ECE661 Fall 2024: Homework 7

TA: Rahul Deshmukh (deshmuk5@purdue.edu)
Due Date: Midnight, 28 Oct 2024
Late submissions will be accepted with penalty: -10 points per-late-day, up to 5 days.

Turn in typed solutions via BrightSpace. Additional instructions can be found at BrightSpace.

1 Theory Question

Give free rein to your imagination and conceive of a new texture detector. Obviously, you can draw ideas from the five different types of detectors covered in class: GLCM, LBP, Gabor Filter, Gram Matrix, and CN. (The last one stands for texture characterization with Channel Normalization parameters as presented in Section 6.2 of the "Texture and Color" tutorial. [1]) As you are thinking of new possibilities for texture characterization, do not forget the pyramid representation of an image. You could think of your own texture detector that captures information in some or all of the octaves of the pyramid.

For motivation, please note that there does not exist a single texture detector that works well for all possible texture types. Besides, all texture detectors have parameters that must be tuned for the images you are interested in. What that implies is that the last word has not yet been said on how best to characterize texture in images.

We are expecting an algorithmic description (with or without psuedo code and diagram) of your novel texture detector and your reasons as to why such a texture descriptor will perform well. Feel free to attach examples of texture images for which you expect your texture descriptor to perform particularly well. (If you are able to cobble together a prototype implementation and show its performance on some set of texture images, your instructor will be very impressed.)

2 Introduction

In this homework, you will first explore different ways of representing the textures in images. In Lecture 16, you have learned about using Local Binary Pattern (LBP) histograms as texture descriptors. And subsequently in Lecture 17, Prof. Kak has introduced the notion of style in the context of CNN-based style transfer. Note that the terms texture and style are used interchangeably in computer vision.

If you are looking at last year's solutions, note that those solutions are based on first converting color images to grayscale images and then applying the texture operators. For this year's solutions, we would like to use the Hue channel instead of grayscale images. That is, you will first need to convert the RBG images into the HSI representation and then separate out the hue channel for all downstream processing. You need to write your own implementation for RGB to HSI/V/L conversion. Feel free to consult the Watershed module in [1] for an example of such conversion.

For the programming tasks, you will be asked to implement both the LBP-based and the CNN-based texture descriptor extraction procedures. You will then use the resulting texture descriptors to train image classifiers and report their performances on the test data.

To summarize, the programming tasks in this homework include:

1. Implement your own routines for extracting LBP histograms as texture descriptors.
2. Given a pretrained CNN encoder, implement your own routines for extracting the Gram Matrix based texture descriptor.
3. For extra credits, also implement the channel normalization parameter based texture descriptor.
4. For each type of texture descriptor you implement, train a weather classifier and quantitatively demonstrate its performance on a test set.

3 Extracting Style from Convolutional Features

The term of neural style transfer is coined by Gatys et. al. in their famous paper [2]. The authors proposed a method that can separately extract the textural information, i.e. style, and the semantically meaningful structures, i.e. content, from the convolutional features produced by an encoder CNN. Subsequently, by rendering the content in the style of a style reference image in an image synthesis process, style transfer can be achieved.

What is particularly interesting to us is the style representation used in [2] and it is derived from the Gram matrix. The Gram matrix captures the inter-correlations among the individual feature maps associated with each convolutional layer of a neural network. It is computed as follows.

Let $\boldsymbol{F}^l$ denote the vectorized feature map of shape (N_l, M_l) at the l^{th} layer of an encoder CNN, where N_l is the number of channels and $M_l = W_l \times H_l$ is the number of spatial locations. The Gram matrix at layer l is simply calculated using the following matrix multiplication:

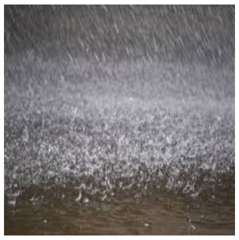
$$\boldsymbol{G}^l = \boldsymbol{F}^l \cdot \boldsymbol{F}^{l^T}, \tag{1}$$

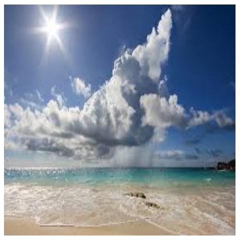
where $\boldsymbol{G}^l \in \mathbb{R}^{N_l \times N_l}$. Since it is symmetric, we need retain only the upper triangular part and vectorize it to be our Gram matrix based texture descriptor vector $\boldsymbol{v}_{gram} = triu(\boldsymbol{G}^l) \in \mathbb{R}^{N_l(N_l+1)/2}$.

