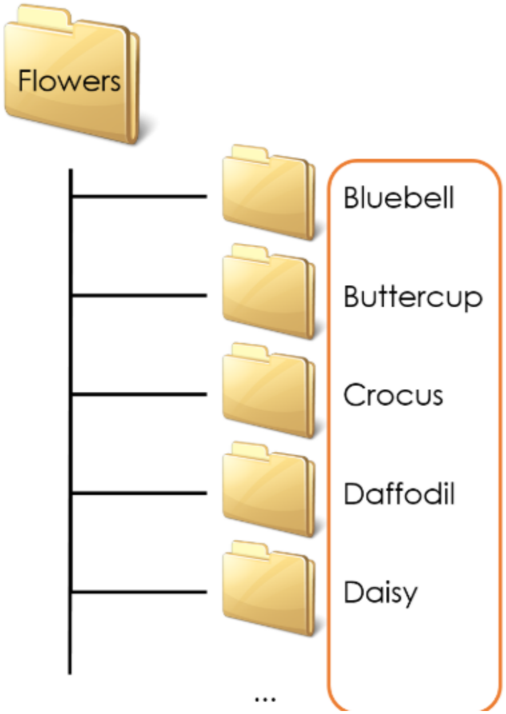


As with any indexed assignment in MATLAB, you can combine these steps into one line.

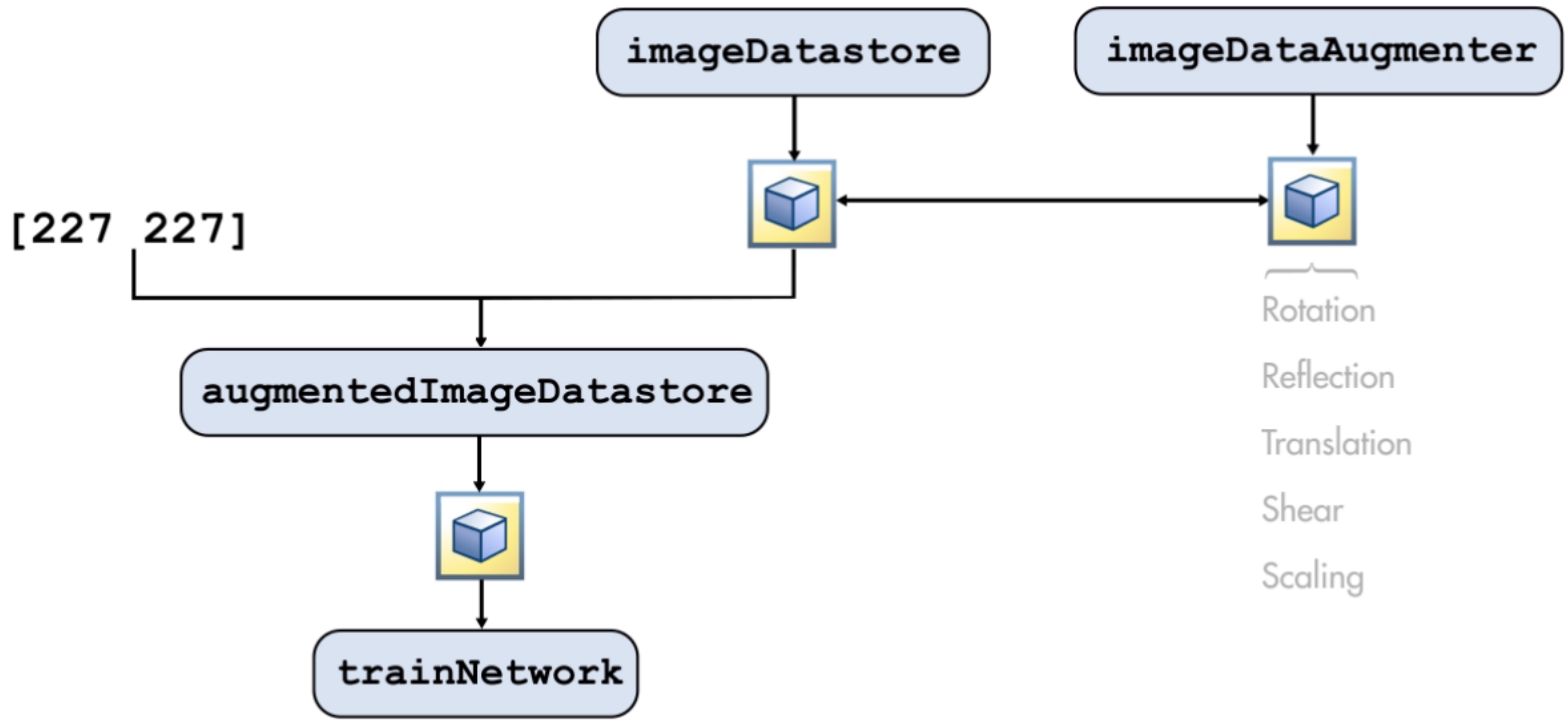
Data

Labeling training images

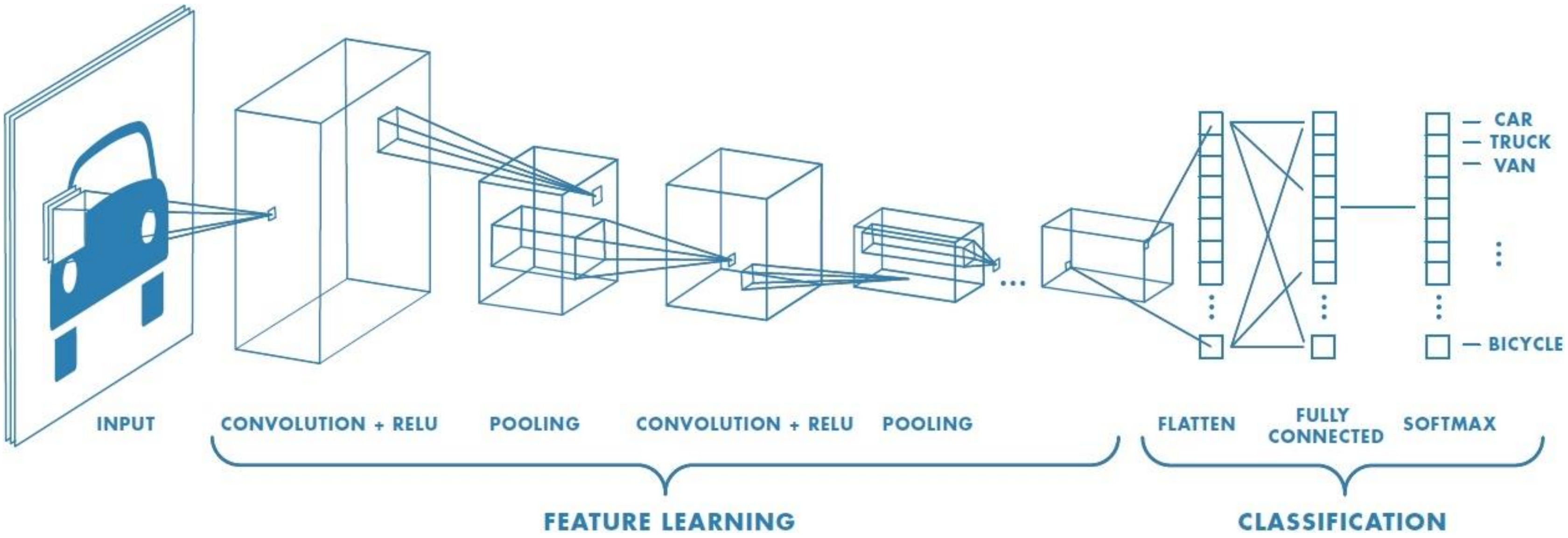
When training a network, you need to provide known labels for the training images. The `Flowers` folder contains 12 subfolders, each of which contains 80 images of one type of flower. The name of the folder can therefore be used to provide the labels needed for training.



