**Labeling and gTruth for SSD and YOLO Convolutional State Recognition**

* First attempt – recognizing just the ZERO state
* Second attempt – add 3 classes: left, ZERO, right and have it learn basic location of pendulum
* Third attempt – N classes with ZERO, left X degrees, right X degrees
* Best comparison is SSDV1, for cars – included with updates in <https://github.com/sbsiewertcsu/ERP/tree/master/MATLAB>
* Best MATLAB references:
  + <https://www.mathworks.com/help/vision/ug/get-started-with-the-image-labeler.html>
  + <https://www.mathworks.com/help/vision/ref/imagelabeler-app.html>

1. Run “imageLabeler” in MATLAB and import PPM 24-bit images, label each state (I just did “zero”) and save the session file. Note that you must export a “table” to MATLAB and then save that to a file as “groundTruth” or similar name. The table should look like this:

>> SSDV1pend

ans =

4×2 table

imageFilename zero

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

{'C:\Users\sbsiewert\Documents\MATLAB\pendTruth\temp-10282022162323-282.ppm'} {1×4 double}

{'C:\Users\sbsiewert\Documents\MATLAB\pendTruth\temp-10282022162325-473.ppm'} {1×4 double}

{'C:\Users\sbsiewert\Documents\MATLAB\pendTruth\temp-10282022162327-663.ppm'} {1×4 double}

{'C:\Users\sbsiewert\Documents\MATLAB\pendTruth\temp-10282022162329-853.ppm'} {1×4 double}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Training an SSD Object Detector for the following object classes:

\* zero

Training on single GPU.

Initializing input data normalization.

|=======================================================================================================|

| Epoch | Iteration | Time Elapsed | Mini-batch | Mini-batch | Mini-batch | Base Learning |

| | | (hh:mm:ss) | Loss | Accuracy | RMSE | Rate |

|=======================================================================================================|

