

Announcement

- Assignment 8 is out. Due in 1.5 weeks Dec, 5, 2019.
- Train a CNN to categorize images as X or O.
- Template code in Matlab provided.
 - Not mandatory to use Matlab (more on this later).

Assignment 8 ConvNet: X's and O's

Says whether a picture is of an X or an O





K>> size(example_image)

ans =

116 116

What is provided:

- 1. Dataset of 900 images each of two categories
- 2. Template code for training and evaluating a CNN in MATLAB





Root folder must contain the training_data folder for the template to work.

BasicCNNtemplate.m

training_data







BasicCNNTemplate.m overview

- 1. Configure the execution of the code.
- 2. Load and prep the data
- 3. Setup the CNN architecture
- 4. Train the Network
- 5. Test the performance of the CNN
- 6. Plotting code.

1. Configure the execution of the code.

doTraining	= true;			
<pre>% Set these flags show.wrong_classif show.filter show.feature_maps</pre>	to inspect and plot ied = false; = false; = true;	the network (Note:	<pre>optimized for scre % wrong classified % filters(weights) % feature maps</pre>	en resolution (1920x1200)) images

2. Load and prep the data

Create an image datastore object

IMDS = imageDatastore('training_data','IncludeSubfolders',true,'FileExtensions','.bmp','LabelSource','foldernames');

example_image = readimage(IMDS,1); % read one example image from the datastore.

```
% Uncomment the line below to display the example_image.
% imshow(example_image);
```

```
numChannels = size(example_image,3);
numImageCategories = size(categories(IMDS.Labels),1);
```

Get channel info and # of label categories

% get color information - The images are single channel in th % Two image categories in our dataset.

```
% Create the training and testing datasets.
% Split ImageDatastore labels by proportions
training_propotion = 0.7;
[trainingDS,validationDS] = splitEachLabel(IMDS,training_propotion,'randomize');
```

LabelCntTr = countEachLabel(trainingDS); LabelCntVa = countEachLabel(validationDS); % load lable information

630 samples in training, 270 in validation for proportion =0.7





datastore















Root folder must contain the training_data folder for the template to work.

BasicCNNtemplate.m

training_data



2. Load and prep the data

Create an image datastore object

IMDS = imageDatastore('training_data','IncludeSubfolders',true,'FileExtensions','.bmp','LabelSource','foldernames');

example_image = readimage(IMDS,1); % read one example image from the datastore.

```
% Uncomment the line below to display the example_image.
% imshow(example_image);
```

```
numChannels = size(example_image,3);
numImageCategories = size(categories(IMDS.Labels),1);
```

Get channel info and # of label categories

% get color information - The images are single channel in th % Two image categories in our dataset.

```
% Create the training and testing datasets.
% Split ImageDatastore labels by proportions
training_propotion = 0.7;
[trainingDS,validationDS] = splitEachLabel(IMDS,training_propotion,'randomize');
```

LabelCntTr = countEachLabel(trainingDS); LabelCntVa = countEachLabel(validationDS); % load lable information

630 samples in training, 270 in validation for proportion =0.7



3. Setup the CNN architecture

%% Setup of the CNN architecture. if doTraining % Convolutional layer parameters filterSize = [10 10]; You can change the filter size or even try multiple filter sizes numFilters = 16; Number of filters usually a power of 2 % An image input layer inputs 2–D images to a network and applies data normalization. % The size of the layer is the same as the number of pixels in our % input images. inputLayer = imageInputLayer(size(example_image),'Name','Input'); % no data augmentation

Create the input layer which simply reads the 116x116 bmp image.

Note the use of the 'Name', 'layer name' args

Data Preprocessing



(Assume X [NxD] is data matrix, each example in a row)



Crop to Symmetric Aspect Ratio









Pixel wise mean and std deviation



Zero Center Normalization

- Subtract mean
- Divide by std dev



3. Setup the CNN architecture

You need to specify the layers in the architecture

```
middleLayers = [
    % The first convolutional layer has a bank of numFilters filters of size filterSize.
    % A symmetric padding of 4 pixels is added.
    convolution2dLayer(...)
    % Next add the ReLU layer:
    reluLayer('Name', 'ReLu1')
   % Follow it with a max pooling layer that has a 5x5 spatial pooling area
    % and a stride of 2 pixels. This down-samples the data dimensions.
    maxPooling2dLayer(...)
    % Repeat the 3 core layers to complete the middle of the network.
    % This time use 32 filters instead of 16.
    % Repeat the 3 core layers one more time
    % This time change symmetric padding to 2 for the convolution, and
    % the stride to 3 for the maxpoolinglayer.
```

3. Setup the CNN architecture

Example architecture

1	'Input'	Image Input	116x116x1 images with 'zerocenter' normalization
2	'Conv1'	Convolution	16 10x10x1 convolutions with stride [1 1] and padding [4 4 4 4]
3	'ReLu1'	ReLU	ReLU
4	'Pool1'	Max Pooling	5x5 max pooling with stride [2 2] and padding [0 0 0 0]
5	'Conv2'	Convolution	32 10x10 convolutions with stride [1 1] and padding [4 4 4 4]
6	'ReLu2'	ReLU	ReLU
7	'Pool2'	Max Pooling	5x5 max pooling with stride [2 2] and padding [0 0 0 0]
8	'Conv3'	Convolution	32 10x10 convolutions with stride [1 1] and padding [2 2 2 2]
9	'ReLu3'	ReLU	ReLU
10	'Pool3'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
11	'FC'	Fully Connected	2 fully connected layer
12	'Softmax'	Softmax	softmax
13	'Classificatio	on' Classification O	utput crossentropyex

3. Setup the CNN architecture – Useful functions

Convolution and Fully Connected Layers

Layer	Description
convolution2dLayer	A 2-D convolutional layer applies sliding convolutional filters to the input.
convolution3dLayer	A 3-D convolutional layer applies sliding cuboidal convolution filters to three- dimensional input.
groupedConvolution2dLayer	A 2-D grouped convolutional layer separates the input channels into groups and applies sliding convolutional filters. Use grouped convolutional layers for channel-wise separable (also known as depth-wise separable) convolution.
transposedConv2dLayer	A transposed 2-D convolution layer upsamples feature maps.
transposedConv3dLayer	A transposed 3-D convolution layer upsamples three-dimensional feature maps.
fullyConnectedLayer	A fully connected layer multiplies the input by a weight matrix and then adds a bias vector.

3. Setup the CNN architecture – Useful functions

Activation Layers

Layer	Description
	A ReLU layer performs a threshold operation to each element of the input, where any value less than zero is set to zero.
leakyReluLayer	A leaky ReLU layer performs a threshold operation, where any input value less than zero is multiplied by a fixed scalar.
<pre>clippedReluLayer</pre>	A clipped ReLU layer performs a threshold operation, where any input value less than zero is set to zero and any value above the <i>clipping ceiling</i> is set to that clipping ceiling.
eluLayer	An ELU activation layer performs the identity operation on positive inputs and an exponential nonlinearity on negative inputs.
J tanhLayer	A hyperbolic tangent (tanh) activation layer applies the tanh function on the layer inputs.
preluLayer (Custom layer example)	A PReLU layer performs a threshold operation, where for each channel, any input value less than zero is multiplied by a scalar learned at training time.

