Daniel Kim
Jason Ligon
CPSC 585

User Manual (Fiat Visio)

1. Introduction

1.1 Purpose

The user manual explains the use and operation of our code and allows for any parties to setup and execute our code successfully. The first draft of this document was written on May 1, 2017 and finalized on May 18, 2017. The authors of this document are: Daniel Kim and Jason Ligon. Daniel was responsible for sections 3, 4 and part of 5 whereas Jason was responsible for sections 1, 2, and also part of 5.

1.2 Definitions, Acronyms and Abbreviations

There will be some use of acronyms and abbreviations used throughout the document in order to provide brevity and more concise reading. The following acronyms and abbreviations are:

- CNN - convolutional neural network
- NN - artificial neural network
- R-CNN - region-based convolutional neural network
- MOT - Multiple Object Tracking database
    - MOT16 - Multiple Object Tracking database from year 2016

2. Hardware Configuration

The only hardware requirements required to run our code are the same requirements to run MATLAB, as that is the environment that our code uses. The requirements from the MATLAB state:

- Processors: any Intel or AMD x86-64 processor
- Memory (RAM): 2GB, 4GB with Simulink
- Disk Space: 4-6GB

However, due to the implementation and usage of NN being computationally intensive, the following recommended requirements are:

- Processor: any 4 core Intel or AMD x86-64 processor
- Memory (RAM): 4GB
- Graphics: dedicated graphics card with at least 1GB GPU memory
- Disk Space: 7GB

3. System Parameters

Our code does not require for any parameters to be entered during runtime. However, the file and folder locations of the training images and test images are hardcoded into our code and scripts and must be set to proper values when running the code on machines that the code was not written on. Both ROICreate.m and RCNN.m require folder paths to be set properly for the code to execute. At the end of the execution of ROICreate.m, a table with the title “Traffic” will be created, and this table must be saved as a .m file to be used in the RCNN.m code.