Data

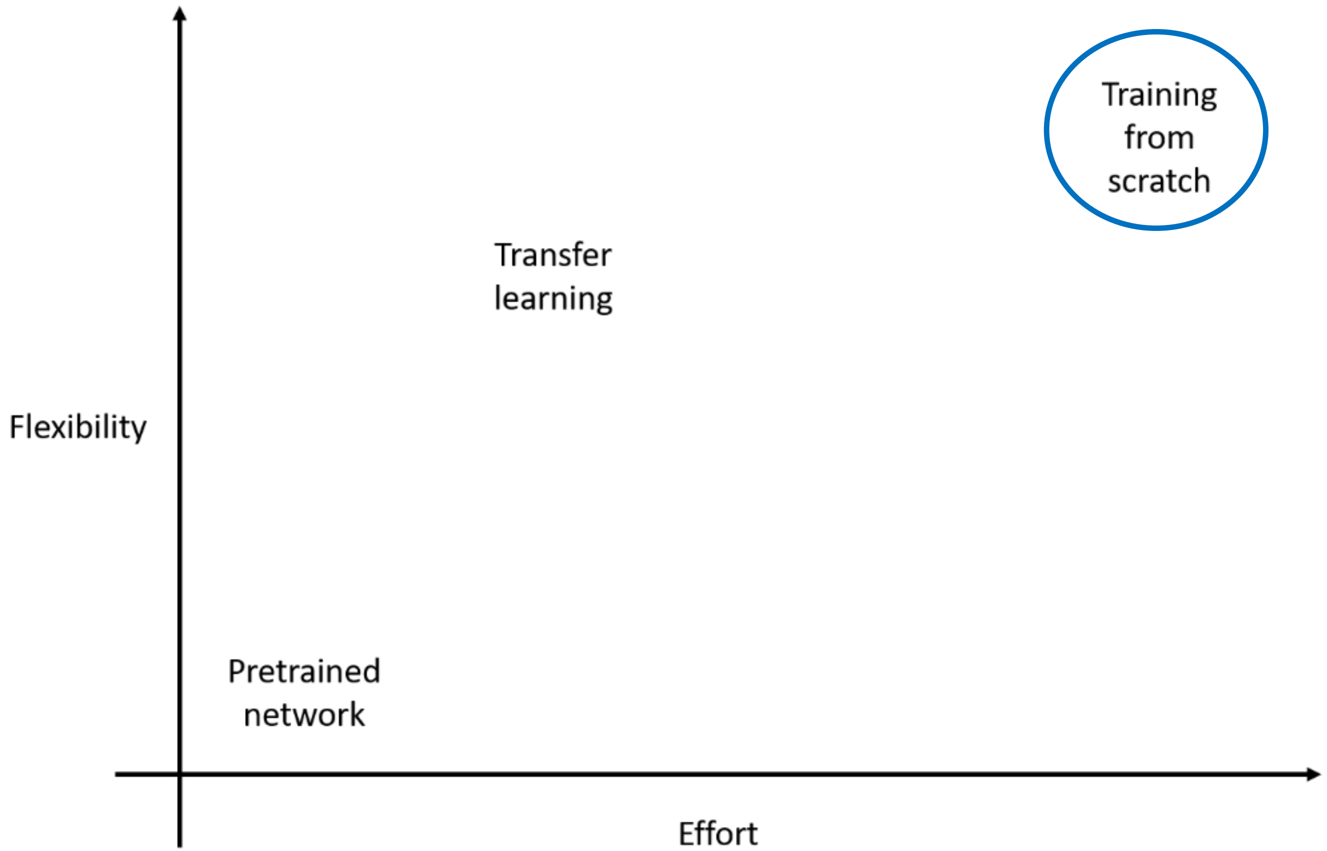

imageDatastore
imageDataAugmenter
[227 227]
augmentedImageDatastore
trainNetwork
Rotation
Reflection
Translation
Shear
Scaling

Transfer learning



See code: TransfL_Alexnet1.m (Flowers)