3. Setup the CNN architecture – Useful functions

Pooling and Unpooling Layers

Layer	Description
averagePooling2dLayer	An average pooling layer performs down-sampling by dividing the input into rectangular pooling regions and computing the average values of each region.
averagePooling3dLayer	A 3-D average pooling layer performs down-sampling by dividing three- dimensional input into cuboidal pooling regions and computing the average values of each region.
globalAveragePooling2dLayer	A global average pooling layer performs down-sampling by computing the mean of the height and width dimensions of the input.
globalAveragePooling3dLayer	A global average pooling layer performs down-sampling by computing the mean of the height, width, and depth dimensions of the input.
maxPooling2dLayer	A max pooling layer performs down-sampling by dividing the input into rectangular pooling regions, and computing the maximum of each region.
<pre>maxPooling3dLayer</pre>	A 3-D max pooling layer performs down-sampling by dividing three- dimensional input into cuboidal pooling regions, and computing the maximum of each region.
maxUnpooling2dLayer	A max unpooling layer unpools the output of a max pooling layer.

3. Setup the CNN architecture

Final layers already defined – need not change

```
finalLayers = [
   % % Add a fully connected layer with the same number of neurons as
   % the number of image categories.
                                                            Fully connected layer
   fullyConnectedLayer(numImageCategories, 'Name', 'FC')
   % Add the softmax loss layer and classification layer.
   % The final layers use the output of the fully connected layer to compute the categorical
   % probability distribution over the image classes. During the training
   % process, all the network weights are tuned to minimize the loss over this
   % categorical distribution.
                                                        Softmax layer
    softmaxLayer('Name', 'Softmax');
   classificationLayer('Name', 'Classification')
   ];
                                     Cross entropy classification loss
layers = [
   inputLayer
   middleLayers
                   All layers are stacked together
   finalLayers
    ];
```

```
4.
CNN
Training
```

```
%% Train the Network
   %Initialize the first convolutional layer weights using
   % normally distributed random numbers with standard deviation of 0.0001.
    % This helps improve the convergence of training.
    layers(2).Weights = 0.0001 * randn([filterSize numChannels numFilters]);
                                             Initial weights have been provided
   % Set the network training options
    % Try Momentum option 0.1 and 0.9 - Which is Better ?
   % Try LearningRate 0.01, and 0.001 - What is the difference ?
   % Try 10-20 Maxepochs
    opts = trainingOptions('sgdm', ...
        'Momentum', 0, ...
                                             You have to try out different values
        'InitialLearnRate', 0, ...
        'LearnRateSchedule', 'piecewise', ...for Momentum, Learning Rates and
        'LearnRateDropFactor', 0.5, ...
                                             MaxEpochs
        'LearnRateDropPeriod', 10, ...
        'L2Regularization', 0.004, ...
        'MaxEpochs', 0, ...
        'MiniBatchSize', 64, ... % 64 for Quadro
        'Verbose', true,...
        'Plots', 'training-progress');
   % Train a network.
                                              Training happens here
    rng('default');
                                              Should take ~ 10mins on a CPU
    rng(123); % random seed
    XONet = trainNetwork(trainingDS, layers, opts);
    save('XONet.mat','XONet');
```



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



'convl' 'relul' 'norml' 'pooll' 'conv2' 'relu2' 'norm2'	Convolution ReLU Cross Channel Normalization Max Pooling Convolution ReLU	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0] ReLU cross channel normalization with 5 channels per element 3x3 max pooling with stride [2 2] and padding [0 0 0 0] 256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2] Polly
'relul' 'norml' 'pooll' 'conv2' 'relu2' 'norm2'	ReLU Cross Channel Normalization Max Pooling Convolution ReLU	ReLU cross channel normalization with 5 channels per element 3x3 max pooling with stride [2 2] and padding [0 0 0 0] 256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2] Polly
'norml' 'pooll' 'conv2' 'relu2' 'norm2'	Cross Channel Normalization Max Pooling Convolution ReLU	cross channel normalization with 5 channels per element 3x3 max pooling with stride [2 2] and padding [0 0 0 0] 256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2] Polly
'pool1' 'conv2' 'relu2' 'norm2'	Max Pooling Convolution ReLU	3x3 max pooling with stride [2 2] and padding [0 0 0 0] 256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2] Polly
'conv2' 'relu2' 'norm2'	Convolution ReLU	256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
'relu2' 'norm2'	ReLU	PotII
'norm2'	2 (C)	Relo
1000121	Cross Channel Normalization	cross channel normalization with 5 channels per element
poorz	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
'relu3'	ReLU	ReLU
'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
'relu4'	ReLU	ReLU
'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
'relu5'	ReLU	ReLU
'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
'fc6'	Fully Connected	4096 fully connected layer
'relu6'	ReLU	ReLU
'drop6'	Dropout	50% dropout
'fc7'	Fully Connected	4096 fully connected layer
'relu7'	ReLU	ReLU
'drop7'	Dropout	50% dropout
'fc8'	Fully Connected	1000 fully connected layer
'prob'	Softmax	softmax
'output'	Classification Output	crossentropyex with 'tench', 'goldfish', and 998 other classes
	'norm2' 'pool2' 'conv3' 'relu3' 'conv4' 'relu4' 'conv5' 'relu5' 'pool5' 'fc6' 'fc7' 'fc7' 'fc7' 'fc7' 'fc8' 'prob' 'output'	<pre>'norm2' Cross Channel Normalization 'pool2' Max Pooling 'conv3' Convolution 'relu3' ReLU 'conv4' Convolution 'relu4' ReLU 'conv5' Convolution 'relu5' ReLU 'conv5' Convolution 'relu5' ReLU 'pool5' Max Pooling 'fc6' Fully Connected 'relu6' ReLU 'drop6' Dropout 'fc7' Fully Connected 'relu7' ReLU 'drop7' Dropout 'fc8' Fully Connected 'relv7' Softmax 'output' Classification Output</pre>

m x n x 3 image	Image Features			
--------------------	----------------	--	--	--

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	'relul'	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	Belling Eastures
12	'conv4'	Convolution	TIQCEZ COULEStride [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'	Ralu	ReLU
19	'drop6'	Dropout	50% dropout
20	'£c7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	'1C8'	Fully Connected	1000 fully connected layer
24	'prob'	Softmax	softmax
25	'output'	Classification Output	crossentropyex with 'tench', 'goldfish', and 998 other classes



