### CSSE463 – Sunset detector

Joseph Jagiella, Weimu Song

[jagielj1@rose-hulman.edu](mailto:jagielj1@rose-hulman.edu), [songw@rose-hulman.edu](mailto:songw@rose-hulman.edu)

#### Abstract

Classifying an image is an easy task for a human, but is a very difficult problem for a machine. A machine must be trained to distinguish particular objects and patterns from a background. We have trained our program to detect whether or not the image is of a sunset. This is done by first extracting features, such as mean and variance of color, from the image that will be analyzed. Next, these results are classified as sunset or non-sunset. The system is trained to classify images based upon a set of images that are known to be either sunsets or non-sunsets. In order to improve accuracy without fitting the training set too closely, the system is then validated against a different training set. The accuracy of such a classifier is heavily dependent upon the size and quality of the training set. Our training set consists of only 800 sunset images and 800 non-sunset images, but its accuracy on the test data, a further 1000 images, is 88% using our first method, and 94% using our second method.

**1. Introduction**

Classifying objects in a scene is one of the most important problems in Image Recognition. An accurate classifier enables many useful possibilities, such as road sign recognition for self-driving cars. Detecting a sunset is more difficult than detecting an object, because a sunset affects other objects in the scene, rather than being its own object. Detecting only the sun, an object, would be easier than detecting the sun and its light affecting the rest of the image in a certain way.



**Figure (1.1).** A sunset.

For example, Figure 1.1 is considered to be a sunset despite the sun itself not being present in the image. The only way for the classifier to detect whether the image is of a sunset is to detect how the light of the sun is affecting the rest of the image.



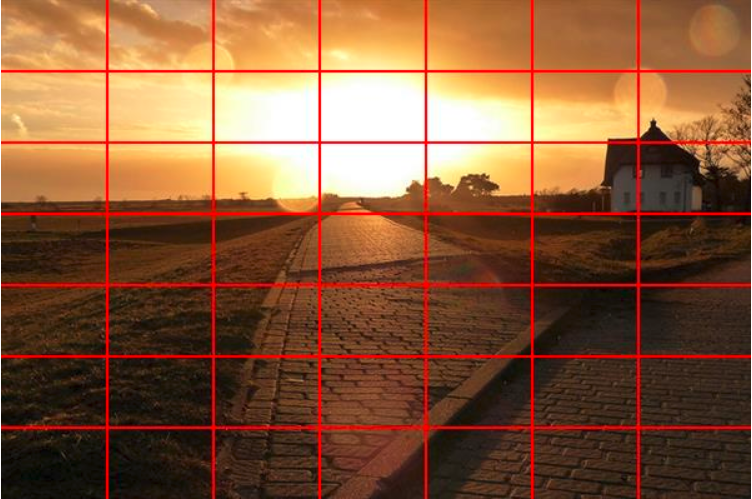
**Figure (1.2).** A sunset.

However, Figure 1.2 is also considered a sunset, even though the majority of the image is not affected the way Figure 1.1 is. There is a wide variety of ways a sunset can be presented in an image, so a simple way to classify them will not be accurate.

Our proposed solution is to train a classifier based upon a data set of sunset and non-sunset images, so that it will learn all of the aspects of a sunset image, not just the few that are simple to explicitly define. This is a good solution to many binary classifier applications, as what is being detected can be calibrated based upon the training data. This sunset detector could be reused to detect something else. Once the classifier is trained, it can simply be given an image, and it will determine whether that image is what it is trained to look for or not, and provide a number that represents how close to the classification threshold it was, which can be described as its confidence.

**2. process**

First, the image must be broken up into sections. This is better than analyzing the image as a whole, as it retains a certain amount of spatial information. For example, if a sunset is usually above the bottom of the image, such as in Figure 1 and Figure 2, it will be possible for the classifier to distinguish this if it knows what section of the image it is analyzing. We break an image into a 7x7 grid, starting from the top left corner, and ignore any leftover pixels on the right and bottom edges. This results in a maximum of 6 rows and columns of the image that are ignored, which has minimal effect on its classification [1].



**Figure (2.1).** A sunset broken into a 7x7 grid.

Once the image is broken into a 7x7 grid, we extract features from each block. We convert the RGB values of the image into LST space, where L = R + G + B, S = R – B, and T = R – 2\*G + B. LST was found to perform better than RGB for our purposes.