Training from scratch

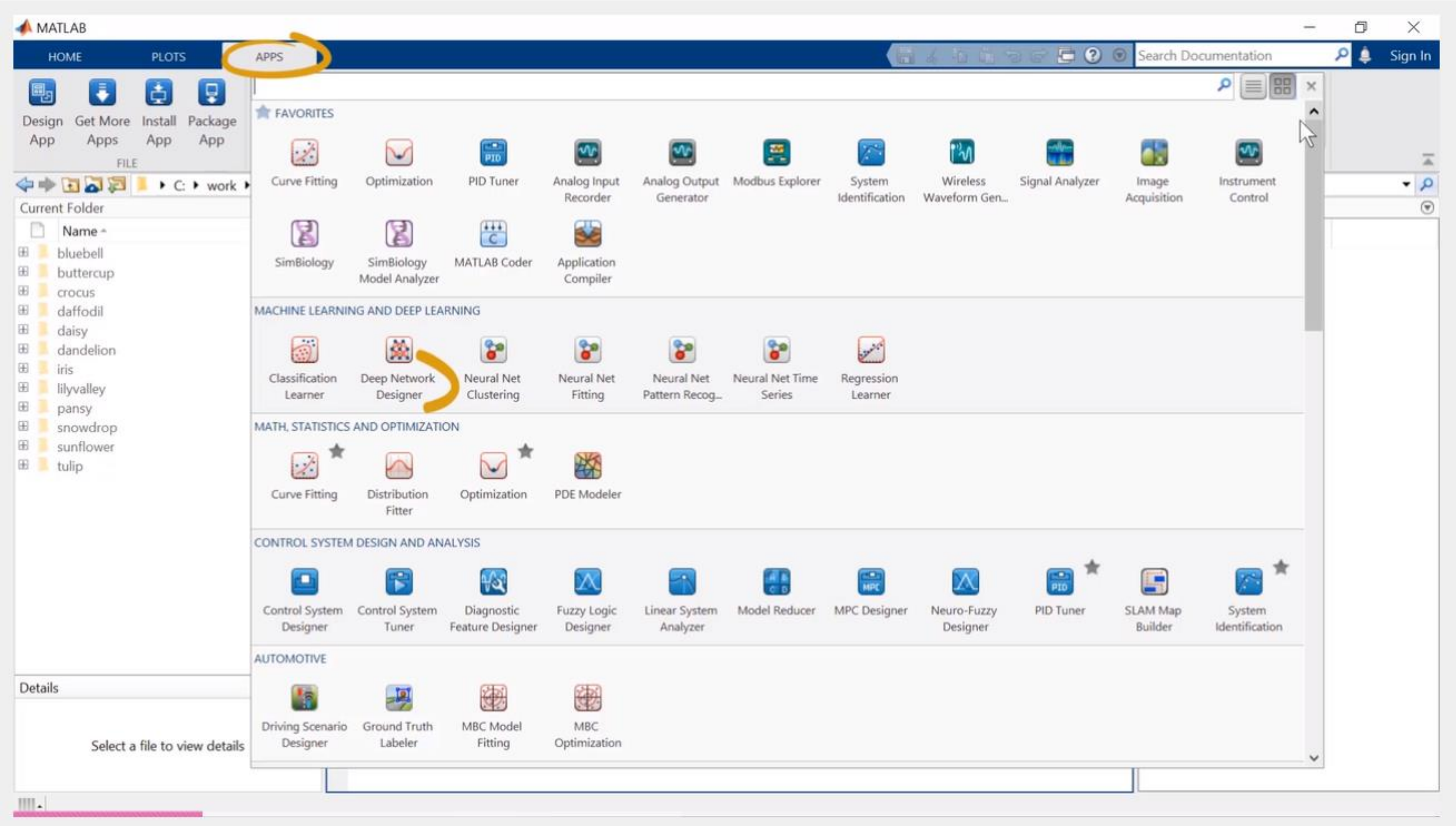


Training from scratch (Layers)

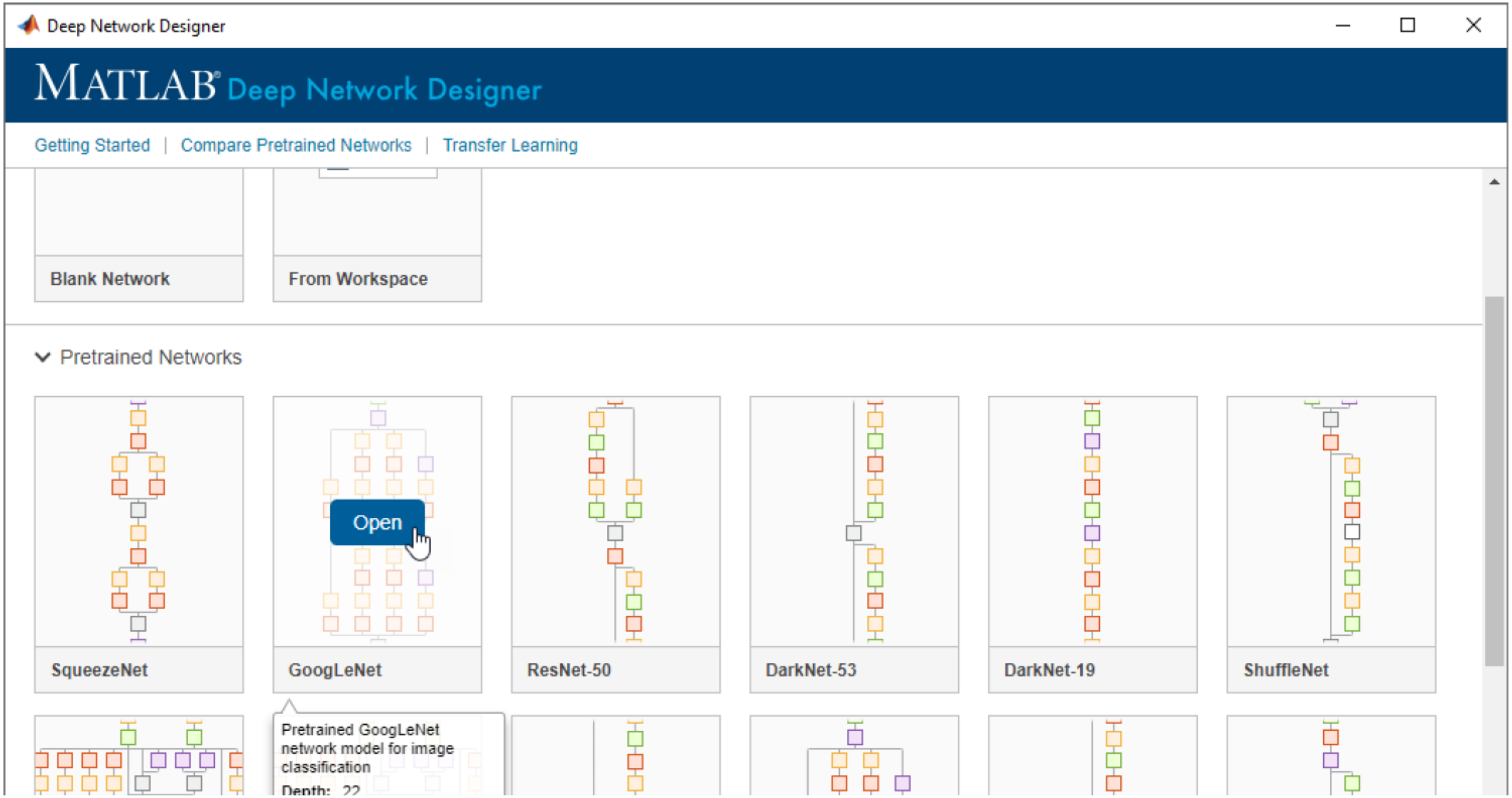
Example 1: Scratch_Digits.m

analyzeNetwork(net)

