SUNSET DETECTOR

Chirag Sirigere, Connie Zhu

CSSE463 Image Recognition

27 July 2021

ABSTRACT

Classifying an image as a sunset or not using an SVM is quite difficult because the right kernel scale, box constraint, and threshold values must be chosen. This project entails converting 3200 different RGB images (1600 from the train, 1000 from the test, and 600 from the validation datasets) to LST color space, segmenting the images into 49 sections and gathering the mean and standard deviation of each segment of each LST color band. The features (mean and standard deviation) are then standardized to a scale of 0 to 1 and processed through an SVM based on the train dataset to determine if the test dataset contains sunsets (+1) or not (-1) using a correctly fitted ROC curve value from the threshold values. The baseline LST-SVM has an accuracy of 89.6% on the test dataset but is overfitted based on the validation dataset. In a second part of the project, two CNNs are used to classify the images, one from squeezenet for transfer learning and one from scratch in Keras. A few observations were made in this project: Both CNNs performed better than the LST-SVM with accuracies of 95% and marginal FPR versus TPR ROC curves (nearly 0:0.9). The squeezenet CNN, however, provided a more stable validation loss function than the from-scratch Keras CNN.

1. INTRODUCTION

This project is a continuation of Matthew Boutell's research [1]. With the rapid growth of digital image data, it has become an increasingly urgent need for computers to automatically understand images. As an important research topic of understanding the image, image classification and recognition (mainly scene classification) has received attention. Image classification has a wide range of applications in many fields of computer vision, such as content-based image retrieval, remote sensing image analysis, or video surveillance. After decades of development, although many encouraging results have been achieved in image classification, it is still a very challenging problem due to the complexity of the problem itself.

Image classification is widely used in content-based image retrieval. The purpose of content-based image retrieval is to search in the image database and find the corresponding image that meets the query conditions based on the content information or the specified query criteria under the premise of a given query image.

Content-based image retrieval refers to the query condition itself is an image, or a description of the image content. The indexing method is to extract the underlying features, and then calculate and compare the distance between these features and the query condition to determine how similar the two pictures are.

Similar to the content-based image retrieval system, we detect sunsets by extracting features from each sunset image and putting those features into an SVM to calculate the distance between the features of different images to predict whether an image can be classified as sunset. We also run two different CNNs on the raw RGB images.

More applications of this method would be classifying flowers in images or objects soaring through the sky, and these tasks are not easy to accomplish because any small deviation in the color spaces of the segmented images or the SVM model will make the classifier worse. Everything from the color space to the RBF kernel scale needs to be taken into consideration to make the perfect model to predict a sunset.

Having learned how to visualize and model a two-dimensional dataset with an SVM in this image recognition class, modeling a multidimensional dataset ($d > 3$) becomes difficult because it is hard to visualize the data. Even with a matrix of multivariate plots, it is time consuming and impossible to visualize 294 features.

Our task is to recognize and classify sunset images from the website http://sunset.csse.rose-hulman.edu with a zipped file of images. Figure 1 shows an image of a sunset from the train dataset with warm colors in the sky, dark colors on the ground, and a large horizon

with no interfering objects in the foreground. Figure 2 shows an image of a non-sunset from the train dataset with a bridge as the foreground and no warm, natural colors in the sky that could indicate a setting sun.

Figure 1: An image of a sunset.

Figure 2: An image of a non-sunset.

2. PROCESS

By converting the images from RGB to LST space, we can extract the mean and standard deviation of each band of a segmented image to be passed through an LST-SVM. A few different preprocessing approaches were taken on the CNN classifier.

The unzipped folder has three subfolders titled train, test, and validate and each of these folders has two folders with non-sunset images and sunset images. The number of non-sunset and sunset images in the train, test, and validate sets are 800, 500, and 300, each, giving a total of 1600, 1000, and 600 images in the train, test, and validate datasets, respectively.

2a. LST-SVM Preprocessing

We first converted the RGB images to LST space and divided each image into 49 regions using a 7 x 7 grid. Provided are the RGB to LST color space conversion equations (Equations 1 to 3):

$$L = R + G + B \quad (1)$$

$$S = R - B \quad (2)$$

$$T = R - 2G + B \quad (3)$$

The problem with splitting an image into a 7 x 7 grid was that some images had uneven dimensions. The solution was to remove the last row or column depending on the mod operator of the row or column and the block size of 7. If either the mod of the row or the column and the block size of 7 resulted in 0, the end index of the dimension would be removed. Removing the last index of a dimension is fine because a single row or column would not affect the entire feature significantly. Also, the images have enough pixels and information at the ends that a one-pixel continuation of a dimension is negligible. Shown in Figure 3 is an example of a 7 x 7 segmented image of the bridge from Figure 2.

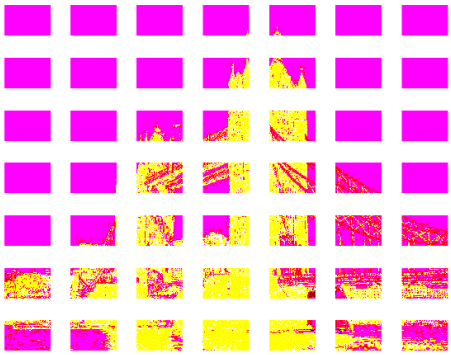

Figure 3: The LST space image of the bridge from Figure 2.

We then computed the mean and standard deviation of each LST color band. This gives us 294 features for 49 regions x 3 bands x 2 (mean and variance). The 1600 sunset and non-sunset images from the train dataset were combined with the 1000 sunset and non-sunset images from the test dataset giving a 2600 row by 294 column feature matrix.

Theoretically, the ranges of values of the LST space are L: [0, 765], S: [-255, 255], T: [-510, 510]. Since the values have different weights, the matrix was then

normalized between 0 and 1 based on the 0-1 feature normalization method provided by Equation 4:

$$(featureValue - minimumValue) \,/\, maximumValue \quad (4)$$

The 2600-row feature matrix is then split back into the 1600-row train and 1000-row test feature matrices. The train feature matrix is passed through an SVM to determine the best kernel scale and box constraint values that output the lowest false positive rate and the highest accuracy and true positive rate on the validation dataset. The best kernel scale and box constraint values were set to be used on the test dataset with a variable threshold value. Again, the best threshold value that outputted the lowest false positive rate and the highest accuracy and true positive rate was considered.

