PROJECT 2: SUNSET DETECTOR

Graham Fuller
Jacob Oakes

Rose-Hulman Institute of Technology
Email: fullerga@rose-hulman.edu
oakesja@rose-hulman.edu

ABSTRACT

In this study we created a sunset detection application to classify images as sunsets or non-sunsets. Our process of accomplishing this included extracting and normalizing 294 features from each image from a set of training images. We used these features and their classifications to train SVMs and neural nets. For both types of classifiers we varied their input parameters and thresholds until we achieved the model with highest accuracy.

1. INTRODUCTION

Image recognition can have a big influence on the way we process images. With just some simple knowledge from an image we can determine if it is a sunset or not and from that process the image accordingly. For example, if we know that an image contains a sunset, we can detect the sky area and brighten the other parts of the image to bring out important details someone may not normally see in the original.

Detecting a sunset is a challenge because images of the sun setting are not clear and distinguished. The components to a sunset are a lot more complicated than, say an orange. There are clouds, birds, water and many other elements you need to take into account, as seen in Figure 0.

Fig. 0: Hard image to classify as sunset

The proposed solution is interesting because it learns from only test sunset images how to classify almost all sunset images. No other base information about the images was needed to create a model that classifies so well.

2. EXTENSION

For our extension we chose to use neural nets as our training model instead of SVM. There are numerous different neural net architectures that can be used along with a number of layers that can be used for each. We experimented with two different neural net architectures and performed tests with layer sizes for each. These experiments can be read more in depth in Section 4.4. Our hopes were to find a neural net with similar or better results than SVM. A comparison of the results can be found in Section 4.5.

3. FEATURE EXTRACTION

To extract features from each image, we first broke up the image into a 7x7 grid so that there were 49 sections as seen in Figure 1.

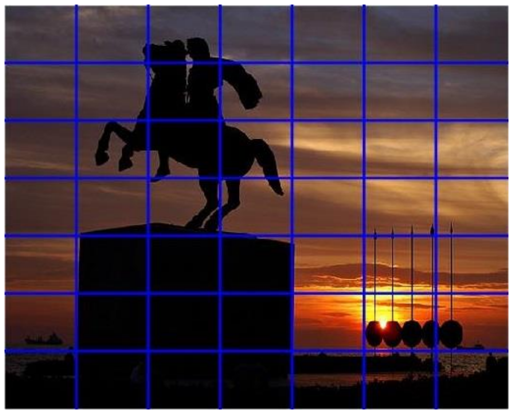


Fig. 1: Picture of grid overlaying image

In order to make each grid cell equal in size, extra pixels in each dimension were cut off on the edge if needed. At most the image would lose 6 rows of pixels on the ends. This part of the image can be considered negligible because it is so small. Next, we calculated the LST values from RGB values using the following formulas:

$$L = R + G + B$$
$$S = R - B$$
$$T = R - 2G + B$$

The range of each value is as follows:

L: 0 to 3
S: -1 to 1
T: -2 to 2

This is because each value RGB has a range from 0 to 1. Then from the first formula you can get 1+1+1 and 0+0+0. From the second formula you get 1-0 and 0-1. From the third formula you get 1-2*0+1 and 0-2*1+0.

To normalize the data we went feature by feature and used the following equation, where min = min value in the cell and max = max value in the cell:

$$(\text{FeatureValue}-\text{min})/(\text{max}-\text{min})$$

We used the normalized data to better compare feature values to one another.

4. EXPERIMENTAL SETUP

For this experiment we used two sets of images; the training set and the testing set.

4.1 Training set

The training set consisted of 499 images. There were 223 regular sunsets, 276 non-sunsets.

4.2 Testing set

The training set consisted of 605 images. There were 222 regular sunsets, 241 non-sunsets, 92 hard sunsets and 50 hard non-sunsets.

4.3 SVM process

First, we extracted the features and normalized them for all of the images. Next, we tried to find the best sigma parameter for our SVM, which used the RBF kernel function. We chose not to experiment with linear and polynomial functions because with 294 features it was very unlikely that our data was linearly separable. The only other choice we had was the RBF full function. We were unable to figure out what parameters to pass to it so we stuck with the RBF kernel function.

For the RBF kernel function we knew the sigma parameter had to be less than $\sqrt{294}$ because after normalizing the values the maximum distance between to images for a feature is one. Since there are 294 features, the maximum distance for the margin is just the square of all the sums between the features. Because of this we varied the sigma parameter from 1 to $\sqrt{294}$ to try and find the best sigma value based on accuracy. As seen in Figure 2.