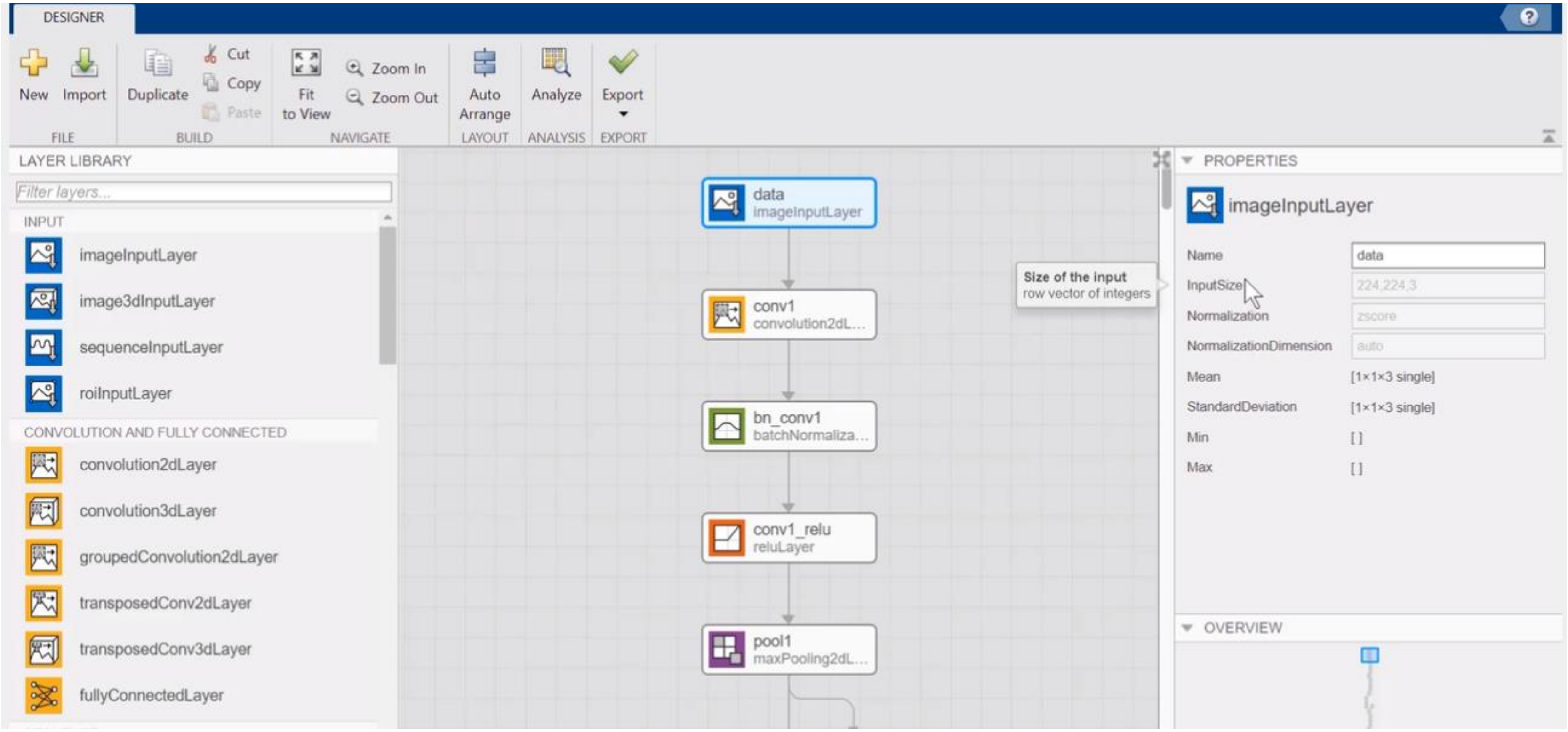
Deep Network Designer



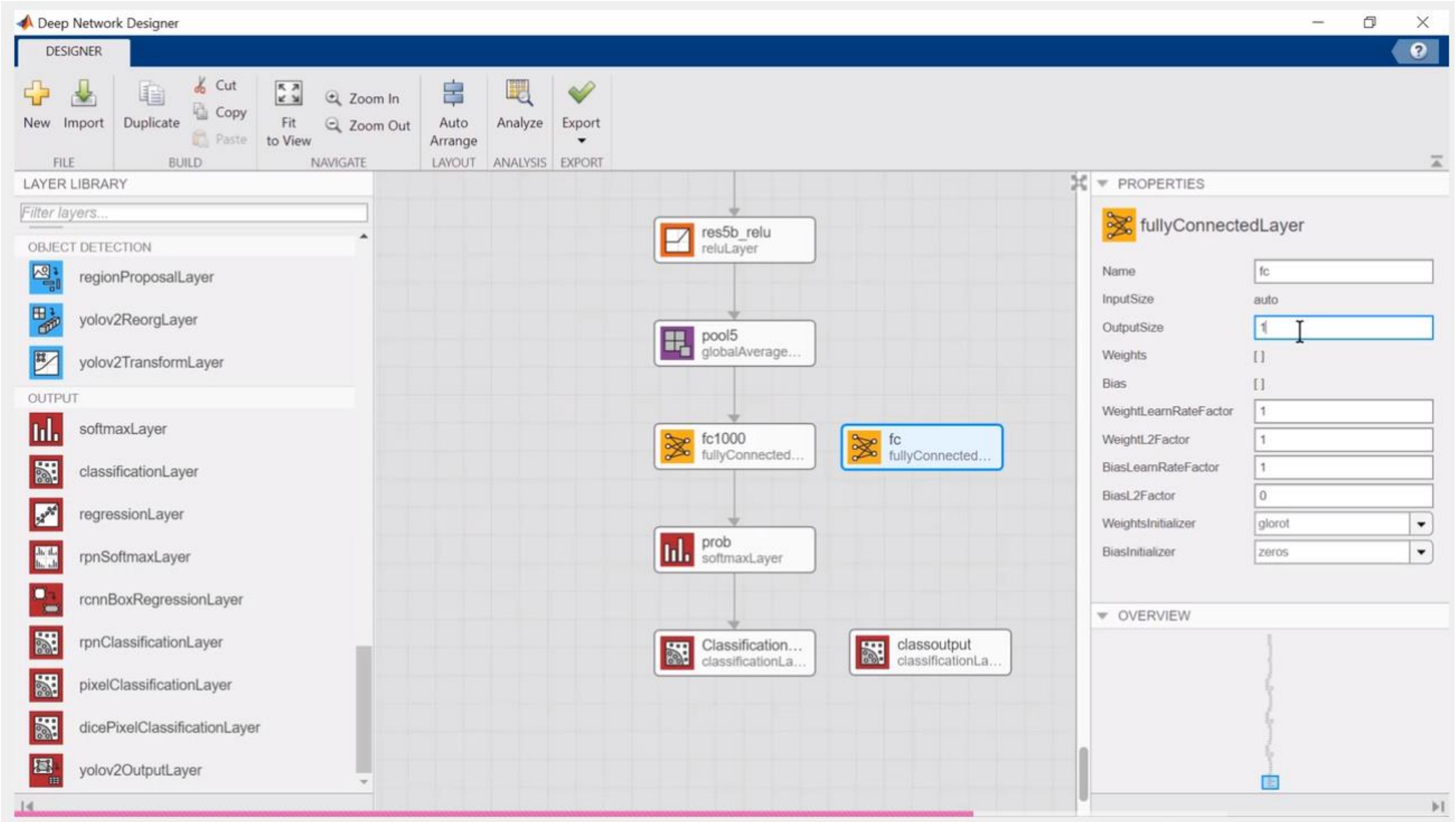
Deep Network Designer



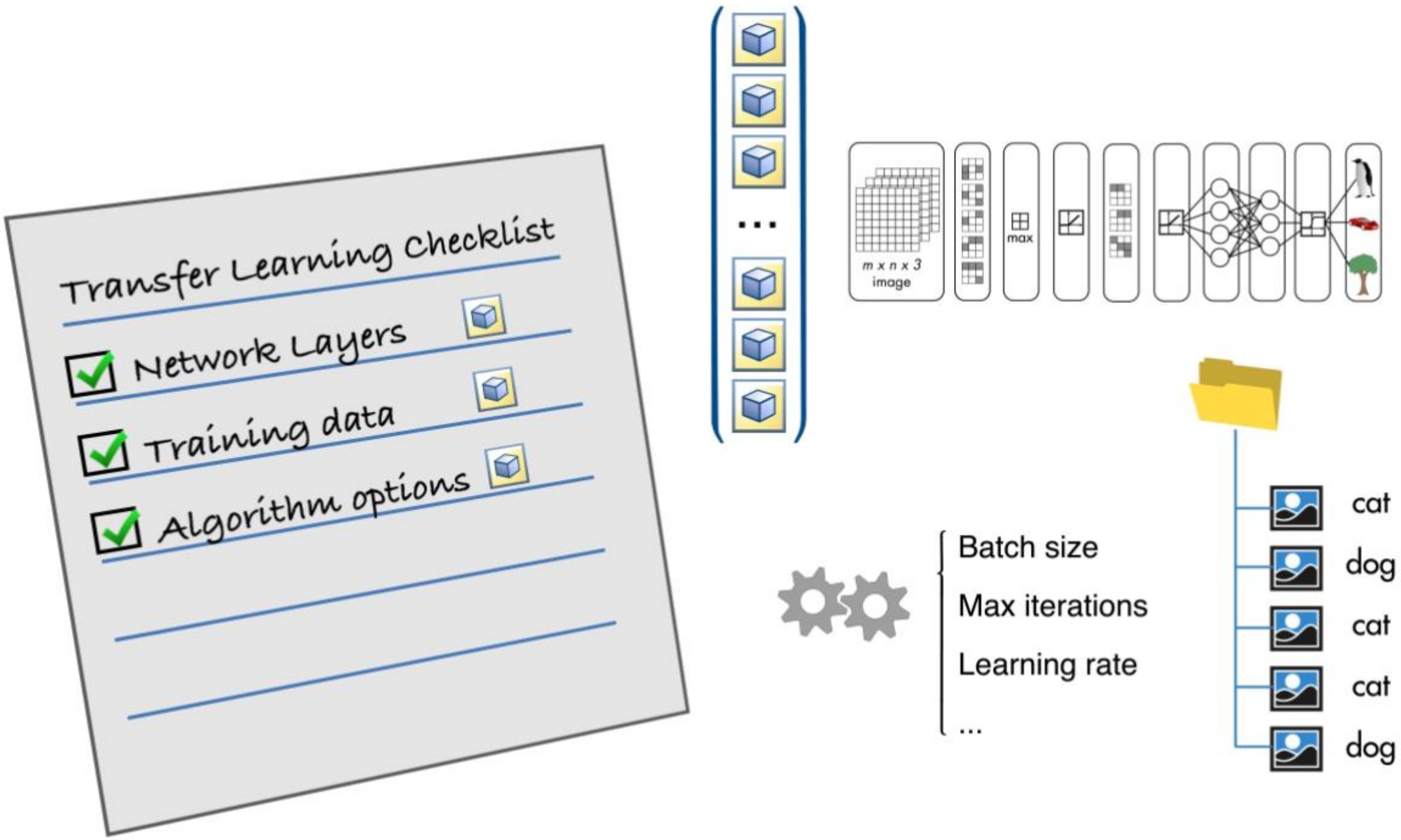
Deep Network Designer



Deep Network Designer

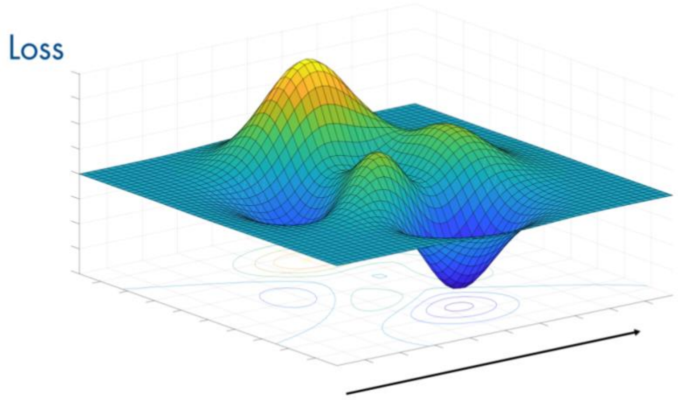