2b. CNN Preprocessing

Two different processes were taken by each teammate: Chirag Sirigere used transfer learning while Connie Zhu used a CNN from scratch in Keras.

The images were not converted to any color space from their original RGB color bands but were rescaled to 227 x 227 pixels and were either randomly disoriented or not, to be inputted to the squeezenet network skeleton for transfer learning.

The plan for transfer learning was to modify the last convolution layer and the classification output layer and do the following experiments:

1. Run with all layers frozen up to the last convolution layer and 10 mini batches with a maximum of 6 epochs.
2. Run with all layers frozen up to the last convolution layer and 20 mini batches with a maximum of 6 epochs.
3. Run with all layers unfrozen, having the algorithm relearn every weight with 20 mini batches, with a maximum of 6 epochs and disoriented input images to remove the sunset and clouds on top.
4. Run experiment 3 but with 10 epochs instead of 6 epochs.

For the CNN made in Keras, the images were normalized from [0, 255] to [0, 1] and the images were randomly disoriented.

3. CLASSIFICATION

3a. SVM Classifier

Support Vector Machine (SVM) is a supervised learning model which analyzes and classifies data. Given a set of training datas, each training data is pre-labeled as belonging to one or the other of the two categories. Then, the SVM builds a model that assigns new data to one of the two categories. The SVM model puts the data as points in space so that the mapping makes the data of individual categories separated by the widest possible interval, the margin. This margin is a line or nonlinear function plotted in the space that contains support vectors. Support vectors are points that keep the margin from being too distanced. The SVM maps the new data to the same space and predicts the category they belong to, based on which side they fall on. Using Equation 5 for the Radial-Basis Function (RBF),

$$K(x, x_i) = exp(-\frac{||x - x_i||^2}{2\sigma^2}) \quad (5)$$

the hyperparameter, σ, controls how fast the equation tapers off, in other words, the width of the Gaussian curve.

3b. CNN Classifier

Convolutional Neural Networks are a type of feedforward neural network generally composed of these types of layers: the input layer, a convolutional layer, a ReLU layer, a pooling layer, a fully connected classification layer, and a SoftMax function layer. These layers not only classify an image or signal, but they also learn features in the image or signal, which is what makes CNNs useful. In that order, the individual layers have the following purposes:

1. The input layer is an entire dataset of images or signals, all rescaled to fit into the neural network architecture.
2. The convolutional layer usually is a 3 x 3 filter with tens of convolution filters that take each image and learn the edges and color gradients of the image.
3. The ReLU layer is a nonlinear transfer function that changes the value of the input to a new value based on Equation 6 below.

$$g(x) = max(x, 0) \quad (6)$$

4. The pooling layer down samples the number of data points from a previous

layer by applying a striding function that captures the maximum or average value in each segment of the previous layer.

Layers 2, 3, and 4 are often grouped into the feature learning subset of layers since they learn and determine the different features in the image.

5. The fully-connected classification layer is a neural network of neurons classifying the various outputs of the previous layers.
6. The SoftMax function normalizes the outputs of the fully-connected classification layer to the range [0, 1] given by Equation 7 below.

$$s(x_i) = \frac{exp(x_i)}{\sum_{k=1}^{d} exp(x_k)} \quad (5)$$

4. EXPERIMENTAL SETUP

Originally, the non sunset and sunset images were taken from the photo sharing website Flickr. At least 2000 sunset images were taken from the website: https://www.flickr.com/groups/sunsetcentral/pool/ Another approximately 2000 non sunset images were selected from the following website: https://www.flickr.com/groups/photography-club-of-flickr/pool/. For the training dataset, 800 images for each sunset and non-sunset were taken, along with 500 images for each sunset and non-sunset in the testing dataset. For the validation dataset, 300 sunset images and 300 non sunset images are selected. Each RGB image has dimensions ranging from 300 pixels to 650 pixels with a resolution of 96 dpi.

5. RESULTS

5a. LST-SVM Results

There were two ways to determine the best SVM kernel scale and box constraint values: automatically and manually optimizing for k and bc.

For the LST-SVM on the validation set, two for loops were used to determine each combination of kernel scale value from 1 to 10 and box constraint value from 1 to 10. Shown in Table 1 of the Appendix, we noticed a jump in nearly twice the true positive rate between the two bolded rows resulting in the need for another iteration of different numbers.

Using the 'OptimizeHyperparameters' option of 'auto', we were able to find the second-best kernel scale value and box constraint value of 2.8 and 2.5, respectively. The accuracy of the SVM on the validation set with kernel scale of 2.8 and box constraint of 2.5 achieved 84% accuracy with a false positive rate of 0.293 and a near perfect true positive rate of 0.97. The problem with this was the relatively higher FPR compared to a kernel scale and box constraint value of 2 and 1, respectively. Figure 33 in the Appendix shows the iterations of kernel scale values and box constraint values.

This called for a further fine tuning of the hyper parameters. Using the two for loops, our objective was to keep the TPR and accuracy above 80% while keeping FPR below 20% because if the accuracy were less than 80%, say 75%, one would be better off randomly guessing if an image has a sunset or not with 50% accuracy. This was achieved with the kernel scale value of 1.4 and 1, shown in Table 2 of the Appendix. One downside to this is that the ratio of support vectors to the number of observations in the train dataset is nearly 0.7, possibly overfitting the validation dataset compared to the other hyperparameters with ratios of nearly 0.4.

After tuning the hyperparameters, the LST-SVM was run on the test dataset with the kernel scale and box constraint values of 1.4 and 1. Figure 4 shows the ROC curve of the test set with different decision threshold values. From the indicated accuracy, 89.6% at a threshold value of 0, the train dataset has overfit the test dataset.

