UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
School of Professional & Executive Development

Tools for Artificial Intelligence with MATLAB, initiation (TAIM)

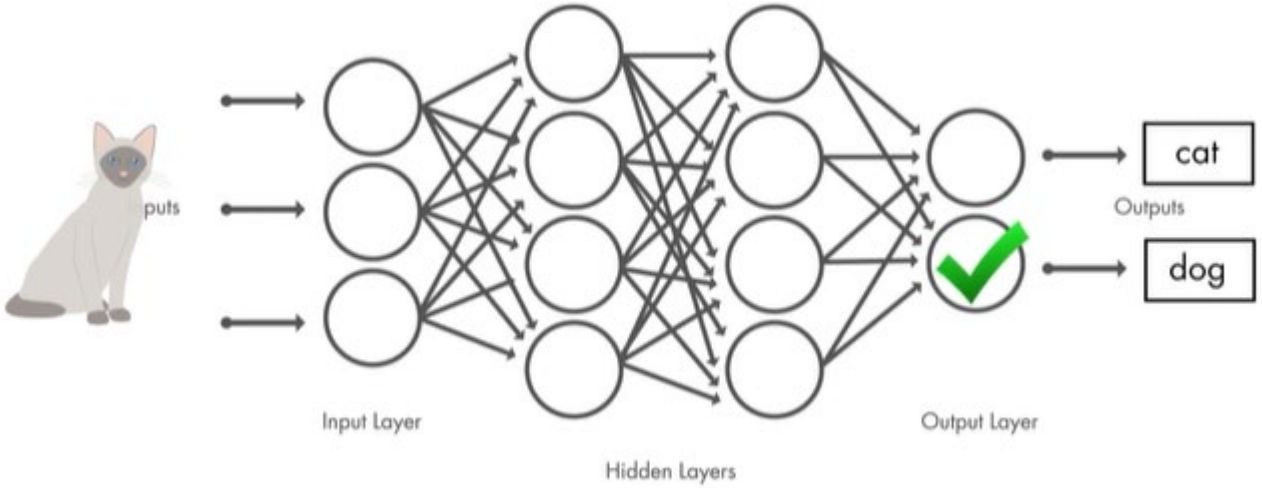
José Antonio Lázaro

Deep Learning

Barcelona, 4, February 2025

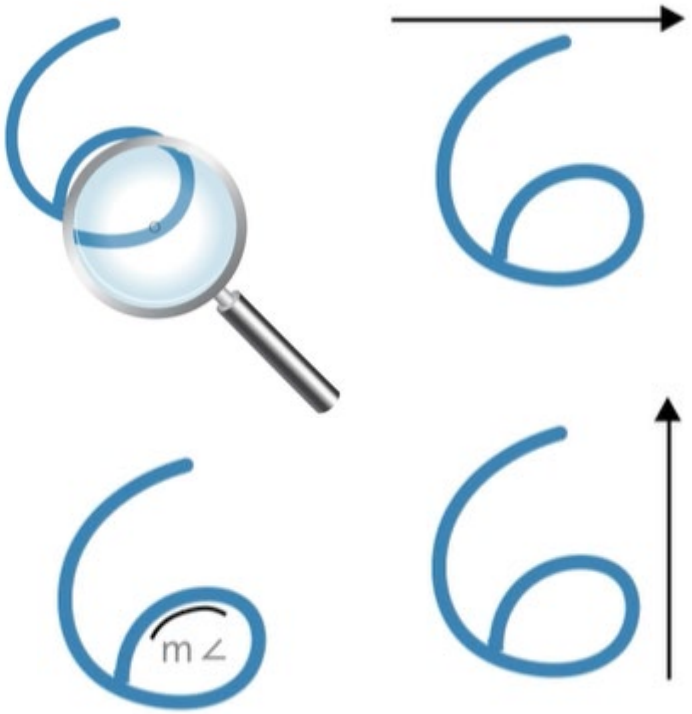
What is it?

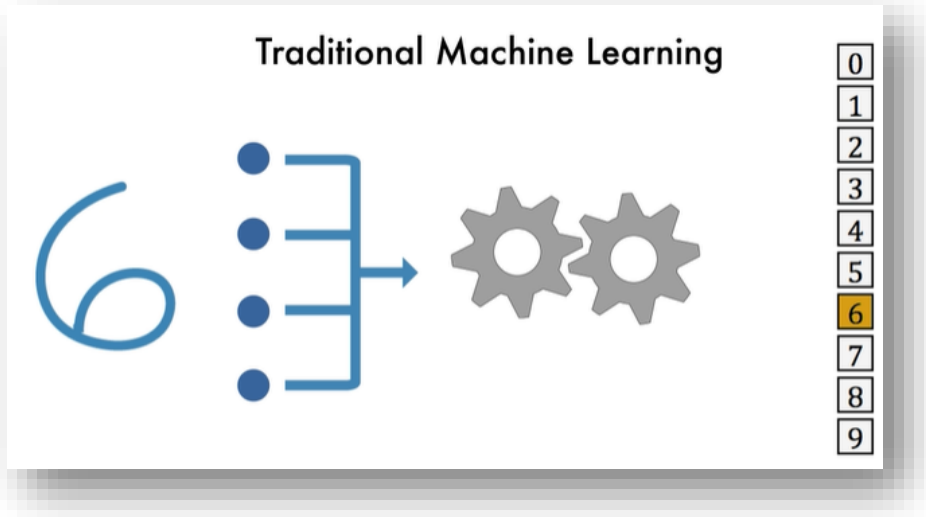
- Deep learning is a Machine learning technique that uses Deep Neural Networks (networks with many layers) to make predictions.



Comparisson with Machine Learning

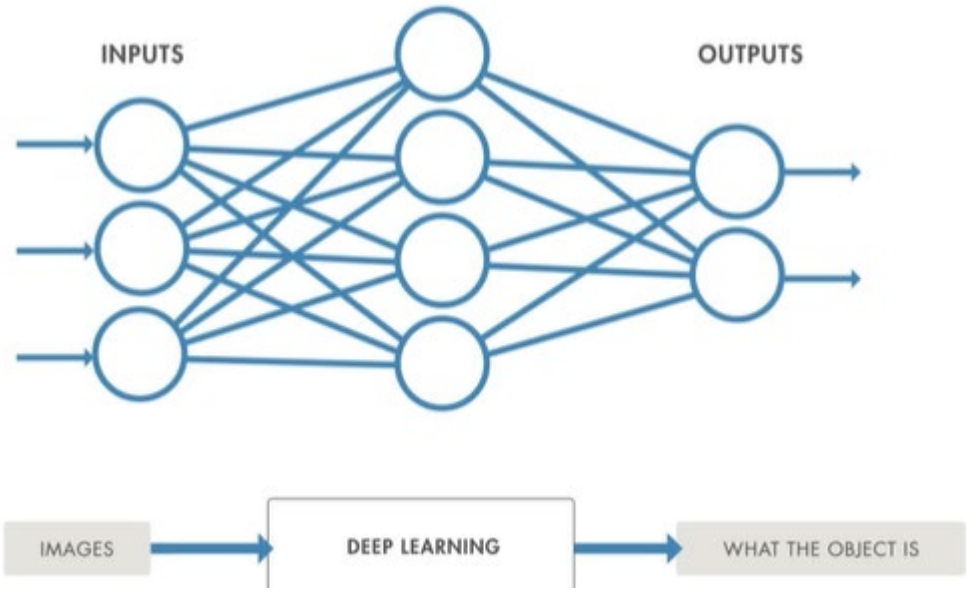
- Deep learning is a Machine learning technique that uses Deep Neural Networks (networks with many layers) to make predictions.





Comparisson with Machine Learning

- Deep learning goal is to perform end-to-end learning.
- The images themselves are the inputs.
- Features & Classification are learned directly from the images.

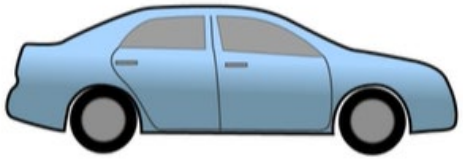


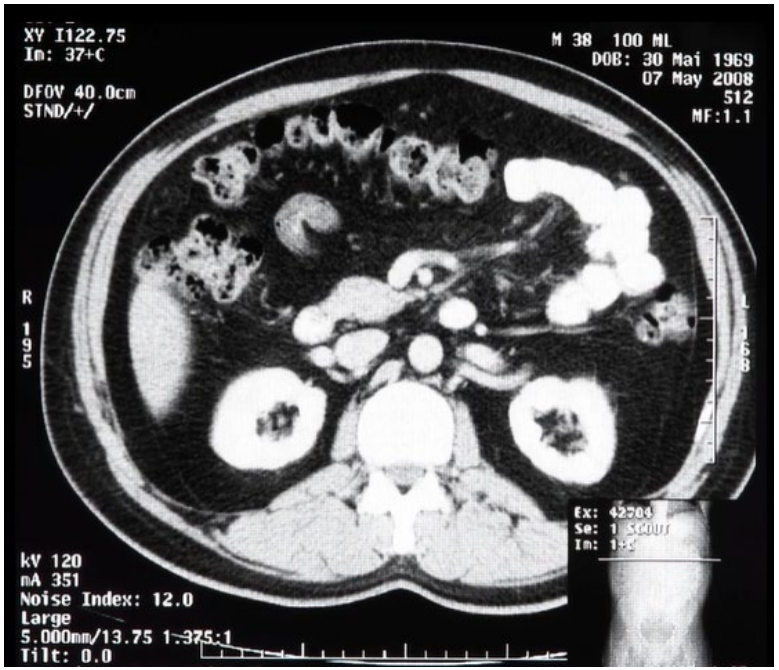
What for?

- Extracting information from Image.

E.g.:

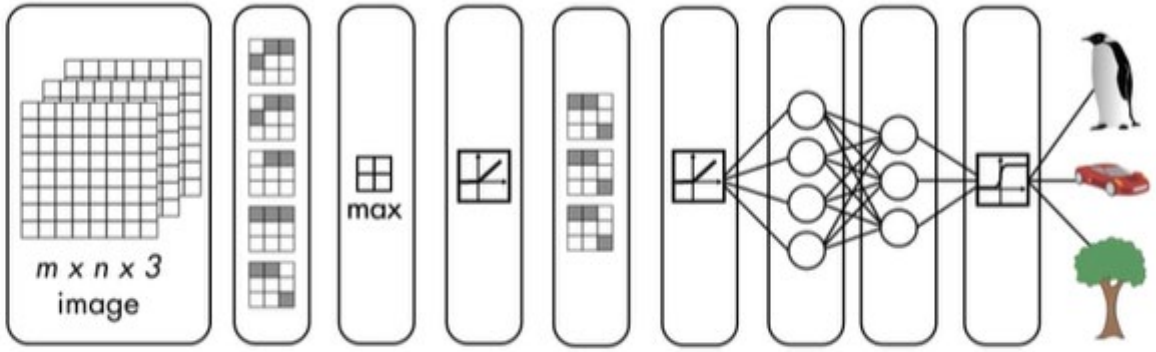
- Hand-written text recognition
- Autonomous Vehicles
- Medical preliminary diagnosis from the image





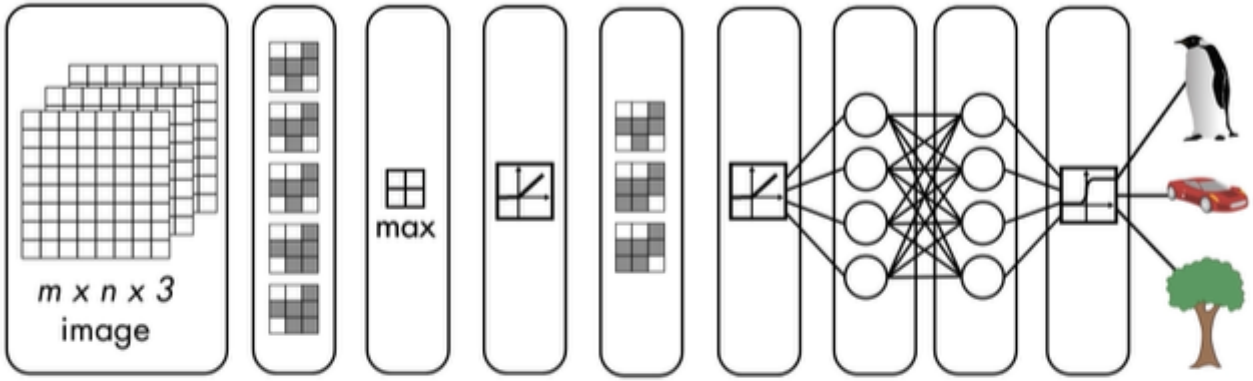
General Scheme

- Extracting information from data.



General Scheme

- As in previous applications, Matlab will help implementing the mathematical formulas.



$$E(\theta) = -\sum_{i=1}^{n}\sum_{j=1}^{k} t_{ij} \ln y_j(x_i, \theta)$$

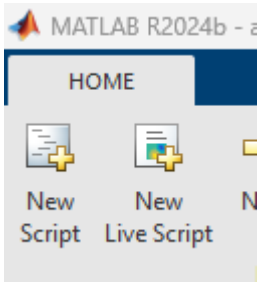
$$\theta_{\ell+1} = \theta_\ell - \alpha \nabla E(\theta_\ell) + \gamma(\theta_\ell - \theta_{\ell-1})$$

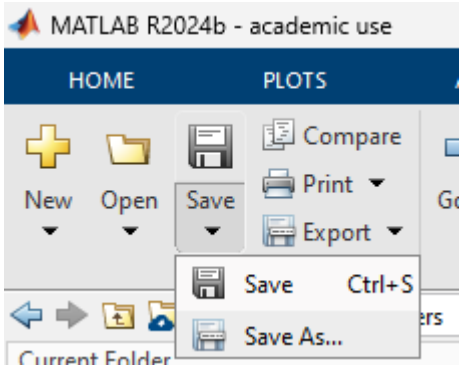
$$y_r(x) = \frac{\exp\,(a_r(x))}{\sum_{j=1}^{k} \exp\,(a_j(x))}$$

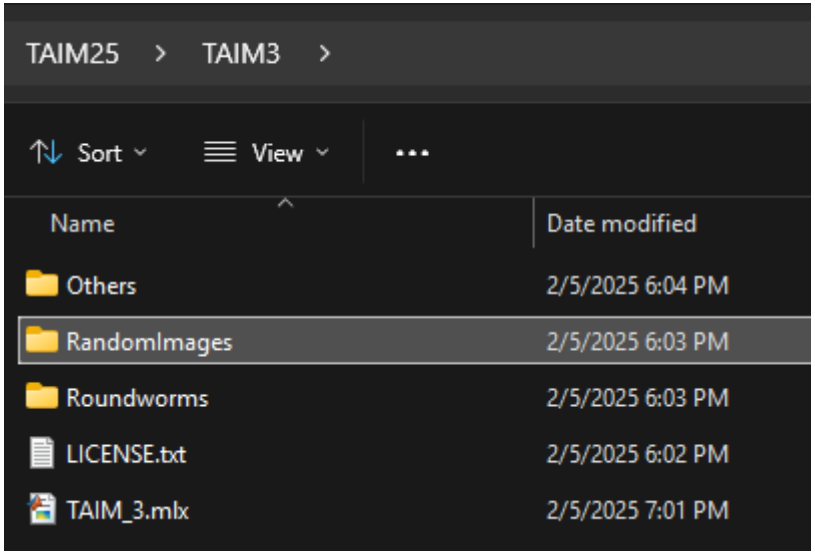
Let's go

Open Matlab

- Do "New Live Script"
- Go "Live Editor"
- And "Save as"
- Create a new Folder (E.g. "**TAIM3**")
- NO SPACES at the NAME