Training


Transfer Learning Checklist
Network Layers
Training data
Algorithm options
m x n x 3
image
max
Batch size
Max iterations
Learning rate
...
cat
dog
cat
cat
dog

Optimization parameters

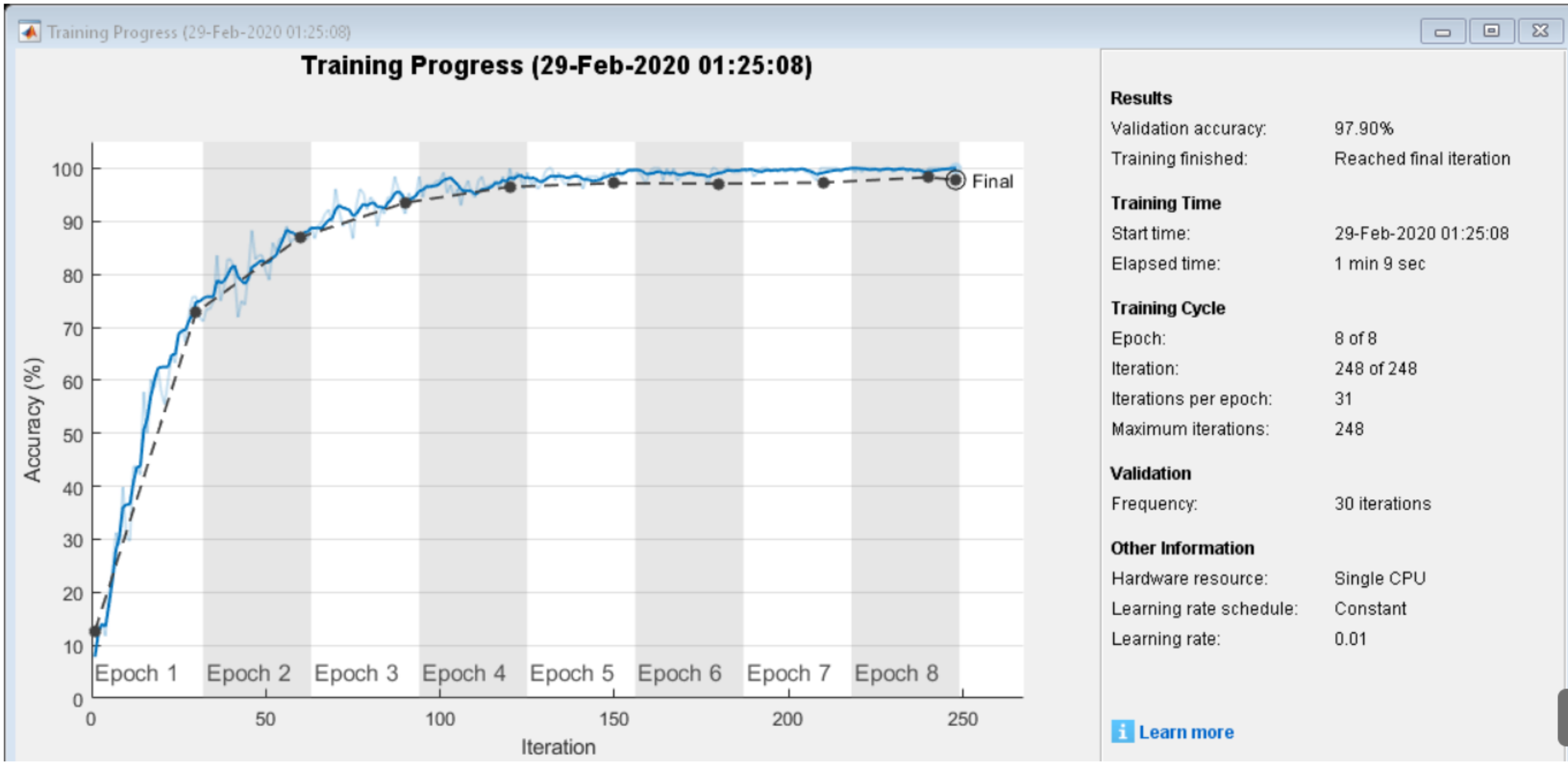
See options code.



