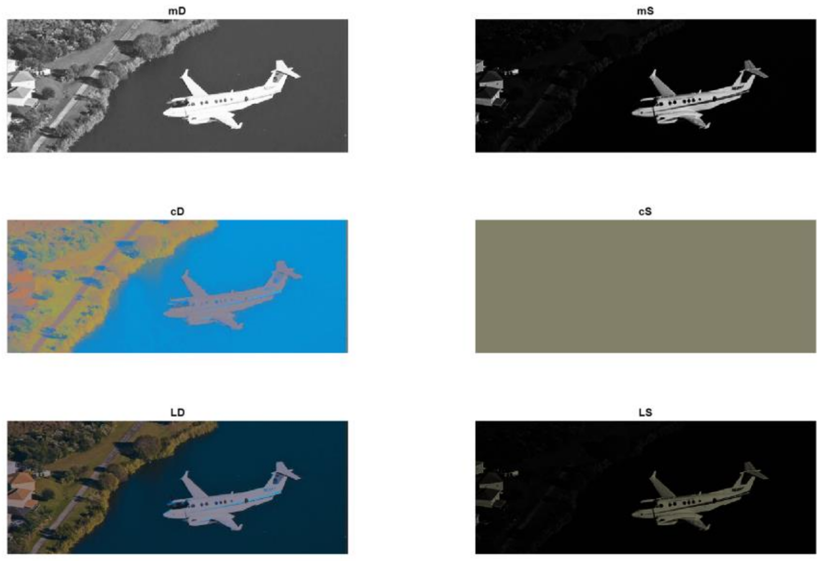
CIVIL 6415 Midterm Project

Elliot Lee

Submitted to Dr. Yilmaz



1. Program Environment

a) Windows

Using Matlab, execute "main_light.m". In order to run the full pipeline, Ubuntu 16.04 is required.

b) Ubuntu 16

Required Softwares/packages:

- git (assumed default)
- Matlab (assumed default)
- python2 (assumed default)
- python2 packages

```
$ chmod +x pythonReady.sh
$ sudo sh pythonReady.sh
```

- Tensorflow 1.3.1 or later (skip if already installed)

```
$ sudo apt-get install python-pip python-dev
$ sudo pip install tensorflow       (this assumes pip2)
```

- Keras (skip if already installed)

```
$ sudo pip install keras            (this assumes pip2)
```

or

```
$ sudo pip --no-cache-dir  install keras   (if memory error)
```

2. Running the Full-Pipe-Line Program on Ubuntu 16.

Using Matlab, execute "main_full.m"

If there is any hardware incompatibility or any circumstance that the program does not run, please let me know. I will bring a hardware to run.

3. Program Structure

A. Pipe Line

Since Matlab super pixel function is convenient, Matlab is applied to extract the relevant data regarding an image and its super pixel. The data is then saved as text files so that python application can use it for optimization using Keras with Tensorflow backend. The optimization result is then saved as text file so that Matlab can parse and plot the result images. This pipe line is shown in Figure 1.

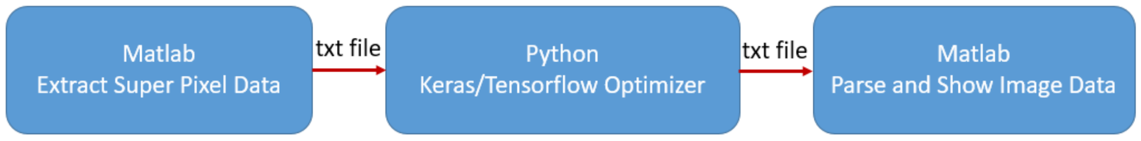


Figure 1: Program Pipe Line

As it can be understood, in this assignment, Matlab is used for data management.

B. Optimizer Structure

There are 4 different neural networks which outputs mD, cD, and mS based on the observed pixel values while cS output is based on initial random RGB value.

$$mD_{ij} = NN_1(R_{ij}, G_{ij}, B_{ij})$$
$$mC_i = NN_2(R_{ij}, G_{ij}, B_{ij})$$
$$mS_{ij} = NN_3(R_{ij}, G_{ij}, B_{ij})$$
$$mS = NN_3(R, G,)$$

where i is ith super pixel, j is jth pixel in the super pixel, NN1 and NN3 are fully connected layer with 2 hidden layers, NN2 is long short term memory with 100 hidden steps, and NN3 is fully connected layer with 1 hidden layer.