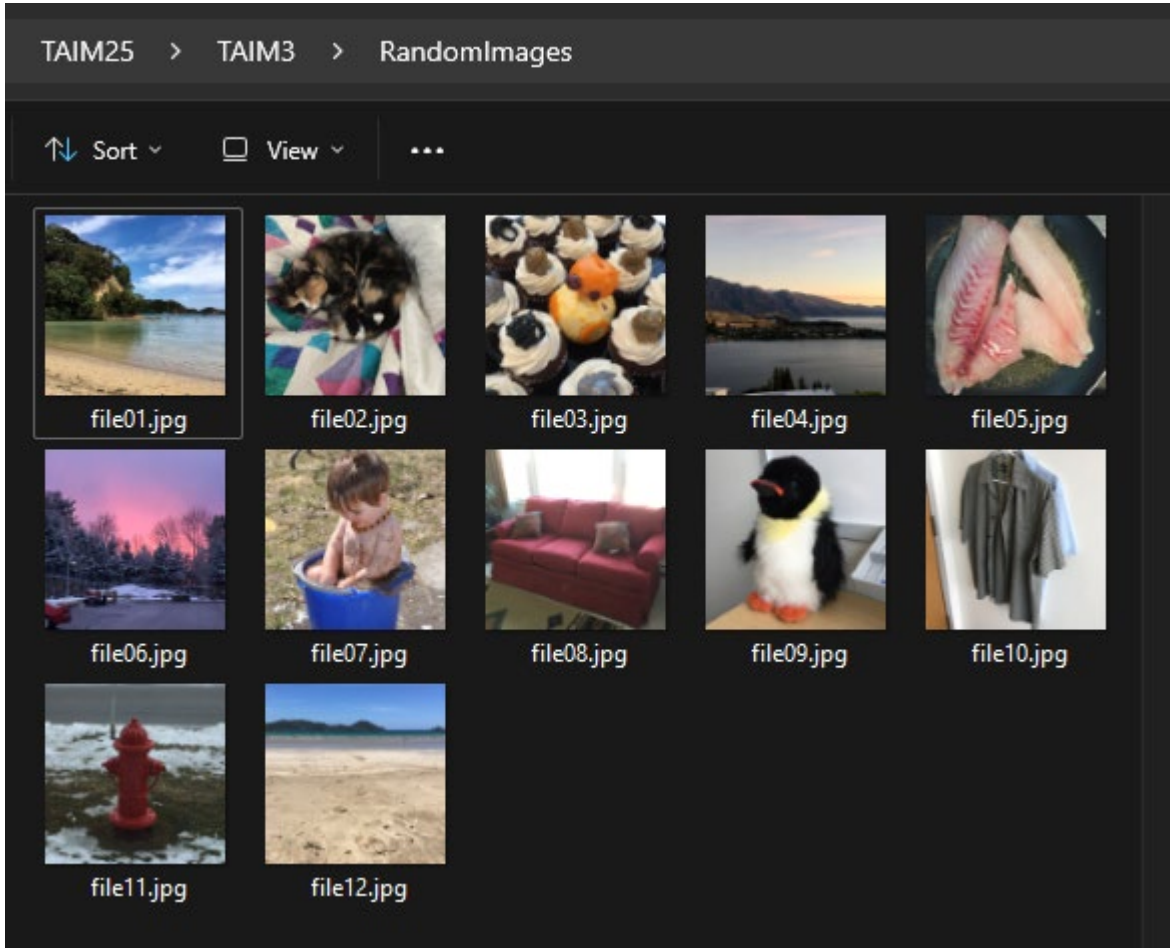






Let's start Preparing DATA

- Download and paste to your "TAIM3" the unzipeed folder "**RandomImages**"



Let's start Preparing DATA

- Download and paste to your "TAIM3" the unzipeed folder "**RandomImages**"

```
All images in this archive are licensed under the Creative Commons By-Attribution License, available at:
https://creativecommons.org/licenses/by/2.0/
The photographers are listed below, thanks to all of them for making their work available, and please be sure to credit them for any use as per the license.

daisy/7568630428_8cf0fc16ff_n.jpg CC-BY by A Guy Taking Pictures - https://www.flickr.com/photos/80901381@N04/7568630428/
daisy/7410356270_9dff4d0e2e_n.jpg CC-BY by martinak15 - https://www.flickr.com/photos/martinaphotography/7410356270/
daisy/4286053334_a75541f20b_m.jpg CC-BY by jenny downing - https://www.flickr.com/photos/jenny-pics/4286053334/
daisy/10770585085_4742b9dac3_n.jpg CC-BY by Thangaraj Kumaravel - https://www.flickr.com/photos/kumaravel/10770585085/
daisy/8759177308_951790e00d_m.jpg CC-BY by Marina del Castell - https://www.flickr.com/photos/marinadelcastell/8759177308/
daisy/4131565290_0585c4dd5a_n.jpg CC-BY by jenny downing - https://www.flickr.com/photos/jenny-pics/4131565290/
daisy/3504430338_77d6a7fab4_n.jpg CC-BY by Dhilung Kirat - https://www.flickr.com/photos/dhilung/3504430338/
daisy/3084924076_4d5c5711af_m.jpg CC-BY by Bùi Linh Ngân - https://www.flickr.com/photos/linhngan/3084924076/
daisy/2642408410_61545fdc83_n.jpg CC-BY by Dennis Jarvis - https://www.flickr.com/photos/archer10/2642408410/
daisy/8710109684_e2c5ef6aeb_n.jpg CC-BY by Jon Bunting - https://www.flickr.com/photos/84744710@N06/8710109684/
daisy/2612704455_efce1c2144_m.jpg CC-BY by Swaminathan - https://www.flickr.com/photos/araswami/2612704455/
daisy/8021540573_c56cf9070d_n.jpg CC-BY by martinak15 - https://www.flickr.com/photos/martinaphotography/8021540573/
```

licensed under the Creative Commons By-Attribution License, available at: https://creativecommons.org/licenses/by/2.0/

Let's start coding

- Let's load and see images by:

"img1 = imread("file01.jpg")

imshow(img1)"

How you would show some other images? → Do it!

```
img1 = 227×227×3 uint8 array
img1(:,:,1) =

    90    90    89    87    85    84    83    82    79    80    82    85    89
    91    91    90    88    87    85    84    84    80    81    82    84    88
    93    92    91    90    89    88    87    87    82    82    82    83    85
    95    94    94    93    92    91    91    90    86    85    84    84    85
    97    97    96    96    95    95    95    94    92    91    87    86    88
    99    99    99    99    98    98    98    98    96    95    94    91    91
   100   100   101   101   101   101   101   101   101    99    98    98    95
   101   101   102   102   102   102   102   103   103   103   101   101    98
   103   102   115   105   106   113   108   104    94    95    96    97    99
   112   100   105   108   109   110   112    96    98   111   105   102   104
   120   132   120   120   112   102    96   123   120    94   107   103   109
   128    90   118   119   117   109    51    84    85    63    95    63    81
   119   100   128   117   124   120    75    64    59    81    79    72    60
   134   145    67    77    82    62    71    89   110    92    73    48    50
```

Using Neural Networks

- Starting “from scratch” is hard.
- Pretrained networks have been trained by experts to classify images into hundreds of categories.
- We will 1st load a pretrained network into MATLAB and then use the network with new images.

```
% Create a copy of a pretrained deep network
net = imagePretrainedNetwork
net =
dlnetwork with properties:

Layers: [68x1 nnet.cnn.layer.Layer]
Connections: [75x2 table]
Learnables: [52x3 table]
State: [0x3 table]
InputNames: {'data'}
OutputNames: {'prob_flatten'}
Initialized: 1

View summary with summary.

% By default, the function provides a deep network named
“dlnetwork”.
% This network was trained on a data set called ImageNet,
% which contains 1000 classes, or categories.
```

Using Neural Networks

Let's ask this "dlnetwork" to predict file01 to file03 and see the result.

```
% You can use the minibatchpredict function to predict the subject of an image.
% The result is a vector of prediction scores, one for each class that the network net can predict.
% scores = minibatchpredict(net,img)

s1 = minibatchpredict(net,img1)
```

s1 = *1x1000 single row vector*
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 …

Using Neural Networks

% plot(s) to see what do we have obtained
% How do we know what means the result?
% Yes, we forgot to "load" the "classNames" (1000 classes, or categories)
% of “dlnetwork”

% We can do by:

```
[net,classes] = imagePretrainedNetwork
net =
dlnetwork with properties:

Layers: [68x1 nnet.cnn.layer.Layer]
Connections: [75x2 table]
Learnables: [52x3 table]
State: [0x3 table]
InputNames: {'data'}
OutputNames: {'prob_flatten'}
Initialized: 1

View summary with summary.

classes = 1000x1 string
"tench"
"goldfish"
"great white s…
"tiger shark"
"hammerhead"
"electric ray"
"stingray"
"cock"
"hen"
"ostrich"
```

Using Neural Networks

```
% Now the file01 is:
c1 = scores2label(s1,classes)
```

c1 = categorical
seashore

```
% Now to predict file02 and file03 and see the result:

img2 = imread("file02.jpg");
imshow(img2)

s2 = minibatchpredict(net,img2)

c2 = scores2label(s2,classes)
```

c2 = categorical
Gordon setter

Using Neural Networks

```
img3 = imread("file03.jpg");
imshow(img3)

s3 = minibatchpredict(net,img3)

c3 = scores2label(s3,classes)
```

c3 = categorical
bakery

Other Neural Networks?

Let's try another networl to predict file01 to file03 and see the result (E.g. "GoogLeNet")

```
[net,classes] = imagePretrainedNetwork("googlenet")
net =
dlnetwork with properties:

Layers: [143x1 nnet.cnn.layer.Layer]
Connections: [169x2 table]
Learnables: [116x3 table]
State: [0x3 table]
InputNames: {'data'}
OutputNames: {'prob'}
Initialized: 1

View summary with summary.

classes = 1000x1 string
"tench"
"goldfish"
"great white s…
"tiger shark"
"hammerhead"
"electric ray"
"stingray"
"cock"
"hen"
"ostrich"
```

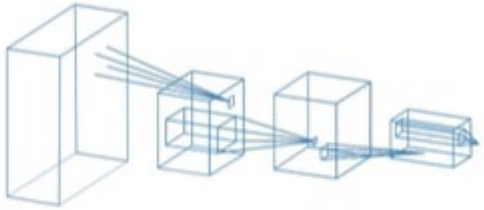
```
% GoogLeNet has the same prediction
% for the seashore and couch, but predicts that the cat image
is a different dog breed.
```

```
% You can try different pretrained networks
% by looking at the model name arguments in the
imagePretrainedNetwork
% documentation
```

Other Neural Networks?

MATLAB

Deep Learning Toolbox

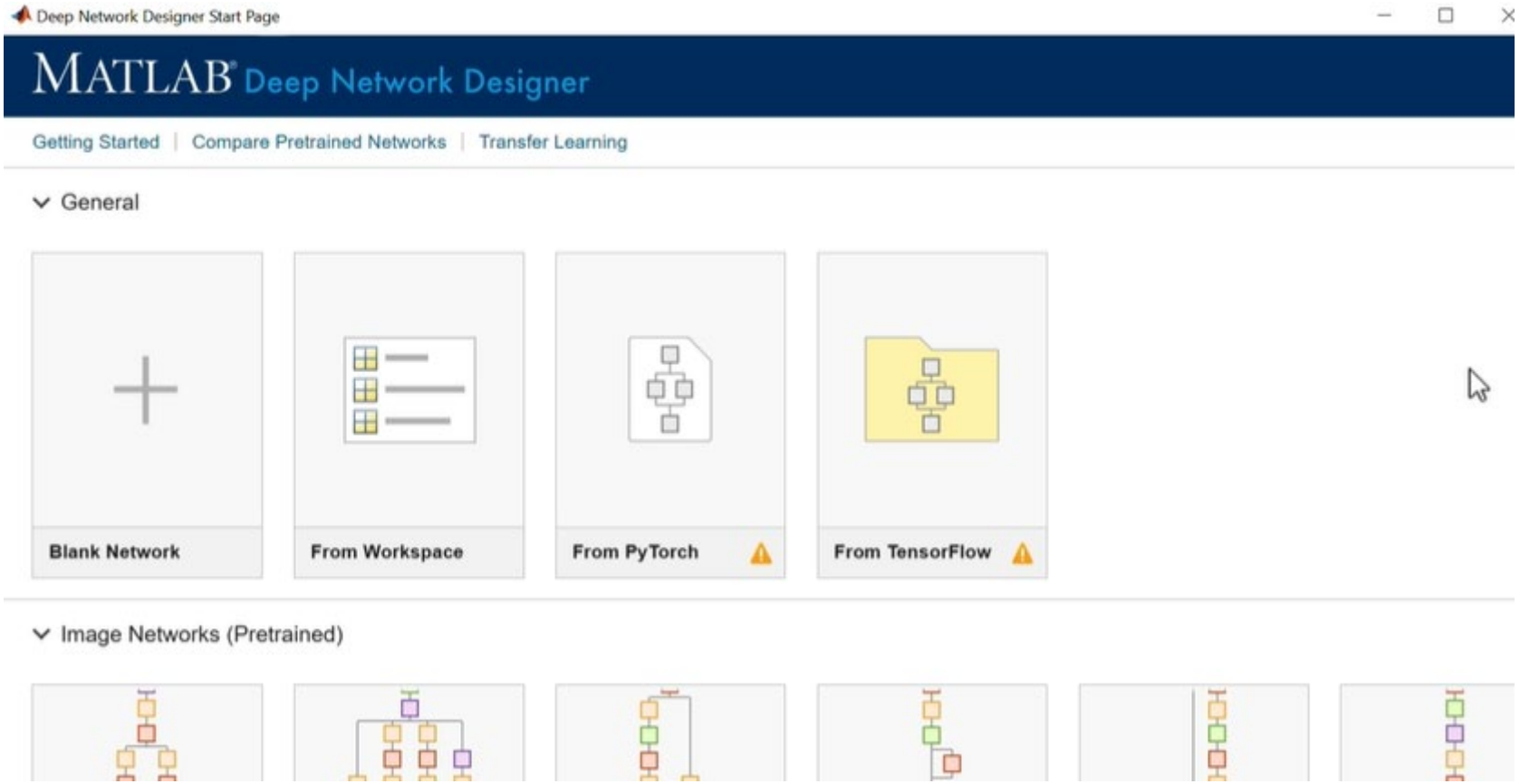


Pretrained Network

- Matlab provides access to many of them
- In local Version, you will need to install most of them
- In **Matlab Online**, you have them all preinstalled.