2 'conv1' Convolution 96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0 3 'relu1' REU REU 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 11 'relu3' ReLU ReLU 12 'conv4' Convolution 384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 13 'relu4' ReLU ReLU 14 'conv5' Convolution 256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 15 'relu4' ReLU ReLU 16 'pool5' <t< th=""><th>1</th><th>'data'</th><th>Image Input</th><th>227x227x3 images with 'zerocenter' normalization</th></t<>	1	'data'	Image Input	227x227x3 images with 'zerocenter' normalization
3 'relu!' 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' Convolution 256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2 7 'relu2' ReLU ReLU 8 'norm2' Convolution 256 5x5x48 convolutions with stride [1 1] and padding [2 0 0 0] 10 'conv3' Convolution 384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 11 'relu4' ReLU ReLU 12 'conv4' Convolution 384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 13 'relu4' ReLU ReLU 14 'conv5' Convolution 256 3x3x192 convolutions with stride [2 2] and padding [0 0 0] 15 'relu4' ReLU ReLU 14 'conv5' Convolution 256 3x3x192 convolutions with stri	2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
4'norml'Cross Channel Normalizationcross channel normalization with 5 channels per element5'pool1'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]6'conv2'Convolution256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2 27'relu2'ReLUReLU8'norm2'Cross Channel Normalizationcross channel normalization with 5 channels per element9'pool2'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]10'conv3'Convolution384 3x3x256 convolutions with stride [1 1] and padding [1 1 111'relu3'ReLUReLU12'conv4'Convolution384 3x3x192 convolutions with stride [1 1] and padding [1 1 113'relu4'ReLUReLU14'conv5'Convolution256 5x3x192 convolutions with stride [1 1] and padding [1 1 115'relu5'ReLUReLU16'pool5'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0]17'fc6'Fully Connected4096 fully connected layer18'relu6'ReLUReLU19'drop6'Dropout50% dropout20'fc7'Fully Connected4096 fully connected layer21'relu7'ReLUFol23'fc8'Fully Connected1000 fully connected layer24'propSoftmaxSoftmax25'output'Classification OutputCorossentropyex with 'tench', 'goldfish', and 998 oth	3	'relu1'	ReLU	ReLU
5'pool1'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]6'conv2'Convolution256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 27'relu2'ReLUReLU8'norm2'Cross Channel Normalizationcross channel normalization with 5 channels per element9'pool2'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]10'conv3'Convolution384 3x3x256 convolutions with stride [1 1] and padding [1 1 111'relu3'ReLUReLU12'conv4'Convolution384 3x3x192 convolutions with stride [1 1] and padding [1 1 113'relu4'ReLUReLU14'conv5'Convolution256 3x3x192 convolutions with stride [1 1] and padding [1 1 115'relu5'ReLUReLU16'pool5'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0]17'fc6'Fully Connected4096 fully connected layer18'relu5'ReLUReLU19'drop6'Dropout50% dropout20'fc8'Fully Connected4096 fully connected layer21'relu7'ReLUReLU22'fc8'Fully Connected23'fc8'Fully Connected24probSoftmax25'output'Classification Output25'output'Classification Output	4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
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 11 'relu3' ReLU ReLU 12 'conv4' Convolution 384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 13 'relu4' ReLU ReLU 14 'conv5' Convolution 256 3x3x192 convolutions with stride [1 1] and padding [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	5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
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] 11 'relu3' ReLU ReLU 12 'conv4' Convolution 384 3x3x192 convolutions with stride [1 1] and padding [1 1 1] 13 'relu4' ReLU ReLU 14 'conv5' Convolution 256 3x3x192 convolutions with stride [1 1] and padding [1 1 1] 15 'relu5' ReLU ReLU 16 'pool5' Max Pooling 3x3 max pooling with stride [2 2] and padding [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 'relu7' Fully Connected	6	'conv2'	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
8'norm2'Cross Channel Normalizationcross channel normalization with 5 channels per element9'pool2'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]10'conv3'Convolution384 3x3x256 convolutions with stride [1 1] and padding [1 1 111'relu3'ReLUReLU12'conv4'Convolution384 3x3x192 convolutions with stride [1 1] and padding [1 1 113'relu4'ReLUReLU14'conv5'Convolution256 3x3x192 convolutions with stride [1 1] and padding [1 1 115'relu5'ReLUReLU16'pool5'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]17'fc6'Fully Connected4096 fully connected layer18'relu6'ReLUReLU19'drop6'Dropout50% dropout20'fc7'Fully Connected4096 fully connected layer21'relu7'ReLUReLU23'fc8'Fully Connected24PDODSOLUMAX25'output'Classification Output	7	'relu2'	ReLU	ReLU
9'pool2'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]10'conv3'Convolution384 3x3x256 convolutions with stride [1 1] and padding [1 1 1]11'relu3'ReLUReLU12'conv4'Convolution384 3x3x192 convolutions with stride [1 1] and padding [1 1 1]13'relu4'ReLUReLU14'conv5'Convolution256 3x3x192 convolutions with stride [1 1] and padding [1 1 1]15'relu5'ReLUReLU16'pool5'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]17'fc6'Fully Connected4096 fully connected layer18'relu6'ReLUReLU19'drop6'Dropout50% dropout20'fc7'Fully Connected4096 fully connected layer21'relu7'ReLUReLU23'fc8'Fully Connected24propSOTLMAX25'output'Classification Output	8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
10'conv3'Convolution384 3x3x256 convolutions with stride [1 1] and padding [1 1 1]11'relu3'ReLUReLU12'conv4'Convolution384 3x3x192 convolutions with stride [1 1] and padding [1 1 1]13'relu4'ReLUReLU14'conv5'Convolution256 3x3x192 convolutions with stride [1 1] and padding [1 1 1]15'relu5'ReLUReLU16'pool5'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0]17'fc6'Fully Connected4096 fully connected layer18'relu6'ReLUReLU19'drop6'Dropout50% dropout20'fc7'Fully Connected4096 fully connected layer21'relu7'ReLUReLU23'fc8'Fully Connected1000 fully connected layer24probSOTLMAXSOTLMAX25'output'Classification Outputcrossentropyex with 'tench', 'goldfish', and 998 other classes	9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
11'relu3'ReLUReLU12'conv4'Convolution384 3x3x192 convolutions with stride [1 1] and padding [1 1 1]13'relu4'ReLUReLU14'conv5'Convolution256 3x3x192 convolutions with stride [1 1] and padding [1 1 1]15'relu5'ReLUReLU16'pool5'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]17'fc6'Fully Connected4096 fully connected layer18'relu6'ReLUReLU19'drop6'Dropout50% dropout20'fc7'Fully Connected4096 fully connected layer21'relu7'ReLUReLU23'fc8'Fully Connected1000 fully connected layer23'fc8'Fully ConnectedSoftimax24propSoftimaxSoftimax25'output'Classification Outputcrossentropyex with 'tench', 'goldfish', and 998 other classes	10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
12'conv4'Convolution384 3x3x192 convolutions with stride [1 1] and padding [1 1 113'relu4'ReLUReLU14'conv5'Convolution256 3x3x192 convolutions with stride [1 1] and padding [1 1 115'relu5'ReLUReLU16'pool5'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]17'fc6'Fully Connected4096 fully connected layer18'relu6'ReLUReLU19'drop6'Dropout50% dropout20'fc7'Fully Connected4096 fully connected layer21'relu7'ReLUReLU23'fc8'Fully Connected1000 fully connected layer23'fc8'Fully Connected1000 fully connected layer24propSOTCHMAXSOTCHMAX25'output'Classification Outputcrossentropyex with 'tench', 'goldfish', and 998 other classes	11	'relu3'	ReLU	ReLU
13'relu4'ReLUReLU14'conv5'Convolution256 3x3x192 convolutions with stride [1 1] and padding [1 1 115'relu5'ReLUReLU16'pool5'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]17'fc6'Fully Connected4096 fully connected layer18'relu6'ReLUReLU19'drop6'Dropout50% dropout20'fc7'Fully Connected4096 fully connected layer21'relu7'ReLUReLU23'fc8'Fully Connected1000 fully connected layer23'fc8'Fully Connected1000 fully connected layer24probSoftmaxSoftmax25'output'Classification Outputcrossentropyex with 'tench', 'goldfish', and 998 other classes	12	'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
14'conv5'Convolution256 3x3x192 convolutions with stride [1 1] and padding [1 1 115'relu5'ReLUReLU16'pool5'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]17'fc6'Fully Connected4096 fully connected layer18'relu6'ReLUReLU19'drop6'Dropout50% dropout20'fc7'Fully Connected4096 fully connected layer21'relu7'ReLUReLU23'fc8'Fully Connected1000 fully connected layer23'fc8'Fully Connectedsoitunax24propSoitunaxsoitunax25'output'Classification Outputcrossentropyex with 'tench', 'goldfish', and 998 other classes	13	'relu4'	ReLU	ReLU
15'relu5'ReLUReLU16'pool5'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]17'fc6'Fully Connected4096 fully connected layer18'relu6'ReLUReLU19'drop6'Dropout50% dropout20'fc7'Fully Connected4096 fully connected layer21'relu7'ReLUReLU23'fc8'Fully Connected1000 fully connected layer24probSoftimaxsoftimax25'output'Classification Outputcrossentropyex with 'tench', 'goldfish', and 998 other classes	14	'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
16'pool5'Max Pooling3x3 max pooling with stride [2 2] and padding [0 0 0 0]17'fc6'Fully Connected4096 fully connected layer18'relu6'ReLUReLU19'drop6'Dropout50% dropout20'fc7'Fully Connected4096 fully connected layer21'relu7'ReLUReLU23'fc8'Fully Connected1000 fully connected layer24probSoftmaxsoftmax25'output'Classification Outputcrossentropyex with 'tench', 'goldfish', and 998 other classes	15	'relu5'	ReLU	ReLU
17'fc6'Fully Connected4096 fully connected layer18'relu6'ReLUReLU19'drop6'Dropout50% dropout20'fc7'Fully Connected4096 fully connected layer21'relu7'ReLUReLU22'drop7'Durpout50% dropout23'fc8'Fully Connected1000 fully connected layer24propSortmaxsortmax25'output'Classification Outputcrossentropyex with 'tench', 'goldfish', and 998 other classes	16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
18 'relu6' ReLU ReLU 19 'drop6' Dropout 50% dropout 20 'fc7' Fully Connected 4096 fully connected layer 21 'relu7' ReLU ReLU 22 'drop7' Pully Connected 1000 fully connected layer 23 'fc8' Fully Connected 1000 fully connected layer 24 prob Sortamax 25 'output' Classification Output crossentropyex with 'tench', 'goldfish', and 998 other classes	17	'fc6'	Fully Connected	4096 fully connected layer
19'drop6'Dropout50% dropout20'fc7'Fully Connected4096 fully connected layer21'relu7'ReLUReLU20'decc7'December23'fc8'Fully Connected24probSoftmax25'output'Classification Output25'output'Classification Output	18	'relu6'	ReLU	ReLU
20 'fc7' Fully Connected 4096 fully connected layer 21 'relu7' ReLU ReLU 20 'dec21' Decent 500 decent 23 'fc8' Fully Connected 1000 fully connected layer 24 prob Sortmax sortmax 25 'output' Classification Output crossentropyex with 'tench', 'goldfish', and 998 other classes	19	'drop6'	Dropout	50% dropout
21 'relu7' ReLU 23 'fc8' Fully Connected 24 prop Softmax 25 'output' Classification Output ReLU FOUNDATION For the softmax Crossentropyex with 'tench', 'goldfish', and 998 other classes	20	'fc7'	Fully Connected	4096 fully connected layer
23 'fc8' Fully Connected 1000 fully connected layer 23 'fc8' Fully Connected 1000 fully connected layer 24 prob Sortmax sortmax 25 'output' Classification Output crossentropyex with 'tench', 'goldfish', and 998 other classes	21	'relu7'	ReLU	ReLU
23 'fc8' Fully Connected 1000 fully connected layer 24 prop Sortmax sortmax 25 'output' Classification Output crossentropyex with 'tench', 'goldfish', and 998 other classes	20	1 days 71	Despect	E00 deepent
24 prob Sortmax Sortmax 25 'output' Classification Output crossentropyex with 'tench', 'goldfish', and 998 other classes	23	'fc8'	Fully Connected	1000 fully connected layer
25 'output' Classification Output crossentropyex with 'tench', 'goldfish', and 998 other classes	44	Prop.	SULLINAX	SOLLINAX
	25	'output'	Classification Output	crossentropyex with 'tench', 'goldfish', and 998 other classes