4 Understanding the Dataset

For this homework you will be using the outdoor multi-class weather data [3]. This dataset has a little over 1000 images divided into four weather categories. The four categories are: cloudy, rain, shine and sunrise. The last 50 images from each category are used to create the test set for this homework.

(a) Cloudy


(b) Rain


(c) Shine

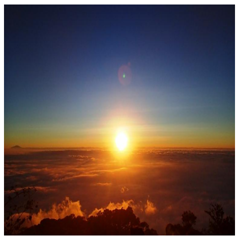

(d) Sunrise

Figure 1: Sample input images.

5 Programming Tasks

Now we ask you to implement an image classification framework that classifies images based on the texture descriptors discussed below.

5.1 Extracting Texture Descriptors

Local Binary Pattern

Implement **your own LBP descriptor** extraction algorithm to obtain a histogram feature vector for each image in the database. You can refer to Prof. Kak's implementation [1]. You have to use the Hue channel of the HSI images for this task. For visualization, you should plot the LBP histogram feature vector of at least one image from each class.

Gram Matrix

You are provided with the source code (`vgg_and_resnet.py`) for the following pretrained image encoding networks:

1. A customized **VGG-19**[4] based network. The customized network has been truncated to the `relu5_1` layer. This customized network takes in a RGB image and will give you an output feature map of size `[512, h = H/16, w = W/16]`.

2. A dual resolution **ResNet-50**[5] based network. The network has been truncated to the `conv3_x` layer of ResNet. This network takes in a RGB image and will give you a coarse and fine resolution feature map. The feature maps will be of sizes: Coarse feature map `[1024, h = H/16, w = W/16]` and Fine feature map `[512, h = H/8, w = W/8]`.

Using the output feature maps of the three configurations (VGG19, ResNet-50 Coarse, and ResNet-50 Fine) you need to implement **your own Gram matrix based descriptor** extraction routines.

For visualization, you should plot the 2D Gram Matrix for at least one image from each class for the three configurations. Comment on your Gram matrix visualizations. Are all feature channels strongly correlated?

5.2 Building An Image Classification Pipeline

Once you have implemented your own texture descriptor extraction routines, you can now build your own weather classification pipeline. Here are the recommended steps:

1. Preprocess all training and testing images by resizing them to (256, 256) for the sake of computational efficiency. Also you should make sure they all have three channels i.e. RGB.
2. Extract the feature vectors for all images. Together they form a feature matrix of size (S, C), where S is the number of samples/images and C is the dimension of a feature vector. Do this for all types of descriptor and both the training and testing images.
3. Train a SVM multi-class classifier to fit the training data. You are free to use open-source implementations from OpenCV or scikit-learn.
4. Apply your trained classifier on the testing feature matrix. Experiment with different choices of parameters (e.g. R and P in LBP). Record your best classification accuracy as well as the full confusion matrix.
5. Finally, for each trained classifier, display at least one correctly classified and one mis-classified image. Make sure that you indicate the predicted and ground-truth class labels (Cloudy/Rain/Shine/Sunrise) for each image.

5.3 Implementation Notes

1. For LBP, you should first convert your RGB images into grayscale. You should also further downsize your images to (64, 64) for more computationally feasible LBP calculations.
2. Since the number of channels in the VGG output that you have access to is $N_l = 512$. This means that your Gram matrix will be of size 512×512. Since, that makes for too many parameters for texture characterization, downsample the Gram matrix so that it is of size 32×32 and then dimensionality of the Gram descriptor will be $32(32 + 1)/2 = 528$.

3. Once you have extracted all the feature matrices, don't forget to save them to disk. You can use `numpy.savez_compressed` and load with `numpy.load`.
4. You can use the same `conda` environment previously set up for SuperGlue. You may need to install additional packages such as scikit-learn.
5. Your confusion matrix will be a 4×4 matrix since there are 4 classes. Rows correspond to the actual class labels while columns correspond to the predicted class labels. Note that a perfect confusion matrix will be a diagonal matrix with all diagonal values equal to 50 for our test set.