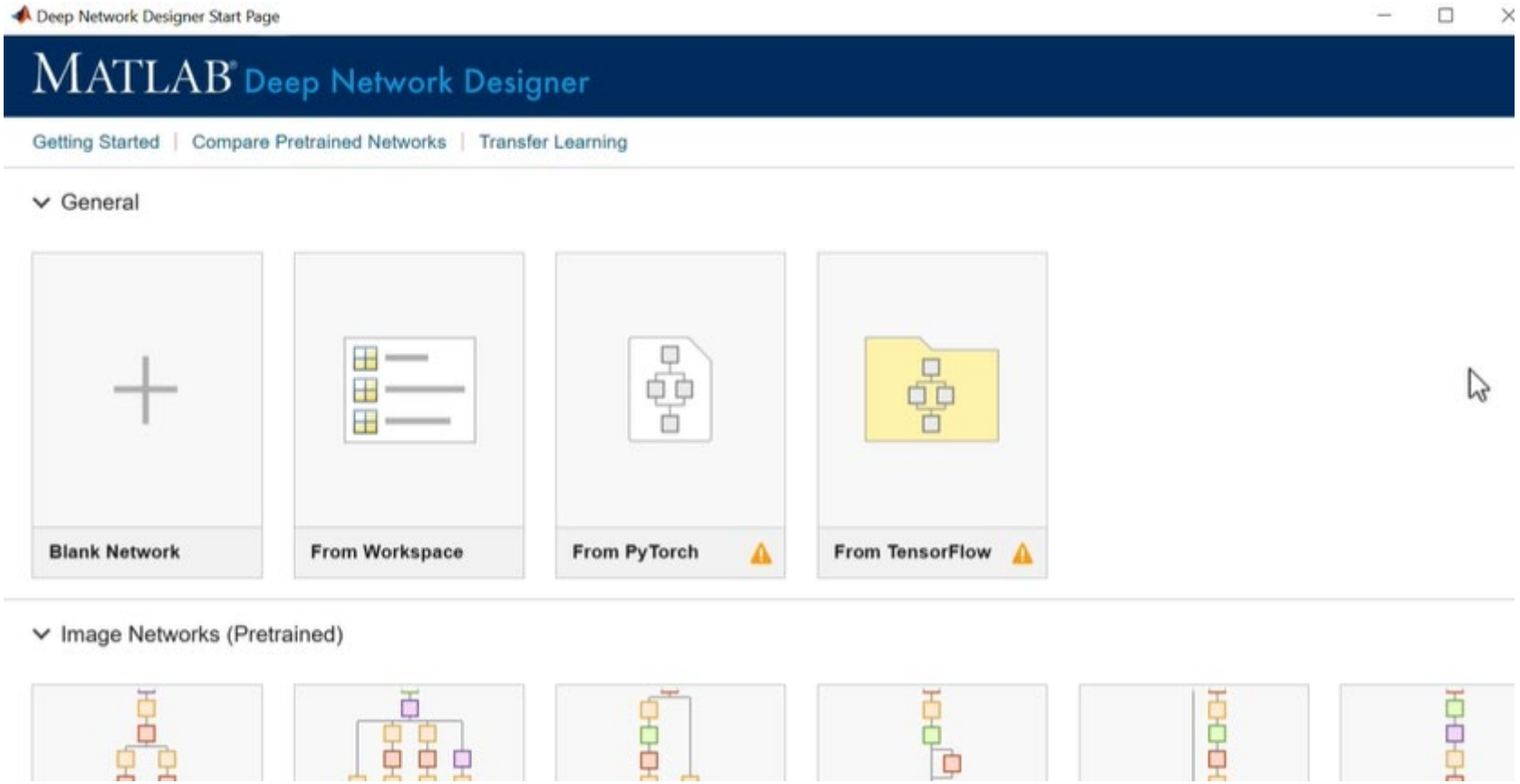
Yes we have also a Matlab App: Classification Learner

- Following similar approach as with the Curve Fitter App



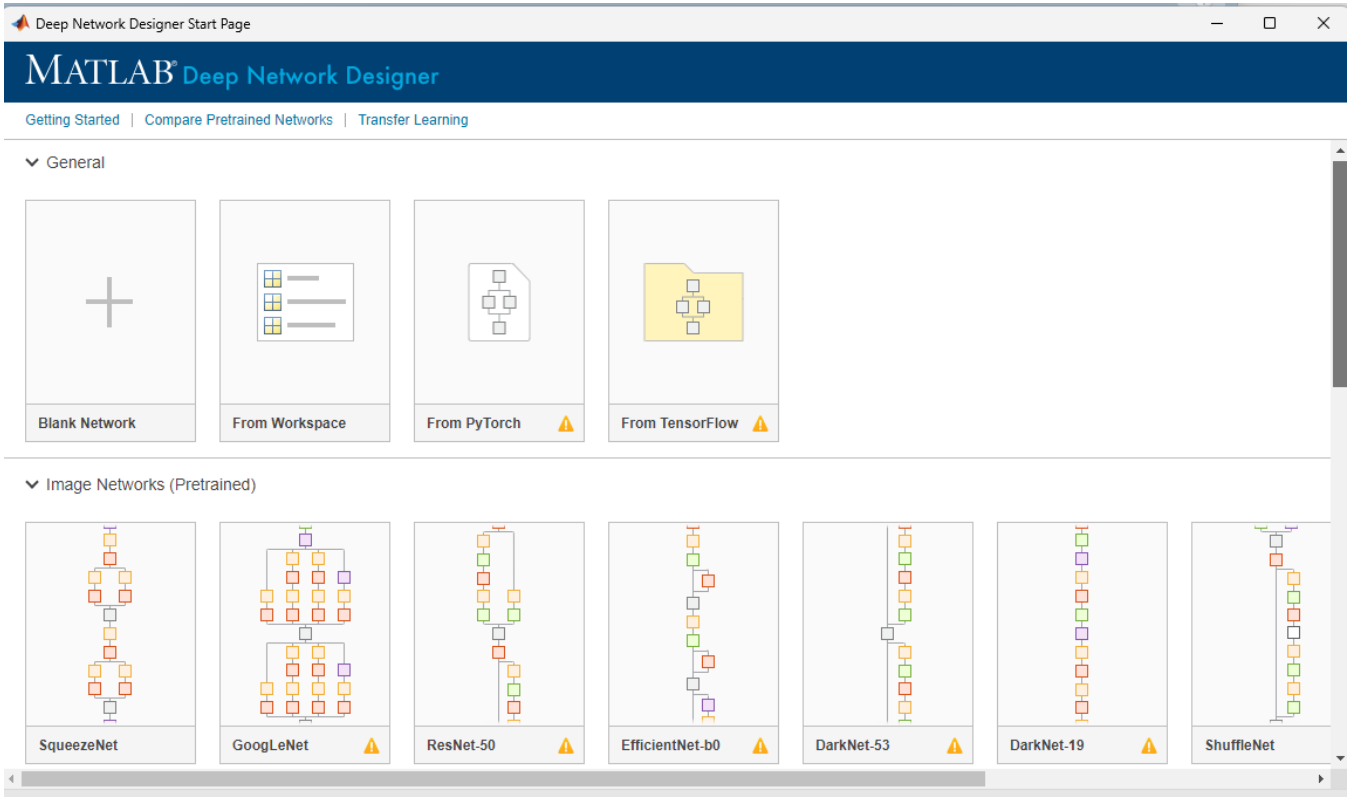
Yes we have also a Matlab App: Classification Learner

- Following similar approach as with the Curve Fitter App



Yes we have also a Matlab App: Classification Learner

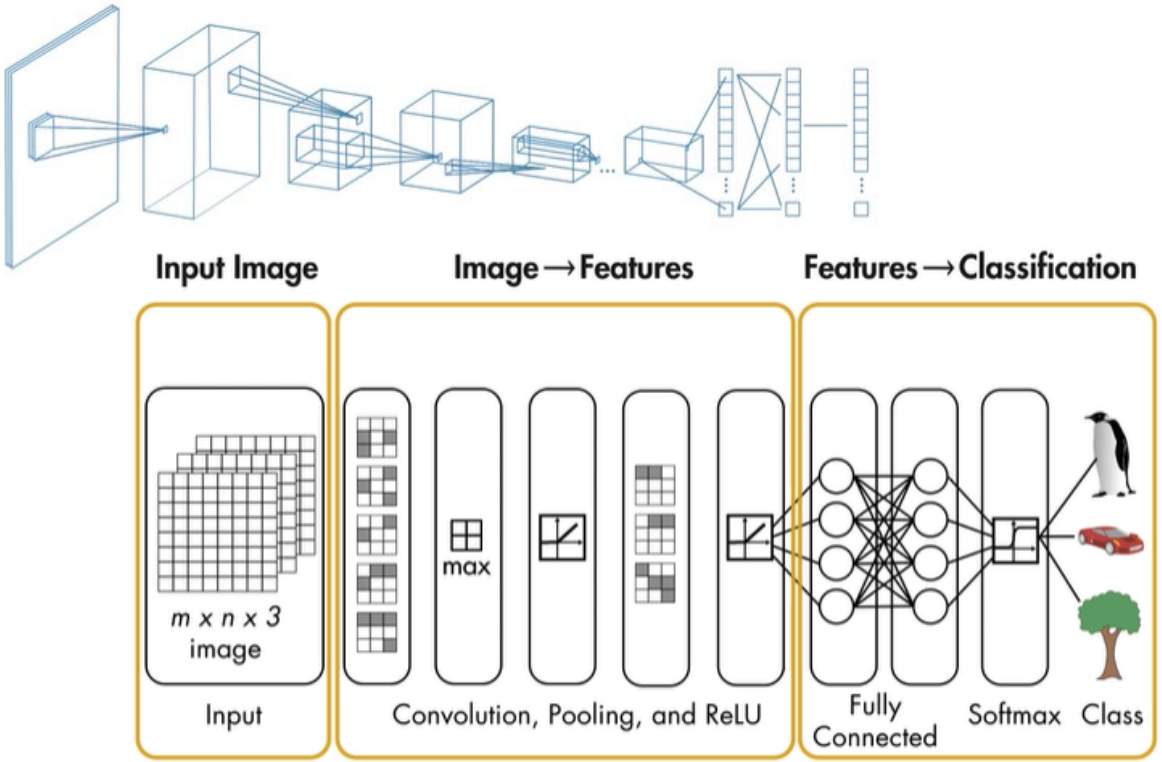
- SqueezeNet is included with your installation, so it can be used right away. Networks with the warning icon need to be installed, so click on the network you want to try. This opens the **Add-On Explorer**.



- To avoid multiple installations, we will use the **Matlab Online**

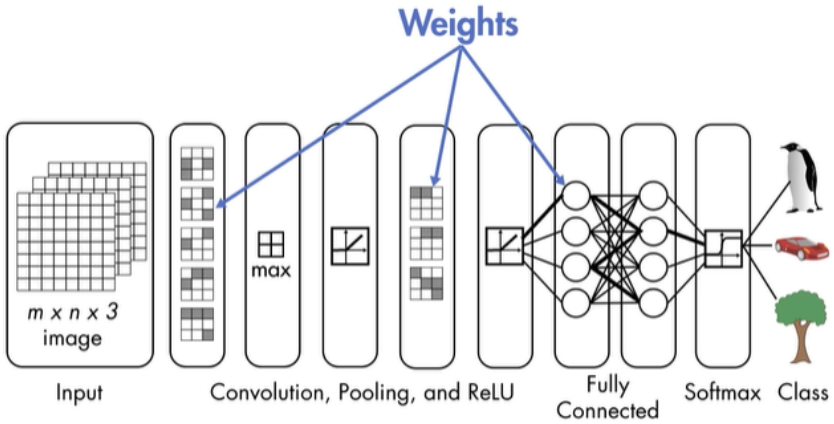
Convolutional Neural Network

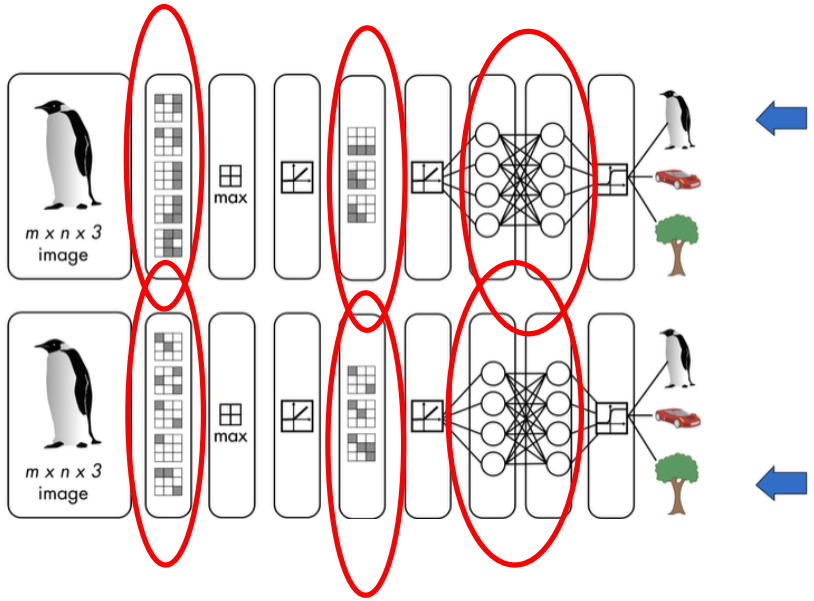
- CNNs are specialized neural networks for inputs with a two-dimensional structure (as images)
- Network architecture:
  - Various layers (key part of the network architect)
  - Layers contain parameters (“weights”)
  - The “weights” are learned by training on labeled data.
- Output layer of a CNN provides predictions for each possible class.



Convolutional Neural Network

- CNNs are specialized neural networks for inputs with a two-dimensional structure (as images)
- Network architecture:
  - Various layers (key part of the network architect)
  - Layers contain parameters (“weights”)
  - The “weights” are learned by training on labeled data.
- Output layer of a CNN provides predictions for each possible class.

