IMAGE RECOGNITION SUNSET DETECTOR FINAL REPORT

Shunfan Du, Songyu Wang

{dus, wangs4}@rose-hulman.edu

ABSTRACT

The report describes image recognition models we developed to identify if a given image contains a sunset scene. First, it introduces how we use LST color model to extract each image's features by cropping the image into 49 equally spaced sections and calculating each section's mean and standard deviation. Second, the report explains how we choose hyper-parameters and the threshold to optimize our Support Vector Machine (SVM) classification using the training set and validation set. Third, the report concludes the optimized results that we got from both predicting the validation set and the test set. The validation set's accuracy is 93.5%, and the test set's accuracy is 90.4%. Last, the report brings the rationales behind some misclassified images and the future work that we plan to do. Also, we realized that SVM has its limitation as we are not sure about what would be the best features to extract. Therefore, we introduced Convolutional Neural Network(CNN) to help us extract feature using AlexNet. Furthermore, inspired by the feature extraction CNN, we build a transfer learning CNN using AlexNet and GoogLeNet. The feature extraction CNN's accuracy is 96.33% on the validation set and 93.6% on the test set. The transfer learning CNN using AlexNet has an accuracy of 93.83% on the validation set and 93.8% on the test set, while the transfer learning CNN using GoogLeNet has an accuracy of 92.17% on the validation set and 94% on the test set.

1. INTRODUCTION

In the sunset detector project, we attempt to find an SVM and CNNs to allow computers to auto-detect if there is a sunset scene in given images. A group of images from Flickr is used to train the SVM [1]. Those images are labeled in two categories, sunset, and non-sunset. After training, we can feed an image to the SVM or CNN, and it should be able to tell the user if the given image is a sunset scene or not with relatively high accuracy. Scene classification can be very useful in different areas. If the subset SVM or CNN is a success, we can apply the same principle to more images in order to detect objects in different categories. For example, we can use a similar SVM or CNN to detect if an image contains human beings for security purpose.

Unlike detecting oranges, which is straightforward to do as oranges are always round and orange, detecting sunset is more challenging because there exists so many variables and features to consider: First, each sunset scene has different colors and brightness, some sunset scenes are yellow or even white while most of them are orange. Second, Each sunset scene might be along with or covered by different things, such as the sun, clouds, trees, mountains, buildings, and sea (Fig. 1).

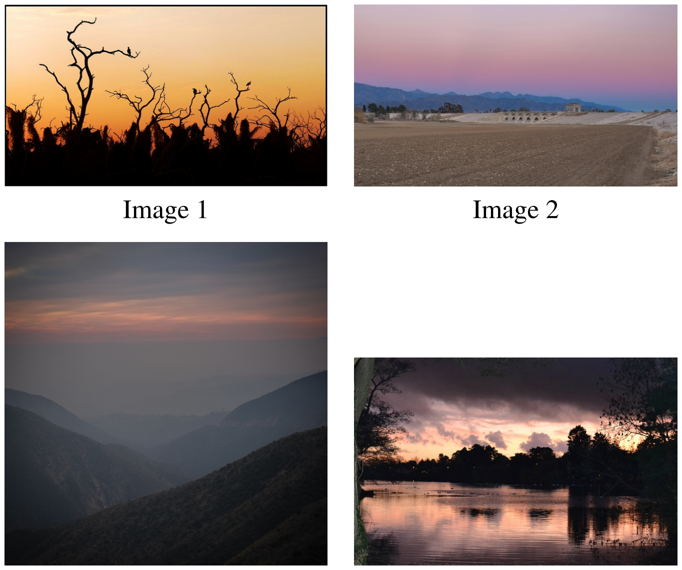


Fig. 1. Challenging Images

Our proposed solution has the following interesting designs: We label images into different categories. Using SVM, the algorithm extracts some features relate to partial image color and uses them to learn what the sunset-like features are. The result classifiers are the baseline SVM. Then, we adjust it with a different set of images to optimize the accuracy. The optimized SVM will be the result. All the random images we put into the optimized SVM can be recognized with high accuracy. For feature extracting CNN, we use pre-trained CNN layers to extract feature instead of using the feature we subjectively decide. After retrieving the features, we use them to train a SVM as mentioned above. For tranfer learning CNN, we use a pre-trained CNN and replace the last layers to detect sunset sence. In a more general context of image recognition, we can use this type of SVM or CNN to classify images for future analysis.