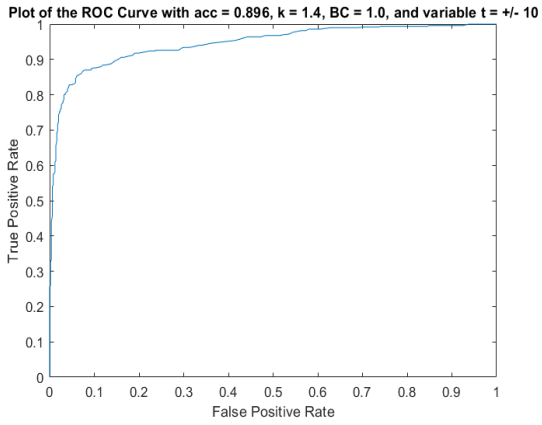

Figure 4: The ROC curve of the test dataset.

5b. CNN Results

The transfer learning progress graphs of accuracy and loss can be seen in the Appendix. The following figures show the ROC curve of the various test dataset outputs by varying the threshold between 0 and 1.

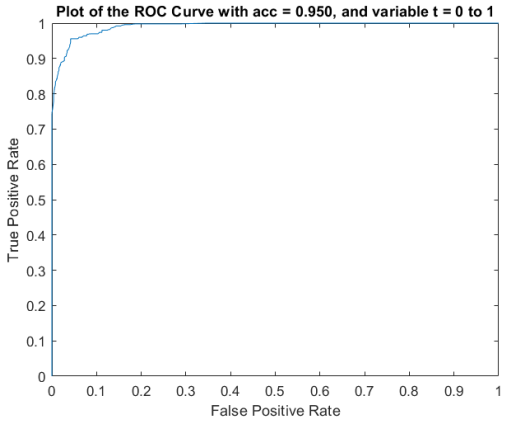

Figure 5: ROC curve for Section 2.3.1. Its respective transfer learning progress graph of accuracy and loss can be seen in Figure 29 of the Appendix.

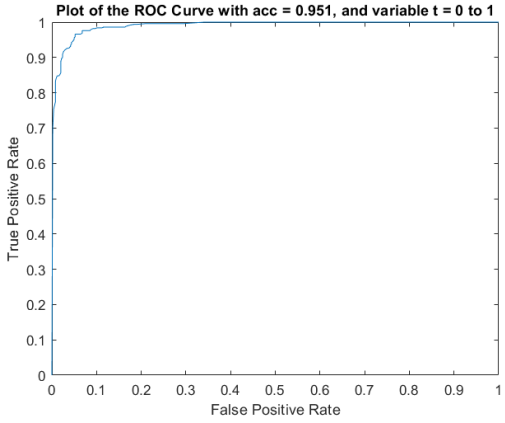

Figure 6: ROC curve for Section 2.3.2. Its respective transfer learning progress graph of accuracy and loss can be seen in Figure 30 of the Appendix.

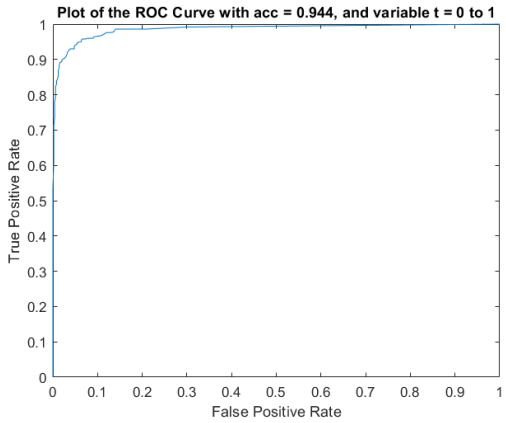

Figure 7: ROC curve for Section 2.3.3. Its respective transfer learning progress graph of accuracy and loss can be seen in Figure 31 of the Appendix.

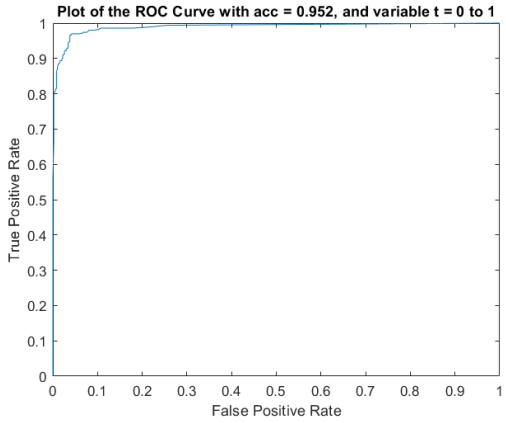

Figure 8: ROC curve for Section 2.3.4. Its respective transfer learning progress graph of accuracy and loss can be seen in Figure 32 of the Appendix.

The structure of the Keras neural network contains 3 layers of convolutional layer 32, 3x3 filters with an activation function of ReLU. 3 layers of pooling layers with a kernel size of 3x3. 1 flatten layer. 2 dense layers with 32 neutrons and activation function of ReLU. 1 dense layer of 1 neuron and an activation function of sigmoid.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	multiple	896
max_pooling2d (MaxPooling2D)	multiple	0
conv2d_1 (Conv2D)	multiple	9248
max_pooling2d_1 (MaxPooling2	multiple	0
conv2d_2 (Conv2D)	multiple	9248
max_pooling2d_2 (MaxPooling2	multiple	0
flatten (Flatten)	multiple	0
dense (Dense)	multiple	16416
dense_1 (Dense)	multiple	1056
dense_2 (Dense)	multiple	33

Total params: 36,897
Trainable params: 36,897
Non-trainable params: 0

Figure 9: The structure of CNN.

In addition to the Keras CNN structure, the accuracy of this model reached 95%. One problem with this is that the test dataset was overfitted because of the rise in validation loss. Although an auto stop was enforced, the validation loss still rose. Compared to the squeezenet CNN for transfer learning, the Keras CNN would be considered unstable because the Keras CNN's validation loss rose considerably.

Compared to the LST-SVM, the accuracy is nearly 5% higher and the ROC curves of the CNNs show that there is some leeway for the FPR versus TPR, approaching nearly 90% TPR with nearly 0% FPR, while the LST-SVM gives nearly a 55% TPR for 0% FPR. Even though the LST-SVM models the training dataset quickly in less than two minutes, the high accuracy and low TPR is undesirable. At the cost of the time taken to train and apply the network to the test dataset, the CNN is better than the LST-SVM as shown in the ROC curves and validation accuracies.