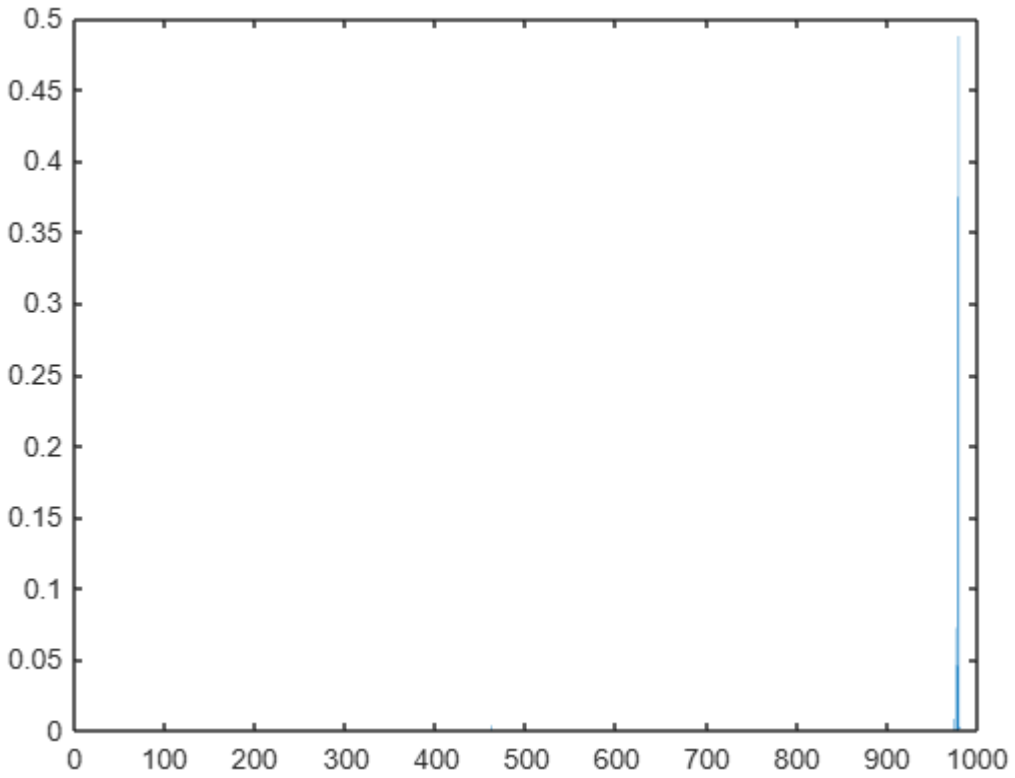




Examining Predictions

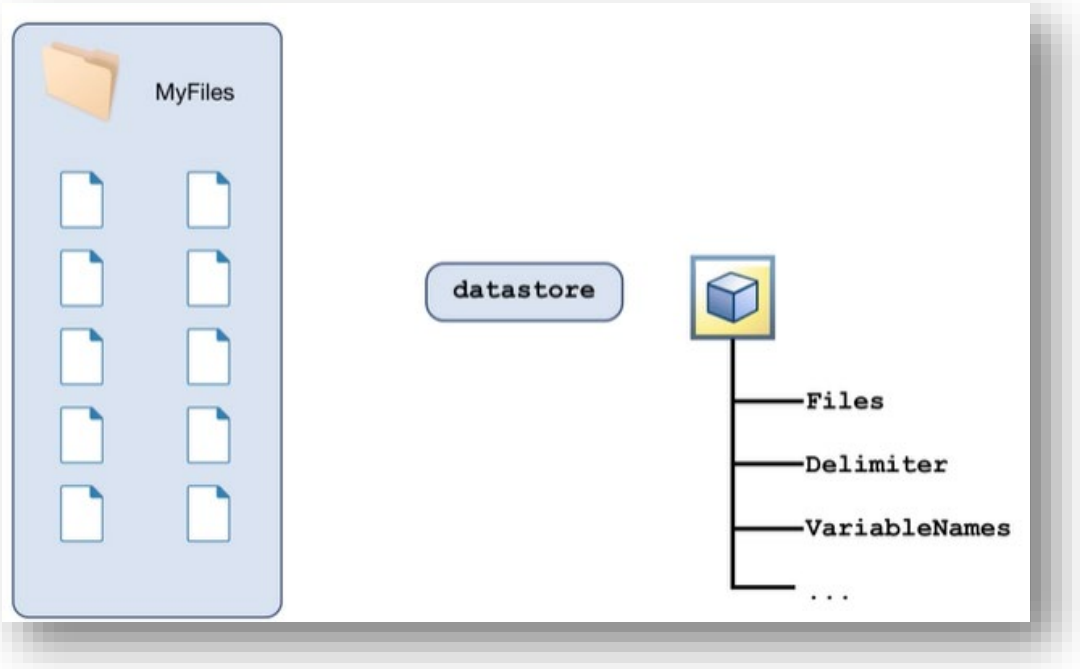
```
% Let's analize predictions for the file01.jpg
scores = minibatchpredict(net,img1)
```

```
% Let's draw a "bar“ plot
bar(scores)
```



Let's go for a "datastore" as in ML

Now is your turn to do it for file02.jpg and file03.jpg



Create a Datastore for images

ls ("./RandomImages/*.jpg")

./RandomImages/file01.jpg ./RandomImages/file05.jpg ./RandomImages/file09.jpg ./RandomImages/file02.jpg
./RandomImages/file06.jpg ./RandomImages/file10.jpg ./RandomImages/file03.jpg ./RandomImages/file07.jpg
./RandomImages/file11.jpg ./RandomImages/file04.jpg ./RandomImages/file08.jpg ./RandomImages/file12.jpg

imds = imageDatastore("./RandomImages/*.jpg")

```
imds =
  ImageDatastore with properties:

                      Files: {
                             '/MATLAB Drive/TAIM25/TAIM3/RandomImages/file01.jpg';
                             '/MATLAB Drive/TAIM25/TAIM3/RandomImages/file02.jpg';
                             '/MATLAB Drive/TAIM25/TAIM3/RandomImages/file03.jpg'
                              ... and 9 more
                             }
                    Folders: {
                             '/MATLAB Drive/TAIM25/TAIM3/RandomImages'
                             }
    AlternateFileSystemRoots: {}
                   ReadSize: 1
                     Labels: {}
      SupportedOutputFormats: ["png"    "jpg"    "jpeg"    "tif"    "tiff"]
         DefaultOutputFormat: "png"
                     ReadFcn: @readDatastoreImage
```

Create a Datastore for images

```
fname = imds.Files

img = readimage(imds,7);
imshow(img)

scores = minibatchpredict(net,imds)

preds = scores2label(scores,classes)

max(scores,[],2)
```

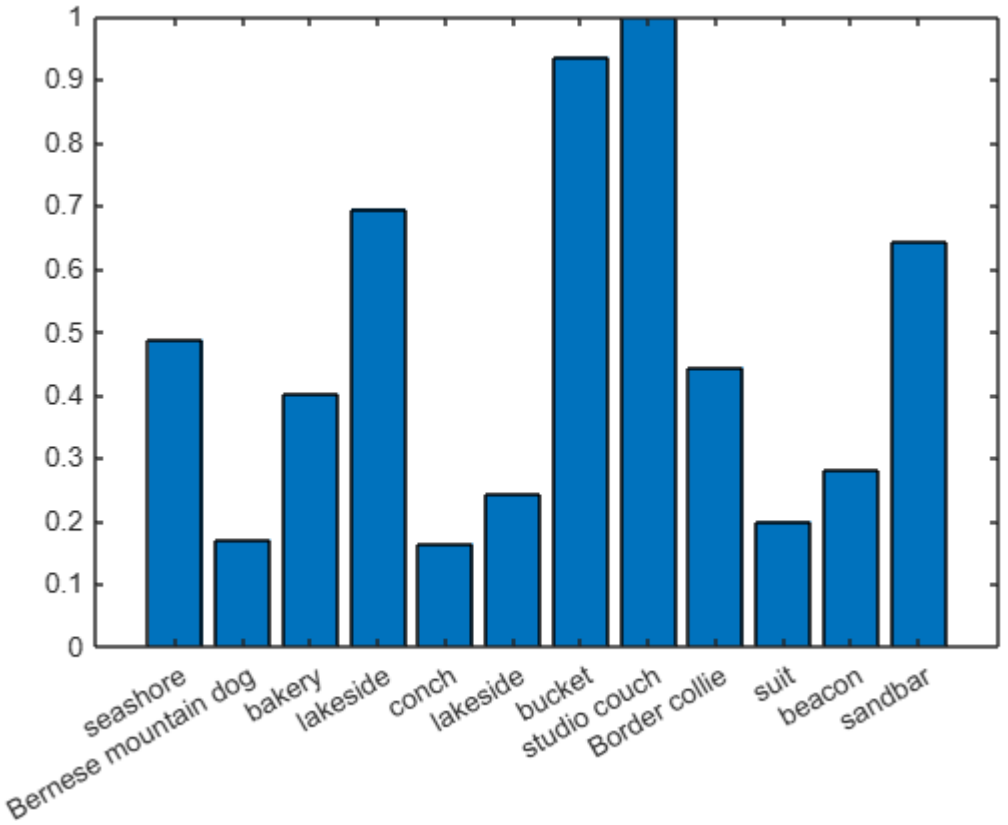
```
ans = 12x1 single column vector
    0.4867
    0.1700
    0.4020
    0.6952
    0.1643
    0.2414
    0.9361
    0.9991
    0.4436
    0.1995
    ⋮
```

```
preds = 12x1 categorical
    seashore
    Bernese mountain d…
    bakery
    lakeside
    conch
    lakeside
    bucket
    studio couch
    Border collie
    suit
      ⋮
```

Create a Datastore for images

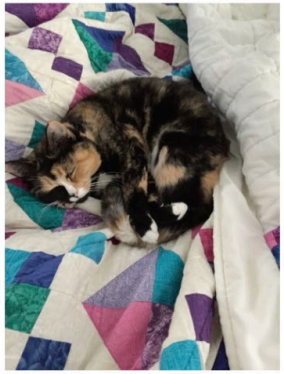
bar(max(scores,[],2))

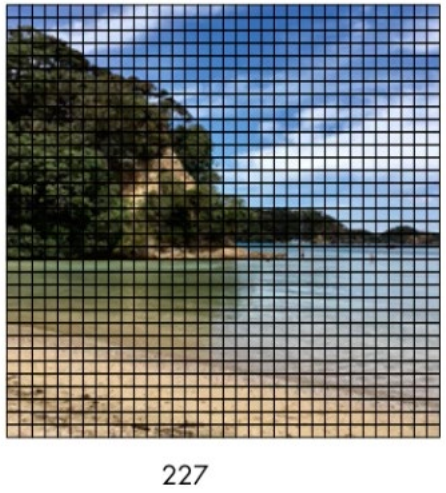
xticklabels(preds)

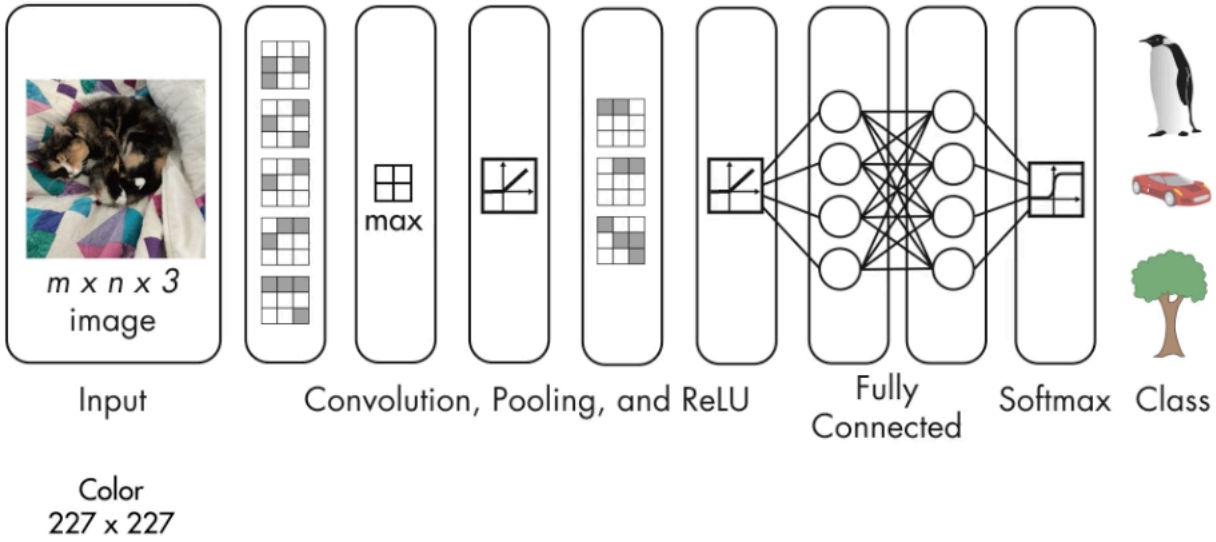


Adjusting Input Images

- What if the images does not have the “standard” characteristics expected by the pretrained NN?



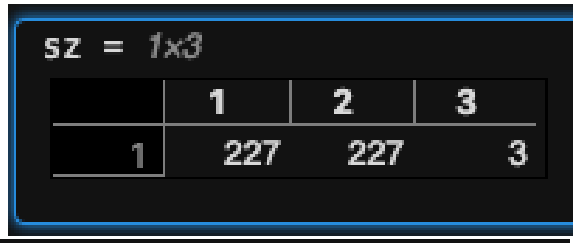




Adjusting Input Images: going to code

Adjust Input Images

```
% Let's see initial size of image file01.pjg
img = imread("file01.jpg");
imshow(img)
sz = size (img)
```




```
% On the other hand, let's see the main
characteristics of the Network used:
summary(net)
```

```
Initialized: true

Number of learnables: 6.9M

Inputs:
   1   'data'   224x224x3 images
```

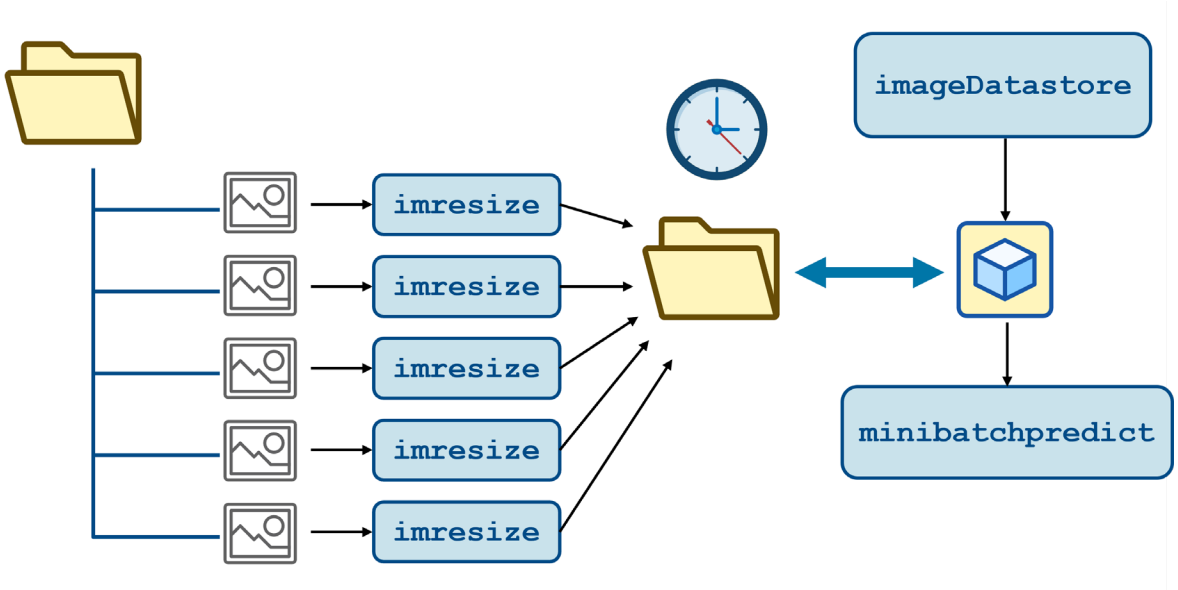
```
% As initial size does not match, let's resize the image to the demands of
% the network
img = imresize (img, [224 224])
imshow (img)
```

```
Initialized: true

Number of learnables: 6.9M

Inputs:
   1   'data'   224x224x3 images
```

```
% Finally, does it changes the result. Let's hope not!
s = minibatchpredict(net,img)
c = scores2label(s,classes)
```

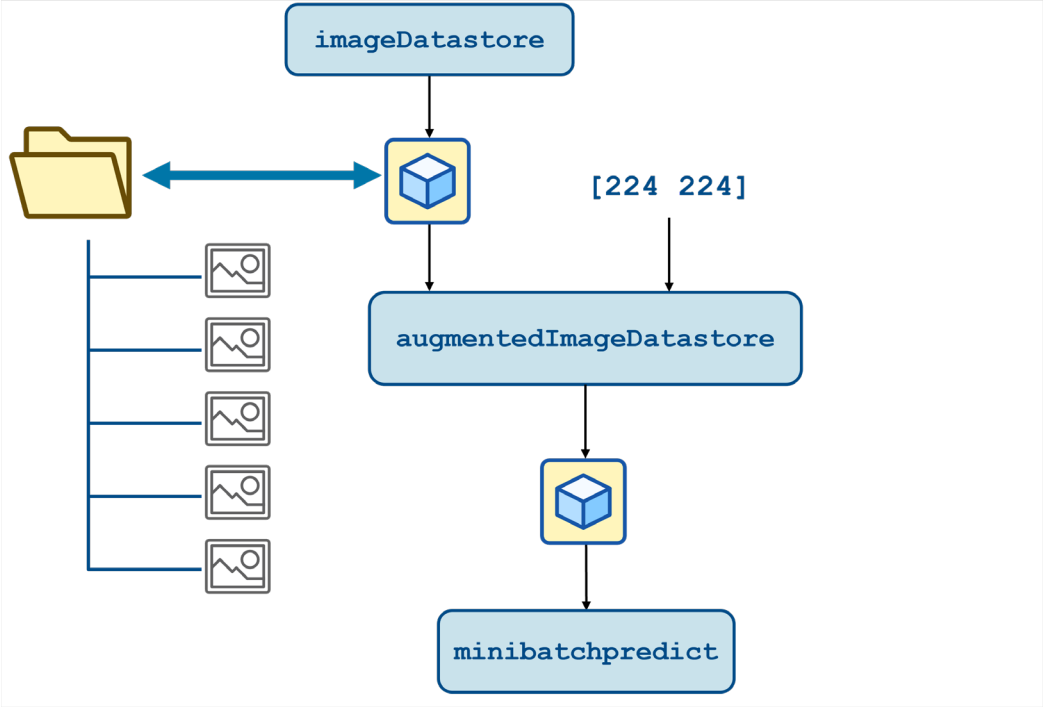
Adjusting Input Images: Now over the “datastore”

- To use an image datastore, you also need to save the processed images to a folder.
- For large data sets, saving a duplicate of your files can take up a lot of space.