2. FEATURE EXTRACTION

After receiving an image, our program first converts the RGB color model to LST color model using the Fig. 2 formula for every pixel. The theoretical range for each band is also listed in Fig. 2.

$$L = R + G + B$$
$$0 < L < 765$$
$$S = R - B$$
$$-255 < S < 255$$
$$T = R - 2G + B$$
$$-510 < T < 510$$

Fig. 2. RGB to LST Formula

After getting the LST color model, we crop the image into 49 sections using a 7×7 grid (Fig. 3). If the size of the image is not divisible by 7, we will ignore the extra pixels. According to Boutell, cropping the edge of the image could make the image a better match for sunset scenes [2].

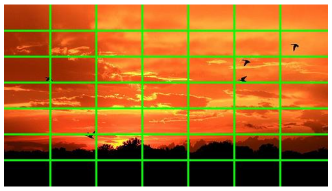


Fig. 3. Grid Example

For every pixel from each color band and each section in an image, we calculate the mean value and standard deviation. With the mean value, we can know the normalized color schemes of each section. With standard deviation value, we can know how various those pixels' color are and the changes in the shades of colors in each section. From Fig. 2, we know each band has its own range, so their mean value and standard deviation value also have their own range. For SVM to work, we need to weight them equally, so we decide to standardize them into a (0,1) range. Since there are three bands, the result should contain 6 values for each section (L mean, L stand deviation, S mean, S stand deviation, T mean and T stand deviation). And there is 49 section in an image, 294 feature values will be returned for every image in the image set.

In order to train SVM, we need two matrices. Our training image set contains 1600 images. Putting all the extracted features into one matrix, we have a 1600×294 feature matrix. The second matrix is a label matrix. All the image labels are imported to this 1600×1 matrix to tell SVM if the given image from the image set is a sunset. Those two matrices are the final input we use to train the SVM.

3. CLASSIFICATION

3.1. SVM Classification

SVM will create a model based on the inputs matrices. After learning the labels, SVM will find some features from feature matrix as supports vectors and draw a boundary based on those chosen support vectors. If the vectors are linear separable, we can attempt to find the margin between vectors in two category and maximize it by adjusting support vectors. If the vectors are not linear separable, based on Cover's Theorem, we know that those vectors are possibly separable in higher dimension.Since our dataset are random images, so they are unlikely to be linear separable, we use kernel function, Gaussian Radial-basis function (RBF), to model the SVM. The boundary,either in two dimension or higher dimension, will be the identifier to separate two categories. If a new input is given, SVM will find out which side the input will land based on the boundary and output corresponding result.

Once we have a baseline SVM, we need to experiment different hyper-parameters including kernel scale and box constraint. The hyper-parameters that have the most accurate result and use the fewer support vectors would be the best fit. Moreover, we could try different thresholds to find out the best threshold that results in highest accuracy. The details are described in Section 5.1.

4. EXPERIMENTAL SETUP

4.1. SVM Setup

At first, we have a training data set that contains 800 sunset images and 800 non-sunset images. The approximate range of resolution is $(300 \sim 700) \times (300 \sim 700)$. We use this data set to train the baseline SVM mentioned at the beginning of Section 3.1. Once we get the baseline SVM, we use a validate data set to adjust SVM parameters. The validate data set contains 300 sunset image and 300 non-sunset images. The approximate range of resolution is $(300 \sim 700) \times (300 \sim 700)$. After all the hyper-parameters being decided, we use a test image set to find the threshold that can provide the best accuracy. The final result SVM can be generated with the parameters and threshold at this point, and we can know how well our SVM does in identifying sunset scenes. The test image set contains 500 sunset images and 500 non-sunset images. The approximate range of resolution is also $(300 \sim 700) \times (300 \sim 700)$.

4.2. CNN Setup

The setup starts with installing the pretrained networks called AlexNet and GoogLeNet. Since the input image of AlexNet has to be 227×227 and the input image of GoogLeNet has to be 224×224, we used our resizeImageSet function to resize the original image set into two image sets so that they can be used for both AlexNet and GoogLeNet.