For each of the L, S, and T bands, we extract 2 features: the mean and the variance, from each of the blocks of the grid. Assuming the image is broken into a 7x7 grid, this results in a 49x2x3, or 294-dimension feature vector. Our process also automatically standardizes the feature data by centering by the weighted column mean and scaling by the standard deviation in a later step, because the ranges of the L, S, and T bands are inconsistent. L can be between 0 and 765, S can be between -255 and 255, and T can be between -510 and 510.

**3. Classification**

**3a. Support Vector Machines**

For each image in the training set, a 294-dimension feature vector and a value that simply stores whether or not the image is of a sunset is stored in a matrix and array, respectively. These values are used to generate support vectors using a support vector machine (SVM). The SVM automatically standardizes the data to values between 0 and 1. The SVM attempts to fit curves to a plot of the data so as to separate the data to opposite sides of a curve, representing the threshold. A data set with a simple plot, where all of the positives are grouped together and all of the negatives are grouped together, can be separated with a single line. A more complicated data set may need to be separated by hundreds of curves. Once the support vectors that fit the plot are found, these represent the trained system. Test images may now be plotted, and the existing support vectors will determine the classification of the image based upon where in the plot it appears.

**3b. Convolutional Neural Networks**

Alternatively, a Convolutional Neural Network (CNN) can be used for both feature extraction and classification. For example, AlexNet, one of the two CNN’s we used, takes in a 227x227 image, and has 25 layers. We used the first 20 layers for feature extraction, and replaced the last 5 layers with an SVM for classification. We also used the first 22 layers for feature extraction and replaced the last 3 layers with new layers, trained to classify sunsets. We also used GoogleNet, which has 144 layers, and replaced the last 3 layers with new layers, trained to classify sunsets. The 3 new layers were, in both cases, a fully connected layer, a softmax layer, and a classification layer, in that order.

**4. Experimental Setup**

We used a training data set to train the system, a validation set to calibrate the classification to a higher accuracy, and a test set to show results. The training set consists of 800 unique sunset images and 800 unique non-sunset images. The test set consists of 500 unique sunset images and 500 unique non-sunset images. The validation set consists of 300 unique sunset images and 300 unique non-sunset images. The images are, on average, 600x400 pixels, with deviation of up to 200 in each dimension. The system outputs the classification of the image, and a number that represents its score. The closer the 0 the score is, the closer to the classification threshold, and therefore, the less confident the classifier is that it was correct. From the test set of 1000 images, rates of correct and incorrect classifications can be calculated.

To obtain AlexNet and GoogLeNet, we used their respective MATLAB packages.

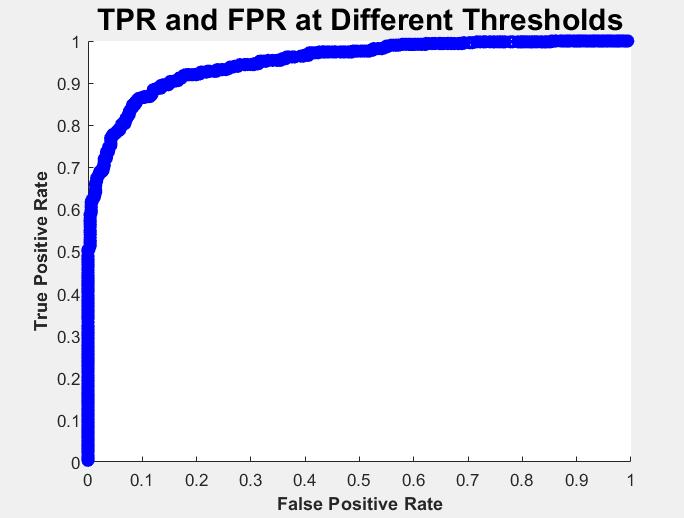
**5. Results**

**5a. SVM Results**

The SVM uses a number of hyperparameters to tune its performance. We found the best values for these hyperparameters by testing the SVM against the validation set, and choosing the values that resulted in the highest accuracy, but lowest number of support vectors at that accuracy. Our SVM performed best with a Box Constraint value of 100, a Kernel Scale of 30, and the RBF Kernel Function. Automatically normalizing the data is also a hyperparameter, which we set to true. Below is the data we used to determine the best Kernel Scale and Box Constraint values.