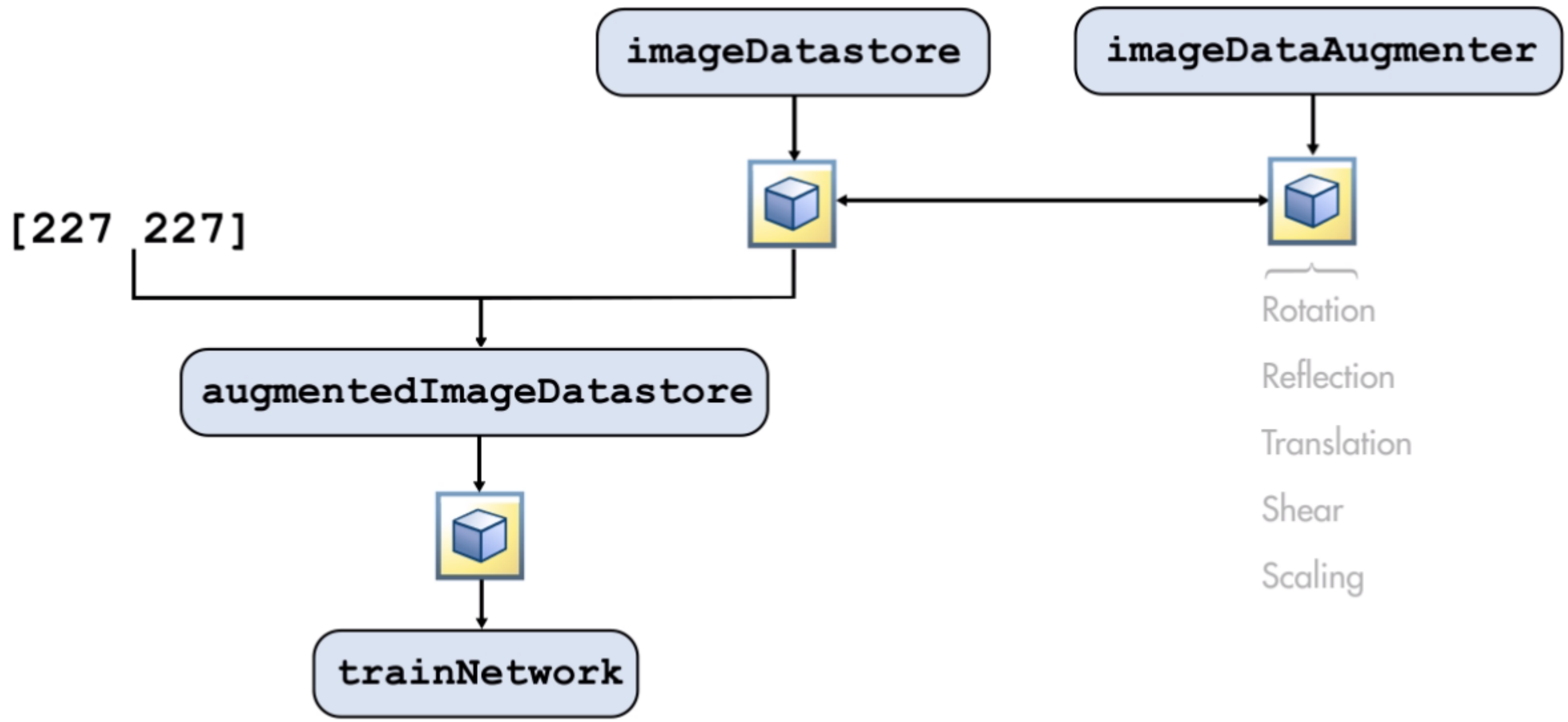
Training

```
>> newnet = trainNetwork(data,layers,options)

Training on single GPU.
Initializing image normalization.
|=========================================================================================|
|     Epoch    |   Iteration  | Time Elapsed |  Mini-batch  |  Mini-batch  | Base Learning|
|              |              |  (seconds)   |     Loss     |   Accuracy   |     Rate     |
|=========================================================================================|
|            1 |            1 |         0.47 |       3.5061 |        7.81% |       0.0010 |
|            3 |           10 |        10.31 |       0.7686 |       75.00% |       0.0010 |
|            5 |           20 |        18.96 |       0.2371 |       92.19% |       0.0010 |
|            8 |           30 |        27.43 |       0.0770 |       97.66% |       0.0010 |
|           10 |           40 |        35.31 |       0.0336 |       99.22% |       0.0010 |
|           13 |           50 |        43.17 |       0.0289 |       99.22% |       0.0010 |
|           15 |           60 |        50.15 |       0.0104 |      100.00% |       0.0010 |
|           18 |           70 |        56.84 |       0.0072 |      100.00% |       0.0010 |
|           20 |           80 |        63.00 |       0.0210 |       99.22% |       0.0010 |
|           23 |           90 |        69.37 |       0.0035 |      100.00% |       0.0010 |
```

Training


Training Progress (29-Feb-2020 01:25:08)
Training Progress (29-Feb-2020 01:25:08)
Accuracy (%)
Iteration
Epoch 1
Epoch 2
Epoch 3
Epoch 4
Epoch 5
Epoch 6
Epoch 7
Epoch 8
Final
Results
Validation accuracy: 97.90%
Training finished: Reached final iteration
Training Time
Start time: 29-Feb-2020 01:25:08
Elapsed time: 1 min 9 sec
Training Cycle
Epoch: 8 of 8
Iteration: 248 of 248
Iterations per epoch: 31
Maximum iterations: 248
Validation
Frequency: 30 iterations
Other Information
Hardware resource: Single CPU
Learning rate schedule: Constant
Learning rate: 0.01
Learn more

Data



imageDatastore
imageDataAugmenter
[227 227]
Rotation
Reflection
Translation
Shear
Scaling
augmentedImageDatastore
trainNetwork

Data augmentation in Matlab

imageDataAugmenter

Description

An image data augmenter configures a set of preprocessing options for image augmentation, such as resizing, rotation, and reflection. The imageDataAugmenter is used by an augmentedImageDatastore to generate batches of augmented images. For more information, see Augment Images for Training.