- Pre-Training

Especially, each neural network block is pre-trained for the fast convergence as shown in Figure 2.

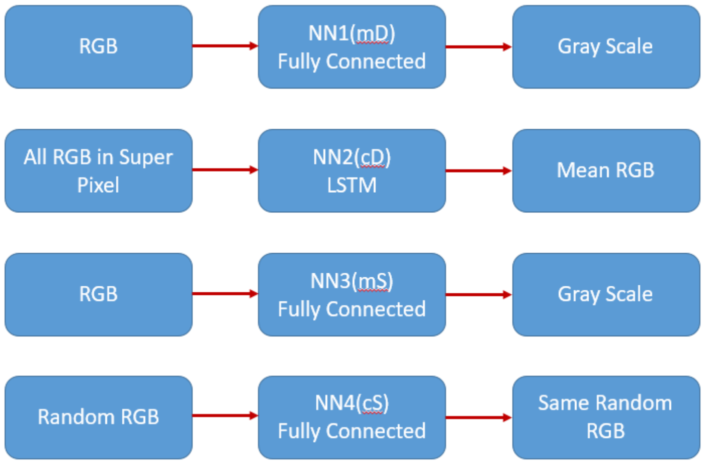


Figure 2: Pre-training Schedule for each neural network estimator.

- Optimization

Next, each output is concatenated into one matrix as shown in Figure 3.

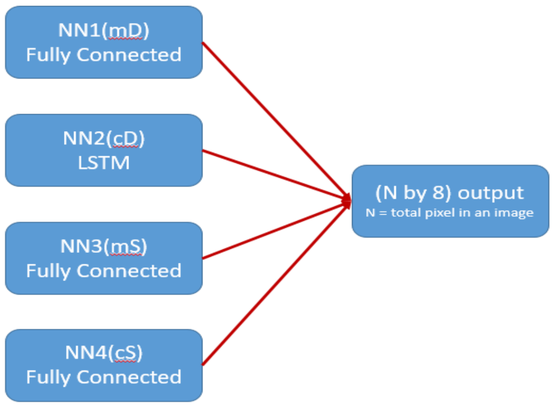


Figure 3: Concatenated Outputs

First column of the concatenated output is mD, 2nd, 3rd, 4th are cD, and mS, and cD correspondingly. Then, simply the loss is calculated using the equation provided from the instruction hadout; it is repeated as below.

$$r_{ij} = m_D(x_{ij})\mathrm{C_D}(\mathrm{X_k}) + \mathrm{m_S}(x_{ij})\mathrm{C_S} - \mathrm{C_L}(\mathrm{x_{ij}})$$

The optimizer used here is Adam, adaptive momentum estimation, which is built in the deep learning libraries for stochastic gradient descent.