|  |  |  |  |
| --- | --- | --- | --- |
| **KernelScale** | **BoxConstraint** | **SupportVectors** | **Accuracy** |
| 14.38 | 100 | 848 | 0.93 |
| 21 | 100 | 620 | 0.93 |
| 21 | 1000000 | 620 | 0.93 |
| 21 | 1 | 673 | 0.93 |
| 22 | 100 | 600 | 0.93 |
| 20 | 100 | 641 | 0.93 |
| 23 | 100 | 588 | 0.92 |
| 24 | 100 | 575 | 0.92 |
| 1000000 | 100 | 1600 | 0.55 |
| 40 | 100 | 505 | 0.91 |
| 35 | 100 | 514 | 0.92 |
| 34 | 100 | 516 | 0.92 |
| 33 | 100 | 515 | 0.92 |
| 32 | 100 | 514 | 0.92 |
| 31 | 100 | 516 | 0.92 |
| **30** | **100** | **517** | **0.93** |
| 29 | 100 | 523 | 0.93 |
| 30.5 | 100 | 516 | 0.92 |
| 50 | 100 | 502 | 0.91 |

The SVM was then run on the test image set. With a default threshold value of 0, the accuracy was 88%. We also tried 1000 different threshold values on the test image set, values from -5 to 5 in iterations of 0.01, in order to see if a non-zero threshold would have better accuracy. Figure 5.1 shows the results.



**Figure (5.1).** SVM results at 1000 different threshold values.

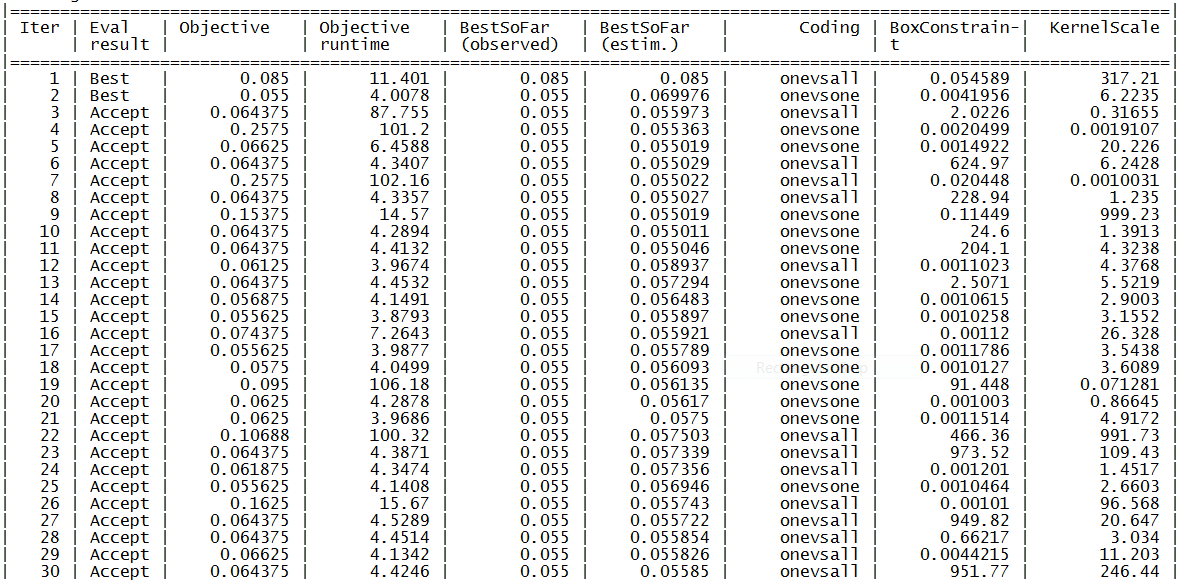
This graph does not show the threshold values for each point, because only one point matters: The point with the highest True Positive Rate while maintaining the lowest False Positive Rate, or more simply, the point with the shortest Euclidean distance to (0,1). This threshold value was found to be 0.16, which resulted in an accuracy of 89%.

**5b. Convolutional Neural Network Results**

We used AlexNet to perform both Feature Extraction and Transfer Learning. We also used GoogLeNet to perform Transfer Learning. These results were markedly better than the SVM results.

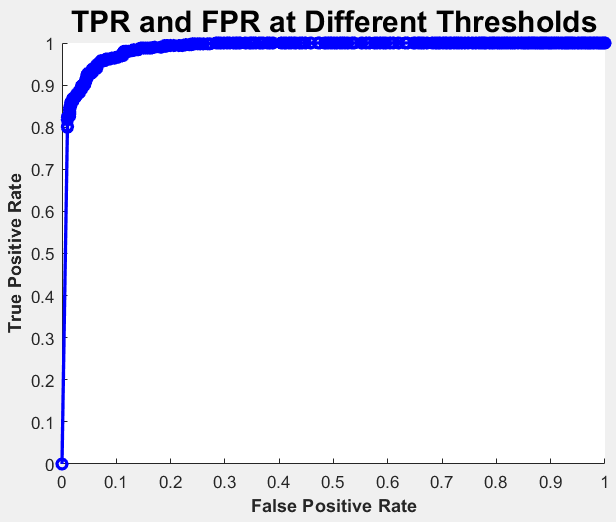
**AlexNet Feature Extraction**

We used the pre-trained AlexNet neural network to extract features from the training set, and used those features to train an SVM. We used the fitcecoc function’s options in MATLAB to automatically find better hyperparameters. This is what it reported:



The best hyperparameters it found were a BoxConstraint of 0.0041956 and a KernelScale of 6.2235. Unfortunately, fitcecoc has no way to report the number of support vectors in the SVM.

The SVM was then run on the test image set. Without the above hyperparameter tuning, the system achieved an accuracy of 93.8%. With hyperparameter tuning, it achieved 93.9% accuracy. These results were found using a threshold value of 0, the default. We also tried 1000 different thresholds from -5 to 5 in iterations of 0.01, in order to see if a non-zero threshold would have better accuracy. Figure 5.2 shows these results.

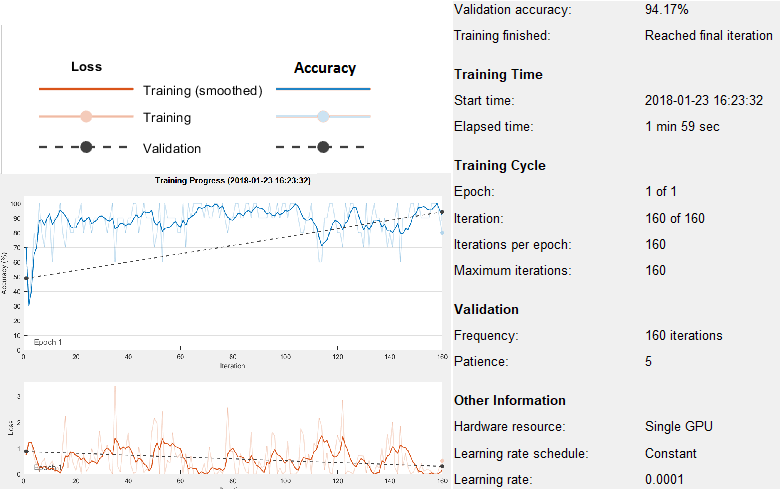


**Figure (5.2).** SVM results at 1000 different threshold values.

The threshold value that resulted in the optimum ratio of true positive rate to false positive rate was -0.62, which resulted in an accuracy of 94.2%.

**AlexNet Transfer Learning**

We used the pre-trained AlexNet neural network, but replaced the last 3 layers with new layers, and trained the network to classify sunsets. The result of training AlexNet can be seen in Figure 5.3.

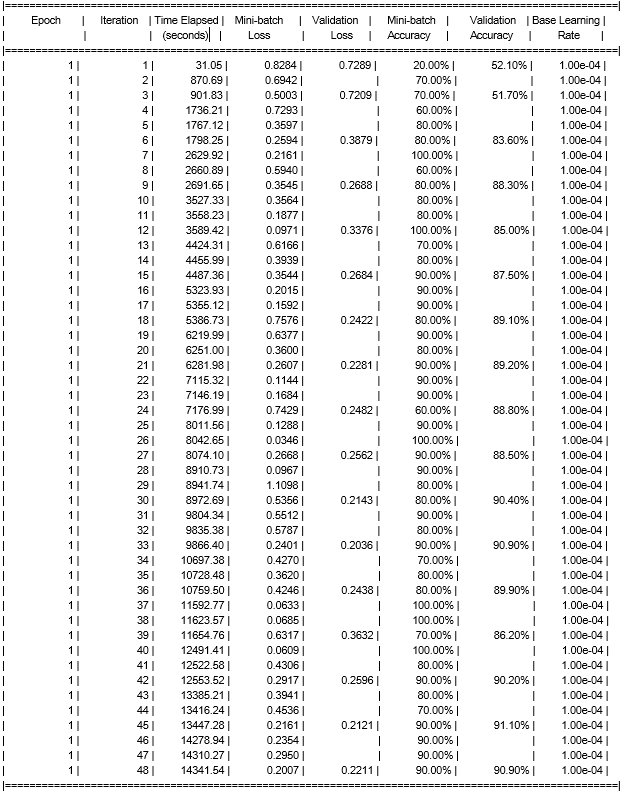


**Figure (5.3).** AlexNet Transfer Learning Results

After training, it must classify the test images. With this procedure, AlexNet classified the test images with 92.8% accuracy.

**GoogLeNet Transfer Learning**

We also used the pre-trained GoogLeNet neural network, which has many more layers than AlexNet, to perform classification. We replaced the last 3 layers with new layers and retrained the network using our training images. These are the results it reported:



Training GoogLeNet took substantially longer than training AlexNet (14341.54 seconds vs 119 seconds), mostly because it has 144 layers compared to AlexNet’s 25. Even with the added complexity, GoogLeNet only achieved 90.9% accuracy on its validation, compared to AlexNet’s 94.17%. After training, we used it to classify the test images. GoogLeNet also classified the test image set with 90.9% accuracy. It is possible that GoogLeNet would perform better if it were to be trained with more epochs, but we did not have time to train a neural network for several weeks, and had no hardware with which to speed up training.

**6. Discussion**

**6a. SVM Discussion**

False Negatives:



1. (b)

The above 2 images were classified as non-sunsets when they are actually sunsets. Image (a) had a score of -0.55, which is very close to being classified as a sunset. It is likely that this classification was wrong because of the unusually purple sky. Image (b) had a score of -1.98, which is fairly distant to the threshold. This is likely because the sun itself is obstructed by branches, and the image has an overall green color.

True Positives:



(c) (d)

The above 2 images were correctly classified as sunsets. Image (c) had a score of 5.24, which is very distant to the threshold, meaning the SVM is very confident that it is a sunset. This is likely because the colors of the image are very consistent with that of a sunset, and possibly also because the sunset mostly occurs in the top half of the image. Image (d) had a score of 0.46, which is very close to the threshold. This is likely because most of the sky is still blue, but part of the image is affected by sunlight without obstruction.

True Negatives:



(e) (f)

The above 2 images were correctly classified as non-sunsets. Image (e) had a score of -5.81, which is very distant to the threshold, so the SVM was very confident that this image is not a sunset. This is likely because the image has none of the colors associated with a sunset, does not possess the gradient pattern a sunset would cast, and has no sun object pictured. Image (f) had a score of -0.45, which is very close to the threshold. This is likely because the buildings are lit up in a way that looks like a sunset is illuminating them, but there is enough clearly black sky to prevent it from being classified as a sunset.

False Positives:



(g) (h)

The above 2 images were classified as sunsets, when they are in fact not sunsets. Image (g) had a score of 0.04, which is very close to the threshold. This is likely because the colors of the image are close to that of a sunset, follow a short gradient, and dominate a large portion of the image. Image (h) had a score of 1.62, which is fairly distant to the threshold. The SVM was fairly confident that Image (h) is a sunset, likely because the image has many of the same colors as the sky and a sunset, and because the image has a smooth gradient between the beach and the water colors.

**6b. Convolutional Neural Network Discussion**

Of the 3 approaches we tried, there were tradeoffs to be made with regards to speed and accuracy. These are the runtimes for each of the approaches:

|  |  |  |
| --- | --- | --- |
| **Net** | **Operation** | **Runtime (s)** |
| AlexNet | Feature Extraction (Training) | 555.83188 (NN)  + 691.5088 (SVM)  = 1247.34068 |
| AlexNet | Feature Extraction (Classification) | 354.9861 (NN)  + 0.033820 (SVM)  = 355.01992 |
| AlexNet | Transfer Learning (Training) | 119 |
| AlexNet | Transfer Learning (Classification) | 22.183746 |
| GoogLeNet | Transfer Learning (Training) | 14341.54 |
| GoogLeNet | Transfer Learning (Classification) | 906.092390 |

Of these options, using AlexNet for Transfer Learning was definitely the fastest. However, it resulted in an accuracy of 92.8%. Using AlexNet for Feature Extraction and an SVM for classification took longer, but resulted in an accuracy of 94.2%. GoogLeNet took the longest by far, and had the lowest accuracy of 90.9%. This is most likely because of the limited amount of time and resources used to train the neural networks. However accurate the neural net classifiers may turn out to be with more training, 94.2% accuracy with only 1600 training images is excellent, and much higher than the SVM-only accuracy of 89%.

Below are some of the classifications made by the AlexNet-SVM combination, as it was the most accurate.

False Negatives:



(a) (b)

The above 2 images were incorrectly classified as non-sunsets, when they are in fact sunsets. Image (a) had a distance value of -1.1789, which is fairly distant to the threshold, so the SVM was fairly confident that the image was not a sunset. This is likely because the sunset in the image is more purple than red, and the image is taken at an extreme upwards angle. Image (b) had a distance value of -0.5643, so the SVM was not very confident that the image was not a sunset. The SVM likely thought the image was not a sunset because the trees obscure much of the image.

True Positives:



(c) (d)

The above 2 images were correctly classified as sunsets. Image (c) had a confidence value of 2.5355, so the SVM was very confident that it is a sunset. This is likely because the image is almost perfectly split, and the upper half has no obstructions or discolorations to obscure the sunset. Image (d) had a confidence value of 0.8018, so the SVM was not very confident that the image is a sunset. This is likely because the image is rather dark and the sunset is obscured by the clouds.

False Positives:



(e) (f)

The above 2 images were incorrectly classified as sunsets, when they are in fact not sunsets. Image (e) had a confidence value of 1.7175, so the SVM was very confident that the image is a sunset. This is likely because the image is very sunny, and the time of day appears to be late in the afternoon. Image (f) had a confidence value of 0.6142, so the SVM was not very confident that the image is of a sunset. It likely thought the image is of a sunset because there is a bright light in the sky, similar to the sun.

True Negatives:



(g) (h)

The above 2 images were correctly classified as non-sunsets. Image (g) had a confidence value of -2.7595, so the SVM was very confident that the image is not of a sunset. This is likely because the image is very bright. Image (h) had a confidence value of -0.5882, so the SVM was not very confident that the image is not of a sunset. This is likely because the image is mostly yellow.

**7. Conclusion**

The classifier turned out to be fairly accurate, with an accuracy of 88%, or 89% with a different threshold, using an SVM. However, this is likely not accurate enough to be truly useful in a practical application. If we had more time to work on the classifier, there are a few things we could do to improve the accuracy.

First and foremost, a larger training set would improve the accuracy of the classifier, because the training is the most important step towards calibrating it, and having more diverse images in its data set allows it to prepare for outliers. 1000 images is a fairly small data set in comparison to the several million that some other learning algorithms use. However, this would take a great deal of time, not only to collect the images, but to train the system with them.

Furthermore, a few techniques were implemented in the original sunset classifier that we did not implement. For example, cropping the image in order to maximize the ratio of sunset area to image area would prevent irrelevant background from skewing results. Repositioning major portions of the image in an effort to keep sunset features in a consistent location would also improve accuracy [1].

Another technique we might implement is to detect if an image is erroneously tinted a different color, such as if the camera lens is green, and adjust the image to a normal hue. This would decrease false negatives, but may increase false positives as well. More research into this matter may improve the results.

With the use of AlexNet to perform Feature Extraction, and the use of an SVM to classify the images, using a certain threshold, we experienced an accuracy of 94.2%. This is much higher than using only an SVM, and may be high enough to be of use for a practical application. Theoretically, using a neural network to also classify images instead of using an SVM should be more accurate, but we did not have time to train the neural networks properly. If we had more time, it is possible we could achieve higher accuracy. We could also spend more time tuning the parameters of the neural networks to improve accuracy.

**8. References**

[1] Matthew Boutell, Jiebo Luo, and Robert T. Gray. Sunset scene classification using simulated image recomposition. *IEEE International Conference on Multimedia and Expo*, Baltimore, MD, July 2003