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	'poo15'	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
- 22	16-01	Pully Concepted	1000 fully connected layer
24	'prob'	Softmax	softmax
20	output	CLASSIFICATION Output	crossencropyex with 'tench', 'goldfish', and 998 other classes



'data'	Image Input	227x227x3 images with 'zerocenter' normalization
'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
'relu1'	ReLU	ReLU
'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
'conv2'	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
'relu2'	ReLU	ReLU
'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
'relu3'	ReLU	ReLU
'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
'relu4'	ReLU	ReLU
'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
'relu5'	ReLU	ReLU
'poo15'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
'fc6'	Fully Connected	4096 fully connected layer
'relu6'	ReLU	ReLU
'drop6'	Dropout	50% dropout
'fc7'	Fully Connected	4096 fully connected layer
'relu7'	ReLU	ReLU
'drop7'	Dropout	50% dropout
'fc8'	Fully Connected	1000 fully connected layer
Immah I	A - Ekman	- Flores
'output'	Classification Output	crossentropyex with 'tench', 'goldfish', and 998 other classes

Training on single GPU. Initializing image normalization.

	Epoch	Iteration	Time Elapsed (seconds)	(Mini-batch Loss		Mini-batch Accuracy)	Base Learning Rate
=== 	1 3	1 10	0.47 10.31	-=- 	3.5061 0.7686	-=- 	7.81% 75.00%	 	



Training on single GPU.

.

Initializing image normalization.