4. Results

Image1

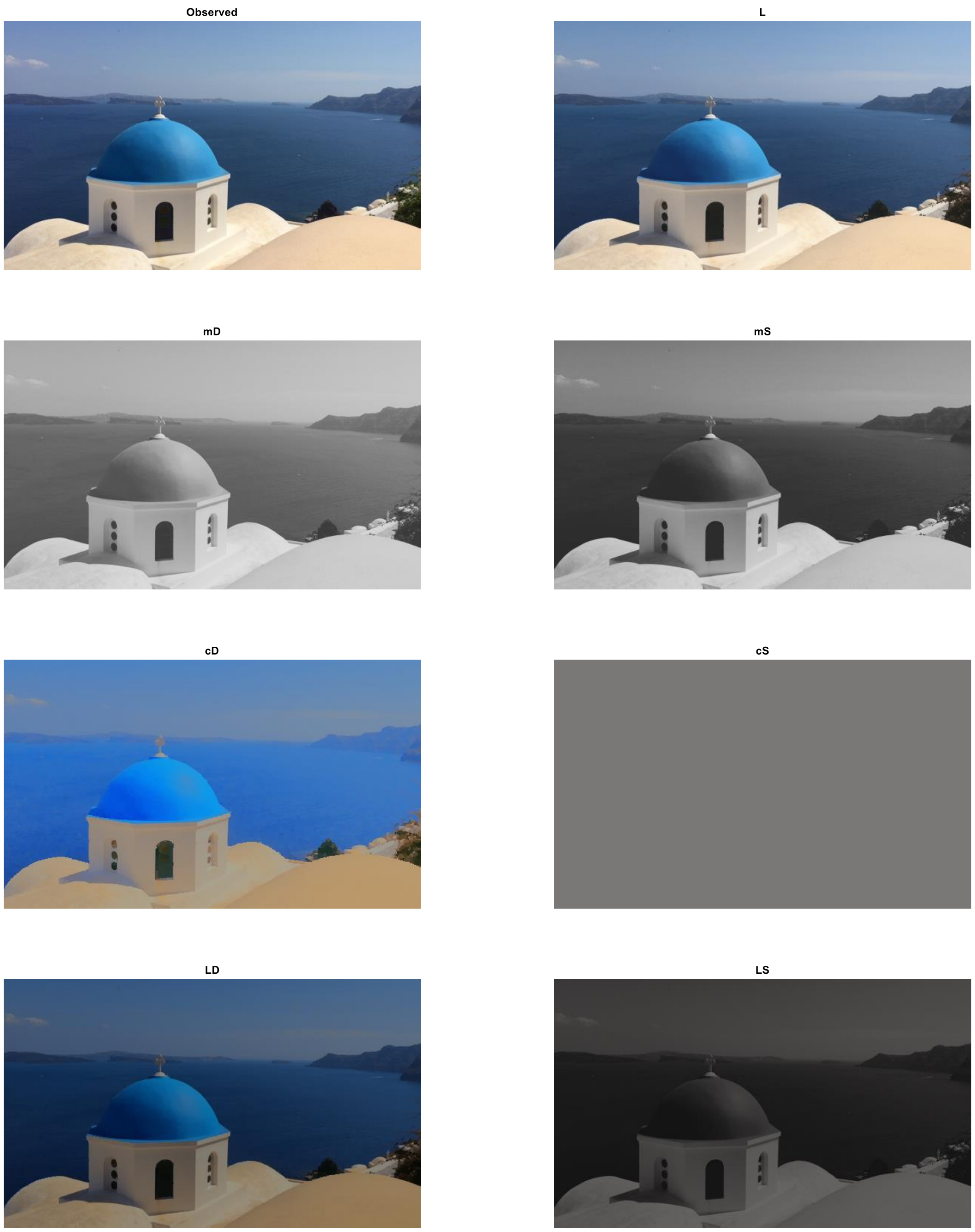


Image2

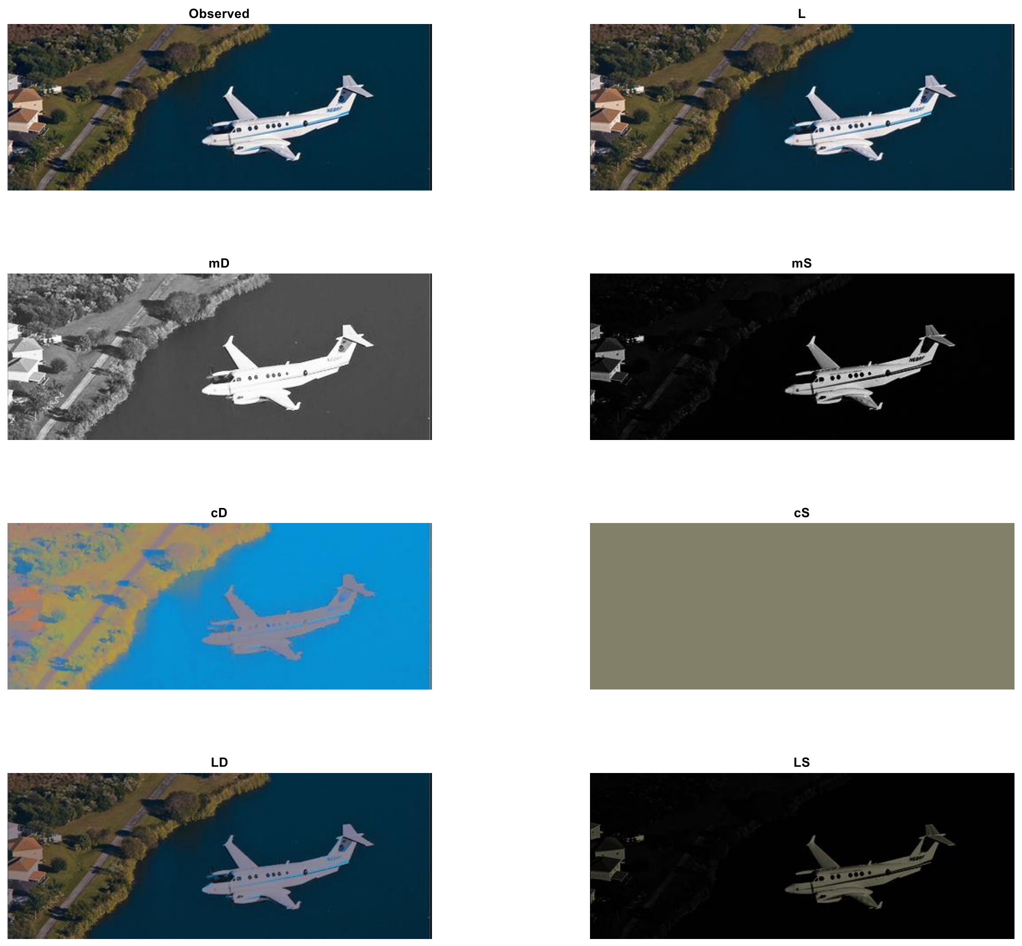
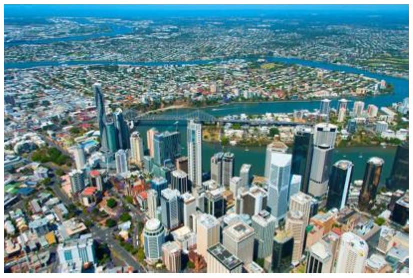
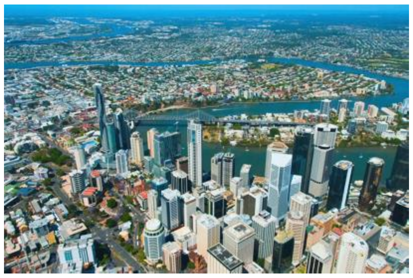


Image3