Syntax

```
aug = imageDataAugmenter
aug = imageDataAugmenter(Name,Value)
```

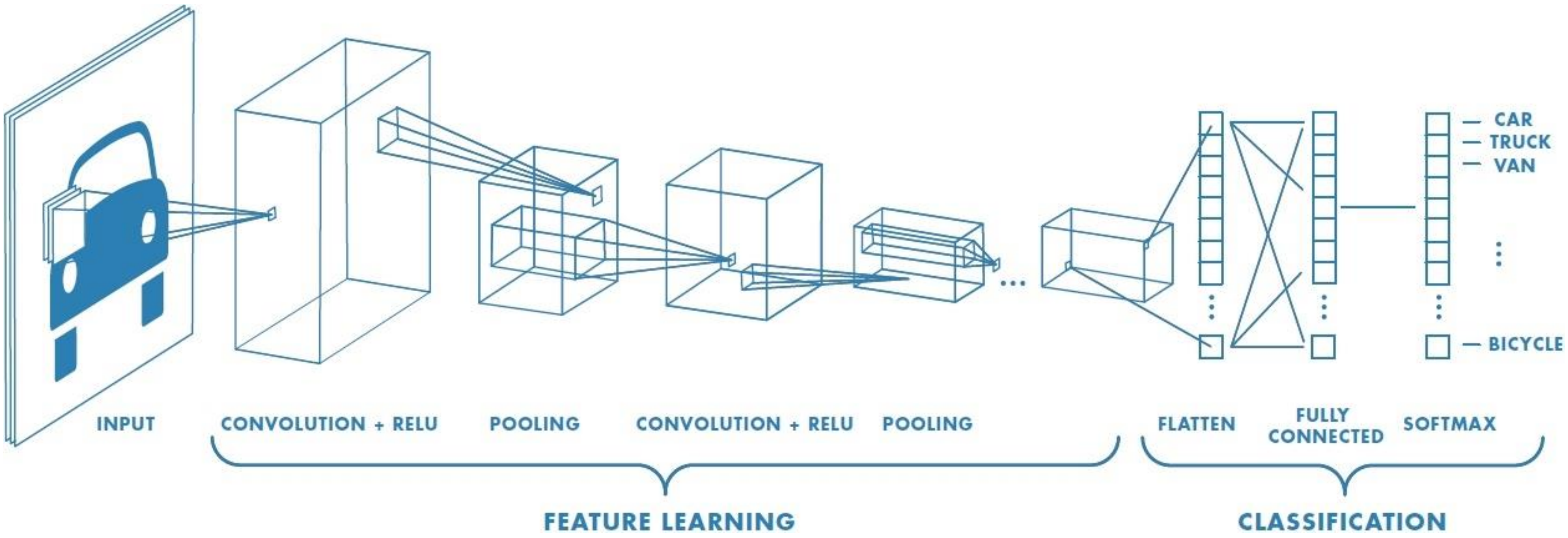
Data augmentation

```
% Create an imageDataAugmenter object that specifies
preprocessing options for image augmentation

[XTrain,YTrain] = digitTrain4DArrayData;

imageAugmenter = imageDataAugmenter( ...
    'RandRotation',[-20,20], ...
    'RandXTranslation',[-3 3], ...
    'RandYTranslation',[-3 3])

imageSize = [28 28 1];

augimds =
augmentedImageDatastore(imageSize,XTrain,YTrain,'DataAug
mentation',imageAugmenter);
```

Training from scratch (Layers)

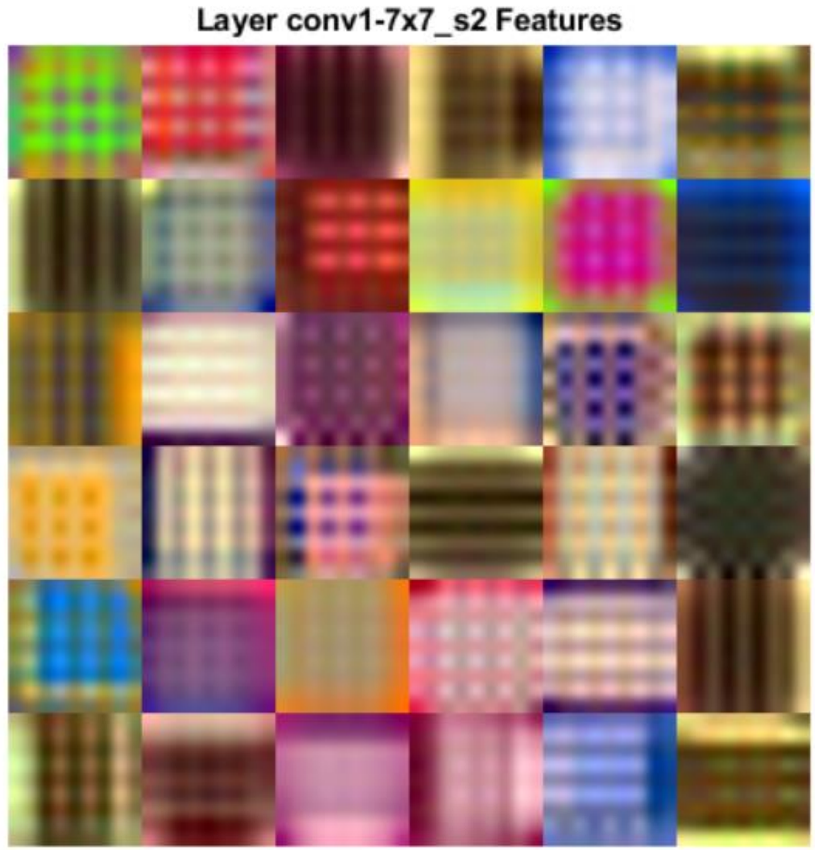
Example 2: Scratch_Digits_Augment.m

Feature extraction



See: Demo_FeatureExtraction.m (cfar10)