| 1 | 1 | 00:00:07 | 1.2463 | 57.40% | 1.95 | 0.0010 |

| 17 | 50 | 00:01:53 | 0.0637 | 96.94% | 0.34 | 0.0010 |

| 34 | 100 | 00:03:40 | 0.0417 | 98.13% | 0.28 | 0.0008 |

| 50 | 150 | 00:05:22 | 0.0324 | 98.57% | 0.24 | 0.0008 |

| 67 | 200 | 00:07:21 | 0.0255 | 98.81% | 0.22 | 0.0006 |

| 84 | 250 | 00:09:07 | 0.0265 | 98.93% | 0.22 | 0.0006 |

| 100 | 300 | 00:10:50 | 0.0213 | 99.05% | 0.20 | 0.0005 |

| 117 | 350 | 00:12:36 | 0.0194 | 99.09% | 0.19 | 0.0005 |

| 134 | 400 | 00:14:22 | 0.0194 | 99.16% | 0.19 | 0.0004 |

| 150 | 450 | 00:16:05 | 0.0170 | 99.18% | 0.18 | 0.0004 |

| 167 | 500 | 00:17:54 | 0.0221 | 99.23% | 0.20 | 0.0003 |

| 184 | 550 | 00:19:43 | 0.0184 | 99.24% | 0.18 | 0.0003 |

| 200 | 600 | 00:21:26 | 0.0143 | 99.26% | 0.16 | 0.0003 |

| 217 | 650 | 00:23:13 | 0.0167 | 99.27% | 0.17 | 0.0002 |

| 234 | 700 | 00:25:01 | 0.0166 | 99.30% | 0.17 | 0.0002 |

| 250 | 750 | 00:26:43 | 0.0187 | 99.31% | 0.19 | 0.0002 |

| 267 | 800 | 00:28:29 | 0.0125 | 99.33% | 0.15 | 0.0002 |

| 284 | 850 | 00:30:19 | 0.0140 | 99.34% | 0.16 | 0.0001 |

| 300 | 900 | 00:32:01 | 0.0142 | 99.36% | 0.16 | 0.0001 |

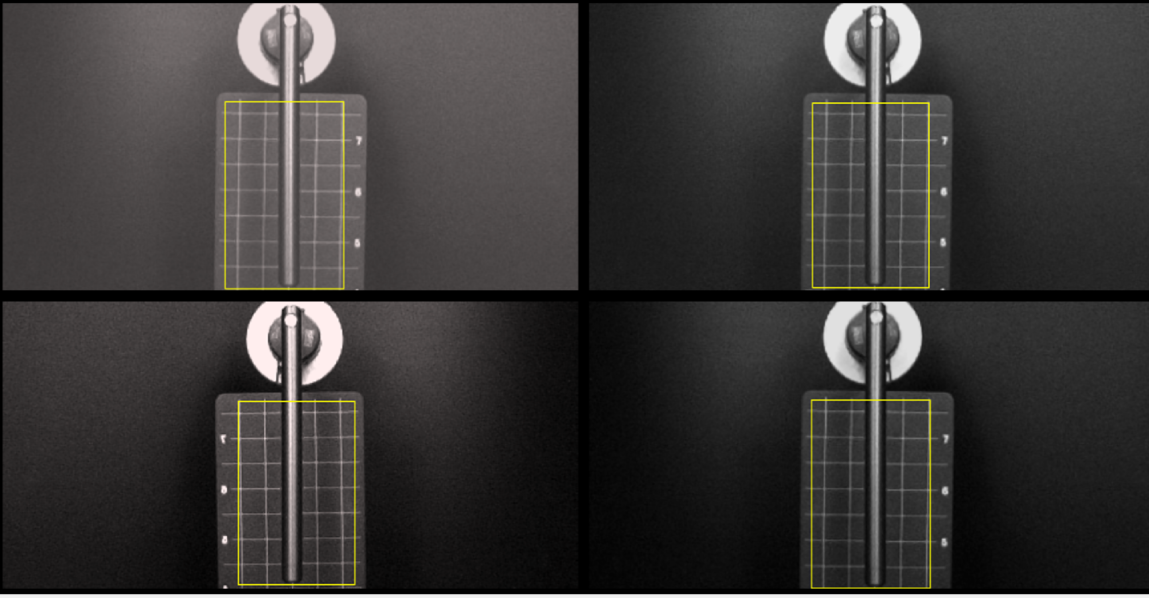
|=======================================================================================================|

Detector training complete.

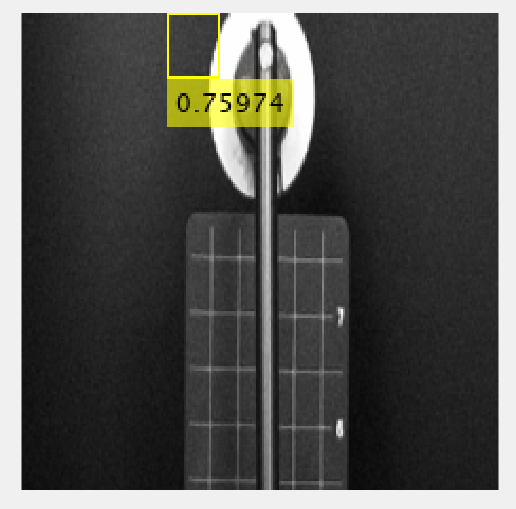
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

>>

The input was this (lighting adjusted to make it generalize to different conditions):



The test image claims 75.974% confidence, but boxes an odd location?