6 Extra Credits (20 points)

Another way of extracting style from the convolutional features is through the channel normalization parameters. This was first explored by Huang et. al. in [6] as they sought a more computationally lightweight approach to neural style transfer. They demonstrated that the channel normalization parameters, i.e. the per-channel means and variances, can sufficiently capture the style information in an image. Subsequently, by aligning the channel normalization parameters of a content image with those from a style image, style transfer can be achieved. This operation is known as the Adaptive Instance Normalization (AdaIN) and it has since played an important role in many image generation frameworks such as the famous StyleGAN [7].

For your implementation, given a feature map $\boldsymbol{F}^l$ of shape (N_l, M_l), the channel normalization parameters can be written as the following per-channel mean and variance values:

$$\begin{aligned} \mu_i^l &= \frac{1}{M_l} \sum_{k=0}^{M_l-1} x_{i,k}^l, \\ \sigma_i^l &= \sqrt{\frac{1}{M_l} \sum_{k=0}^{M_l-1} (x_{i,k}^l - \mu_i^l)^2}, \end{aligned} \tag{2}$$

where $x_{i,k}^l$ denotes the feature value at channel i and location k of the feature map $\boldsymbol{F}^l$. Subsequently, you can simply use the concatenation of the above statistics across all feature channels as your texture descriptor:

$$\boldsymbol{v}_{norm} = (\mu_0, \sigma_0,\ \mu_1, \sigma_1,\ \cdots\ ,\ \mu_{N_l}, \sigma_{N_l}) \in \mathbb{R}^{2N_l}. \tag{3}$$

For the extra credit programming task, implement **your own channel normalization parameter based descriptor** extraction routines. Again, you will be using the convolutional feature maps of VGG-19 and ResNet-50 as explained in Section 5.1

7 Submission Instructions

Include a typed report explaining how did you solve the given programming tasks.

1. Turn in a zipped file, it should include (a) a typed self-contained pdf report with source code and results and (b) source code files (only .py files are accepted). Rename your .zip file as hw7_<First Name><Last Name>.zip and follow the same file naming convention for your pdf report too.

   There should be two items in your submission - (1) Homework PDF (2) ZIP file containing source code (*.py files) and text files (if any).

2. **Submit only once on BrightSpace**. Otherwise, we cannot guarantee that your latest submission will be pulled for grading and will not accept related regrade requests.

3. Your pdf must include a description of

   - Your answer to the theoretical question in Section 1.
   - LBP histogram feature vector of at least one image from each class.
   - Gram matrix plots for at least one image from each class for all descriptors.
   - For all types of descriptor, your performance measures including classification accuracy and confusion matrix.
   - Your comments on the performance of different descriptors and provide reasons as to why one method works better than the other.
   - For all trained classifiers display correctly classified and misclassified image.
   - For extra credit, the classification accuracy and confusion matrix using the channel normalization parameter descriptor.
   - Your source code. Make sure that your source code files are adequately commented and cleaned up.

4. To help better provide feedbacks to you, make sure to **number your figures**.

5. The sample solutions from previous years are for reference only. **Your code and final report must be your own work.**

Homework File Size

For your homework submissions, please ensure that your reports (PDFs) are **under 10MB**.

Some ways to reduce your report sizes are:

- Downsample your input/output image when including in the reports.
- When including plots in your reports, save them as PDFs instead of PNG/JPEG. PDFs are vector formats which are lightweight and do not pixelate when zooming into the plot.

Strike a balance between the file size and image quality displayed in your report. This will help your TA in easy distribution of homeworks for grading without maxing out the disk space. Moreover, this will help you when you upload your homeworks to your individual git repos. You don't want to upload 40 Mb PDF to your git repo!

Finally, do not include the images in your ZIP files for your homework submissions. Ideally, the ZIP file should be under 1MB because it only contains ASCII (*.py /*.txt) files.

References

[1] "Texture and Color Tutorial." `https://engineering.purdue.edu/kak/Tutorials/TextureAndColor.pdf`. 1, 2, 3

[2] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2414–2423, 2016. 2

[3] Gbeminiyi Ajayi, "Multi-class Weather Dataset for Image Classification." `http://dx.doi.org/10.17632/4drtyfjtfy.1`. 3

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 3

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 4

[6] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE international conference on computer vision*, pp. 1501–1510, 2017. 5

[7] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019. 5