## CSSE463: Image Recognition / Sunset Detector

**Work in pairs on this assignment.**

The overall goal of this project is to implement a system similar to the *baseline* classifier described in my ICME paper (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; found in the Papers section of Moodle and the website). You will write MATLAB code that will extract features from sets of images, train a classifier to classify images using them, and report the results in a short conference-paper quality submission. In part 2, you’ll try applying deep learning to solve the same problem.

## PART 1: CLASSICAL IMAGE RECOGNITION (SVM)

**Deliverables:**

1. **All MATLAB code**.

2. **Report.** The following sections mimic the organization of many conference papers. *This will be good practice for your* *term project and for writing such papers. Other details may be found in the rubric in this folder:* ***look at it****.*

*Abstract:* Briefly summarize the entire paper, including background and approximately 1 sentence each for features, classifier, and **numerical** results. Pack many details concisely.

*Section 1*: Introduction. Why do we care about sunset detection (and scene classification as a whole)? Why is sunset detection a challenging problem? Include at least one image that shows how challenging it is. What is interesting about the proposed solution? Set the problem in the more general context of image recognition – how does it fit? Define the scope of the problem clearly, that is, what are the inputs and form of the outputs? Your goal is to grab the attention of the user so they keep reading, so make sure you do this explicitly. This section is mostly non-technical and several paragraphs long.

*Section 2*: Process followed for feature extraction. Describe the process, not merely walking through your code. How did you extract the features? Give lots of details, showing an image with the grid overlaid on it. Give equations for RGB to LST. What are the theoretical ranges for each band? (It is sufficient to discuss the means, I’ll leave analysis of the standard deviations as an option for a particularly motivated team.) Discuss that this is the reason for standardizing the data (hint: most classifiers allow an option to standardize the data – you should use that option if you can; otherwise, you’d need to do it yourself). Describe how you handled images with dimensions that weren’t a multiple of the block size, and why that was OK. Since you are building on the work in the paper above, add a Reference section at the end of the paper, add that paper, and cite it here (if you aren’t sure how, see the many examples of citations in the paper above).

*Section 3*: Classification.

3a. Write a short paragraph about the baseline classifier (SVMs). How do SVMs work?

3b: Write a short paragraph about convolutional neural nets. Describe the architecture: number and type of layers, plus anything else that seems relevant. (**Skip for part 1**)

*Section 4*: Experimental setup. The last sections include details about the process regardless of the data set used. Now include details about the data set: your inputs and outputs. How many images of each type (sunset and non-sunset) did you use in the training, validation, and testing sets? What was the approximate range of resolutions of the images?

*Section 5*: Results.

5a: SVN results. Document the SVN experiments conducted. How did you choose your classifier hyper-parameters (kernel scale and box constraints)? What was the accuracy of the classifier on the validation set once the hyper-parameters were tuned? How many of the training images became support vectors? (Remember that generally you want higher accuracy but fewer support vectors, since too many support vectors means overfitting. You’ll probably want to balance these two.) Include a table, figure, or otherwise show **evidence** of experimenting with hyper-parameters on the validation set. If it is too lengthy, you can move it to an Appendix and reference it in this section. Once you tune your hyper parameters, then run your classifier on the test set and show your results in the form of an ROC curve as you vary your decision threshold away from 0.

5b: CNN results (**skip for part 1**)

*Section 6*: Discussion.

6a: Show sample test-set images that were classified successfully and unsuccessfully by the SVN. Include at least 8 images - 2 of each type (FP, TP, FN, TN) - and discuss for each image why the classification results seem reasonable. (A good way to choose interesting images is to look both at ones in the margin and ones that are far from the margin – use the predict functions’s **score** output to guide you.)

6b: CNN discussion (**skip for part 1**)

*Section 7*: Conclusion. Describe what the next steps would be if you had another 2 weeks to continue working on the problem. What if you had another year? Don’t let this just be an afterthought – consider issues that you identified in the discussion.

**Process and Hints (this is how I would direct a research assistant)**

**Read carefully, this includes other steps you haven’t thought of.**

1. Get the zip file of images here: <http://sunset.csse.rose-hulman.edu/>

The appendix describes the image collection process.