1. Download MATAB and run on your machine to get familiar with code – the path problem we saw is the gTruth data paths, which are generated by imageLabler – I would load the “\*session.mat” files and Export gTruth as a “table” in your environment
   1. Perhaps there is a better solution to the gTruth pathing issue, but I do not know how to change – this work-around has worked for the group so far
   2. Otherwise, the data can just be unzipped in place as long as the paths in gTruth match where the files are
2. Had to recall how we made the cow gTruth – we really somehow made “cowDatasetGroundTruth.mat”, which can be imported by just double clicking and becomes “gTruth” as a MATLAB variable.
   1. We repeated this for eleDatasetGroundTruth.mat and it works the same.
   2. This was based on vehicleDatasetGroundTruth.mat, the starting example.
3. gTruth table should be loaded

**SSD Network Configuration**:

<https://www.mathworks.com/help/vision/ref/ssdlayers.html>

Best references on network configuration:

* <https://www.mathworks.com/help/vision/ug/getting-started-with-ssd.html>
* <https://www.mathworks.com/help/deeplearning/ref/vgg16.html> - much simpler than resnet50
* <https://www.mathworks.com/help/deeplearning/ug/train-deep-learning-network-to-classify-new-images.html>
* <https://www.mathworks.com/help/deeplearning/ug/train-residual-network-for-image-classification.html>

**Training**:

<https://www.mathworks.com/help/vision/ref/trainssdobjectdetector.html>

Most likely want multi-class detector with classes that include zero offset left and right be specific angles.

**MATLAB Deep Learning Examples (Original)**

1. SSD - <https://www.mathworks.com/help/vision/ug/object-detection-using-single-shot-detector.html>

openExample('deeplearning\_shared/ObjectDetectionUsingSSDDeepLearningExample')

1. RCNN - <https://www.mathworks.com/help/vision/ug/object-detection-using-faster-r-cnn-deep-learning.html>

openExample('deeplearning\_shared/DeepLearningFasterRCNNObjectDetectionExample')

1. YOLO V2 - <https://www.mathworks.com/help/vision/ug/train-yolo-v2-network-for-vehicle-detection.html>

openExample('vision/TrainYOLOV2NetworkForVehicleDetectionExample')

1. YOLO V3 - <https://www.mathworks.com/help/vision/ug/object-detection-using-yolo-v3-deep-learning.html>

openExample('deeplearning\_shared/ObjectDetectionUsingYOLOV3DeepLearningExample')

doTraining = true;

C:\Program Files\MATLAB\R2020a\examples\deeplearning\_shared\data

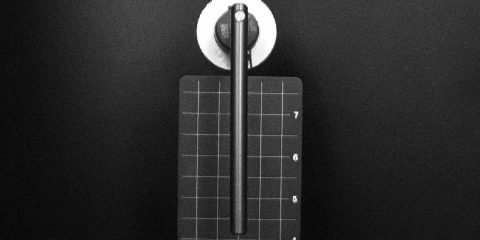


**We need new GroundTruth and new DatasetImages for our objects of interest.**

[**https://www.mathworks.com/help/deeplearning/ug/data-sets-for-deep-learning.html**](https://www.mathworks.com/help/deeplearning/ug/data-sets-for-deep-learning.html)

|  |  |
| --- | --- |
| Vehicle  https://www.mathworks.com/help/deeplearning/ug/vehicle.png | The Vehicle data set consists of 295 images containing one or two labeled instances of a vehicle. This small data set is useful for exploring the YOLO-v2 training procedure, but in practice, more labeled images are needed to train a robust detector.  The images are of size 720-by-960-by-3.  Extract the Vehicle data set. Set dataFolder to the location of the data.  filename = 'vehicleDatasetImages.zip';  dataFolder = fullfile(tempdir,'vehicleImages');  if ~exist(dataFolder,'dir')  unzip(filename,tempdir);  end  Load the data set as a table of file names and bounding boxes from the extracted MAT file and convert the file names to absolute file paths.  data = load('vehicleDatasetGroundTruth.mat');  vehicleDataset = data.vehicleDataset;  vehicleDataset.imageFilename = fullfile(tempdir,vehicleDataset.imageFilename);  Create an image datastore containing the images and a box label datastore containing the bounding boxes using the imageDatastore and boxLabelDatastore functions, respectively. Combine the resulting datastores using the combine function.  filenamesImages = vehicleDataset.imageFilename;  tblBoxes = vehicleDataset(:,'vehicle');  imds = imageDatastore(filenamesImages);  blds = boxLabelDatastore(tblBoxes);  cds = combine(imds,blds);  For an example showing how to process this data for deep learning, see [Object Detection Using YOLO v2 Deep Learning](https://www.mathworks.com/help/deeplearning/ug/object-detection-using-yolo-v2.html). |

