3.1.1 Data Augmentation

Figure 2: Result of flipping an image horizontally

To expand our training set, we used a traditional method of data augmentation. Originally the training set had 2974 images, and we doubled the training set to 5948 images by flipping the images horizontally. Flipping the images horizontally can help prevent a model from biasing towards one side of an image. Figure 2 shows an example of an image after being flipped for reference.

3.1.2 Adaptive Histogram Equalization

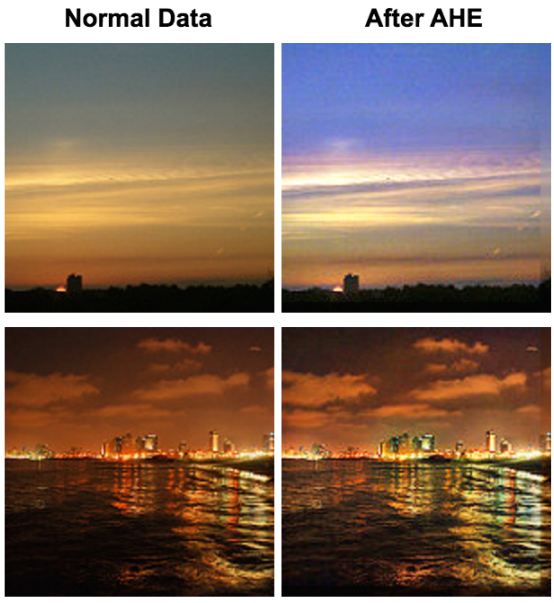

Figure 3: Result of applying Adaptive Histogram Equalization

The dataset contained similar images taken in different dynamic ranges. This problem can be solved by manipulating the histogram of the images. Manipulation of histogram will change the density function which results in a a better dynamic distribution [3]. We applied adaptive histogram equalization (AHE) to all the images in our original dataset to make a new dataset. The adaptive histogram equalization uses multiple histograms in different sections of an image to enhance the contrast of the image. Algorithm 1 describes adaptive histogram equalization and the result can be seen in Figure 3.

Algorithm 1 Adaptive Histogram Equalization

1: **procedure** ADAPTIVE HISTOGRAM EQUALIZATION
Read input image as img
2: Convert img to HSV color space
3: Run CLAHE algorithm on the V (Value) channel of img
4: Convert img back to RGB space
5: **return** img
6: **end procedure**

Contrast Limited Adaptive Histogram Equalization (CLAHE) mentioned in Algorithm 1 is an processing technique to improve contract in images. We used the sci-kit-image library to perform the adaptive histogram equalization for this project [4].

4. Approach

We used TensorFlow along with other python packages such as numpy, matplotlib, skimage, etc to code all the experiments in this project [4], [9], [10]. We applied the following models to our dataset to classify images into their correct categories:

1. Multiclass Support Vector Machine (SVM)
2. Multiclass SVM + HOG + HSV
3. Three Layer ConvNet
4. Five Layer ConvNet
5. AlexNet
6. VGGNet

4.1. Multiclass Support Vector Machine (SVM)

The multiclass Support Vector Machine (SVM) is the generalization of ordinary SVM for multiple classes. It uses the following loss function

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + \Delta)$$

where L_i is the loss for the i^{th} example, s_i is the score for the i^{th} class, y_i is the true label for the i^{th} example, and delta is the margin. Our multi class SVM takes in a numpy vector of flattened raw image data of length 150*150*3 = 67500 (150 being the width and height pixels, and 3 being the RGB channels) and outputs the raw score for each class.