 	Epoch	 	Iteration	 	Time Elapsed (seconds)	 	Mini-batch Loss	1	Mini-batch Accuracy	1	Base Learning Rate
	1	I	1	I	0.47	I	3.5061	I	7.81%	1	0.0010
1	3	1	10	1	10.31	1	0.7686	1	75.00%	1	0.0010
1	5	I.	20	1	18.96	L	0.2371	T	92.19%	I	0.0010
I	8	L	30	L	27.43	L	0.0770	T	97.66%	I	0.0010
I	10	1	40	1	35.31	L	0.0336	L	99.22%	I	0.0010
1	13	I.	50	1	43.17	L	0.0289	1	99.22%	1	0.0010
I	15	1	60	1	50.15	L	0.0104	L	100.00%	I	0.0010
I	18	1	70	1	56.84	1	0.0072	I	100.00%	I	0.0010
1	20	1	80	L	63.00	1	0.0210	T	99.22%	I	0.0010
1	23	1	90	1	69.37	L	0.0035	I	100.00%	T	0.0010
I	25	L	100	1	74.85	L	0.0027	L	100.00%	I	0.0010
1	28	1	110	Ĩ.	81.19	Ĭ.	0.0053	1	100.00%	- È	0.0010

Training on single GPU.

Initializing image normalization.

 	Epoch	1	Iteration	 	Time Elapsed (seconds)	 	Mini-batch Loss	Mini-batch Accuracy	Base Learning Rate
 	1	1	1	1	0.47	1	3.5061	7.81%	0.0010
1	3	1	10	1	10.31	1	0.7686	75.00%	0.0010
1	5	L	20	L	18.96	L	0.2371	92.19%	0.0010
1	8	1	30	L	27.43	L	0.0770	97.66%	0.0010
1	10	1	40	1	35.31	L	0.0336	99.22%	0.0010
1	13	T.	50	1	43.17	L	0.0289	99.22%	0.0010
1	15	T.	60	1	50.15	L	0.0104	100.00%	0.0010
1	18	L	70	1	56.84	L	0.0072	100.00%	0.0010
1	20	1	80	1	63.00	1	0.0210	99.22%	0.0010
1	23	1	90	L	69.37	L	0.0035	100.00%	0.0010
1	25	L	100	1	74.85	L	0.0027	100.00%	0.0010
1	28	1	110	1	81.19	1	0.0053	100.00%	0.0010
	30	I	120	1	86.75	1	0.0045	100.00%	0.0010

Training on single GPU.

Initializing image normalization.

	Epoch	 	Iteration	 	Time Elapsed (seconds)	 	Mini-batch Loss	Mi A	ni-batch ccuracy	Base F	Learning
==== 	1 3 5 8 10 13 15 18 20 23 25		1 10 20 30 40 50 60 70 80 90 100		0.47 10.31 18.96 27.43 35.31 43.17 50.15 56.84 63.00 69.37 74.85		3.5061 0.7686 0.2371 0.0770 0.0336 0.0289 0.0104 0.0072 0.0210 0.0035 0.0027		7.81% 75.00% 92.19% 97.66% 99.22% 99.22% 100.00% 100.00% 99.22% 100.00% 100.00%		0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010
 	28 30	1	110 120	1	81.19 86.75		0.0053 0.0045	1	100.00% 100.00%	l I	0.0010

Training on single GPU.

Initializing image normalization.

 	Epoch		Iteration	Time Elapsed (seconds)	Mini-batch Loss	Mini-batch Accuracy	Base Learning Rate
==== 	1 3 5 8 10 13 15 18 20 23 25 28		1 10 20 30 40 50 60 70 80 90 100 110	0.47 10.37 18.96 27.43 35.37 43.17 50.15 56.84 63.00 69.37 74.85 81.19	3.5061 0.7686 0.2371 0.0770 0.0336 0.0289 0.0104 0.0072 0.0210 0.0035 0.0027 0.0027 0.0053	7.81% 75.00% 92.19% 97.66% 99.22% 99.22% 100.00% 99.22% 100.00% 100.00% 100.00% 100.00% 100.00% 100.00% 100.00%	0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010 0.0010

Training on single GPU.

Initializing image normalization.

	Epoch	1	Iteration	1	Time Elapsed (seconds)	Min	i-batch Loss	 	Mini-batch Accuracy		Base Learning Rate
 	1	1	1	I	0.47		3.5061	1	7.81%	1	0.0010
1	3	1	10	1	10.31		0.7686	1	75.00%	I	0.0010
1	5	1	20	1	18.96		0.2371	1	92.19%	T	0.0010
1	8	I.	30	L	27.43		0.0770	Т	97.66%	I	0.0010
1	10	1	40	L	35.31		0.0336	1	99.22%	1	0.0010
1	13	T.	50	L	43.17		0.0289	1	99.22%	I.	0.0010
1	15	I.	60	I.	50.15		0.0104	L	100.00%	T	0.0010
1	18	1	70	1	56.84		0.0072	L	100.00%	I.	0.0010
1	20	1	80	1	63.00		0.0210	L	99.22%	I	0.0010
1	23	1	90	L	69.37		0.0035	L	100.00%	I.	0.0010
1	25	L	100	T	74.85		0.0027	L	100.00%	I	0.0010
1	28	1	110	1	81.19		0.0053	1	100.00%	1	0.0010
1	30	I	120	I	86.75		0.0045	I	100.00%	i	0.0010







Training Algorithm Options



- Momentum

4. Test the performance of the CNN

```
% calculate the confusion matrix. |
confMat = confusionmat(validationDS.Labels, labels); Compute the confusion matrix
confMat = bsxfun(@rdivide,confMat,sum(confMat,2));
fprintf('Performance on validation set \t\t\t%.4f\n',mean(diag(confMat)));
```

Report the mean accuracy



>> [cm,grp] = confusionmat(yObserved,yPred)









5. Plotting options

- 1. Plot wrongly classifies images from the ValidationDS
- 2. Plot the filters from the Convolution layers
- 3. Plot the feature maps for some of the input images

Figure 65	Figure 66 🛄 🖄		▲ Figure 68 □ □ ☑ ▲ □ □
<u>File Edit View Insert Tools D</u> esktop <u>W</u> indow <u>H</u> elp	■ <u>File Edit View Insert T</u> ools <u>D</u> esktop <u>W</u> indow <u>H</u> elp	ש <u>F</u> ile <u>E</u> dit <u>V</u> iew Insert <u>T</u> ools <u>D</u> esktop <u>W</u> indow <u>H</u> elp ש	Eile Edit View Insert Tools Desktop Window Help > I
🗋 😂 🛃 🌭 🔍 🥄 🧐 🐙 🔏 • 😓 🗖 🖽 🖿 🛄	🗌 🔁 🗃 🖕 🗞 🍳 🖑 🕲 🐙 🔏 • 🗔 🗉 💷 💷	1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	1 2 2 3 3 8 3 3 3 2 3 2 3 2 3 2 3 2 3 3 2 3 4 3 3 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
· · · · · · · · · · · · · · · · · · ·			
100			
80 -			
			1 💽 🏩 🌉 🛃 🔢
60			
40 -			
20			-2
		(0)	
20 40 60 80 100			1 1.
▲ Figure 165 🗆 🗆 🕅	Figure 166	Figure 167	▲ Figure 168
<u>File E</u> dit <u>V</u> iew Insert <u>T</u> ools <u>D</u> esktop <u>W</u> indow <u>H</u> elp	▶ <u>F</u> ile <u>E</u> dit <u>V</u> iew <u>I</u> nsert <u>T</u> ools <u>D</u> esktop <u>W</u> indow <u>H</u> elp	۲ <u>F</u> ile <u>E</u> dit <u>V</u> iew Insert <u>T</u> ools <u>D</u> esktop <u>W</u> indow <u>H</u> elp ۲	Eile Edit View Insert Tools Desktop Window Help N
🗋 🗃 🛃 🔌 🔍 🔍 🖑 🕲 🐙 🔏 • 🗔 🔲 🗉 💷 💷	🗋 🔁 🖬 🖕 🔍 🄍 🕲 🗐 🐙 🔏 • 🗔 🔲 📰 💷 🖽	1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	1 2 2 2 3 8 3 3 3 2 3 2 3 2 3 2 3 2 3 2 3
· · · · · · · · · · · · · · · · · · ·			
	\rightarrow \rightarrow \rightarrow \rightarrow		
100			
80 -			
		A A A 🔍 A 🔨	I 📑 🔊 🥂 🏹 🐹 📰 II
60			
40 -			
20	\rightarrow \rightarrow \rightarrow		
20 40 60 80 100			
Figure 1	Figure 3	Figure 52	■ Figure 365
<u>Eile Edit View Insert Tools Desktop Window H</u> elp	Eile Edit View Insert Tools Desktop Window Help	■ <u>File Edit View Insert T</u> ools <u>D</u> esktop <u>W</u> indow <u>H</u> elp	<u>F</u> ile <u>E</u> dit <u>V</u> iew Insert <u>I</u> ools <u>D</u> esktop <u>W</u> indow <u>H</u> elp ∞
1 🗃 🛃 🦕 🔍 🤍 🖑 🕲 🐙 🔏 • 🗔 🗖 📰 🖿 🛄	🚺 🗂 🗃 🛃 🔌 🗞 🍳 🍳 🖑 🕲 🐙 🔏 - 🗔 🗉 📰 💷 💷	🔄 🔁 🗁 🖂 🌭 🗞 🍳 🖑 🧐 🐙 🔏 - 😓 🗉 🗉	🗋 🗃 🗟 🔈 🍳 🔍 🖑 🕲 🐙 🖌 • 🗔 🗉 💷 🔲
	1 conv layer, weights of filter 1:16	2 conv layer, weights of filter 32	3 conv layer, weights of filter 32
classified as circles m classified as circles m			
100			
80 🗙 80			
60 60			
40 40			
20 20		COMPANY DATABASE LANCES LANCE	
20 40 60 80 100 20 40 60 80 100		200 Ball 200 Ball	10740 E228
		GRANE DOLDS HEADS (ADS	55-65 8535



Input Feature Map

Rectified Feature Map





Rectified Feature Map

Max











ssol 0.4

0.3

0.2

0.1

0

Epoch 1

Epoch 2

20

Epoch 3

60

40

Epoch 4

80

Epoch 5

100

Epoch 6 Epoch 7

120

Results Validation accuracy: N/A Training finished: Reached final iteration **Training Time** Start time: 20-Nov-2019 14:14:55 Elapsed time: 6 min 25 sec **Training Cycle** Epoch: 10 of 10 Iteration: 210 of 210 Iterations per epoch: 21 Maximum iterations: 210 Validation Frequency: N/A N/A Patience: **Other Information** Hardware resource: Single CPU Learning rate schedule: Piecewise 0.001 Learning rate:

Learn more

Epoch 9

180

Epoch 10

200

Epoch 8

160

140



You need to submit:

- 1. Upload a single .zip file with the filename [firstname_lastname_UVA_computing_ID].zip
- 2. The zip file should contain:
 - a. The original training_data folder with the subfolders circles and crosses with the images included (this is < 3.6 Mb)
 - b. Your solution to BasicCNNtemplate.m
 - c. Your best performing network in the form of a .mat file XONet (This file is automatically created when doTraining == true)
 - d. Your responses to the effect of Momentum, InitialTraiingRate, and Epochs on the performance of the network Include supporting plots and accuracy values.
 - i. This can be a PDF with the plots and inferences included.
 - e. Report (with plots) on the architecture, and accuracy of your best performing network:
 - i. Include an image of the layers of the network.
 - ii. Report accuracy (as computed by the template, using the confusion matrix) on the validationDS of your best performing model.
 - iii. Report the chosen values of the hyperparameters of your network.

Submit your best model as a .mat file

Cι	Current Folder								
	Name 🔺								
	training_data								
	BasicCNNsolution.m								
	🕂 XONet.mat								
	🕂 XONet2.mat								

Not mandatory to use Matlab: [Part 1]

1. Use whatever DL framework you are familiar/comfortable with.

2. Provide all your code and include a 'requirements.txt' file to list all the dependencies needed to run the code.

[https://pip.readthedocs.io/en/1.1/requirements.html]

3. You are responsible for generating all the plots required by the assignment.



Not mandatory to use Matlab: [Part 2]

1. Must provide the best performing CNN as a .mat file

2. Use Open Neural Network Exchange (ONNX) standard.

Open Neural Network Exchange (ONNX)

1. Export your CNN from your framework as a ONNX model. Examples:

https://github.com/onnx/tutorials

2. Use importONNXNetwork in Matlab and generate the .mat file





IM & GENET

How a dataset changed deep learning

The Beginning: CVPR 2009



Jia Deng, Wei Dong, Richard Soch	er, Li-Jia Li, Kai Li and Li Fei-Fei Princeton University, USA	-	
filadane, where, rancher, ital.	11. faifailliga.primaton.adu	-	- 11
Chemistic sheets second head	in contraction from the second second		1.64
		14	in the
Abstract	content-based image search and image understanding also-		25.
/ which and	rithms, as well as for providing critical training and bench-	1000	
The explosion of image data on the Internet has the po-	marking data for such algorithms.	100	114
ttial to foster more sophisticated and robust models and	ImageNet uses the hierarchical structure of WordNet [9].	1	111
porithms to index, retrieve, organize and interact with im-	Each meaningful concept in WordNet, possibly described	53	100
es and multimedia data. But exactly how such data can	by multiple words or word phrases, is called a "synonym	20	11
harnessed and organized remains a critical problem. We	set" or "synset". There are around 80,000 noun synsets	22	- 65
roduce here a new database called "ImageNet", a large-	in WordNet. In ImageNet, we aim to provide on aver-		101
ale ontology of images built upon the backbone of the	age 500-1000 images to illustrate each synset. Images of	1.00	ge
the 80.000 consets of WanDet with an average of 500.	as described in Sec. 3.3 Instachet therefore will offer		alu
the support system by without with an average by Soo- 600 clean and full resolution images. This will result in	tens of millions of clearly sorted images. In this name,		60
ns of millions of annotated images organized by the se-	we report the current version of ImageNet, consisting of 12		RC N
antic hierarchy of WordNet. This paper offers a detailed	"subtrees": mammal, bird, fisk, reptile, amphibian, vehicle,		ο.
subsits of ImageNet in its current state: 12 subtrees with	furniture, munical instrument, geological formation, tool,		10
147 synsets and 3.2 million images in total. We show that	flower, fruit. These subtrees contain 5247 synsets and 3.2	10	il.
sageNet is much larger in scale and diversity and much	million images. Fig. 1 shows a snapshot of two branches of		24
ore accurate than the current image datasets. Construct-	the mammal and vehicle subtrees. The database is publicly		- 90
g such a large-scale database is a challenging task. We	available at ht tp://www.image-net.org.		10
scribe the data collection scheme with Amazon Mechan-	The rest of the paper is organized as follows: We first		645
al Turk. Lastly, we ellustrate the usefulness of ImageNet	show that ImageNet is a large-scale, accurate and diverse		200
rough three simple applications in object recognition, in- the classification and automatic object clustering. We have	image database (Section 2). In Section 4, we present a few	5	1
at the scale, accuracy, diversity and hierarchical struc-	simple application examples by exploring the current tria- reNet, mostly the mammal and vehicle subtrees. Our real		6
re of ImageNet can offer unnaralleled consortanities to re-	is to show that ImaneNet can serve as a useful resource for		10
archers in the computer vision community and beyond.	visual recognition applications such as object recognition,	ween	- 20
	image classification and object localization. In addition, the	are of	60
	construction of such a large-scale and high-quality database	andes	
Introduction	can no longer rely on traditional data collection methods.	400	64
	Sec. 3 describes how ImageNet is constructed by leverag-		100
The digital era has brought with it an enormous explo-	ing Amazon Mechanical Turk.	6 the	1.9
in of data. The latest estimations put a number of more	2. Properties of ImageNet	ople.	100
in 3 billion photos on Flickr, a similar number of video	at respectives of mangement	es of	RO
ps on touture and an even targer number for images in a Groude Image Search database. More conhisticated and	ImageNet is built upon the hierarchical structure pro-	ette-	12
bust models and almorithms can be proposed by exploit.	vided by WordNet. In its completion, ImageNet aims to	erve	10
e these images, resulting in better applications for users	contain in the order of 30 minion cleanty tabeled full rese-		100
index, retrieve, organize and interact with these data. But	is written. Images (100-2000 per synetc), At the time outs paper	Re B	
actly how such data can be utilized and organized is a	will be based on the mammal and vehicle subtrees.	s the	1
oblem yet to be solved. In this paper, we introduce a new		sam-	10
age database called "ImageNet", a large-scale ontology	Scale ImageNet aims to provide the most comprehensive	reci-	41
images. We believe that a large-scale ontology of images	and diverse coverage of the image world. The current 12	a for	50
a critical resource for developing advanced, large-scale	subtrees consist of a total of 3.2 million cleanly annotated	z the	100
		outy.	11
	1		16
		100	1.64
of images, but only a solver of GK, images are sublicle acceletate	jects in images should have variable appearances, no	witions.	340
	the second		100
		-	_
how mercany one in a research they have a	where we make a state of the second state of t		and a local division of the

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2009.

IM GENET on Google Scholar



Imagenet: A large-scale hierarchical image database J Deng, W Dong, R Socher, LJ Li, K Li... - Computer Vision and ..., 2009 - ieeexplore.ieee.org Abstract: The explosion of image data on the Internet has the potential to foster more sophisticated and robust models and algorithms to index, retrieve, organize and interact with images and multimedia data. But exactly how such data can be harnessed and organized Cited by 4386 Related articles All 30 versions Cite Save



Imagenet large scale visual recognition challenge O Russakovsky, J Deng, H Su, J Krause... - International Journal of ..., 2015 - Springer Abstract The ImageNet Large Scale Visual Recognition Challenge is a benchmark in object category classification and detection on hundreds of object categories and millions of images. The challenge has been run annually from 2010 to present, attracting participation Cited by 2847 Related articles All 17 versions Cite Save

...and many more.

From IM GENET Challenge Contestants to Startups



V i S E N Z E Simplifying the Visual Web

clarifai





DNNresearch VUNO

"The IM GENET of x"



SpaceNet DigitalGlobe, CosmiQ Works, NVIDIA



MusicNet J. Thickstun et al, 2017



Medical ImageNet Stanford Radiology, 2017



ShapeNet A.Chang et al, 2015



EventNet G. Ye et al, 2015



ActivityNet F. Heilbron et al, 2015

Hardly the First Image Dataset



Segmentation (2001) D. Martin, C. Fowlkes, D. Tal, J. Malik.



KTH human action (2004) I. Leptev & B. Caputo



CAVIAR Tracking (2005) R. Fisher, J. Santos-Victor J. Crowley



CMU/VASC Faces (1998) H. Rowley, S. Baluja, T. Kanade



Sign Language (2008) P. Buehler, M. Everingham, A. Zisserman



Middlebury Stereo (2002) D. Scharstein R. Szeliski



MSRC(2006) Shotton et al. 2006



PASCAL(2007) Everingham et al, 2009



FERET Faces (1998) P. Phillips, H. Wechsler, J. Huang, P. Raus



UIUC Cars (2004) S. Agarwal, A. Awan, D. Roth



CalTech 101/256 (2005) Fei-Fei et al, 2004 GriffIn et al, 2007



(2007) Yao et al, 2007



COIL Objects (1996) S. Nene, S. Navar, H. Murase



3D Textures (2005) S. Lazebnik, C. Schmid, J. Ponce

77	1 Bet	
		10

LabelMe (2005) Russell et al, 2005



Lotus Hill



MNIST digits (1998-10) Y LeCun & C. Cortes



CuRRET Textures (1999) K. Dana B. Van Ginneken S. Nayar

J. Koenderink

2:05	The E	SP Game	0090	15 6 1 2	Dog
	80		Tere Barren 	20-	Leash German
1			Ξ.		Shepard
					 Standing Canine

ESP (2006) Ahn et al, 2006



TinyImage (2008) Torralba et al. 2008

A new way of thinking...

To shift the focus of Machine Learning for visual recognition

from modeling...

...to data. Lots of data.

While Others Targeted Detail...



LabelMe

Per-Object Regions and Labels Russell et al, 2005



Lotus Hill

Hand-Traced Parse Trees Yao et al, 2007

...ImageNet Targeted Scale


IM GENET Goals



Carnivore

- Canine
 - Dog
 - Working Dog
 - Husky



High Resolution

To better replicate human visual acuity

High-Quality Annotation

To create a benchmarking dataset and advance the state of machine perception, not merely reflect it

Free of Charge

To ensure immediate application and a sense of community

Neural Nets are Cool Again!



13,259 Citations Imagenet classification with deep convolutional neural networks A Krizhevsky, I Sutskever, GE Hinton - Advances in neural ..., 2012 - papers.nips.cc Abstract We trained a large, deep convolutional neural network to classify the 1.3 million high-resolution images in the LSVRC-2010 ImageNet training set into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 39.7\% and 18.9\% Cited by 13259 Related articles All 95 versions Cite Save

...And Cooler and Cooler \boldsymbol{J}





"VGG Net"







[Krizhevsky et al. NIPS 2012]

[Szegedy et al. CVPR 2015]

[Simonyan & Zisserman, ICLR 2015] [He et al. CVPR 2016]



"First, we find that the performance on vision tasks still increases linearly with orders of magnitude of training data size."