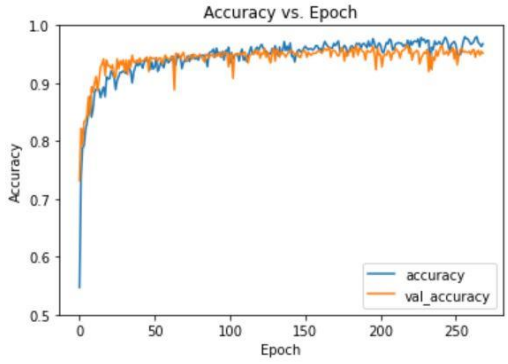

Figure 10: Plot of the validation accuracy

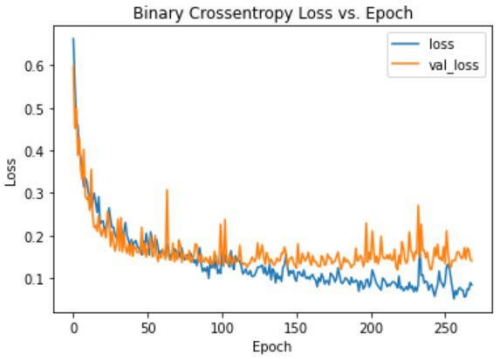

Figure 11: Plot of validation loss

6. DISCUSSION

6a. LST-SVM

Shown below are some images that were classified as true positive, true negative, false positive, and false negative, respectively.

Figure 12 is one of the easier true positive images to classify in the test set because there are only a few colors in the image, and they are all warm. This image clearly has clouds and a horizon with warm colors in the majority of the image, and low brightness.

Figure 12: sunset\13610943374_102cbf1185_z.jpg classified as a true positive with a distance score of 2.4513.

With a distance score of 0.0074859, Figure 13 is one of the harder true positive images to classify in the test set because although the entire image is warm and vibrant with only the horizon being black, there are colorful objects in the foreground that might make it harder for the LST-SVM to classify the image correctly.

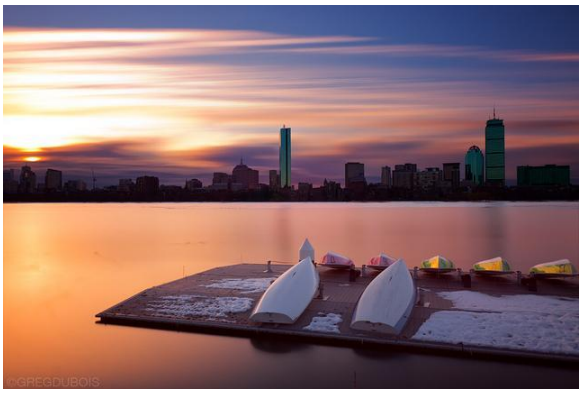

Figure 13: sunset\12858408714_3a2dfa374f_z.jpg classified as a true positive with a distance of 0.0074859.

Figure 14 is an image of a deer, and has no warm colors, streaks of clouds, a city skyline, a sun's reflection, or anything that sunset image has, making this image easy for the LST-SVM to classify the image.

Figure 14:nonsunset\4149765026_8b00482eb8_z.jpg classified as a true negative with a distance of -2.1896.

In this image, there are warm colors from the wooden chairs and wall, but the LST-SVM classified this image as a non-sunset because of the green plants on top of the bench. Normally, a sunset image does not have green in the sky (in the top half of the image), but since this image in Figure 15, has green plants in the top half of the image, this image is classified as a non-sunset.

Figure 15:nonsunset\4096288056_fb57a94967_z.jpg classified as a true negative with a distance of -0.0047136.

Figure 16 shows an image of a surfer with the ocean and an orange backdrop on the top of the image. The LST-SVM has falsely classified this image as a sunset because the top half portion of the image has warm colors, and the bottom half has darker colors.

Figure 16:nonsunset\3988986067_cc4d5d8094_z.jpg classified as a false positive with a distance of 0.65022.

Figure 17 shows an image of a flower in the foreground with more warm colored flowers in the background. The LST-SVM has probably detected the blur of the background flowers as streaks of clouds and the dark background as an ocean horizon or mountains. What makes this image interesting is that the flower in the foreground should make the image be detected as a non-sunset because of the many colors in the region.

Figure 17: nonsunset\4134856200_75f52720ec_z.jpg classified as a false positive with a distance of 0.00097362.

The tree in the upper quarter of the image in Figure 18 blocks the sky, making the LST-SVM classify the image as a non-sunset even with the warm colors in the center horizon of the image.

Figure 18: sunset\13639141365_cf7182808a_z.jpg classified as a false negative with a distance of -0.0030367.

The image in Figure 19 was falsely detected as a non-sunset because of the scattered trees in the portion of the image with the sky and the sun.

Figure 19: sunset\13122476943_9b44e86aa6_z.jpg classified as a false negative with a distance of -1.3203.

A test set accuracy of 89.6% is a questionably high number for the LST-SVM. This is due to the LST-SVM classifying an image as a sunset if the image has warm colors in the top half of the image and dark colors in the bottom half of the image. In addition, it may misclassify an image as a non-sunset if the image has scattered colors in the image, especially in the location of the sky or sun, or may classify an image as non-sunset if it has no warm colors in the image. In short, the LST-SVM memorizes the exact details of the training images and applies those specific details to the entire test dataset. Looking at the 15% FPR and the way the LST-SVM is generated, it makes sense that the LST-SVM is simply looking for regional details of the warm colors and may misclassify images as a sunset.

6b. CNN

Shown below are some images that were classified as true positive (Figure 20 and Figure 21), true negative (Figure 22 and Figure 23), false positive (Figure 24

and Figure 25), and false negative (Figure 26 and Figure 27).

Figure 20 is recognized as sunset even though the color is mainly cold colors. Figure 21 has few warm colors that are detected by the CNN as a sunset.

Figure 20: sunset\12574445733_fd40d5655a_z.jpg

Figure 21: sunset\1414426559_4be2a4877d_z.jpg

Figure 22 correctly classifies the surfer image as a non-sunset image mainly because the CNN was trained on disoriented images, so the warm sky and the ocean waves are differentiated. Figure 23 is clearly not a sunset and was probably easy for the CNN to classify the image based on the bright yellow color.

