# Average Brightness

October 14, 2018

## 0.1 # Day and Night Image Classifier

The day/night image dataset consists of 200 RGB color images in two categories: day and night. There are equal numbers of each example: 100 day images and 100 night images.

We'd like to build a classifier that can accurately label these images as day or night, and that relies on finding distinguishing features between the two types of images!

*Note: All images come from the AMOS dataset (Archive of Many Outdoor Scenes).*

### 0.1.1 Import resources

Before you get started on the project code, import the libraries and resources that you'll need.

```
In [1]: import cv2 # computer vision library
        import helpers

        import numpy as np
        import matplotlib.pyplot as plt
        import matplotlib.image as mpimg

        %matplotlib inline
```

## 0.2 Training and Testing Data

The 200 day/night images are separated into training and testing datasets.

- 60% of these images are training images, for you to use as you create a classifier.
- 40% are test images, which will be used to test the accuracy of your classifier.

First, we set some variables to keep track of some where our images are stored:

```
image_dir_training: the directory where our training image data is stored
image_dir_test: the directory where our test image data is stored
```

```
In [2]: # Image data directories
        image_dir_training = "day_night_images/training/"
        image_dir_test = "day_night_images/test/"
```

### 0.3 Load the datasets

These first few lines of code will load the training day/night images and store all of them in a variable, `IMAGE_LIST`. This list contains the images and their associated label ("day" or "night").

For example, the first image-label pair in `IMAGE_LIST` can be accessed by index: `IMAGE_LIST[0][:]`.

```
In [3]: # Using the load_dataset function in helpers.py
        # Load training data
        IMAGE_LIST = helpers.load_dataset(image_dir_training)
```

### 0.4 Construct a `STANDARDIZED_LIST` of input images and output labels.

This function takes in a list of image-label pairs and outputs a **standardized** list of resized images and numerical labels.

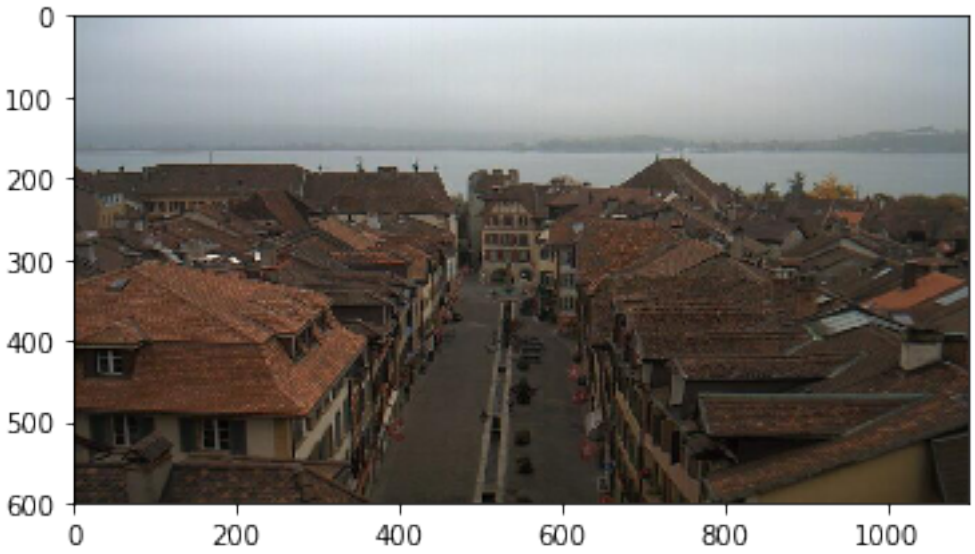
```
In [4]: # Standardize all training images
        STANDARDIZED_LIST = helpers.standardize(IMAGE_LIST)
```

### 0.5 Visualize the standardized data

Display a standardized image from STANDARDIZED_LIST.

```
In [5]: # Display a standardized image and its label

        # Select an image by index
        image_num = 0
        selected_image = STANDARDIZED_LIST[image_num][0]
        selected_label = STANDARDIZED_LIST[image_num][1]

        # Display image and data about it
        plt.imshow(selected_image)
        print("Shape: "+str(selected_image.shape))
        print("Label [1 = day, 0 = night]: " + str(selected_label))
```

```
Shape: (600, 1100, 3)
Label [1 = day, 0 = night]: 1
```




# 1 Feature Extraction

Create a feature that represents the brightness in an image. We'll be extracting the **average brightness** using HSV colorspace. Specifically, we'll use the V channel (a measure of brightness), add up the pixel values in the V channel, then divide that sum by the area of the image to get the average Value of the image.