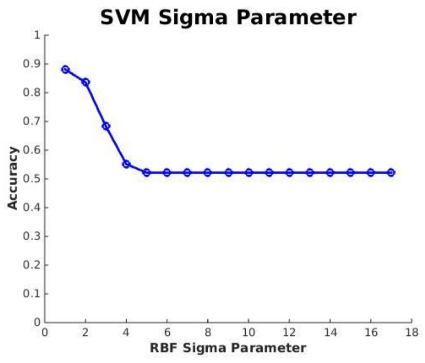


Fig. 2: Graph of tested sigma values

We narrowed down the range until we figured out that the best sigma was somewhere between 0 and 0.5. In the end, we found that the best sigma was 0.007, as seen in Figure 3.

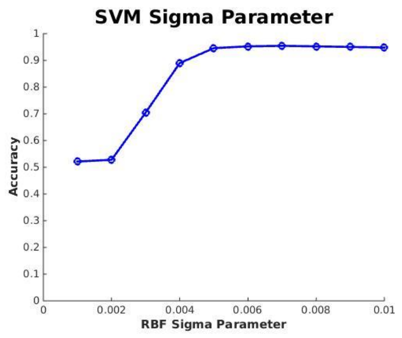


Fig. 3: Graph of tested sigma values with smaller steps

The accuracy for this sigma was 95.46% on the regular testing set, excluding hard images.

After we found the best sigma parameter, we varied the threshold. Inspecting the raw data returned from our SVM, we found the range of the data to be around -3 to 3. This same range was used for the threshold. The ROC curve that was produced by varying the threshold can be seen in Figure 4.

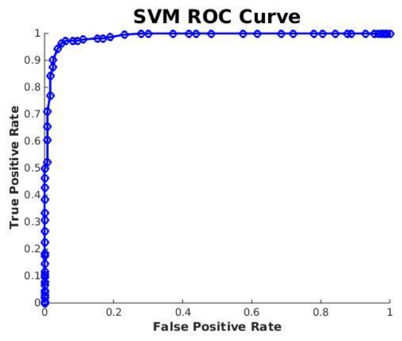


Fig. 4: SVM ROC

For the regular testing set, excluding hard images, we found the best threshold to use to be -0.05 with an accuracy of 95.9%.

When applying our best SVM with a sigma parameter of 0.007 and using a threshold of -0.05 to all of the test images, including the hard ones, we ended up with an accuracy of 88.26%; this SVM had 375 support vectors of the 499 images.

4.4 Neural net process

To find the best neural net we used a similar process to the SVM. However, since neural nets start with random weights, the accuracy for each neural net with the same hidden layer size could be different. For this reason, for each hidden layer size we trained three nets. We decided to choose hidden layer sizes between 0 and 294 since there are 294 features. To start with, we used Matlab's feedforwardnet, starting with a hidden layer of 1 going by 10's until 294. This was very time consuming, so we stopped after a hidden layer size of 61. The average accuracy of the three runs per hidden layer size can be seen in Table 1. All accuracy tests were done with the regular test sets excluding hard images unless specified.

Accuracy	Hidden Layer Size
0.8386	1
0.8459	11
0.8506	21
0.8446	31
0.8494	41
0.8270	51
0.8219	61

Table 1: Feed forward net accuracy

Due to the slow training time of the feed forward nets, we decided to try out Matlab's patternnet. Pattern nets are just feed forward nets built to train and classify inputs according to target classes. They use a different default training function that makes them much faster to train. They also produced a higher accuracy than feed forward nets as well. Due to their speed we were able to test pattern nets with hidden layer size of 1 to 294 with three runs for each hidden layer size. The accuracy vs hidden layer size can be seen in Figure 5. Our best pattern net had an accuracy of 92.27% with 24 hidden layers.

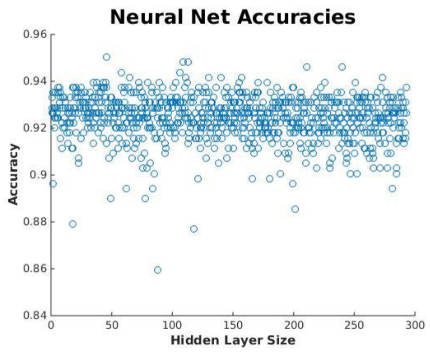


Fig. 5: Pattern net scatter plot of accuracies

Once we found our best pattern net, we varied the threshold just like with our SVM. Neural nets output values between 0 and 1 so the default threshold is 0.5 with anything less than 0.5 not being a sunset and anything greater being a sunset. For this reason, we varied our threshold from 0 to 1. We produced the ROC curve found in Figure 6. The threshold that produced the best accuracy was 0.45 with an accuracy of 95.25%.