Figure 22:nonsunset\3970853125_1c359c85d0_z.jpg

Figure 23:nonsunset\3958305962_23d566fc50_z.jpg

Figure 24 and figure 25 are recognized as sunset even though they are non-sunset. It is because the two figures contained mainly warm colors which makes the system recognize them as sunsets.

Figure 24:nonsunset\4330503230_9dc8a144a_z.jpg

Figure 25:nonsunset\4295420598_57b781a27d_z.jpg

Figure 26 and figure 27 are recognized as non-sunset even though they are sunset. It recognized figure 26 as non-sunset because it did not contain the warm colors which makes the system detect the image as a non-sunset. Figure 27 was possibly classified as a non-

sunset because the portion of the purple sky is small compared to the roads and buildings.

Figure 26:sunset\13547361514_02e23c93d1_z.jpg

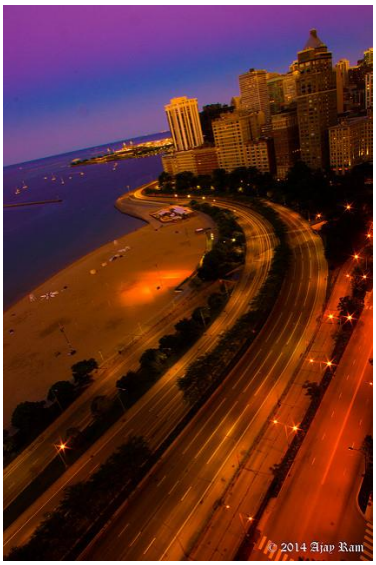

Figure 27: sunset\14400482557_d8c0b904f2_z.jpg

7. CONCLUSION & FUTURE WORK

There were many factors involved in classifying the sunsets. One of those was the number of features used, especially the number of regions the images were split into. If the region sizes were smaller, however, the standard deviation wouldn't be as large, affecting the features. Maybe using a larger database would also help since there might be more sunsets to classify and test on. The images themselves were at a set resolution and were not pixelated as one would see in any image, so this might have affected the overall feature matrix. If the pixels were larger in each segment, the standard deviation would also change. In addition to the physical features of the images, the LST-SVM would also overfit the test dataset. A significant error in a small segment can change the outcome of the classifier. In the future we would analyze the depth of the images using Hough transforms and the individual slopes of the lines could be calculated to find the depth towards the center of the image or where the sun is setting.

The convolutional neural networks reached 95% of accuracy but they can be improved by either adding more layers to the network, retraining the entire training dataset, or performing more data augmentations on the training set. Since the training dataset only contains 1600 images, by performing data augmentation and replicating, the size of the data would increase. Since the convolutional neural network will perform better on large datasets, the accuracy would increase.

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.

APPENDIX

Table 1: Confusion matrix for the validation set indicating the k scale and box constraint with whole number increments. The bolded rows show the jump in the false positive and true positive rate.

k	bc	TP	FP	FN	TN	FPR	TPR	ACC	Support Vector Ratio
1	1	129	11	171	289	0.036666667	0.43	0.696666667	0.91125
1	2	146	15	154	285	0.05	0.486666667	0.718333333	0.91875
1	3	146	15	154	285	0.05	0.486666667	0.718333333	0.92125
1	4	146	15	154	285	0.05	0.486666667	0.718333333	0.92125
1	5	146	15	154	285	0.05	0.486666667	0.718333333	0.92125
1	6	146	15	154	285	0.05	0.486666667	0.718333333	0.92125
1	7	146	15	154	285	0.05	0.486666667	0.718333333	0.92125
1	8	146	15	154	285	0.05	0.486666667	0.718333333	0.92125
1	9	146	15	154	285	0.05	0.486666667	0.718333333	0.92125
1	**10**	**146**	**15**	**154**	**285**	**0.05**	**0.486666667**	**0.718333333**	**0.92125**
2	**1**	**281**	**69**	**19**	**231**	**0.23**	**0.936666667**	**0.853333333**	**0.499375**
2	2	285	75	15	225	0.25	0.95	0.85	0.50375
3	1	288	79	12	221	0.263333333	0.96	0.848333333	0.41875
2	3	287	82	13	218	0.273333333	0.956666667	0.841666667	0.510625
4	1	287	82	13	218	0.273333333	0.956666667	0.841666667	0.41625

Table 2: Confusion matrix for the validation set indicating the k scale and box constraint. The bolded rows show the chosen best and the auto generated best, respectively.

k	bc	TP	FP	FN	TN	FPR	TPR	ACC	Support Vector Ratio
1	1.5	148	16	152	284	0.053333333	0.493333333	0.72	0.91625
1.1	1	178	23	122	277	0.076666667	0.593333333	0.758333333	0.856875
1.1	1.5	192	25	108	275	0.083333333	0.64	0.778333333	0.866875
1.1	2	190	25	110	275	0.083333333	0.633333333	0.775	0.870625
1.1	4	189	25	111	275	0.083333333	0.63	0.773333333	0.875625
….	….	….	….	….	….	….	….	….	….

1.2	1	214	30	86	270	0.1	0.713333333	0.806666667	0.8025
1.2	2.5	226	33	74	267	0.11	0.753333333	0.821666667	0.821875
1.2	3	225	33	75	267	0.11	0.75	0.82	0.824375
1.2	1.5	223	34	77	266	0.113333333	0.743333333	0.815	0.81625
1.2	3.5	225	34	75	266	0.113333333	0.75	0.818333333	0.825
1.2	4	225	34	75	266	0.113333333	0.75	0.818333333	0.825625
1.2	2	225	36	75	264	0.12	0.75	0.815	0.82
1.3	1	238	36	62	264	0.12	0.793333333	0.836666667	0.738125
1.4	**1**	**252**	**45**	**48**	**255**	**0.15**	**0.84**	**0.845**	**0.695**
….	….	….	….	….	….	….	….	….	….
2.8	**2.5**	**291**	**88**	**9**	**212**	**0.293333333**	**0.97**	**0.838333333**	**0.4**
….	….	….	….	….	….	….	….	….	….
2.7	4	293	101	7	199	0.336666667	0.976666667	0.82	0.413125