Adjusting Input Images: Now over the “datastore”

- Alternatively, perform basic preprocessing with the “augmentedImageDatastore" function
  (it takes an image datastore and an image size as inputs and the datastore can be used as input to the minibatchpredict function).



Adjusting Input Images: going to code

Resize Images in a Datastore

% As indicarted, we can create a completed "resized" image data store by:

auds = augmentedImageDatastore([224 224],imds)

```
auds =
  augmentedImageDatastore with properties:

           NumObservations: 12
                     Files: {12x1 cell}
   AlternateFileSystemRoots: {}
              MiniBatchSize: 128
          DataAugmentation: 'none'
      ColorPreprocessing: 'none'
                OutputSize: [224 224]
            OutputSizeMode: 'resize'
       DispatchInBackground: 0
```

```
preds = 12x1 categorical
    seashore
    Bernese mountain d…
    bakery
    lakeside
    chiton
    lakeside
    bucket
    studio couch
    Border collie
    vestment
      ⋮
```

% So that we can process the whole resized collection of images directly:

scores = minibatchpredict(net,auds)

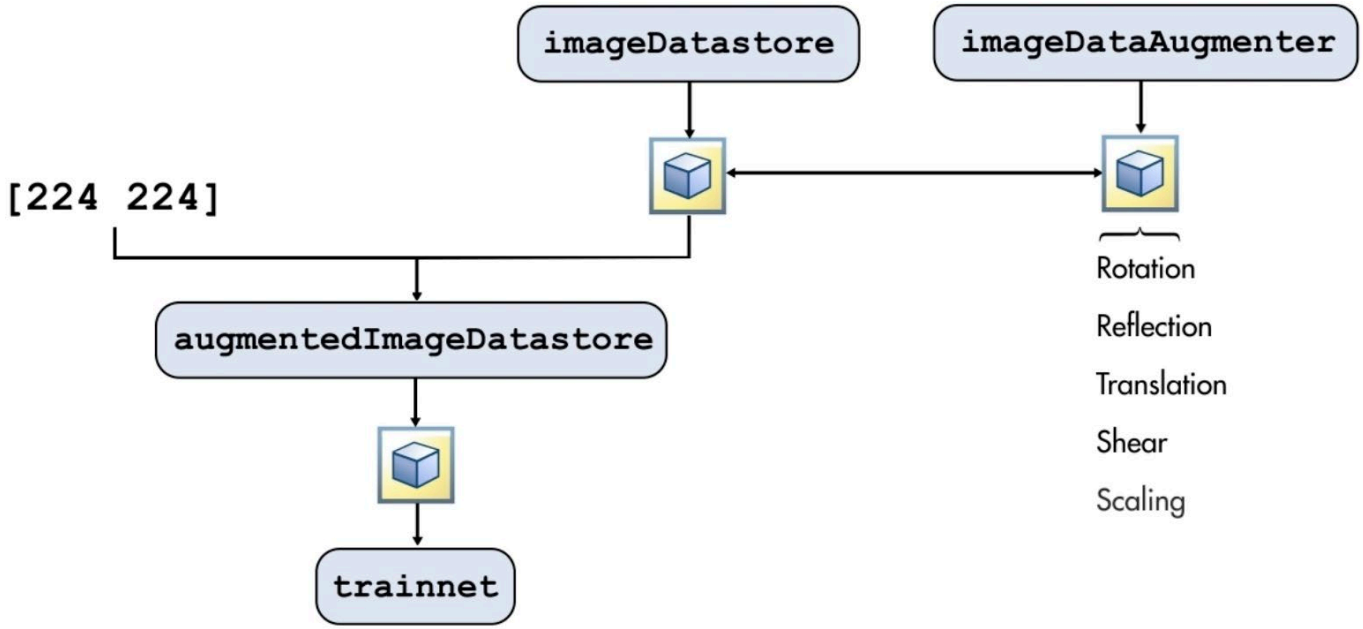
preds = scores2label(scores,classes)

Convert to RGB Images

- Maybe your training images may not be sufficiently varied.
- For example, to build a mobile phone app, you really want the classifier to be able to cope with imperfect images, odd angles, substandard lighting, not well-framed, or cropped, etc.

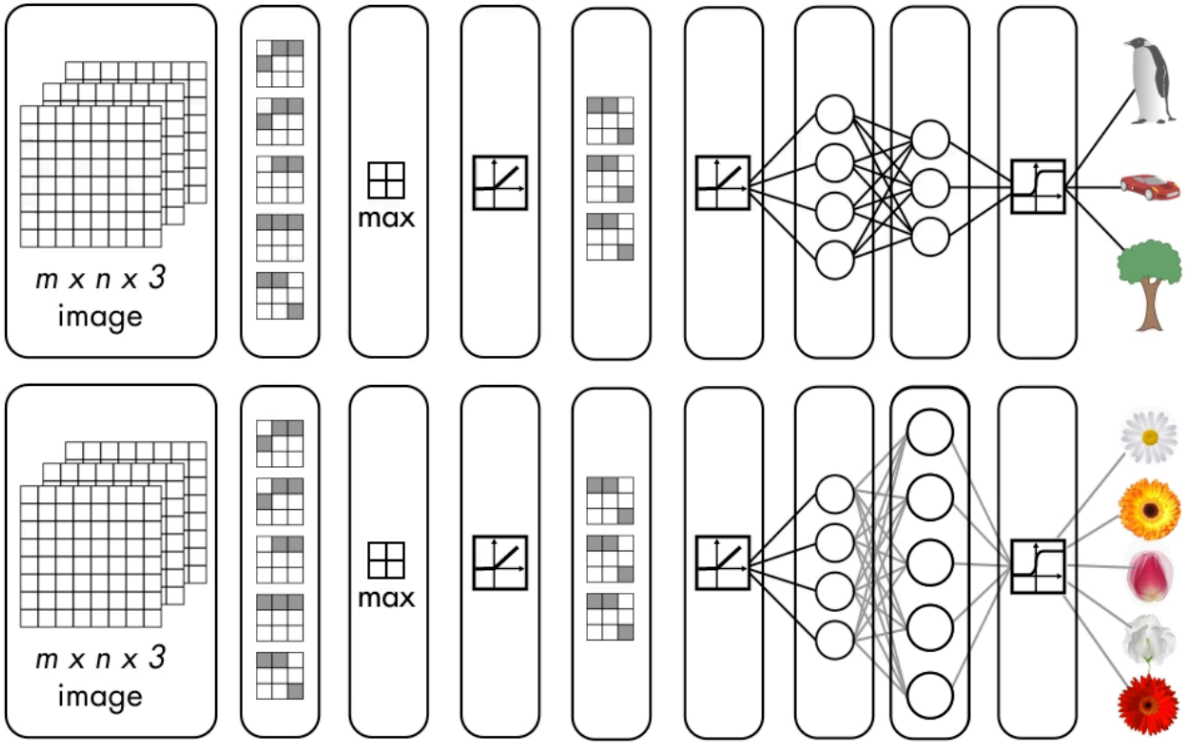
Convert to RGB Images

- Use the “imageDataAugmenter” function to set up a set of transformations to your images to make an augmented training set.
- These transformations include: rotations, reflections, translations, shears, and scaling.
- You then pass the imageDataAugmenter variable to the “augmentedImageDatastore” function as an optional input.

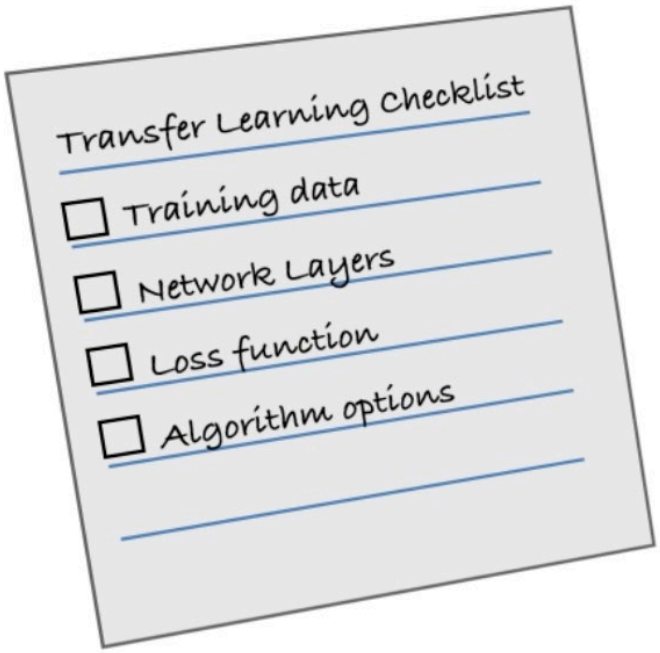


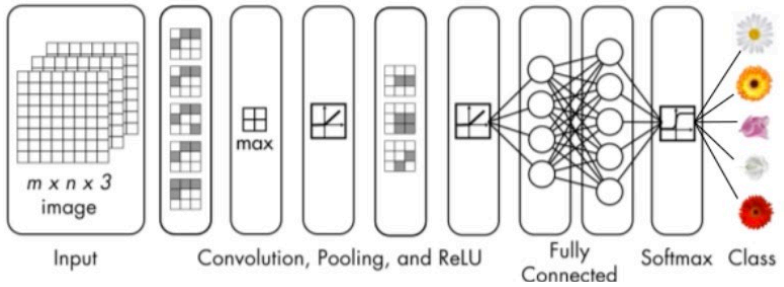
What is Transfer Learning?

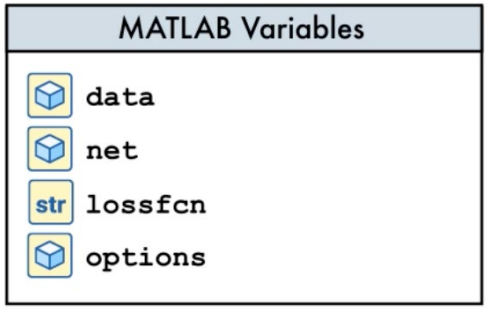
- Pretrained networks can classify a wide range of categories but may not cover your application's classes.
- Transfer learning involves taking a pretrained network, modifying it, and retraining it on new, specific data to fit the desired classification task.



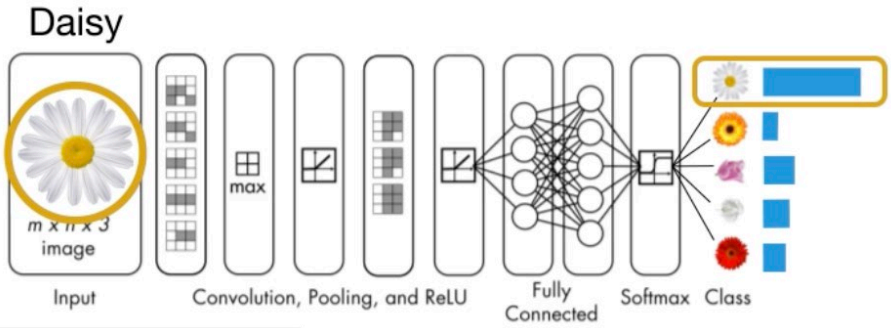
What do we need for Transfer Learning?







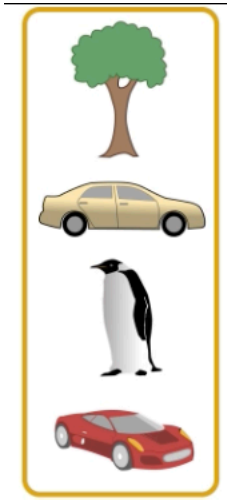
Batch size
Max iterations
Learning rate
...



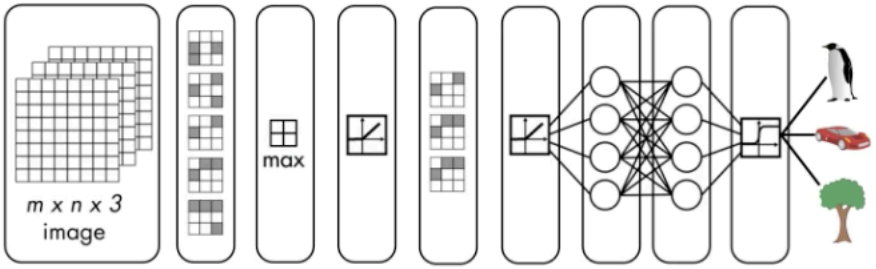
MATLAB Commands

```
newnet = trainnet(data,net,lossfcn,options)
```

As with Machine Learning, we need to slit data for Training and Testing



Training



tree

car

penguin

car

car

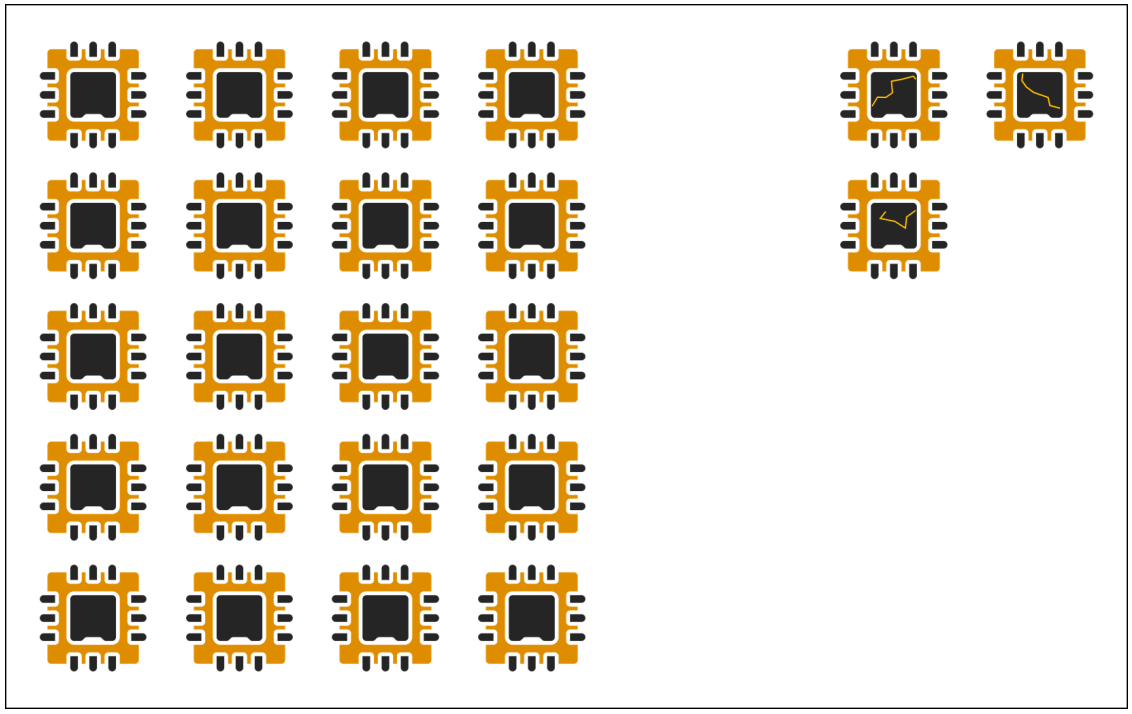
penguin

tree

Testing

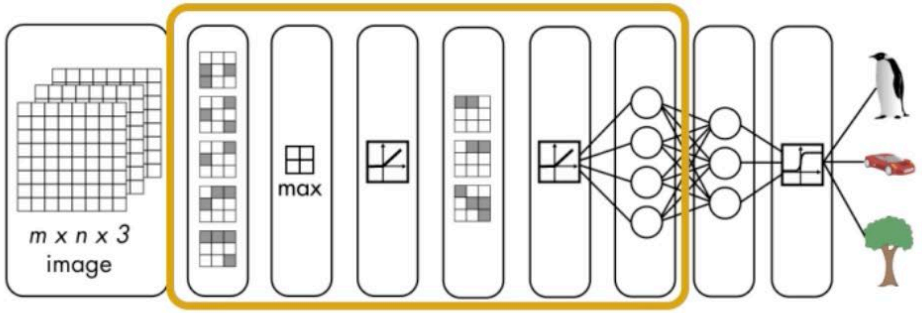
Unbalanced Training Data

- Sometimes, you have many more images of one class than another.
- For example, if you try to detect defects, obtaining many images with no defect is usually easy, but obtaining images with defects is harder.
- In this case, this imbalance can bias the training.
- Splitting the data so that the training images have equal numbers from each class helps to avoid this bias.