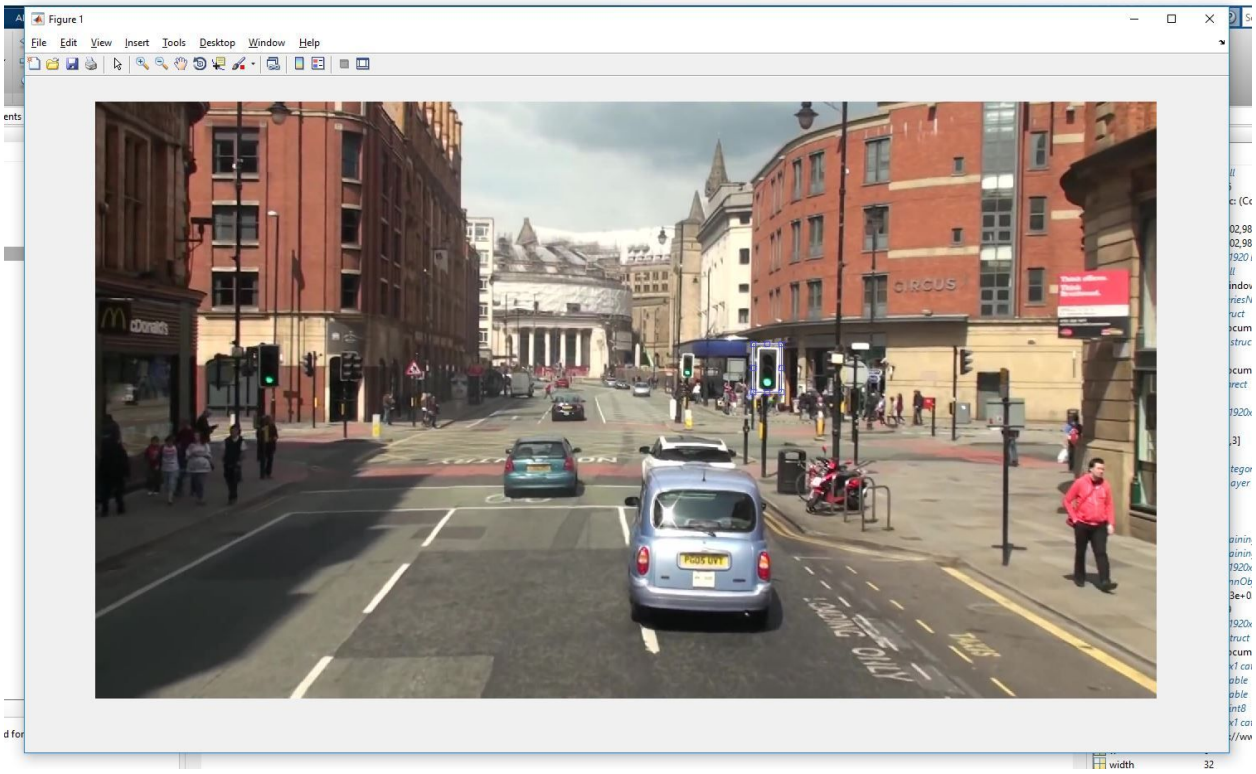
4. Operation Procedure

The proper steps to correctly run our code are listed in the following steps:

1. Open MATLAB and move the source files into the MATLAB folder (the folder on your machine where MATLAB files are saved to by default).
2. Load, but do not run, the ROICreate.m file.
3. If needed, edit the folder path of the folder that the desired training images are stored in. By default, the folder path for the training images used on the development machine is listed.
4. Run the ROICreate.m file in MATLAB and select the ROI around the target object for detection.
5. After the ROI is drawn, the size can be adjusted and moved. When finish, double-click inside the ROI box to proceed to the next training image.
6. Repeat the previous step until each training image is shown and an ROI is selected and the window closes.
7. After ROICreate.m is finished running, proceed to the MATLAB window and save the “Traffic” table created from the code and as a .mat file in the MATLAB folder.
8. Load the RNN.m file and, similar to the ROICreate.m file, if needed, set the folder path of the training and test images that will be used to test the performance of the R-CNN.
9. Run the code and it will automatically download the CIFAR-10 database and create a NN that will be trained with the database.
10. The code will output the accuracy of the pre-trained network and proceed to train the R-CNN using the training images and labels set from the ROICreate.m code.
11. The code will then begin to display the test images one at a time with a ROI drawn around any traffic light that it detects within the image.
12. Click on each image to proceed to the next image, and the this will repeat until all of the test images are shown and the program will then end.

5. Demonstration

The following images shows our code in action and displays some of the various steps outlined the previous (4) Operation Procedure section:



This image shows the ROICreate.m file being executed and is the ROI selection step of the source file. The blue rectangular outline around the traffic light towards the center of the image is the traffic light object that we had chosen to draw a ROI around.

Epoch	Iteration	Time Elapsed (seconds)	Mini-batch Loss	Mini-batch Accuracy	Base Learning Rate
1	50	2.94	2.3023	14.84%	0.001000
1	100	4.21	2.3021	16.41%	0.001000
1	150	5.49	2.3011	10.94%	0.001000
1	200	6.77	2.2966	20.31%	0.001000
1	250	8.05	2.2882	16.41%	0.001000
1	300	9.33	2.1252	26.56%	0.001000
1	350	10.61	2.0211	21.88%	0.001000
2	400	11.89	1.8798	27.34%	0.001000
2	450	13.16	1.8414	21.09%	0.001000
2	500	14.44	1.7822	29.69%	0.001000
2	550	15.72	1.7133	42.97%	0.001000
2	600	16.99	1.7968	34.38%	0.001000
2	650	18.27	1.7024	35.94%	0.001000
2	700	19.54	1.5878	42.19%	0.001000
2	750	20.82	1.5763	40.63%	0.001000
3	800	22.10	1.4098	44.53%	0.001000
3	850	23.38	1.4706	42.19%	0.001000
3	900	24.65	1.4371	48.44%	0.001000
3	950	25.93	1.5837	41.41%	0.001000
3	1000	27.21	1.2616	50.78%	0.001000
3	1050	28.50	1.5773	43.75%	0.001000
3	1100	29.78	1.4143	48.44%	0.001000
3	1150	31.06	1.4690	51.56%	0.001000
4	1200	32.33	1.3689	50.00%	0.001000
4	1250	33.62	1.2556	53.91%	0.001000
4	1300	34.90	1.2656	53.13%	0.001000
4	1350	36.18	1.3506	53.13%	0.001000
4	1400	37.46	1.1166	60.16%	0.001000
4	1450	38.74	1.1195	57.81%	0.001000
4	1500	40.02	1.1375	57.03%	0.001000
4	1550	41.31	1.0727	58.59%	0.001000
5	1600	42.59	1.1207	58.59%	0.001000
5	1650	43.87	1.1122	54.69%	0.001000
5	1700	45.16	1.1744	64.06%	0.001000
5	1750	46.45	0.9955	66.41%	0.001000
5	1800	47.74	1.0010	69.53%	0.001000
5	1850	49.04	1.1675	63.28%	0.001000
5	1900	50.33	0.9754	64.84%	0.001000