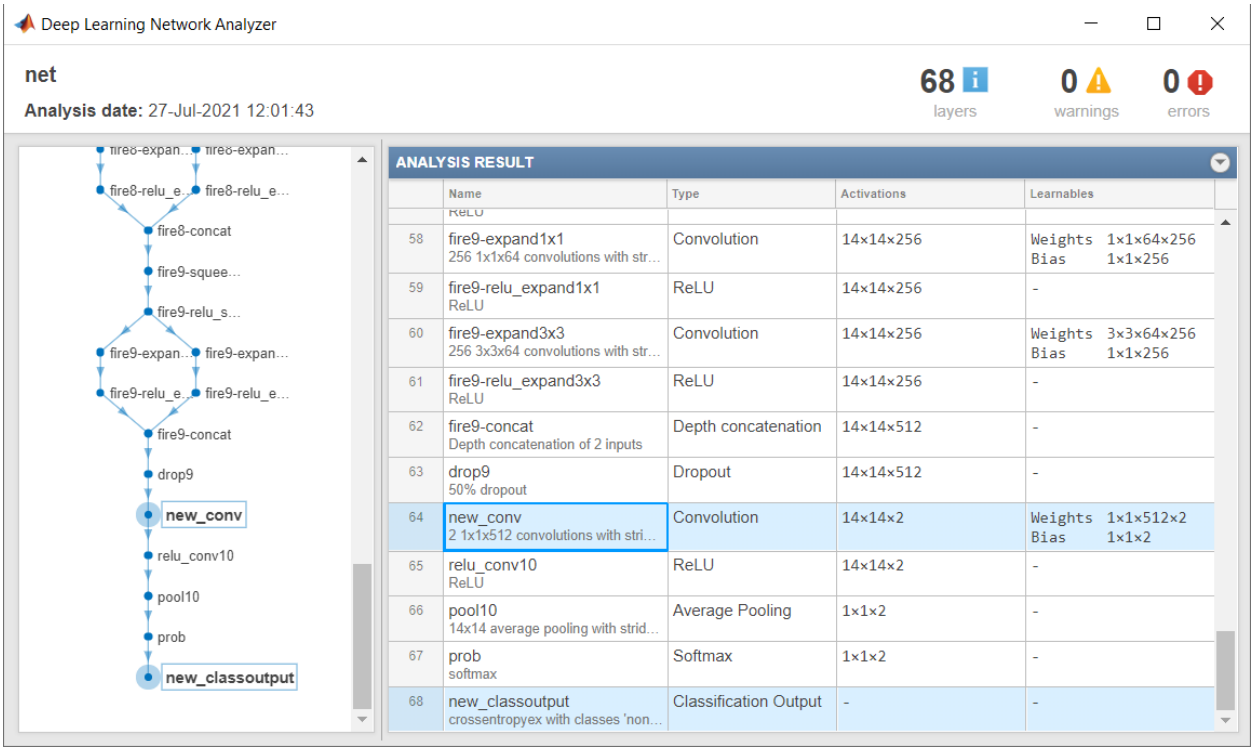

Figure 28: The last few layers of the architecture of the modified squeezenet. The highlighted layers are the layers that were modified from squeezenet's original convolution layer to the new_conv and original classification output to the new_classoutput.

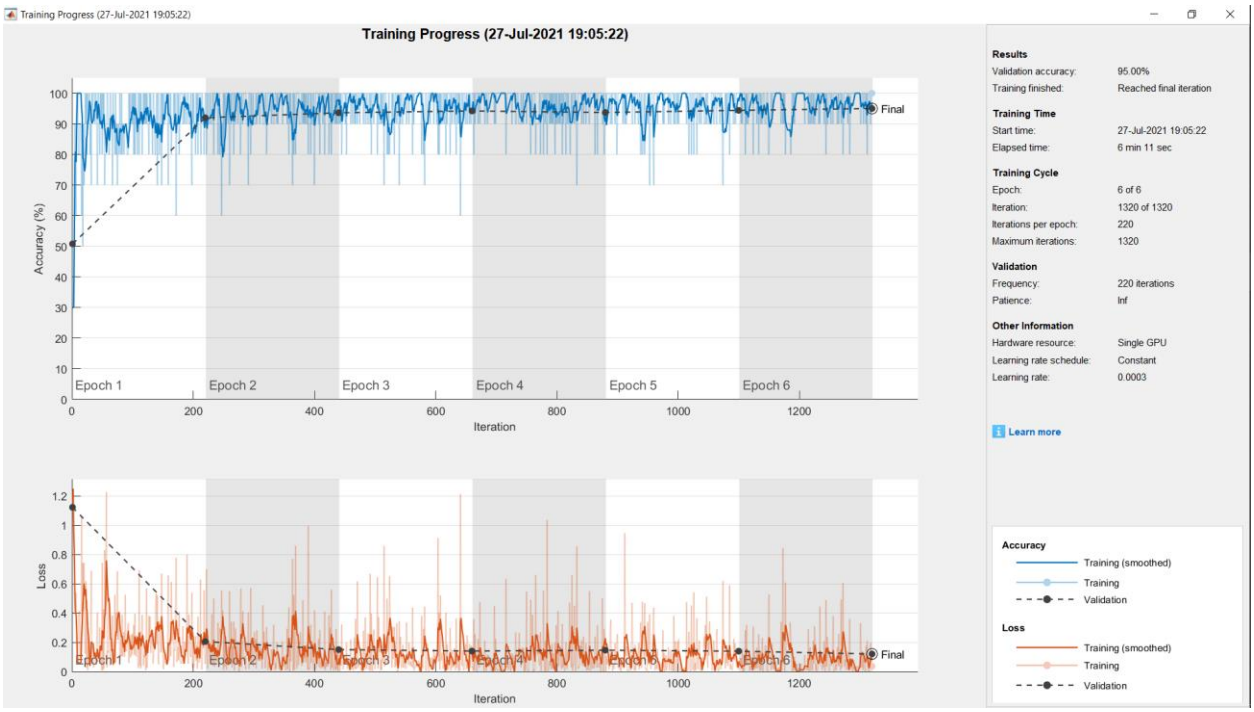

Figure 29: Respective transfer learning progress graph of accuracy and loss from Section 2.3.1. Number of iterations/Number of epochs: 1320/6, Batch size: 10, Learning rate: 0.0003, Training time (Single GPU): 6 minutes 11 seconds.