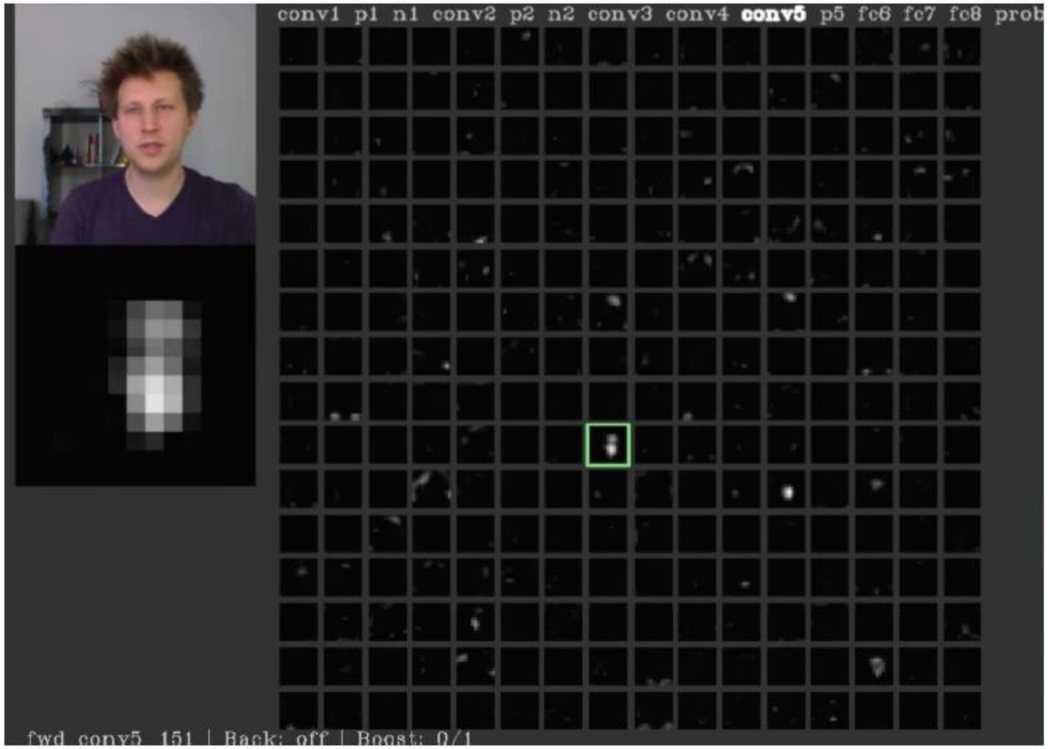
Visualize Features of a CNN



https://www.mathworks.com/help/deeplearning/ug/visualize-features-of-a-convolutional-neural-network.html

Visualize Features of a CNN

conv5 feature map is 128x13x13; visualize as 128 13x13 grayscale images



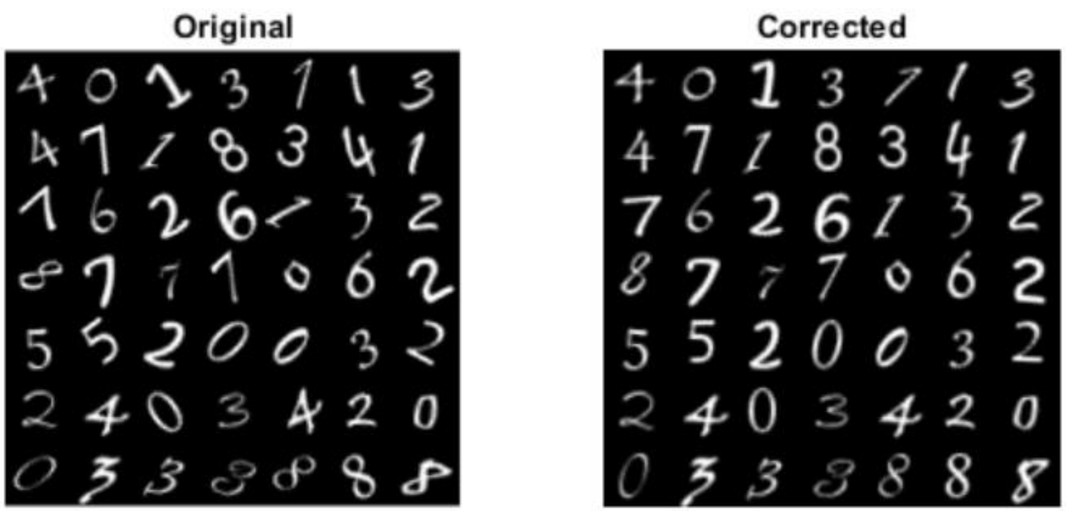
Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

https://microscope.openai.com/models

Convolutional Neural Networks

Classification Regression	CNN - Deep Learning Toolbox™
Object detection	R-CNN - Computer Vision Toolbox™

CNN for regression

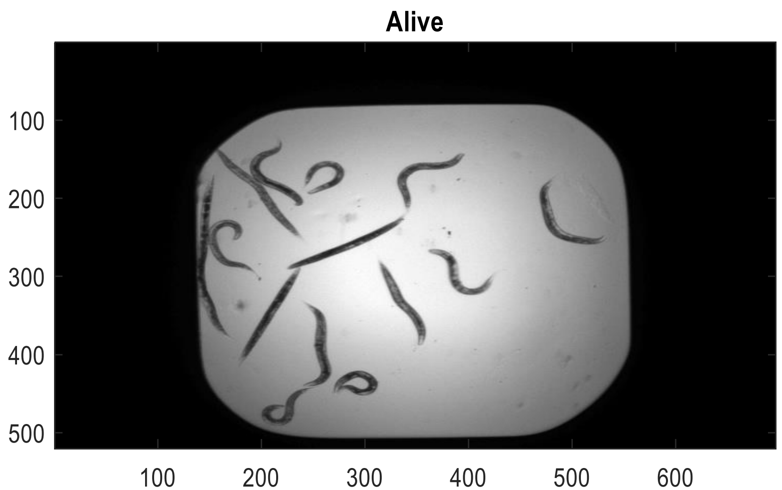


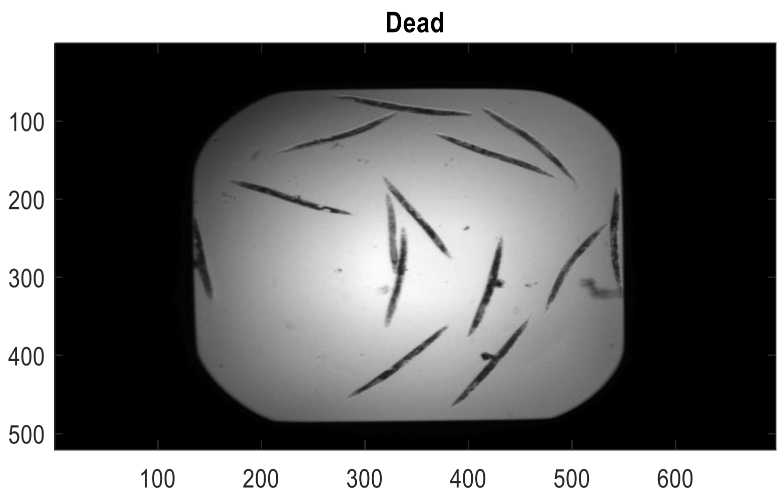
See: Scratch_Digits_Regress.m

Homework 11: Worm classification

- Propose and implement a methodology using CNN to perform a binary classification of the worm database.

- A minimum accuracy of 90% is expected.





Nota: Las etiquetas de clase están en el archivo .csv este lo pueden leer desde matlab con la instrucción: labels= readtable('WormData.csv');