How to do transfer learning at a CNN

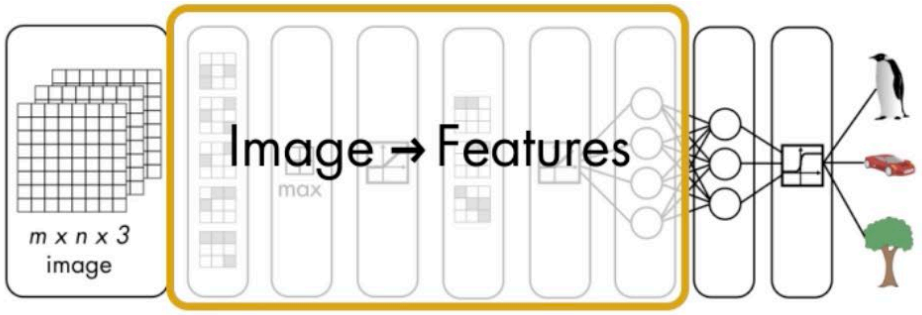
- Most of the layers of the pretrained network are: convolution, pooling, and rectified linear unit layers. These take the original input image and extract various features that can then be used for classification.



1	'data'	Image Input	224x224x3 images with 'zerocenter' normalization
2	'conv1-7x7_s2'	Convolution	64 7x7x3 convolutions with stride [2 2] and padding [3 3 3 3]
3	'conv1-relu_7x7'	ReLU	ReLU
4	'pool1-3x3_s2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 1 0 1]
5	'pool1-norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
6	'conv2-3x3_reduce'	Convolution	64 1x1x64 convolutions with stride [1 1] and padding [0 0 0 0]
7	'conv2-relu_3x3_reduce'	ReLU	ReLU
8	'conv2-3x3'	Convolution	192 3x3x64 convolutions with stride [1 1] and padding [1 1 1 1]
9	'conv2-relu_3x3'	ReLU	ReLU
10	'conv2-norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
11	'pool2-3x3_s2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 1 0 1]
12	'inception_3a-1x1'	Convolution	64 1x1x192 convolutions with stride [1 1] and padding [0 0 0 0]
13	'inception_3a-relu_1x1'	ReLU	ReLU
14	'inception_3a-3x3_reduce'	Convolution	96 1x1x192 convolutions with stride [1 1] and padding [0 0 0 0]
15	'inception_3a-relu_3x3_reduce'	ReLU	ReLU
16	'inception_3a-3x3'	Convolution	128 3x3x96 convolutions with stride [1 1] and padding [1 1 1 1]
17	'inception_3a-relu_3x3'	ReLU	ReLU
18	'inception_3a-5x5_reduce'	Convolution	16 1x1x192 convolutions with stride [1 1] and padding [0 0 0 0]
19	'inception_3a-relu_5x5_reduce'	ReLU	ReLU
20	'inception_3a-5x5'	Convolution	32 5x5x16 convolutions with stride [1 1] and padding [2 2 2 2]
21	'inception_3a-relu_5x5'	ReLU	ReLU
22	'inception_3a-pool'	Max Pooling	3x3 max pooling with stride [1 1] and padding [1 1 1 1]
23	'inception_3a-pool_proj'	Convolution	32 1x1x192 convolutions with stride [1 1] and padding [0 0 0 0]
24	'inception_3a-relu_pool_proj'	ReLU	ReLU

How to do transfer learning at a CNN

- To perform transfer learning, you'll need to modify the network to suit your specific problem. But don't worry. Most of the network will stay the same. You typically change just the last few layers. For example, here's the end of)0

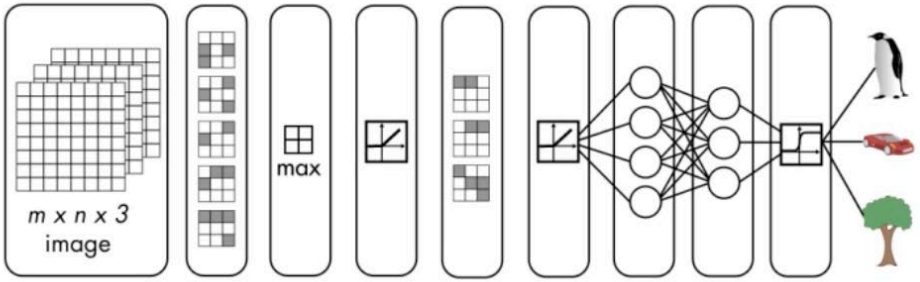


1	'data'	Image Input	224x224x3 images with 'zerocenter' normalization
2	'conv1-7x7_s2'	Convolution	64 7x7x3 convolutions with stride [2 2] and padding [3 3 3 3]
3	'conv1-relu_7x7'	ReLU	ReLU
4	'pool1-3x3_s2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 1 0 1]
5	'pool1-norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
6	'conv2-3x3_reduce'	Convolution	64 1x1x64 convolutions with stride [1 1] and padding [0 0 0 0]
7	'conv2-relu_3x3_reduce'	ReLU	ReLU
8	'conv2-3x3'	Convolution	192 3x3x64 convolutions with stride [1 1] and padding [1 1 1 1]
9	'conv2-relu_3x3'	ReLU	ReLU
10	'conv2-norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
11	'pool2-3x3_s2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 1 0 1]
12	'inception_3a-1x1'	Convolution	64 1x1x192 convolutions with stride [1 1] and padding [0 0 0 0]
13	'inception_3a-relu_1x1'	ReLU	ReLU
14	'inception_3a-3x3_reduce'	Convolution	96 1x1x192 convolutions with stride [1 1] and padding [0 0 0 0]
15	'inception_3a-relu_3x3_reduce'	ReLU	ReLU
16	'inception_3a-3x3'	Convolution	128 3x3x96 convolutions with stride [1 1] and padding [1 1 1 1]
17	'inception_3a-relu_3x3'	ReLU	ReLU
18	'inception_3a-5x5_reduce'	Convolution	16 1x1x192 convolutions with stride [1 1] and padding [0 0 0 0]
19	'inception_3a-relu_5x5_reduce'	ReLU	ReLU
20	'inception_3a-5x5'	Convolution	32 5x5x16 convolutions with stride [1 1] and padding [2 2 2 2]
21	'inception_3a-relu_5x5'	ReLU	ReLU
22	'inception_3a-pool'	Max Pooling	3x3 max pooling with stride [1 1] and padding [1 1 1 1]
23	'inception_3a-pool_proj'	Convolution	32 1x1x192 convolutions with stride [1 1] and padding [0 0 0 0]
24	'inception_3a-relu_pool_proj'	ReLU	ReLU

Image → Features

How to do transfer learning at a CNN

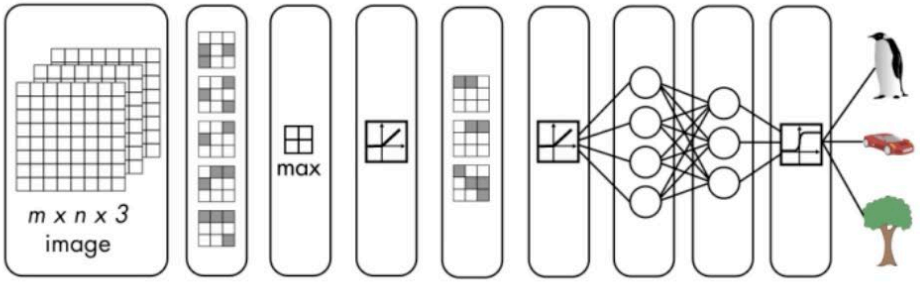
- For transfer learning, you'll need to modify the network to your problem.
- Most of the network will stay the same. (You typically change just the last few layers.)




```
.
.
.
125   'inception_5a-output'            Depth concatenation          Depth concatenation of 4 inputs
126   'inception_5b-1x1'               Convolution                  384 1x1x832 convolutions with stride [1  1] and padding [0  0  0  0]
127   'inception_5b-relu_1x1'          ReLU                         ReLU
128   'inception_5b-3x3_reduce'        Convolution                  192 1x1x832 convolutions with stride [1  1] and padding [0  0  0  0]
129   'inception_5b-relu_3x3_reduce'   ReLU                         ReLU
130   'inception_5b-3x3'               Convolution                  384 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
131   'inception_5b-relu_3x3'          ReLU                         ReLU
132   'inception_5b-5x5_reduce'        Convolution                  48 1x1x832 convolutions with stride [1  1] and padding [0  0  0  0]
133   'inception_5b-relu_5x5_reduce'   ReLU                         ReLU
134   'inception_5b-5x5'               Convolution                  128 5x5x48 convolutions with stride [1  1] and padding [2  2  2  2]
135   'inception_5b-relu_5x5'          ReLU                         ReLU
136   'inception_5b-pool'              Max Pooling                  3x3 max pooling with stride [1  1] and padding [1  1  1  1]
137   'inception_5b-pool_proj'         Convolution                  128 1x1x832 convolutions with stride [1  1] and padding [0  0  0  0]
138   'inception_5b-relu_pool_proj'    ReLU                         ReLU
139   'inception_5b-output'            Depth concatenation          Depth concatenation of 4 inputs
140   'pool5-7x7_s1'                   Global Average Pooling       Global average pooling
141   'pool5-drop_7x7_s1'              Dropout                      40% dropout
142   'loss3-classifier'               Fully Connected              1000 fully connected layer
143   'prob'                           Softmax                      softmax
```

How to do transfer learning at a CNN

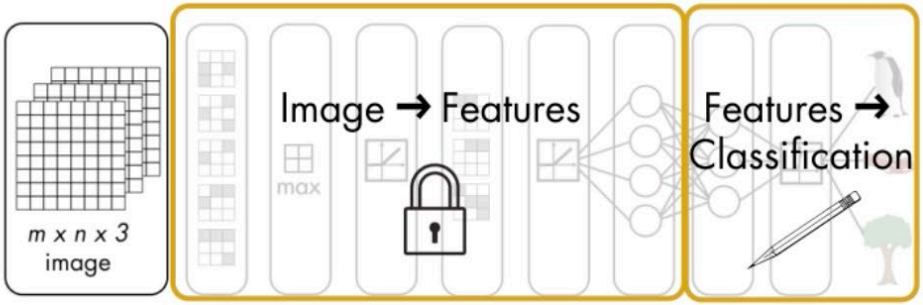
- For example, the end of GoogLeNet, the 142nd layer is a fully connected layer with 1,000 neurons. (It takes the extracted features from the previous layers and maps them to the 1,000 output classes).




```
.
.
.
125   'inception_5a-output'          Depth concatenation          Depth concatenation of 4 inputs
126   'inception_5b-1x1'             Convolution                  384 1x1x832 convolutions with stride [1  1] and padding [0  0  0  0]
127   'inception_5b-relu_1x1'        ReLU                         ReLU
128   'inception_5b-3x3_reduce'      Convolution                  192 1x1x832 convolutions with stride [1  1] and padding [0  0  0  0]
129   'inception_5b-relu_3x3_reduce' ReLU                         ReLU
130   'inception_5b-3x3'             Convolution                  384 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
131   'inception_5b-relu_3x3'        ReLU                         ReLU
132   'inception_5b-5x5_reduce'      Convolution                  48 1x1x832 convolutions with stride [1  1] and padding [0  0  0  0]
133   'inception_5b-relu_5x5_reduce' ReLU                         ReLU
134   'inception_5b-5x5'             Convolution                  128 5x5x48 convolutions with stride [1  1] and padding [2  2  2  2]
135   'inception_5b-relu_5x5'        ReLU                         ReLU
136   'inception_5b-pool'            Max Pooling                  3x3 max pooling with stride [1  1] and padding [1  1  1  1]
137   'inception_5b-pool_proj'       Convolution                  128 1x1x832 convolutions with stride [1  1] and padding [0  0  0  0]
138   'inception_5b-relu_pool_proj'  ReLU                         ReLU
139   'inception_5b-output'          Depth concatenation          Depth concatenation of 4 inputs
140   'pool5-7x7_s1'                 Global Average Pooling       Global average pooling
141   'pool5-drop_7x7_s1'            Dropout                      40% dropout
142   'loss3-classifier'             Fully Connected              1000 fully connected layer
143   'prob'                         Softmax                      softmax
```

How to do transfer learning at a CNN

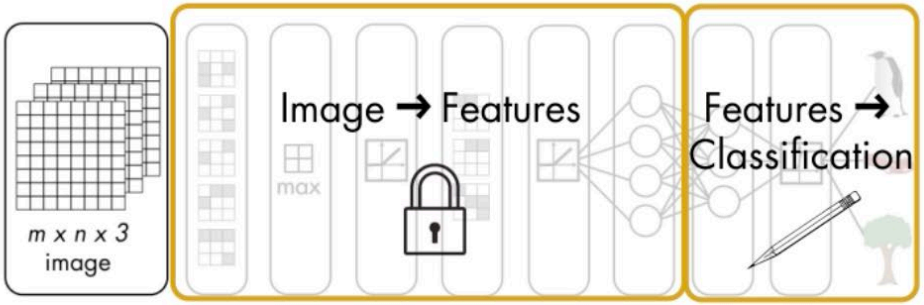
- When changing these layers, the layer graph has the same feature extraction behavior as the pre-trained network.
- But it needs to be trained to map these features to your new image classes.