1. Start by writing your feature extraction code for a single image, that is, calculate the 294-dimension feature vector from the image. You should probably end up encapsulating this as a MATLAB function.

The paper uses Luv space – here, we’ll use LST space. (I have tried LST and RGB spaces and while the difference isn’t huge, LST works slightly better.) The unscaled conversion to LST color space is:

L = R + G + B S = R – B T = R – 2G + B

There is no need to scale them (like dividing L by 3) unless you want to visualize results in this color space. Test your code for bugs (e.g., L should be the grayscale image, and you should have some negatives in S and T). There is sample output on one image in this folder to help you “unit-test” this part of your code. It follows the format below.

1. Write a script that will loop over the whole sets of images (training, validation, and testing sets), extracting features from each one.

If any images cause your script to crash, let me know – it's probably save to delete them if there are only 1-2. Also, if your computer generates any images like Thumbs.db, you can ignore or delete them. Note: you may find MATLAB’s imagedatastore functions to be very helpful for this – they are relatively new and very helpful, especially when you complete part 2. <https://www.mathworks.com/help/matlab/ref/imagedatastore.html>

I also included a couple files with some skeleton code that you may find helpful.

Save the features into a matrix (X). You will also have a vector (Y) of class labels, +1 for sunsets, and -1 for nonsunsets, or otherwise pass the labels into the SVM.

Structure of matrix X:

[img1L11, img1L11, img1S11, img1S11, img1T11, img1T11, img1L12, … ;

img2L11, img2L11, img2S11, img2S11, img2T11, img2T11, img2L12, … ;

…

imgNL11, … ];

As an example, imgnLij means the first moment (mean) of the L band of the block in row i, column j of img n, where i = 1:7, j = 1:7, n = 1:N. Each row of the matrix is the 294 features extracted from a single image. L is second moment (standard deviation); you’ll use standard deviation, although we used variance in the ICME paper… they works about the same. Note that for a 2D matrix M, std(std(M)) does not compute the correct standard deviation! (try it on [3 3; 1 5]) Instead, convert M to a column vector using the **:** operator first, as in std(M(:)).

Note that some of the S and T values should be negative – if not, maybe you forgot to transform to type double to do math?

I give some values for one image in an auxiliary file in this folder, to help you “unit-test” your feature extraction code. Make sure you get the same values.

1. You will then want to normalize your features, because L, S, and T bands have somewhat different scales, as do the means and variances. (From the formulas, what are they?) The easiest method:

a. Look up SVM training parameters for options on standardizing the data.

Other options that you could use if SVM’s didn’t give this option. (This is what we used to have to do and I include it here just because it’s interesting.)

b. Scale each feature type (for example, all the L) so that its max value for any block in the image set = 1 and its min value = 0. To do this for L, you will need to find the max and min of all the L means of all 49 blocks **in all images in all training and testing sets together (don’t normalize each image or set separately)**! [Yes, you should read the ~1000 images from all 4 folders into a ***single*** feature matrix and pass that to the normalize function. Then split it back up when you train and test.] Then the output is out = (orig-min)/(max-min) – you can verify this works. You may use the normalizeFeatures01.m code I gave you to compute this.

c. Scale so that it’s “zero mean, unit variance”. This is another typical approach: make the mean of that single feature type to be 0 and the standard deviation 1 by first subtracting the mean for the feature type, then dividing by its standard deviation. This is a little less sensitive to outliers, although past students have found that options (b) and (c) work roughly the same.

1. (Optional time saver) At this point, you probably have a bunch of matrices of features ({sunset, non} x {train, validate, test}. You can save your results in a file so that you don’t have to keep extracting features every time you turn on your machine or rerun your classifier with a new set of parameters:

save(‘features.mat’, ‘X’, ‘Y’) will do this;

save(‘features.mat’) saves *all* the variables in the workspace.

load(‘features.mat’) will retrieve them back into memory.