5. RESULTS

5.1. SVM Result

To find out the hyper-parameters that could result in an optimal classification, we tried 800 combinations of kernel scales and box constraints on the validation set. After initial observation, we found that the accuracy starts to decrease when the kernel scale is greater than 14. Therefore, the range of the kernel scale that we decide to try is from 1 to 16, while the range of box constraint is from 1 to 50. The index of one attempt that uses kernel scale k and box constraint b would be $50 \cdot (k-1) + b$ in Fig. 4. To pick the hyper-parameters that make classification efficient, we also check the number of support vectors used in each attempt. According to the graph, the attempt index that has the highest accuracy and lowest number of support vectors is 653, which uses a kernel scale of 14 and a box constraint of 3 (Fig. 4). The attempt has an accuracy of 93.5% on validation set, and it uses 862 images (53.9% of all training image set) as support vectors.

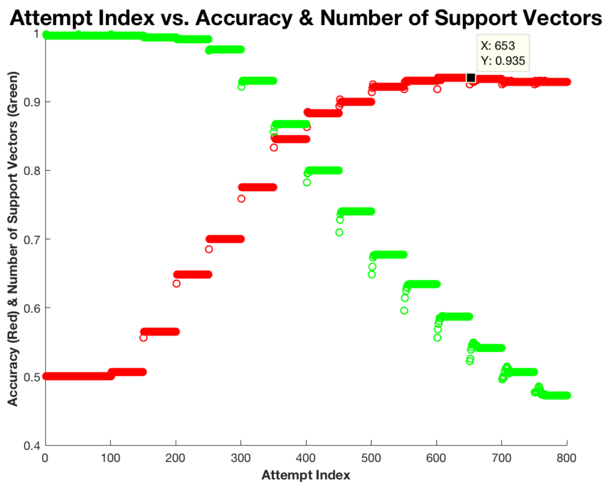


Fig. 4. Attempt Index vs. Accuracy (Red) & Number of Support Vectors (Green)

By default, SVM will use 0 as the threshold in order to identify categories. If the SVM result score is larger than 0, one label will be assigned. If the scores in less than 0, the opposite label will be assigned. Sometimes, use 0 as the threshold may not lead to the optimized result. We need to identify the best threshold to get the optimized accuracy. After examining all the result scores, we know that most of the scores are in the range of -1 to 1. In order to test thresholds as much as possible, we choose to plot all the true positive rate and false positive rate with threshold -5 to 5 to construct ROC curve (Fig. 5).The best threshold should be the point that is closest to (0,1) on the curve. Based on the curve, we found the best threshold is -0.1 in our case.

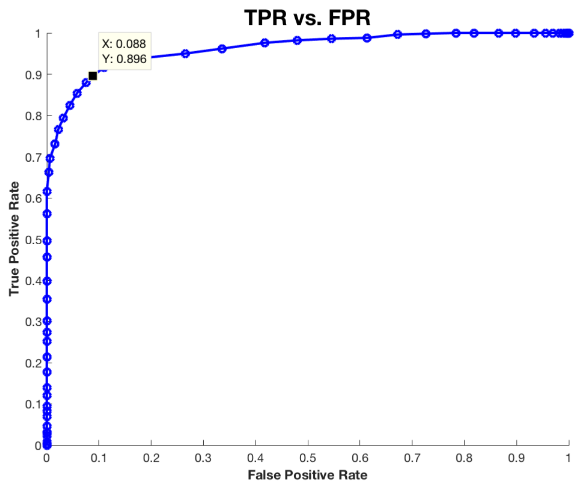


Fig. 5. ROC curve

The final SVM will be generated with hyper-parameters and threshold we selected (Table 1). Moreover, our final SVM has an accuracy of 90.4% on the test set.

Table 1. Final Parameters

Kernel scale used	14
Box constraint used	3
Number of support vectors used	862
Accuracy on the validation set	93.5%
Threshold used	-0.1
Accuracy on the test set	90.4%

5.2. CNN Result

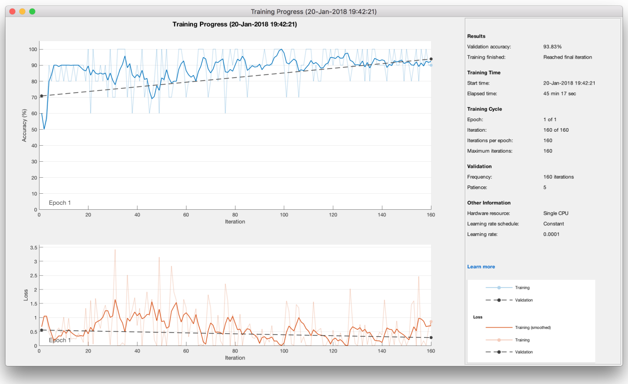
All the result are generated under Mid 2015 MacBook Pro with a single CPU (2.2 GHz Intel Core i7).