C. Sun et al, 2017

Revisiting Unreasonable Effectiveness of Data in Deep Learning Era

Chen Sun¹, Abhinav Shrivastava^{1,2}, Saurabh Singh¹, and Abhinav Gupta^{1,2}

¹Google Research ²Carnegie Mellon University

Abstract

2017

Jul

10

CV

CS.

.02968v1

arXiv:1707

The success of deep learning in vision can be attributed to: (a) models with high capacity; (b) increased computational power; and (c) availability of large-scale labeled data. Since 2012, there have been significant advances in representation capabilities of the models and computational capabilities of GPUs. But the size of the biggest dataset has surprisingly remained constant. What will happen if we increase the dataset size by $10 \times \text{ or } 100 \times$? This paper takes a step towards clearing the clouds of mystery surrounding the relationship between 'enormous data' and deep learning. By exploiting the JFT-300M dataset which has more than 375M noisy labels for 300M images, we investigate how the performance of current vision tasks would change if this data was used for representation learning. Our paper delivers some surprising (and some expected) findings. First, we find that the performance on vision tasks still increases linearly with orders of magnitude of training data size. Second, we show that representation learning (or pretraining) still holds a lot of promise. One can improve performance on any vision tasks by just training a better base model. Finally, as expected, we present new state-of-theart results for different vision tasks including image classification, object detection, semantic segmentation and human pose estimation. Our sincere hope is that this inspires vision community to not undervalue the data and develop collective efforts in building larger datasets.



Figure 1. The Curious Case of Vision Datasets: While GPU computation power and model sizes have continued to increase over the last five years, size of the largest training dataset has surprisingly remained constant. Why is that? What would have happened if we have used our resources to increase dataset size as well? This paper provides a sneak-peek into what could be if the dataset sizes are increased dramatically.

ously, while both GPUs and model capacity have continued to grow, datasets to train these models have remained stangant. Even a 101-layer ReeNet with significantly more

Fine-Grained Recognition



[Gebru, Krause, Deng, Fei-Fei, CHI 2017]





2567 classes 700k images

Case Study: LeNet-5

[LeCun et al., 1998]



Conv filters were 5x5, applied at stride 1 Subsampling (Pooling) layers were 2x2 applied at stride 2 i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC] ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Figure copyright Kaiming He, 2016. Reproduced with permission.

Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture: [227x227x3] INPUT CONV: 96 11x11 filters at stride 4, pad 0 MAX POOL1: 3x3 filters at stride 2 NORM1: Normalization layer CONV2: 265 5x5 filters at stride 1, pad 2 MAX POOL 2: 3x3 filters at stride 2 NORM2: Normalization layer CONV3: 384 3x3 filters at stride 1, pad 1 CONV4: 384 3x3 filters at stride 1, pad 1 CONV5: 265 3x3 filters at stride 1, pad 1 MAX POOL 3: 3x3 filters at stride 2 FC6: Fully connected layer (4096 neurons) FC7: Fully connected layer (4096 neurons) FC8: 1000 neurons (logit scores)



Details/Retrospectives:

- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture: [227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

- [4096] FC6: 4096 neurons
- [4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)



[55x55x48] x 2

Historical note: Trained on GTX 580 GPU with only 3 GB of memory. Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Figure copyright Kaiming He, 2016. Reproduced with permission.

Case Study: ZFNet [Zeiler and Fergus, 2013]



AlexNet but:

```
CONV1: change from (11x11 stride 4) to (7x7 stride 2)
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512
```

ImageNet top 5 error: 15.4% -> 14.8%

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Figure copyright Kaiming He, 2016. Reproduced with permission.

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet) -> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1 and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13 (ZFNet)

-> 7.3% top 5 error in ILSVRC'14



Softmax

FC 4096 FC 4096

Pool

Pool

Input



ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Figure copyright Kaiming He, 2016. Reproduced with permission.

Case Study: ResNet

[He et al., 2015]

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!





Case Study: ResNet [He et al., 2015] ILSVRC 2015 winner (3.6% top 5 error)



Slide from Kaiming He's recent presentation https://www.youtube.com/watch?v=1PGLj-uKT1w

CIFAR-10 experiments





Case Study: ResNet [He et al., 2015]





ection - NVDriveNet detection







Front:

2019 Internationa

0

-

2

car

suv-truck

suv-truck



Rear :

car

Semantic Segmentation

Classification + Localization

Object Detection

Instance Segmentation











Raw input image (left) and input image with labeled ground truth (right).





📣 Ground Truth Labeler						 .	
LABEL					🔁 DemoFolder 者 Dock 🛃 Undoo	ck 🛃 🔚 🔏 🛍 🖆	• = = •
Load Save Import • • Labels • Or Pan	Default Layout Show ROI Labels Show Scene Labels	Algorithm: ☐ Point Tracker ▼ ② Configure Automation	Automate View Label Summary	Export Labels 👻			
FILE MODE	VIEW	AUTOMATE LABELING	SUMMARY	EXPORT			X
ROI Label Definition	01_city_c2s_fcw_10s.r	np4			1 1 20 20 1		
Define new ROI label		_	1				
▶ Vehicle					// 🛃		
▶ Pedestrian		1		F			
TrafficLight			and the state	1	TrafficLigI TrafficLight		
▶ leftLane •		140	A Contraction	b.h.a.			
▶ rightLane ····			1 Conte				
		perstand		Vabiala	Vehicle - right and		
			Venicie di	Venicie			-
Scene Label Definition				1			
			1				
Current Frame Add Label							
O Time Interval Remove Label			leftLane				
Before you can label a scene, begin by defining a Scene Label.	٩						F
	00.00000 05.75487	10.20000 10.20000	0			Zoom In Time I	oterval
	Start Time Current	End Time Max Time	9		<u>لات التا (تتا</u>		

regressionOutputs =

1225×6 **table**

leftLane_a	leftLane_b	leftLane_c	rightLane_a	rightLane_b	rightLane_c	
3.5482e-05	0.0060327	1.7599	-0.00015691	0.030256	-2.0559	
-3.9519e-05	0.014116	1.662	-0.00097636	0.02979	-2.0749	
-6.778e-07	-0.00063158	1.776	-7.0963e-05	0.0024721	-1.9428	
-0.00023646	0.0088324	1.8188	-0.00050391	-0.0015166	-1.973	
-0.00055867	0.012996	1.8074	-8.6643e-05	0.00098652	-1.935	
0 00000000	A AAAAA	1 7015	0 00005000	0 011000	1 0000	

Lane Detection with Deep Learning



Canny Edge Detection



Perspective Transformation of an Image



The 'S' channel, or Saturation, with binary activation



A few more thresholds (left) for activation, with the resulting perspective transformation



Sliding windows and a decent-looking result



- Perspective transformation is fairly specific to the camera
- Gradient and color thresholds only work in a small set of conditions
- Slow 5-8 fps
0.27 meters left of center Radius of Curvature = 3.0(m)





One of the new labels — a lane image



Activation maps of the first few layers











Top left: Input – Perspective Transformed Image Output – Six polynomial coefficients

Top right: Input – Road Image Output – Six polynomial coefficients

Bottom left: Input – Road Image Output – Lane in 'G' color channel

Principles of modeling for CPS – Fall 2017