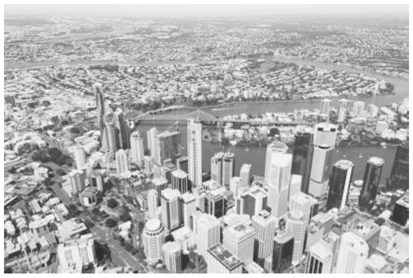
Observed



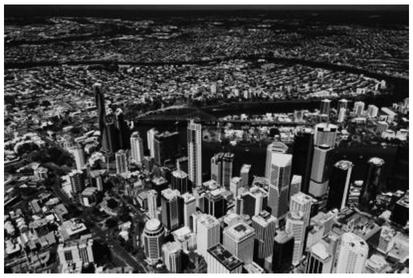
L



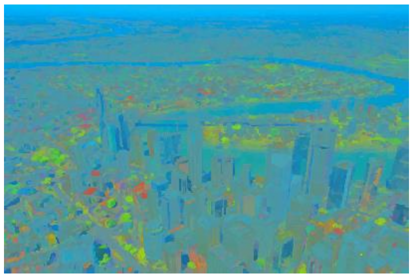
mD



mS

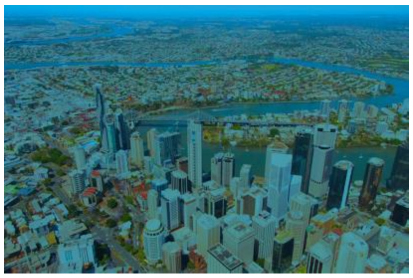


cD



cS

LD



LS