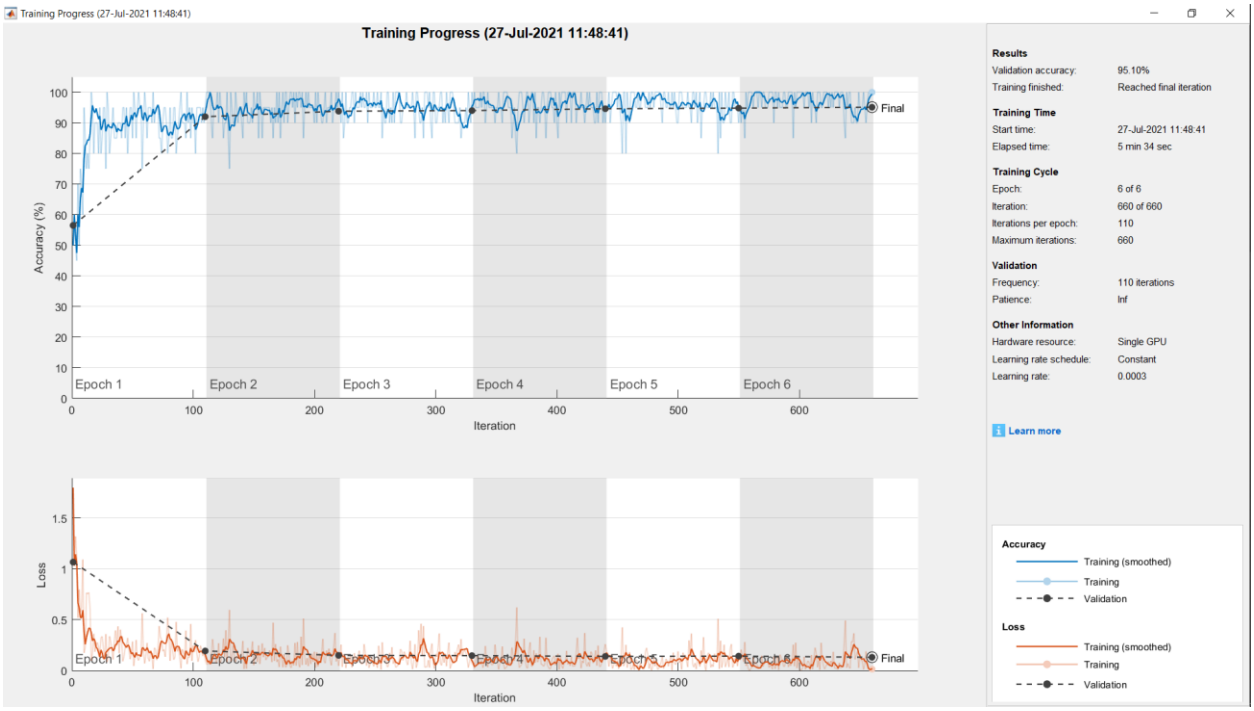

Figure 30: Respective transfer learning progress graph of accuracy and loss from Section 2.3.2. Number of iterations/Number of epochs: 660/6, Batch size: 20, Learning rate: 0.0003, Training time (Single GPU): 5 minutes 34 seconds.

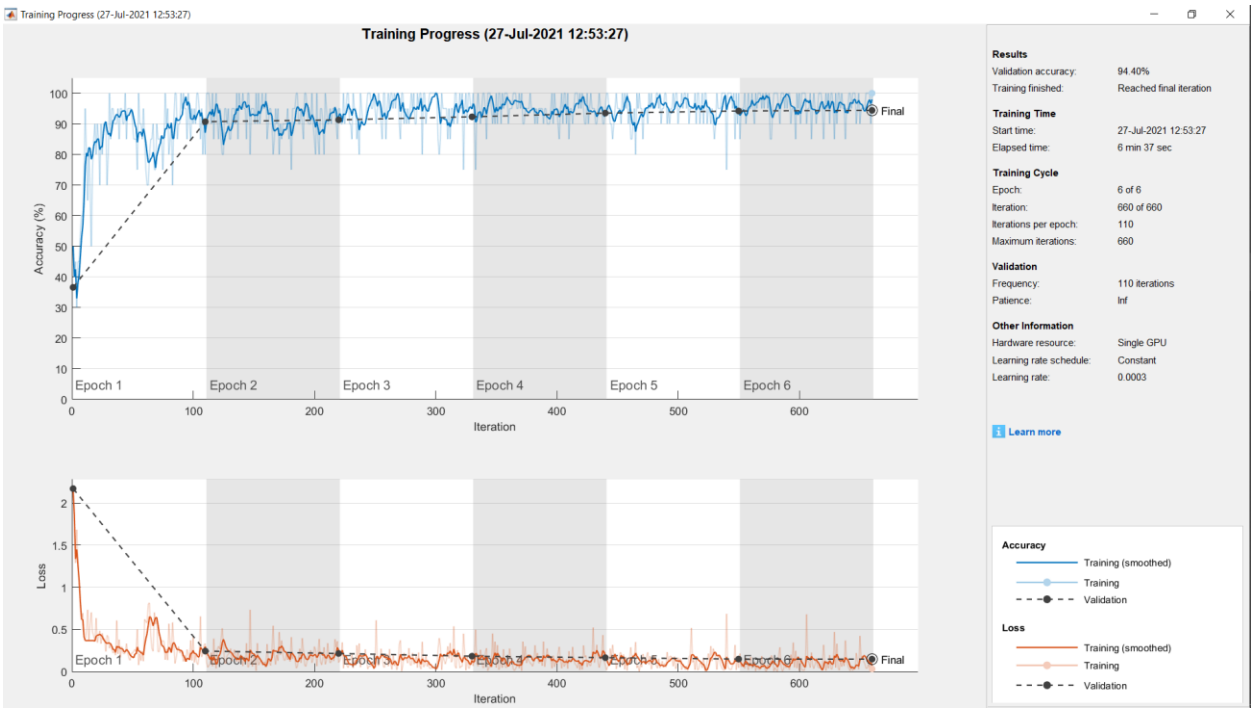

Figure 31: Respective transfer learning progress graph of accuracy and loss from Section 2.3.3. Number of iterations/Number of epochs: 660/6, Batch size: 20, Learning rate: 0.0003, Training time (Single GPU): 6 minutes 37 seconds.