18	6700	175.40	0.4198	85.16%	0.000010
18	6750	176.72	0.4154	84.38%	0.000010
18	6800	178.02	0.5649	84.38%	0.000010
18	6850	179.33	0.5418	81.25%	0.000010
18	6900	180.65	0.5262	85.16%	0.000010
18	6950	181.96	0.5410	78.91%	0.000010
18	7000	183.28	0.4930	80.47%	0.000010
19	7050	184.59	0.5451	83.59%	0.000010
19	7100	185.89	0.4719	81.25%	0.000010
19	7150	187.18	0.6232	78.91%	0.000010
19	7200	188.49	0.6470	78.91%	0.000010
19	7250	189.80	0.3901	86.72%	0.000010
19	7300	191.14	0.4625	85.16%	0.000010
19	7350	192.46	0.4099	85.94%	0.000010
19	7400	193.76	0.3936	86.72%	0.000010
20	7450	195.06	0.5344	81.25%	0.000010
20	7500	196.36	0.4561	85.94%	0.000010
20	7550	197.65	0.5820	82.03%	0.000010
20	7600	198.95	0.3895	85.94%	0.000010
20	7650	200.25	0.5070	82.03%	0.000010
20	7700	201.55	0.5389	84.38%	0.000010
20	7750	202.85	0.4247	87.50%	0.000010
20	7800	204.17	0.4414	85.94%	0.000010
21	7850	205.47	0.5596	78.91%	0.000010
21	7900	206.78	0.6002	78.13%	0.000010
21	7950	208.07	0.4109	84.38%	0.000010
21	8000	209.38	0.5347	81.25%	0.000010
21	8050	210.68	0.4359	84.38%	0.000010
21	8100	211.98	0.3330	86.72%	0.000010
21	8150	213.27	0.5254	79.69%	0.000010
22	8200	214.58	0.3645	85.16%	0.000010
22	8250	215.89	0.4614	82.81%	0.000010
22	8300	217.19	0.4698	78.91%	0.000010
22	8350	218.51	0.4386	87.50%	0.000010
22	8400	219.82	0.4995	79.69%	0.000010
22	8450	221.13	0.3810	87.50%	0.000010
22	8500	222.44	0.3777	88.28%	0.000010
22	8550	223.75	0.4736	85.16%	0.000010
23	8600	225.08	0.4678	82.03%	0.000010
23	8650	226.38	0.4131	84.38%	0.000010
23	8700	227.68	0.4059	85.94%	0.000010
23	8750	228.98	0.5558	84.38%	0.000010

```
|           36 |        13800 |       360.58 |       0.3976 |       85.16% |     0.000000 |
|           36 |        13850 |       361.91 |       0.5309 |       82.03% |     0.000000 |
|           36 |        13900 |       363.23 |       0.4212 |       85.94% |     0.000000 |
|           36 |        13950 |       364.53 |       0.3278 |       87.50% |     0.000000 |
|           36 |        14000 |       365.83 |       0.5067 |       80.47% |     0.000000 |
|           37 |        14050 |       367.13 |       0.3564 |       87.50% |     0.000000 |
|           37 |        14100 |       368.43 |       0.4501 |       82.03% |     0.000000 |
|           37 |        14150 |       369.74 |       0.4600 |       78.13% |     0.000000 |
|           37 |        14200 |       371.07 |       0.4346 |       87.50% |     0.000000 |
|           37 |        14250 |       372.39 |       0.4974 |       79.69% |     0.000000 |
|           37 |        14300 |       373.68 |       0.3735 |       87.50% |     0.000000 |
|           37 |        14350 |       374.99 |       0.3688 |       87.50% |     0.000000 |
|           37 |        14400 |       376.29 |       0.4642 |       85.16% |     0.000000 |
|           38 |        14450 |       377.61 |       0.4670 |       82.81% |     0.000000 |
|           38 |        14500 |       378.92 |       0.4023 |       86.72% |     0.000000 |
|           38 |        14550 |       380.23 |       0.4060 |       85.16% |     0.000000 |
|           38 |        14600 |       381.57 |       0.5467 |       85.16% |     0.000000 |
|           38 |        14650 |       382.89 |       0.5263 |       81.25% |     0.000000 |
|           38 |        14700 |       384.20 |       0.5150 |       84.38% |     0.000000 |
|           38 |        14750 |       385.50 |       0.5267 |       80.47% |     0.000000 |
|           38 |        14800 |       386.81 |       0.4823 |       80.47% |     0.000000 |
|           39 |        14850 |       388.10 |       0.5323 |       85.16% |     0.000000 |
|           39 |        14900 |       389.40 |       0.4564 |       80.47% |     0.000000 |
|           39 |        14950 |       390.70 |       0.6140 |       79.69% |     0.000000 |
|           39 |        15000 |       392.00 |       0.6294 |       78.91% |     0.000000 |
|           39 |        15050 |       393.29 |       0.3781 |       88.28% |     0.000000 |
|           39 |        15100 |       394.59 |       0.4526 |       84.38% |     0.000000 |
|           39 |        15150 |       395.88 |       0.4022 |       87.50% |     0.000000 |
|           39 |        15200 |       397.18 |       0.3914 |       86.72% |     0.000000 |
|           40 |        15250 |       398.47 |       0.5185 |       81.25% |     0.000000 |
|           40 |        15300 |       399.79 |       0.4523 |       87.50% |     0.000000 |
|           40 |        15350 |       401.11 |       0.5650 |       80.47% |     0.000000 |
|           40 |        15400 |       402.42 |       0.3862 |       87.50% |     0.000000 |
|           40 |        15450 |       403.71 |       0.4959 |       80.47% |     0.000000 |
|           40 |        15500 |       405.04 |       0.5291 |       84.38% |     0.000000 |
|           40 |        15550 |       406.37 |       0.4136 |       86.72% |     0.000000 |
|           40 |        15600 |       407.67 |       0.4302 |       85.16% |     0.000000 |
|=========================================================================================|

accuracy =

    0.7601
```