125	'inception_5a-output'	Depth concatenation	Depth concatenation of 4 inputs
126	'inception_5b-1x1'	Convolution	384 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]
127	'inception_5b-relu_1x1'	ReLU	ReLU
128	'inception_5b-3x3_reduce'	Convolution	192 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]
129	'inception_5b-relu_3x3_reduce'	ReLU	ReLU
130	'inception_5b-3x3'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
131	'inception_5b-relu_3x3'	ReLU	ReLU
132	'inception_5b-5x5_reduce'	Convolution	48 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]
133	'inception_5b-relu_5x5_reduce'	ReLU	ReLU
134	'inception_5b-5x5'	Convolution	128 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
135	'inception_5b-relu_5x5'	ReLU	ReLU
136	'inception_5b-pool'	Max Pooling	3x3 max pooling with stride [1 1] and padding [1 1 1 1]
137	'inception_5b-pool_proj'	Convolution	128 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]
138	'inception_5b-relu_pool_proj'	ReLU	ReLU
139	'inception_5b-output'	Depth concatenation	Depth concatenation of 4 inputs
140	'pool5-7x7_s1'	Global Average Pooling	Global average pooling
141	'pool5-drop_7x7_s1'	Dropout	40% dropout
142	'loss3-classifier'	Fully Connected	1000 fully connected layer
143	'prob'	Softmax	softmax

How to do transfer learning at a CNN

- When you train with new data, the network will learn that mapping as well as refine the feature extraction to be slightly more specific to your application.



125	'inception_5a-output'	Depth concatenation	Depth concatenation of 4 inputs
126	'inception_5b-1x1'	Convolution	384 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]
127	'inception_5b-relu_1x1'	ReLU	ReLU
128	'inception_5b-3x3_reduce'	Convolution	192 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]
129	'inception_5b-relu_3x3_reduce'	ReLU	ReLU
130	'inception_5b-3x3'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
131	'inception_5b-relu_3x3'	ReLU	ReLU
132	'inception_5b-5x5_reduce'	Convolution	48 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]
133	'inception_5b-relu_5x5_reduce'	ReLU	ReLU
134	'inception_5b-5x5'	Convolution	128 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
135	'inception_5b-relu_5x5'	ReLU	ReLU
136	'inception_5b-pool'	Max Pooling	3x3 max pooling with stride [1 1] and padding [1 1 1 1]
137	'inception_5b-pool_proj'	Convolution	128 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]
138	'inception_5b-relu_pool_proj'	ReLU	ReLU
139	'inception_5b-output'	Depth concatenation	Depth concatenation of 4 inputs
140	'pool5-7x7_s1'	Global Average Pooling	Global average pooling
141	'pool5-drop_7x7_s1'	Dropout	40% dropout
142	'loss3-classifier'	Fully Connected	1000 fully connected layer
143	'prob'	Softmax	softmax

Results of transfer learning a CNN

- When done, the newly trained network with layers like the input network, but with updated weights is given.
- A text display showing the progress of the training as:

```
newnet = trainnet(data,net,"crossentropy",options)
```

Iteration	Epoch	TimeElapsed	LearnRate	TrainingLoss	TrainingAccuracy
1	1	00:00:09	0.01	3.1155	26.562
10	2	00:01:13	0.01	0.6069	78.125
20	4	00:02:20	0.01	0.14191	92.969
30	6	00:03:26	0.01	0.034291	99.219
40	8	00:04:31	0.01	0.0052891	100
50	10	00:05:37	0.01	0.011322	100
60	12	00:06:42	0.01	0.0033386	100
70	14	00:07:47	0.01	0.00092616	100
80	16	00:08:53	0.01	0.0087101	99.219
90	18	00:09:55	0.01	0.0055897	100
100	20	00:10:57	0.01	0.0036221	100
110	22	00:12:00	0.01	0.0016095	100
120	24	00:13:05	0.01	0.00037103	100
130	26	00:14:10	0.01	0.00018275	100
140	28	00:15:14	0.01	0.00015373	100
150	30	00:16:21	0.01	0.0001372	100

```
Training stopped: Max epochs completed
```

Results of transfer learning a CNN

- Accuracy is the percentage of the training images that the network classifies correctly. (Expected increasing during your training).
- However, accuracy doesn't measure how confident the network is about each classification.

```
newnet = trainnet(data,net,"crossentropy",options)
```

Iteration	Epoch	TimeElapsed	LearnRate	TrainingLoss	TrainingAccuracy
1	1	00:00:09	0.01	3.1155	26.562
10	2	00:01:13	0.01	0.6069	78.125
20	4	00:02:20	0.01	0.14191	92.969
30	6	00:03:26	0.01	0.034291	99.219
40	8	00:04:31	0.01	0.0052891	100
50	10	00:05:37	0.01	0.011322	100
60	12	00:06:42	0.01	0.0033386	100
70	14	00:07:47	0.01	0.00092616	100
80	16	00:08:53	0.01	0.0087101	99.219
90	18	00:09:55	0.01	0.0055897	100
100	20	00:10:57	0.01	0.0036221	100
110	22	00:12:00	0.01	0.0016095	100
120	24	00:13:05	0.01	0.00037103	100
130	26	00:14:10	0.01	0.00018275	100
140	28	00:15:14	0.01	0.00015373	100
150	30	00:16:21	0.01	0.0001372	100

Training stopped: Max epochs completed

Results of transfer learning a CNN

- The loss measures how far from a perfect prediction the network is (totaled over the set of training images).
- This should decrease towards zero as the training proceeds.

```
newnet = trainnet(data,net,"crossentropy",options)
```

Iteration	Epoch	TimeElapsed	LearnRate	TrainingLoss	TrainingAccuracy
1	1	00:00:09	0.01	3.1155	26.562
10	2	00:01:13	0.01	0.6069	78.125
20	4	00:02:20	0.01	0.14191	92.969
30	6	00:03:26	0.01	0.034291	99.219
40	8	00:04:31	0.01	0.0052891	100
50	10	00:05:37	0.01	0.011322	100
60	12	00:06:42	0.01	0.0033386	100
70	14	00:07:47	0.01	0.00092616	100
80	16	00:08:53	0.01	0.0087101	99.219
90	18	00:09:55	0.01	0.0055897	100
100	20	00:10:57	0.01	0.0036221	100
110	22	00:12:00	0.01	0.0016095	100
120	24	00:13:05	0.01	0.00037103	100
130	26	00:14:10	0.01	0.00018275	100
140	28	00:15:14	0.01	0.00015373	100
150	30	00:16:21	0.01	0.0001372	100

```
Training stopped: Max epochs completed
```

Results of transfer learning a CNN

- You may often see the accuracy plateau at its maximum value, while the loss value continues to decrease.
- This is good. It means that the network is still getting better at differentiating between the categories, even if the final classification may not be changing.

```
newnet = trainnet(data,net,"crossentropy",options)
```

Iteration	Epoch	TimeElapsed	LearnRate	TrainingLoss	TrainingAccuracy
1	1	00:00:09	0.01	3.1155	26.562
10	2	00:01:13	0.01	0.6069	78.125
20	4	00:02:20	0.01	0.14191	92.969
30	6	00:03:26	0.01	0.034291	99.219
40	8	00:04:31	0.01	0.0052891	100
50	10	00:05:37	0.01	0.011322	100
60	12	00:06:42	0.01	0.0033386	100
70	14	00:07:47	0.01	0.00092616	100
80	16	00:08:53	0.01	0.0087101	99.219
90	18	00:09:55	0.01	0.0055897	100
100	20	00:10:57	0.01	0.0036221	100
110	22	00:12:00	0.01	0.0016095	100
120	24	00:13:05	0.01	0.00037103	100
130	26	00:14:10	0.01	0.00018275	100
140	28	00:15:14	0.01	0.00015373	100
150	30	00:16:21	0.01	0.0001372	100

Training stopped: Max epochs completed

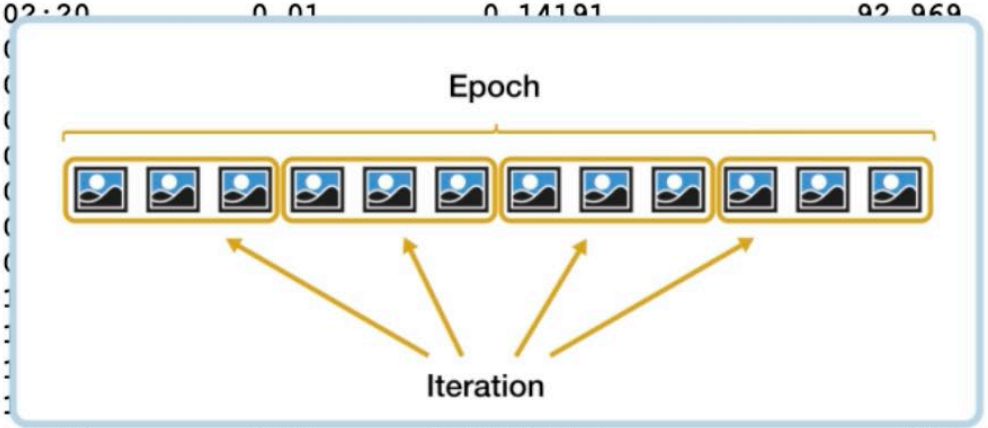
Results of transfer learning a CNN

- At end of the table, it can be sees what caused training to stop.
- Usually due to the number of times the network has processed the training data.

```
newnet = trainnet(data,net,"crossentropy",options)
```

Iteration	Epoch	TimeElapsed	LearnRate	TrainingLoss	TrainingAccuracy
1	1	00:00:09	0.01	3.1155	26.562
10	2	00:01:13	0.01	0.6069	78.125
20	4	00:02:20	0.01	0.14191	92.969
30	6	00:(			
40	8	00:(			
50	10	00:(			
60	12	00:(			
70	14	00:(			
80	16	00:(			
90	18	00:(			
100	20	00:1			
110	22	00:1			
120	24	00:1			
130	26	00:1			
140	28	00:15:14	0.01	0.00015373	100
150	30	00:16:21	0.01	0.0001372	100

```
Training stopped: Max epochs completed
```



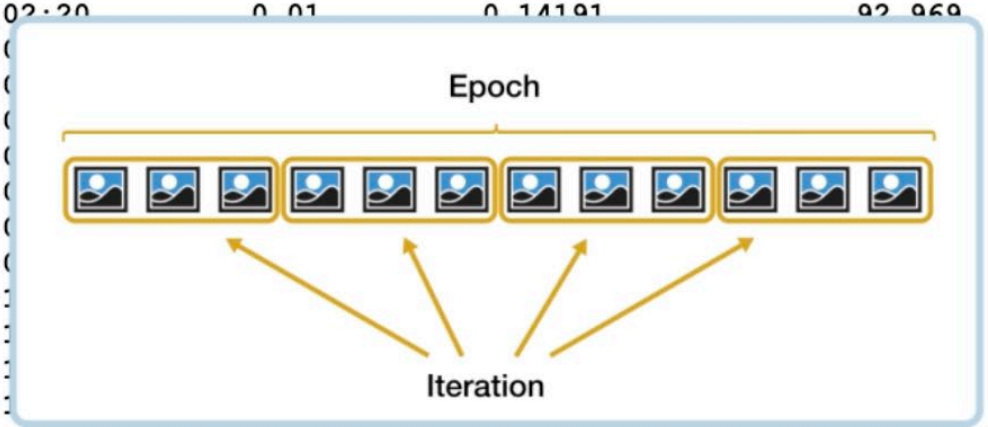
Results of transfer learning a CNN

- At each iteration, the network training uses a subset of the training images, known as a mini-batch, to update the weights. Once the network has trained on the whole training data set, it has completed one epoch. To train the network for more or less time, adjust the maximum epochs option.

```
newnet = trainnet(data,net,"crossentropy",options)
```

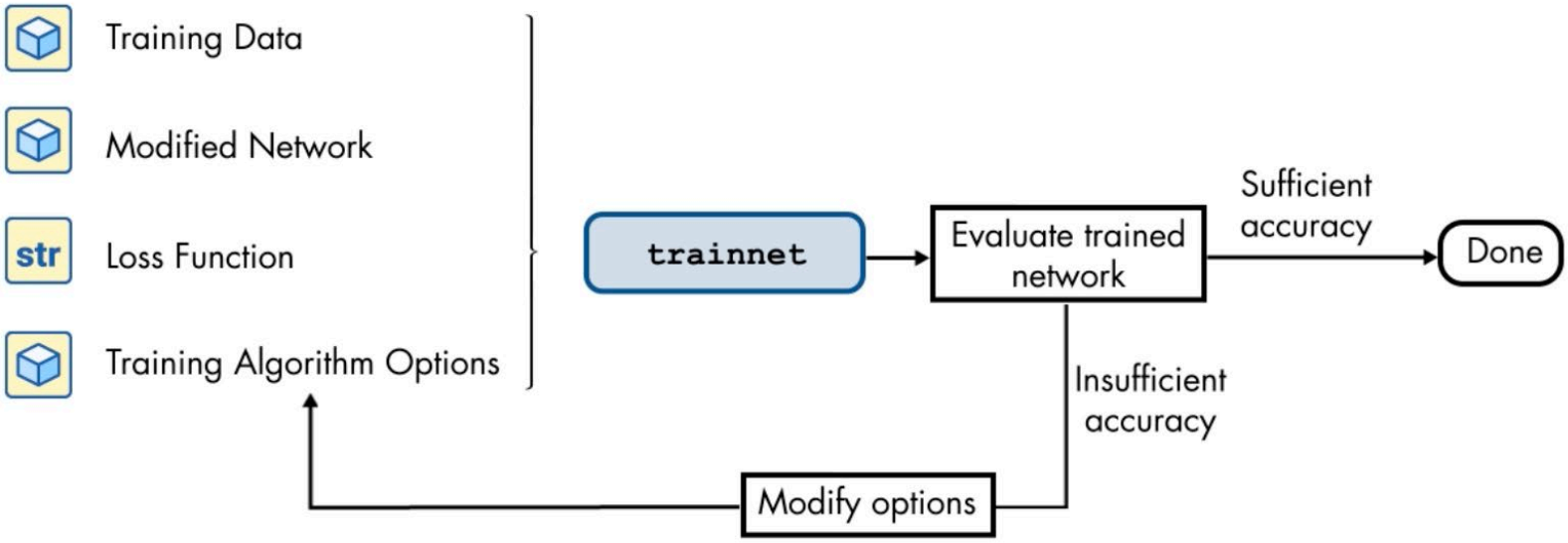
Iteration	Epoch	TimeElapsed	LearnRate	TrainingLoss	TrainingAccuracy
1	1	00:00:09	0.01	3.1155	26.562
10	2	00:01:13	0.01	0.6069	78.125
20	4	00:02:20	0.01	0.14191	92.969
30	6	00:(			
40	8	00:(			
50	10	00:(			
60	12	00:(			
70	14	00:(			
80	16	00:(			
90	18	00:(			
100	20	00:1			
110	22	00:1			
120	24	00:1			
130	26	00:1			
140	28	00:15:14	0.01	0.00015373	100
150	30	00:16:21	0.01	0.0001372	100

```
Training stopped: Max epochs completed
```



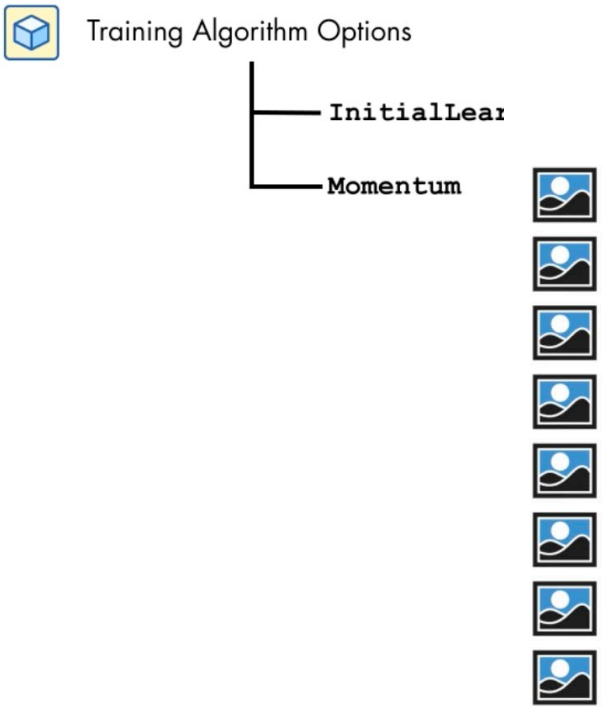
What if the accuracy is not enough?