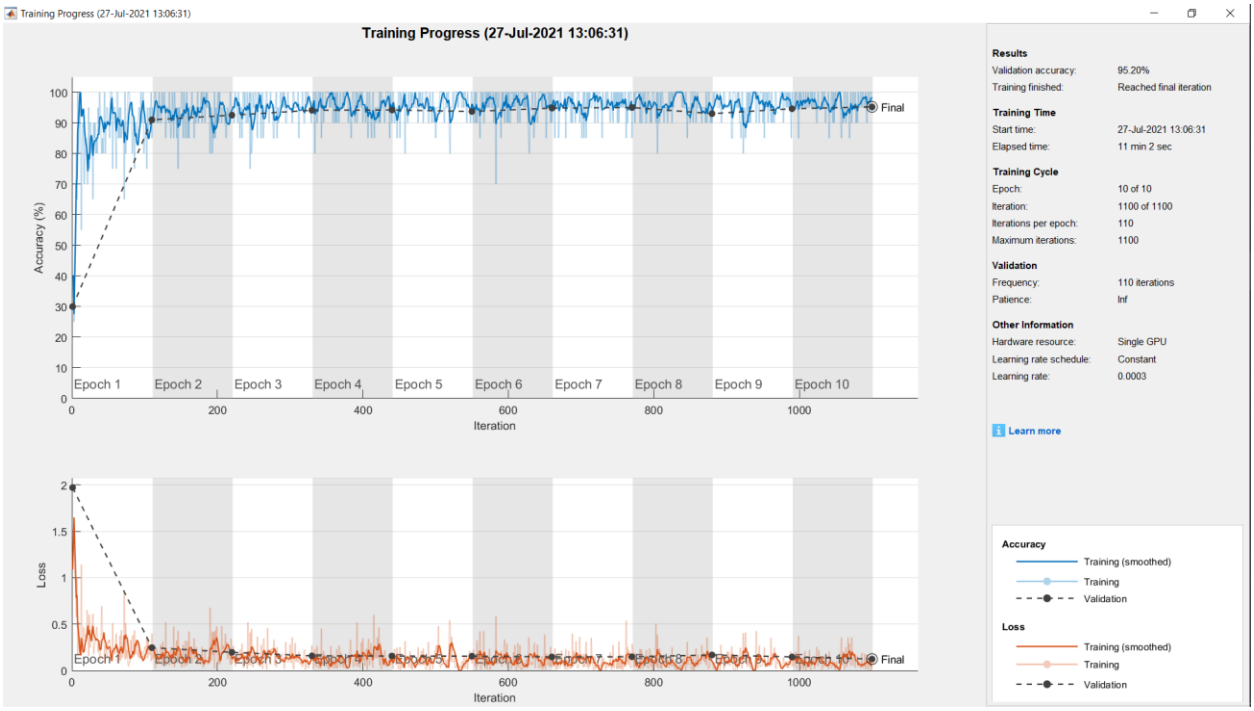

Figure 32: Respective transfer learning progress graph of accuracy and loss from Section 2.3.4. Number of iterations/Number of epochs: 1100/10, Batch size: 20, Learning rate: 0.0003, Training time (Single GPU): 11 minutes 2 seconds.

Iter	Eval result	Objective	Objective runtime	BestSoFar (observed)	BestSoFar (estim.)	BoxConstraint	KernelScale
1	Best	0.22938	2.7167	0.22938	0.22938	0.036271	22.793
2	Accept	0.49313	1.6052	0.22938	0.24765	0.0011098	0.0034167
3	Accept	0.48313	2.1975	0.22938	0.2294	19.293	0.072866
4	Best	0.165	0.80802	0.165	0.16502	109.58	209.82
5	Accept	0.23125	0.87319	0.165	0.17574	0.098698	999.68
6	Best	0.14063	0.50183	0.14063	0.14869	643.64	167.7
7	Best	0.135	0.54856	0.135	0.13498	989.13	154.97
8	Accept	0.13813	0.77433	0.135	0.13516	990.02	56.464
9	Accept	0.17438	0.68632	0.135	0.13518	968.95	959.06
10	Accept	0.13688	0.4647	0.135	0.13536	993.92	103.46
11	Accept	0.135	0.65575	0.135	0.13535	982.72	118.81
12	Accept	0.13563	0.57728	0.135	0.13527	999.74	121.89
13	Accept	0.13563	0.40721	0.135	0.13531	986.32	140.07
14	Best	0.13125	2.3214	0.13125	0.13135	987.88	15.173
15	Accept	0.14	1.3848	0.13125	0.13128	931.53	5.995
16	Accept	0.145	2.4183	0.13125	0.13571	993.6	11.33
17	Accept	0.13813	0.6297	0.13125	0.13576	962.46	66.513
18	Accept	0.48875	1.7136	0.13125	0.13582	978.9	0.001005
19	Accept	0.39563	0.98926	0.13125	0.13583	0.0010007	1.1176
20	Accept	0.23125	0.83379	0.13125	0.13588	0.001	130.17
Iter	Eval result	Objective	Objective runtime	BestSoFar (observed)	BestSoFar (estim.)	BoxConstraint	KernelScale
21	Accept	0.29563	1.1545	0.13125	0.13605	987.59	0.71034
22	Accept	0.48313	2.0247	0.13125	0.13588	994.83	0.009104
23	Best	0.10625	0.6358	0.10625	0.10632	2.8107	2.476
24	Accept	0.11375	0.7084	0.10625	0.10642	12.873	3.9276
25	Accept	0.1125	0.67611	0.10625	0.10646	19.923	2.4746
26	Accept	0.10875	0.67321	0.10625	0.10683	4.3367	2.9814
27	Accept	0.10875	0.66249	0.10625	0.10735	4.5757	2.3055
28	Accept	0.11	0.6399	0.10625	0.10817	4.4303	2.617
29	Accept	0.10813	0.59118	0.10625	0.10725	1.1571	2.6791
30	Accept	0.1325	0.43784	0.10625	0.10728	59.271	25.457

Figure 33: Iterations taken to choose the best kernel scale value and the best box constraint value for the LST-SVM.