1. Train the classifiers, experimenting with kernel scale and box constraints. Warning: **this takes time**; you should automate this process using a script and looping. <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf> has a technique worthy of consideration. There may be MATLAB svm functions that can do this for you – I’m new to this library and haven’t figured out yet what it can do. Calculate accuracy on the regular validation set for each hyper-parameter choice so you can check which are best. **Save transcripts of these runs, so you have a record for your report: you need to demonstrate that you tried to tune your parameters.** Don’t expect your accuracy to be the same as the baseline I reported in the paper. Accuracy can vary quite a bit depending on how hard the set of images is. Accuracy could be lower, because one has a smaller training set, and thus haven’t filled the feature space as well, or it could be higher because the sunsets are more “obvious”.

You should consider the number of support vectors in your analysis. Find out how many you have – sometimes you can change the kernel scale and get by with fewer support vectors while keeping the accuracy fairly constant. You’ll probably want to balance them.

How will you compare these? If you have tested them for a whole range of scales and box constraints and saved the accuracies and numbers of support vectors for each in matrices, you can look at the tables or even plot them using **mesh** (for example, **mesh(numSupportVectors)**.

1. Create an ROC curve. What do you vary to get all the points that comprise the curve? Don’t vary the parameters of the SVM. Instead, vary the threshold between what you call a sunset and a non-sunset.   
     
   Here is a reminder of ROC curves from class, applied to SVMs. There are 2 outputs from predict: [label, score] = predict(…). label is +/1 as you saw in lab and score is a real number (distance from margin, pos or negative, depending on which side of the default threshold it's on). This default threshold between classes is 0, but you can move it, say in increments of 0.1.   
     
   Let’s say we classify 4 images, and y1 = [3.2, -1.3, 0.03, -0.01]   
     
   If the threshold (call it t) = 0, then the first and third are detected as sunsets; if -0.1, then the first, third, and fourth, and if +0.1, then only the first would be considered a sunset. (Make sure you understand this.)  
     
   In the general case, assume that strong sunsets have large positive outputs, strong non-sunsets have large negative outputs, and some weak examples of each are near 0 (hard to classify since ambiguous).   
     
   With a high threshold, say t = 5, then all (or nearly all) of the images with output above 5 should be sunsets, so you get few false positives. Thus your false positive rate (FPR) is very low, which is good. However, your true positive rate (TPR, i.e., recall) will be very low as well, since all those true sunsets with output below 5 get called non-sunsets.   
     
   In the other extreme case, t = -5, you get high TPR (good), but a high FPR as well (bad). With t = 0, (the default), you get a balance.   
     
   So once you tune your SVM’s hyper-parameters on the validation set, you run the SVM once on the test set and it outputs **score**. You then write code to classify the set of images based on a threshold. To get the curve, put that code in a loop, and calculate the true positive rate and a false positive rate for each threshold. Then graph TPR vs. FPR via the ROC plotting code I gave you (roc.m).
2. Look at some of your successes and failures and include at least 2 example images of each type in your report. Hypothesize why you got some of them right and some wrong. If a particular result doesn’t make sense, it’s OK to say that too, I’ve come across my share of mystifying images; the point is that you give it some thought. Then discuss future work.
3. Write it up in a report, as described above. Some people, including myself, like to write as they go.

(Part 2 on next page)

## PART 2: DEEP LEARNING (CNN)

In this part, you will experiment with using a convolutional neural network (CNN) to perform classification of sunset images. As you know, CNN’s perform both feature extraction and classification of images.

**Deliverables:**

1. **All MATLAB code**.

2. **Report.**

a. Address any feedback on your draft of part 1 by adding/rewriting as needed.

b. Modify your paper to include CNNs. First, add sections 3b, 5b, and 6b. Second, modify the other sections (abstract, introduction, conclusion) to include your CNN work.

**Process and Hints: Overall Idea**

It takes time to train and tune the hyper-parameters of deep neural networks. Thus, we should try to leverage the work of other researchers if possible. There are 4 options:

1. One possibility would be to use a pre-trained network like AlexNet, GoogleNet, ResNet, or VGG-19 to classify sunsets. (All of these are available to you in MATLAB.) Unfortunately, while they are all trained to recognize the same 1000 image and object classes as each other, none of those classes is *sunset*.
2. CNN for *feature extraction*. The next option is to use the feature extraction part of one of the pre-trained CNNs and then use the features extracted for each image as inputs for a SVM.
3. The third option is to use *transfer learning*. Transfer learning in general means re-using most of a pre-trained CNN and just replacing and re-training a few layers. In image recognition, we re-use the feature-extraction part of the CNN and just replace and re-train the fully-connected layers. Train this network is usually faster (on the order of hours instead of days or weeks) since it only needs to learn the weights of those last few layers.
4. The final option is to build and train a CNN from scratch, using a similar structure as these other CNNs.