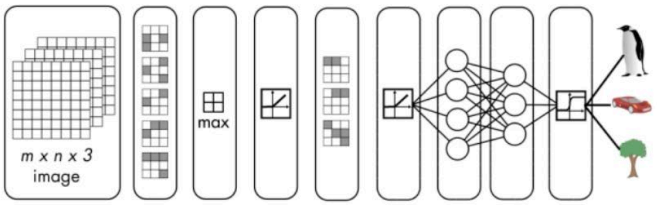
- If the performance is good enough, then congratulations,
- Often the results you get when you first train the network aren't as good as you need.
- What do you do?



What if the accuracy is not enough?

- If the performance is good enough, then congratulations,
- Often the results you get when you first train the network aren't as good as you need.
- What do you do?



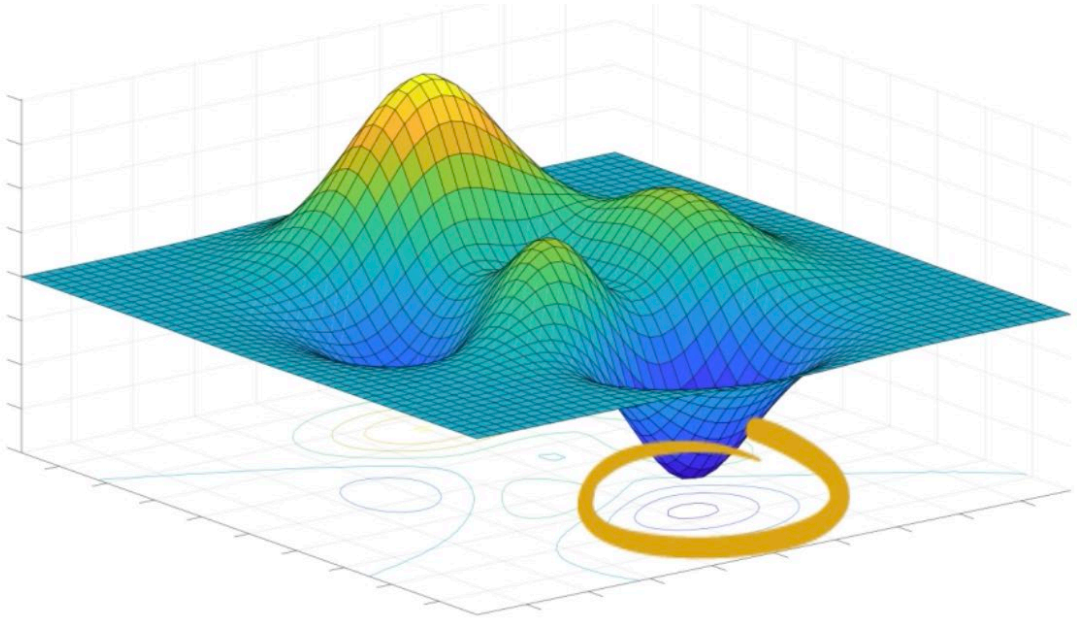


Prediction	Reality
65% tree	penguin
90% car	car
70% penguin	penguin
55% car	tree
80% car	car
50% penguin	car
95% tree	tree
75% tree	penguin

Loss = Difference

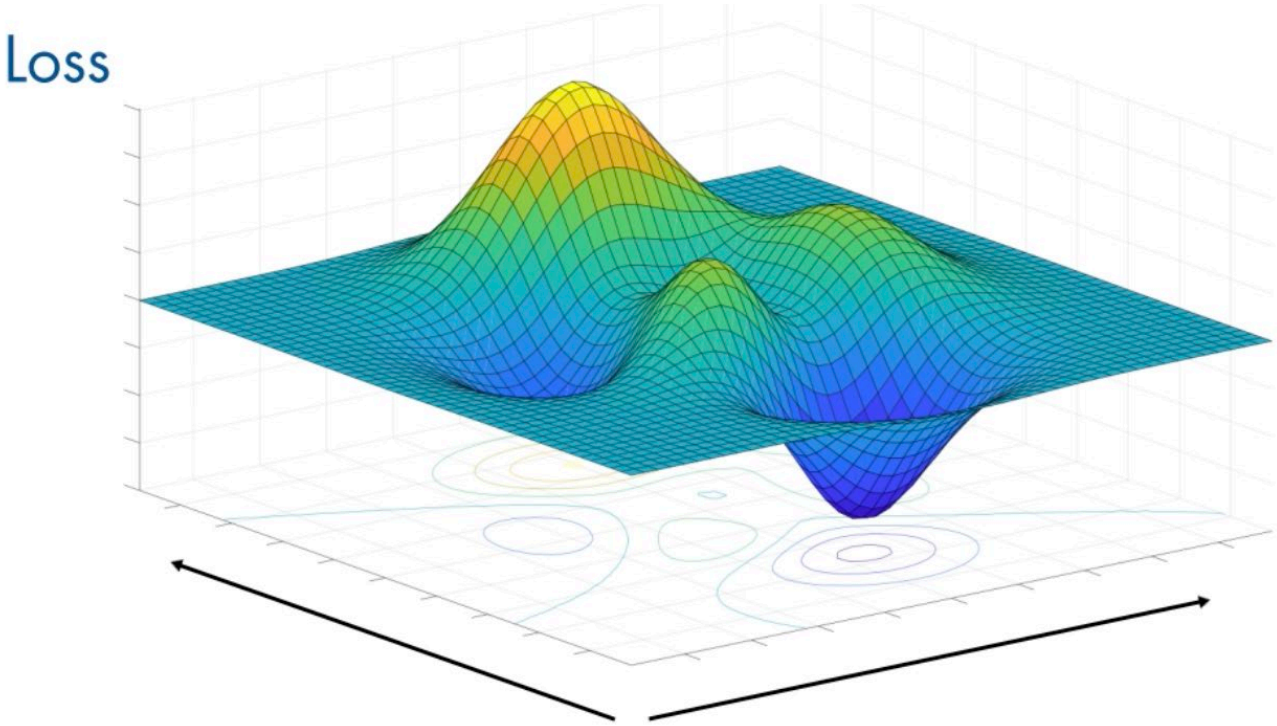
How to visualize the training process?

- Common way to visualize the training process is to imagine that you're trying to find the bottom of a valley.



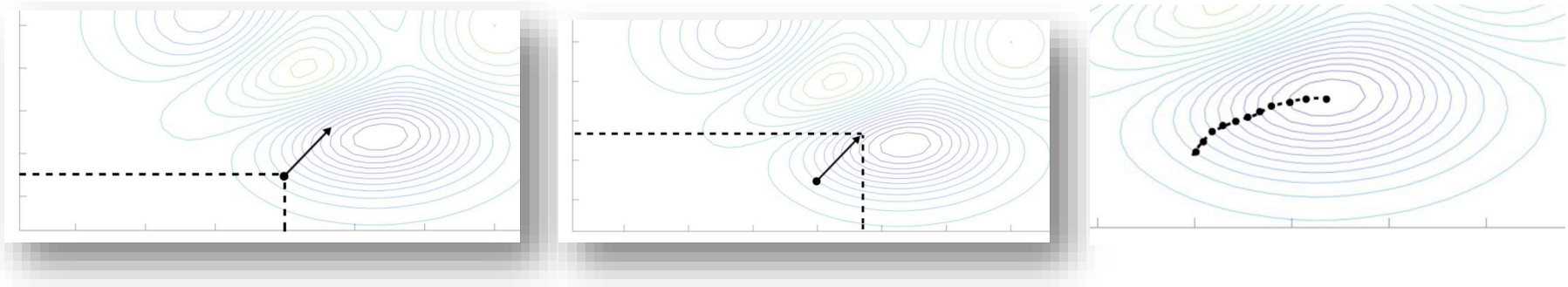
How to visualize the training process?

- The height of the ground is the value of the loss function.
- Your north-south location is one weight value, and your east-west location is another. Of course, real networks have thousands of weights, not just two, but the principle is the same.



How to visualize the training process?

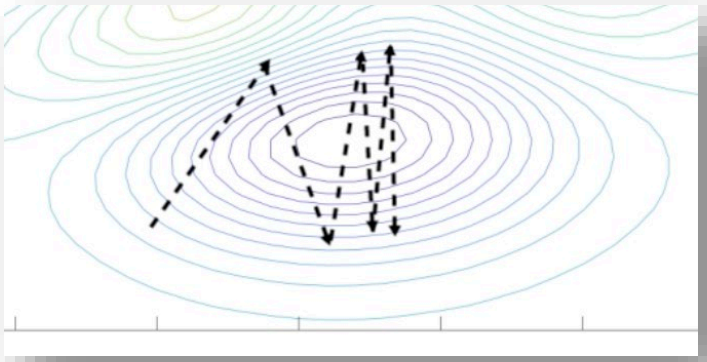
- The gradient descent algorithm works by looking at the height of the ground at a few places around your current position and using that to figure out the gradient or direction that the ground slopes down the steepest.

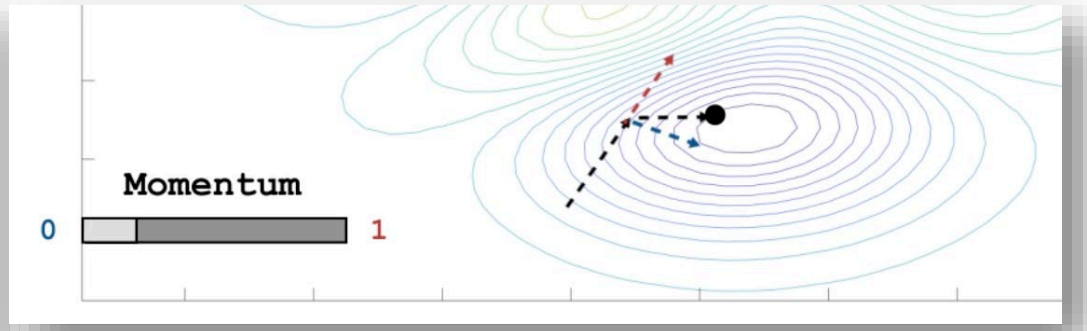


- Then you simply take a step in that direction, which brings you to a new set of weight values.
- But how big a step do you take?
- The size of that step is the learning rate.
- If the learning rate is too small, you'll need to take a lot of steps to get to the bottom, and that takes time.
- But if the learning rate is too big, you might end up jumping around all over the place.

How to visualize the training process?

- If your weights have blown up and your network is giving terrible performance or even not working at all, the first thing you should do is divide your learning rate by 10 and try again.

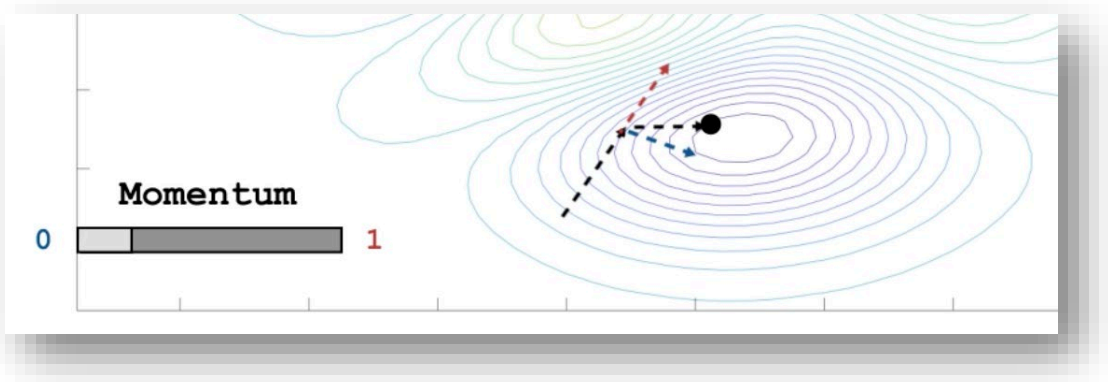




- Gradient descent with momentum tries to stop the jumping around without having to take really small steps.
- You start off going straight down the direction of the gradient

How to visualize the training process?

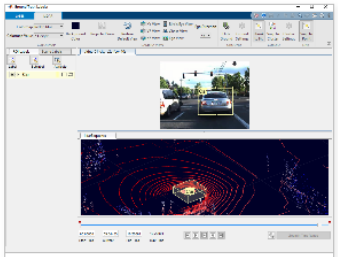
- Instead, you turn toward that direction, similar to the way you would if you were a mass traveling in your original direction but now being pulled in the new direction by a force.
- Your next step is therefore in a direction that's a weighted average of the previous direction and the new direction. That weighting is set by the momentum option.



More resources (if you want)

- You have also examples at:
- https://uk.mathworks.com/help/deeplearning/examples.html?category=applications&exampleproduct=all&newonly=&s_tid=CRUX_lftnav

- And Apps for diferent Applications
- https://uk.mathworks.com/help/deeplearning/referencelist.html?type=app&s_tid=CRUX_topnav&category=applications



Ground Truth Labeler
The Ground Truth Labeler app enables you to label ground truth data in multiple videos, image sequences, or lidar point clouds.

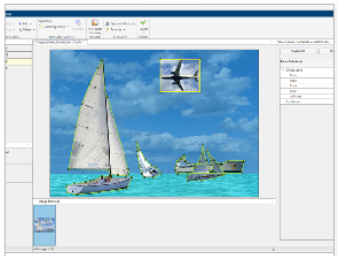
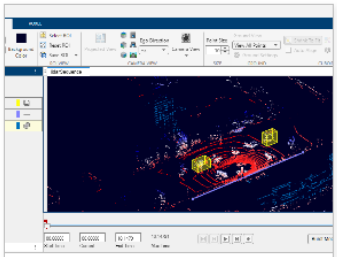
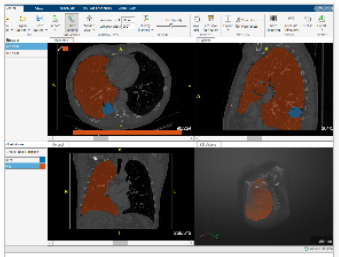


Image Labeler
The Image Labeler app enables you to label ground truth data in a collection of images. Using the app, you can:



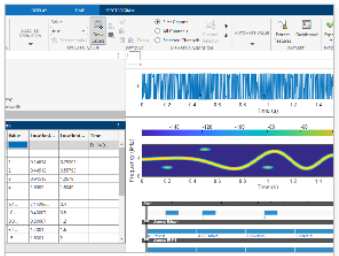
Lidar Labeler
The **Lidar Labeler** app enables you to label objects in a point cloud or a point cloud sequence. The app reads point cloud data from PLY, PCAP, LA...

Since R2020b



Medical Image Labeler
The Medical Image Labeler app enables you to label ground truth data in medical images. Using the app, you can:

Since R2022b



Signal Labeler
The Signal Labeler app is an interactive tool that enables you to label signals for analysis or for use in machine learning and deep learning...