5.2.1. Transfer Learning Using AlexNet

Unfortunately, AlexNet does not support identifying sunset feature, so we need to modify the last three layers before using it. To transfer layers from AlexNet to our new network, we extracted all the layers excluding the last three from AlexNet into our network. Afterwards, we appended a fully connected layer, a softmax layer, and a classification output layer into our network so that it is able to classify if the image is sunset scene or not. Since our laptop does not have too much computing power, we decide to use MaxEpochs=1 in our training option.

According to Fig. 6, it took us 45 minutes and 17 seconds to train the network using the training set, and the accuracy of the network on the validation set is 93.83%.

The technique we used to find the threshold has already been discussed in Section 5.1, so we will not repeat them in this section. With the threshold that we selected for higher accuracy, the network has an accuracy of 93.8% on the test set.



```
Results
Validation accuracy:        93.83%
Training finished:          Reached final iteration

Training Time
Start time:                 20-Jan-2018 19:42:21
Elapsed time:               45 min 17 sec

Training Cycle
Epoch:                      1 of 1
Iteration:                  160 of 160
Iterations per epoch:       160
Maximum iterations:         160

Validation
Frequency:                  160 iterations
Patience:                   5

Other Information
Hardware resource:          Single CPU
Learning rate schedule:     Constant
Learning rate:              0.0001
```

Fig. 6. Output from MATLAB's trainNetwork() function (AlexNet)

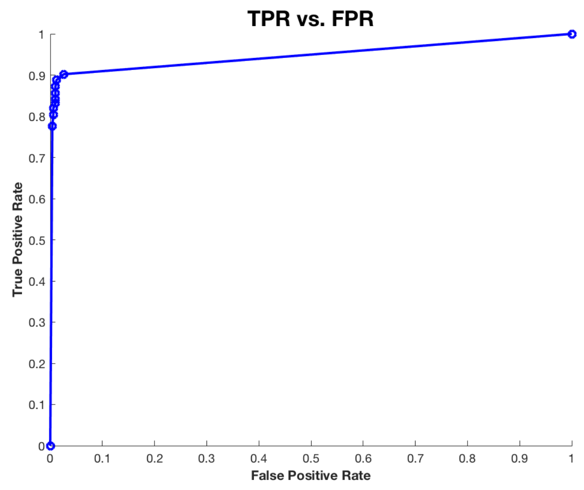


Fig. 7. ROC curve on the test set (AlexNet)

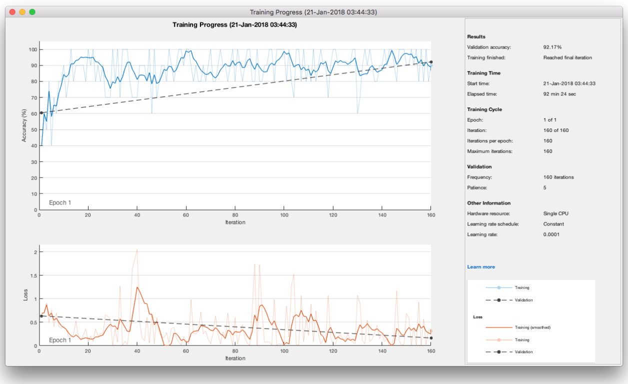
Table 2. AlexNet's Result

Time to train	45 min 17 sec
Time to classify the validation set	2 min 59 sec
Time to classify the test set	4 min 56 sec
Accuracy on the validation set	93.83%
Threshold used	0.1
Accuracy on the test set	93.8%

5.2.2. *Transfer Learning Using GoogLeNet*

Similar to the process of transferring layers from AlexNet, we extracted all the layers excluding the last three from GoogLeNet into our new network. Furthermore, we connected a fully connected layer, a softmax layer, and a classification output layer after the layer 'pool5-drop_7x7_s1'.

According to Fig. 8, it took us 92 minutes and 24 seconds to train the network using the training image set, and the accuracy of the network on the validation set is 92.17%. With the threshold that we selected for higher accuracy, the network has an accuracy of 94% on the test set.



```
Results
Validation accuracy:        92.17%
Training finished:          Reached final iteration

Training Time
Start time:                 21-Jan-2018 03:44:33
Elapsed time:               92 min 24 sec

Training Cycle
Epoch:                      1 of 1
Iteration:                  160 of 160
Iterations per epoch:       160
Maximum iterations:         160

Validation
Frequency:                  160 iterations
Patience:                   5

Other Information
Hardware resource:          Single CPU
Learning rate schedule:     Constant
Learning rate:              0.0001
```

Fig. 8. Output from MATLAB's trainNetwork() function (GoogLeNet)

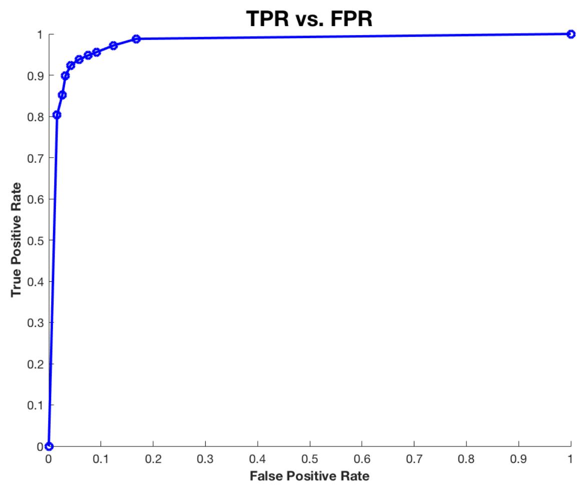


Fig. 9. ROC curve on the test set (GoogLeNet)

Table 3. GoogLeNet's Result

Time to train	92 min 24 sec
Time to classify the validation set	9 min 31 sec
Time to classify the test set	15 min 48 sec
Accuracy on the validation set	92.17%
Threshold used	0.5
Accuracy on the test set	94%

5.2.3. *Feature Extraction Using AlexNet*

This method is similar to the SVM technique we discussed in this paper. The only difference is how we extract features from the images. In the SVM approach, we chose what features we want to use and extracted the features by ourselves. However, in the CNN approach, we used a CNN to extract feature for us. For this problem, we used a pre-trained CNN, AlexNet, and we extracted 'fc7' layer as our feature extracting layer. After feeding images into that layer, we can receive features of the images from it. After having all the features, we could use them to train an SVM to detect sunset scene. Details can be found at 3.1.

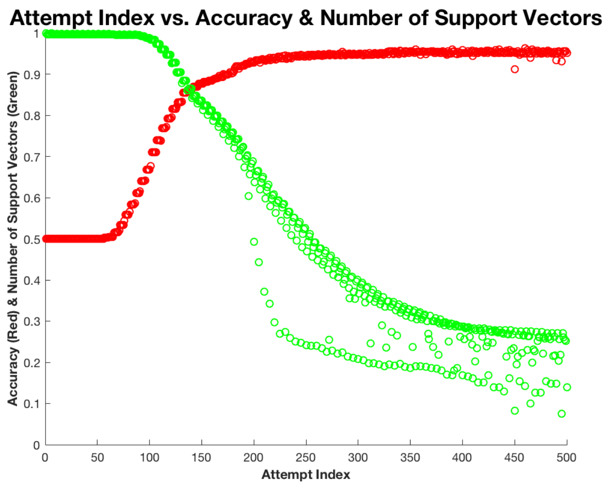


Fig. 10. Attempt Index vs. Accuracy (Red) & Number of Support Vectors (Green)

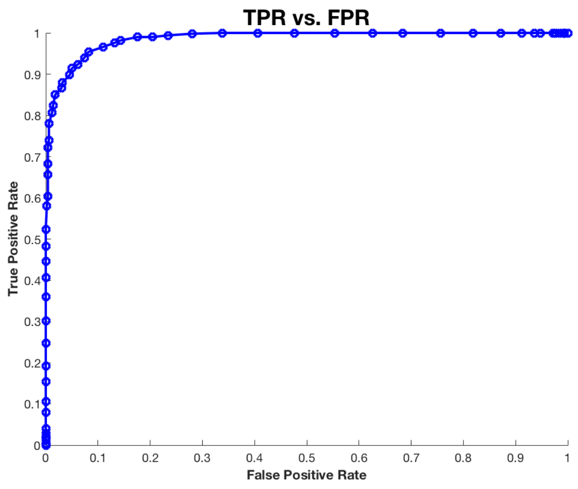


Fig. 11. ROC curve on the test set

Table 4. Feature Extraction's Result

Time to extract features from the train set	9 min 3 sec
Time to extract features from the validation set	3 min 29 sec
Time to extract features from the test set	5 min 49 sec
Time to run fitcsvm and predict on the test set	1 sec
Kernel scale used	92
Box constraint used	1
Support vectors used	228
Accuracy on the validation set	96.33%
Threshold used	-0.3
Accuracy on the test set	93.6%

From Table 2, 3 and 4, we can reach some conclusions. Table 2 and Table 3 demonstrates the performance difference between using two different pretrained networks in the transfer learning. It took GoogLeNet about twice much time to train comparing to AlexNet, but its accuracy on the test set is only 0.2% higher than AlexNet's. Therefore, AlexNet is the most efficient solution to detect the sunset scenes. According to Table 2 and 4, we know that feature extraction CNN can be run about five times faster than the transfer learning CNNs, and the accuracy is about the same as or even better than transfer learning CNNs. For this problem, it is preferable to use feature extraction CNN, which is combined with an SVM.

6. DISCUSSION

6.1. SVM Discussion

Below are some examples of correct or incorrect prediction our SVM made.

Images from Fig. 12 are wrongfully identified as non-sunset scene. Image 1's score is close to the margin, which means that our SVM is not confident about recognizing it as a non-sunset scene. It is reasonable because the sky in image 1 is mainly black while most sunset scenes should have a yellow or orange background. Besides the black color, it is a classical sunset. Our SVM cannot recognize image 2 as a sunset scene because trees in that image cover most of the sky.

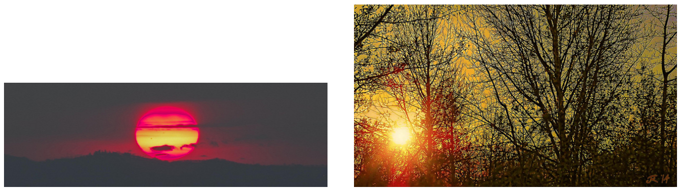


Image 1 (close to the margin) Image 2 (far from the margin)

Fig. 12. False Negative

Images from Fig. 13 are wrongfully identified as sunset scene. Image 1's score is close to the margin, which means that our SVM is not confident about recognizing it as a sunset scene. It makes sense because the main color of this image is orange, even though the background of the image does not look like a sky. Our SVM is confident about recognizing image 2 as a sunset scene because it has a black background that is filled with some yellow colors. Moreover, the flash light looks like the sun as well.

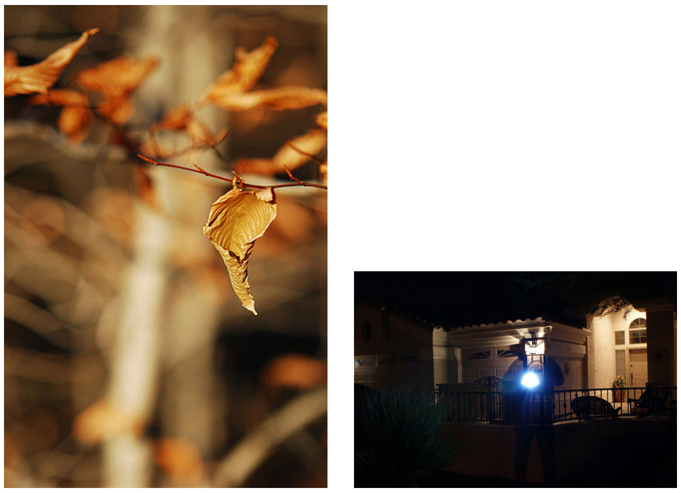


Image 1 (close to the margin) Image 2 (far from the margin)

Fig. 13. False Positive

According to Fig. 14. Image 1 is successfully identified as non-sunset which is true. However, due to the red-orange image scheme, this image is close to the sunset/non-sunset margin. Unlike Image 1, Image 2 is also a non-sunset scene, but the color scheme is green/white, which is not very possible to be considered as sunset scene.

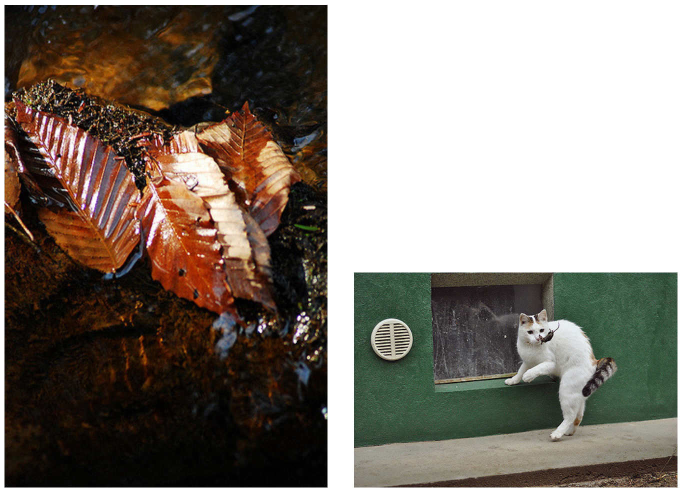


Image 1 (close to the margin) Image 2 (far from the margin)

Fig. 14. True Negative

According to Fig. 15, The two sunset scenes are identified successfully as sunset scene. Image 1's red/orange colors are mostly covered by the blue cloud. That is why it is close to the margin. Image 2 is a classical orange/black sunset scene, so it is far from margin because SVM is confident.

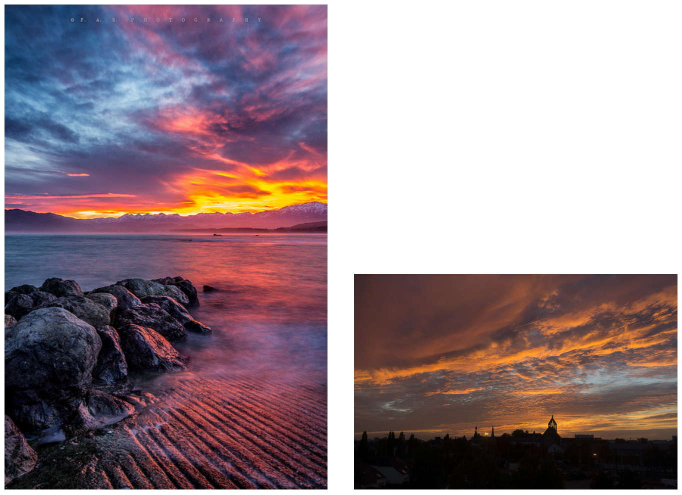


Image 1 (close to the margin) Image 2 (far from the margin)

Fig. 15. True Positive

There is a limitation for SVM. Right now, we subjectively decide to use mean and standard deviation as extracted feature, but we do not know if they are the best choice. If there are better features we can use to get more accuracy results, we should find them. CNN can help us find other features that may produce a better SVM.

6.2. CNN Discussion

The example images provided by CNNs are very similar to the images in section 6.1. All the CNNs performed as expected. Based on section 6.1, CNN may be a better choice. However, CNN takes very long time to train, which is a limitation. Though we could use pretrained CNNs to reduce the training time, it is still longer than using SVM. For CNN that is based on different pretrained network, the accuracy and the runtime could be a tradeoff that we need to consider when we want to decide which pretrained network should be used.

7. CONCLUSION

In short-term, we plan to use recomposition schemes, which were introduced by Boutell [2]. For example, we could flip each image in the training set to increase the number and diversity of the training images. Moreover, as Boutell discovered, people who have different experiences in photography may take photos in different ways. Some images may include a sunset view at corners. Therefore, we should crop four corners of the images and add them to training set as well.

In long-term, we can experience different pretrained CNNs and determine which CNN can produce higher accuracy. Also, we can attempt to train a new CNN instead of using pretrained CNNs, which may give us a better result.

8. REFERENCES

[1] Collected by Matthew R. Boutell, “Images,” .

[2] Robert T. Gray Matthew Boutell, Jiebo Luo, “Sunset scene classification using simulated image recomposition,” *IEEE,10.1109/ICME.2003.1220848*, 2003.