You can read about these options here: <https://www.mathworks.com/help/nnet/ug/pretrained-convolutional-neural-networks.html> That page, along with pages it links to, describe how to download and use the common pre-trained networks and even how to import models from other deep learning frameworks like Caffe and Keras/Tensorflow. You should have learned how to do this in the lab.

I will not dictate what options you do. I have tried options (2) and (3) and think they are the most straightforward. Students who have taken the Deep Leaning class might want to try option (4) to compare MATLAB’s deep learning features with those of Keras. (However, MATLAB isn’t installed on the gauss or hinton servers, so you’d need to use another machine to train your networks.)

Here are some hints from my experience. For feature extraction, you’ll want to use the activations of one of the last layers in the network as your features. Learn how to use the **activations** function. The links above teach about this.

For transfer learning, make sure to capture and include the figure that shows accuracy on the training and validation sets for your report. They are formatted like in Figure 1.



**Figure 1:** Example of output from MATLAB’s **trainNetwork()** function.

Which pre-trained network works best? Does the feature extraction technique with an SVM have any chance at matching the accuracy of a CNN built by using transfer learning? You have a lot of flexibility here within these requirements:

1. You must try at least two of options (2) - (4) above.
2. You must compare the runtime and accuracy of at least two pre-trained networks at least once (for feature extraction or transfer learning).
3. For each option you try, document your process and results clearly like you did in Part 1, for example showing how you used the validation set (as in, CNN training figures or evidence of SVM hyper-parameter tuning) and reporting the accuracy and generating an ROC curve for the test set. (You’ll have to consider how to generate an ROC curve from a CNN output. Hint: you could use the output of the CNN’s softmax or the activations on the layer before the softmax.)

**Appendix: Image collection process**

Flickr is a photosharing website that has APIs for image download. The typical user is a semi-professional photographer.

I started with **sunset images** from this Flickr group: <https://www.flickr.com/groups/sunsetcentral/pool/> . The group contained 19,379 photos taken by over 1600 photographers. A single human user examined thumbnails of the first 2491 photos (enough so that over 2000 remained after pruning). 460 photos that were clearly not detectable sunsets were pruned (52 of which on a second pass looking at medium-resolution images):

black and white photos,   
those without sun or at least warm-colored sky present (for example, those presumably of a late sunrise or early sunset),  
those of a night sky,  
those of a red sun on a gray/black background,   
close-ups of only the sun  
those of the sun reflected off mountains or buildings but taken pointing away from the setting sun,   
those with provocative foreground content,   
those with strong photographic effects (e.g., taken through sunglasses or in a rear-view mirror) or post-capture effects like photo montages or thick borders added.

The remaining 2032 photos include some with large foreground objects, weakly-colored or very dark skies; the intent was not to make an ideal or easy set, but a fair set. Admittedly, this pruning was done quickly, and some remaining photos will be questionable as sunsets.

I used this group as a representative collection of general **nonsunset photos**: <https://www.flickr.com/groups/photography-club-of-flickr/pool/>

It has 9,585 photos taken by almost 400 photographers. A single human user examined thumbnails of the first 3374 photos (enough so that over 2000 remained after pruning). 1347 photos were pruned (125 on the second pass):

sunsets (a small fraction),  
black and white photos (surprisingly many),   
those with provocative foreground content,   
those with strong photographic effects (e.g., taken through sunglasses or in a rear-view mirror) or post-capture effects photo montages or like thick borders added (a surprisingly large number).

Like the sunsets, the remaining 2027 photos could be questionable for some reason.

I took the first 800 images in each set for training, the next 300 for validation, and the next 500 images for a test set. (I reserved the final 400+ images for future use.)

You are welcome to report to me any photos that you believe should not be in the dataset in the future.