For the Pendulum (Synchronome) the images were capture with the FLIR high speed camera in PGM format and look like this:



For the resnet50 Deep Learning architecture, they must be converted into 24-bit PPM grayscale done with:

images = 'C:\Users\siewerts\Documents\MATLAB\pendTruth\PGM';

pgmfiles=dir(fullfile(images, '\\*.pgm\*'));

cd C:\Users\siewerts\Documents\MATLAB\pendTruth\PGM

fullFileName = fullfile(theFiles(k).folder, baseFileName);

for i=1:length(pgmfiles)

imshow(pgmfiles(i).name)

end

% this is a 256x256x1 grayscale image

gimage = imread(pgmfiles(i).name);

% this is a 256x256x3 grayscale image

cimage = repmat(gimage,[1 1 3]);

for i=1:length(pgmfiles)

pgmfullfile=fullfile(pgmfiles(i).folder, pgmfiles(i).name);

gimage = imread(pgmfullfile);

cimage = repmat(gimage,[1 1 3]);

imshow(cimage)

end

ppmfiles=pgmfiles;

for i=1:length(pgmfiles)

pgmfiles(i).name

end

for i=1:length(pgmfiles)

ppmfiles(i).name(length(pgmfiles(i).name)-1)='p';

%length(pgmfiles(i).name)

end

for i=1:length(ppmfiles)

ppmfiles(i).name

length(pgmfiles(i).name)

end

for i=1:length(pgmfiles)

pgmfullfile=fullfile(pgmfiles(i).folder, pgmfiles(i).name);

gimage = imread(pgmfullfile);

cimage = repmat(gimage,[1 1 3]);

cimagefullfile=fullfile('C:\Users\siewerts\Documents\MATLAB\pendTruth\PGM24', ppmfiles(i).name);

imshow(cimage)

imwrite(cimage, cimagefullfile);

end

**MATLAB version and toolboxes used**

>> ver

-----------------------------------------------------------------------------------------------------

MATLAB Version: 9.8.0.1451342 (R2020a) Update 5

MATLAB License Number: 40937346

Operating System: Microsoft Windows 10 Pro Version 10.0 (Build 18362)

Java Version: Java 1.8.0\_202-b08 with Oracle Corporation Java HotSpot(TM) 64-Bit Server VM mixed mode

-----------------------------------------------------------------------------------------------------

MATLAB Version 9.8 (R2020a)

Simulink Version 10.1 (R2020a)

Aerospace Blockset Version 4.3 (R2020a)

Aerospace Toolbox Version 3.3 (R2020a)

Automated Driving Toolbox Version 3.1 (R2020a)

Computer Vision Toolbox Version 9.2 (R2020a)

Control System Toolbox Version 10.8 (R2020a)

DSP System Toolbox Version 9.10 (R2020a)

Deep Learning Toolbox Version 14.0 (R2020a)

Embedded Coder Version 7.4 (R2020a)

GPU Coder Version 1.5 (R2020a)

Image Acquisition Toolbox Version 6.2 (R2020a)

Image Processing Toolbox Version 11.1 (R2020a)

MATLAB Coder Version 5.0 (R2020a)

Parallel Computing Toolbox Version 7.2 (R2020a)

Partial Differential Equation Toolbox Version 3.4 (R2020a)

ROS Toolbox Version 1.1 (R2020a)

Robotics System Toolbox Version 3.1 (R2020a)

Sensor Fusion and Tracking Toolbox Version 1.3 (R2020a)

Signal Processing Toolbox Version 8.4 (R2020a)

Statistics and Machine Learning Toolbox Version 11.7 (R2020a)

Symbolic Math Toolbox Version 8.5 (R2020a)

Vehicle Dynamics Blockset Version 1.4 (R2020a)

>>