The past 3 images show the training process of the CIFAR-10 dataset with our CNN that will serve to act as our pre-trained network. Here we can see that the accuracy of the performance of the CIFAR trained network is 76.01%.

```
Step 2 of 3: Training a neural network to classify objects in training data...

|=========================================================================================|
|     Epoch    |   Iteration  | Time Elapsed |  Mini-batch  |  Mini-batch  | Base Learning|
|              |              |   (seconds)  |     Loss     |   Accuracy   |     Rate     |
|=========================================================================================|
|           25 |           50 |        19.70 |       0.0053 |      100.00% |     0.001000 |
|           50 |          100 |        38.82 |       0.0003 |      100.00% |     0.001000 |
|           75 |          150 |        58.05 |       0.0109 |       98.98% |     0.001000 |
|          100 |          200 |        76.97 |       0.0006 |      100.00% |     0.001000 |
|=========================================================================================|

Network training complete.

Step 3 of 3: Training bounding box regression models for each object class...100.00%...done.

R-CNN training complete.
**********************************************************************
```

The image above shows the training portion on our R-CNN with our traffic light training dataset which uses the pre-trained network previously seen.

This image shows one of the test images that was processed by our R-CNN along with the ROI drawn around the network's detection of a traffic light object.

This image shows another successful traffic light detection and also demonstrates our model's invariance to scale when compared to the previous test image.

This image shows yet another traffic light detection and also demonstrates our model's invariance to lighting compared to the two previous test image examples.