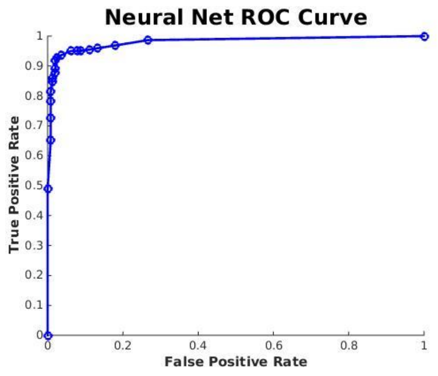


Fig. 6: Pattern net ROC

When applying our best pattern net with a threshold of 0.45 to all of the test images, including the hard ones, we ended up with an accuracy of 87.44%.

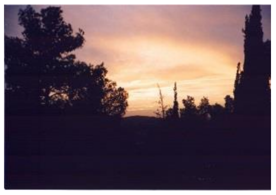
4.5 Results Comparisons

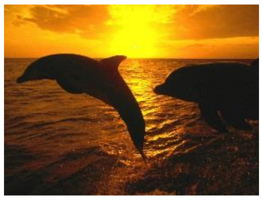
By comparing the neural net ROC in Figure 6 to the SVM ROC in Figure 4, it can be seen that both results are very similar. Our best SVM slightly outperformed with an overall accuracy of 88.26% accuracy for all images which is 0.82% better than our best neural net accuracy.

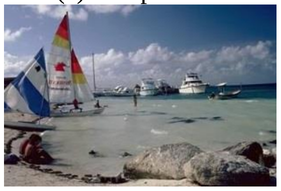
5. CONCLUSION

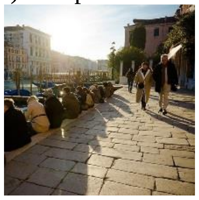
In our test we found images that were classified as false positives, true positives, false negatives and true negatives, as seen in Figure 7. These were for both our best SVM and neural net.

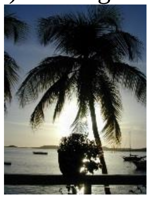
Fig. 7: Classified images

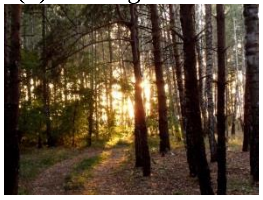

(a) true positive

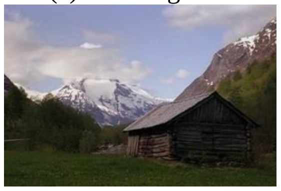

(b) true positive hard

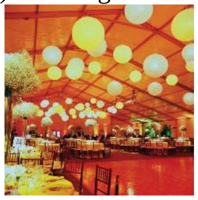

(c) true negative


(d) true negative hard


(e) false negative


(f) false negative hard


(g) false positive


(h) false positive hard

The true positive images in Figure 7a and 7b are determined as sunsets because in each image the sunset is clearly visible and the rest of the image is very plain.

The true negatives images in Figure 7c and 7d are determined as non-sunsets because in each image there is no sunset in the image and the other objects in the image don't look like sunsets.

The false negative images in Figure 7e and 7f are thought not to be sunsets because in each image there is too much blocking the sunset in the image. In both images trees are blocking the sunset.

The false positive images in Figure 7a and 7b are thought to be sunsets because the images contain objects that look like sunsets. In the first image the mountain looks a little like the sun setting in some of the darker images and in the second image the lanterns look like suns.

Given two more weeks, we would find more training data to train our model. That way our results for the testing data would be more accurate. With a wider selection of images we will be able to recognize more sunsets.

Given a year, we would try to detect objects that were in the way of the sunset and remove them from the image so we can analyze the sky more completely. This would help in the case of Figure 7e and 7f since there are trees in the way.

In almost every sunset picture the sun is the brightest part of the image. Another idea would be to take advantage of this fact and instead of using the entire image, only take a set size box around the brightest area of the picture. This way we focus more on the actual sunset and less on the rest of the picture.