## 1.1 RGB to HSV conversion

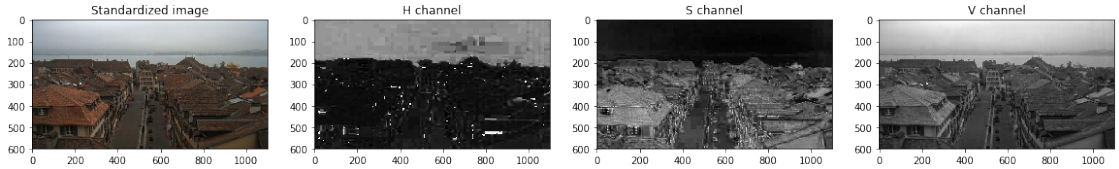
Below, a test image is converted from RGB to HSV colorspace and each component is displayed in an image.

```
In [6]: # Convert and image to HSV colorspace
        # Visualize the individual color channels

        image_num = 0
        test_im = STANDARDIZED_LIST[image_num][0]
        test_label = STANDARDIZED_LIST[image_num][1]

        # Convert to HSV
        hsv = cv2.cvtColor(test_im, cv2.COLOR_RGB2HSV)

        # Print image label
        print('Label: ' + str(test_label))

        # HSV channels
        h = hsv[:,:,0]
        s = hsv[:,:,1]
```

```
        v = hsv[:,:,2]

        # Plot the original image and the three channels
        f, (ax1, ax2, ax3, ax4) = plt.subplots(1, 4, figsize=(20,10))
        ax1.set_title('Standardized image')
        ax1.imshow(test_im)
        ax2.set_title('H channel')
        ax2.imshow(h, cmap='gray')
        ax3.set_title('S channel')
        ax3.imshow(s, cmap='gray')
        ax4.set_title('V channel')
        ax4.imshow(v, cmap='gray')
```

```
Label: 1
```

```
Out[6]: <matplotlib.image.AxesImage at 0x7fc65b1bac50>
```




---

### 1.1.1 Find the average brightness using the V channel

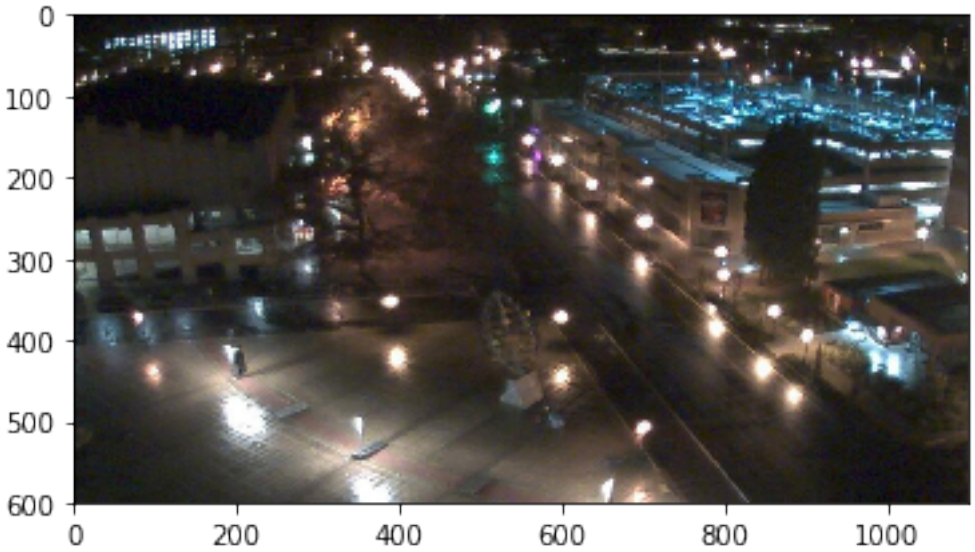
This function takes in a **standardized** RGB image and returns a feature (a single value) that represent the average level of brightness in the image. We'll use this value to classify the image as day or night.

```
In [8]: # Find the average Value or brightness of an image
        def avg_brightness(rgb_image):

            # Convert image to HSV
            hsv = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2HSV)

            # Add up all the pixel values in the V channel
            sum_brightness = np.sum(hsv[:,:,2])

            ## TODO: Calculate the average brightness using the area of the image
            # and the sum calculated above
            area = 600*1100.0

            avg = sum_brightness / area

            return avg
```

In [9]:
```
# Testing average brightness levels
# Look at a number of different day and night images and think about
# what average brightness value separates the two types of images

# As an example, a "night" image is loaded in and its avg brightness is displayed
image_num = 190
test_im = STANDARDIZED_LIST[image_num][0]

avg = avg_brightness(test_im)
print('Avg brightness: ' + str(avg))
plt.imshow(test_im)
```

```
Avg brightness: 71.7448015152
```

Out[9]: <matplotlib.image.AxesImage at 0x7fc65b086320>




In [ ]: