ECE661: Homework 7

Fall 2022
Due Date: 11:59pm, Nov 02, 2022

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

1 Theory Questions

1. The reading material for Lecture 16 presents three different approaches to characterizing the texture in an image: 1) using the Gray Lcale Co-Occurrence Matrix (GLCM); 2) with Local Binary Pattern (LBP) histograms; and 3) using a Gabor Filter Family. Explain succinctly the core ideas in each of these three methods for measuring texture in images. (You are not expected to write more than a dozen sentences on each).

2. With regard to representing color in images, answer Right or Wrong for the following questions and provide a brief justification for each (no more than two sentences):

   (a) RGB and HSI are just linear variants of each other.

   (b) The color space L*a*b* is a nonlinear model of color perception.

   (c) Measuring the true color of the surface of an object is made difficult by the spectral composition of the illumination.

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 term texture and style will be used interchangeably from this point on.

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 the 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 all the individual feature channels. It is computed as follows. Let $\boldsymbol{F}^l$ denote the vectorized feature map of shape (N_l, M_l) at the lth 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 can retain only the upper triangular part and vectorize it to be our Gram matrix based texture descriptor vector $\boldsymbol{v}_{gram} \in \mathbb{R}^{N_l^2}$.

4 Programming Tasks

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

4.1 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.

4.2 Extracting Texture Descriptors

4.2.1 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]. For visualization, you should plot the LBP histogram feature vector of at least one image from each class.

4.2.2 Gram Matrix

Implement **your own Gram matrix based descriptor** extraction routines. You will be provided with a pretrained VGG-19 based network as the encoder (`vgg.py`) and use its output features from the layer `relu5_1` to extract your Gram matrix descriptors. For visualization, you should plot the 2D Gram Matrix for at least one image from each class. Comment on your Gram matrix visualizations. Are all feature channels strongly correlated?

4.3 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 both 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.

4.4 Implementation Notes

1. For LBP, you should first convert your RGB images into grayscale. You should also further downsizing your images to (64, 64) for more computationally feasible LBP calculations.

2. Since the dimensionality of the original Gram descriptor can be very high, *e.g.* $\boldsymbol{v}_{gram} \in \mathbb{R}^{512^2}$ for feature maps with 512 channels, use a random subset of values (*e.g.* $C = 1024$) in $\boldsymbol{v}_{gram}$ as your Gram descriptor vector for this homework.

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.

5 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 [4] 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 [5].

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 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:

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

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 from the same layer of the VGG-19 based network.

6 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.

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 questions 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 both types of descriptor, your performance measures including classification accuracy and confusion matrix.
   - Your comments on the performance of different descriptors.
   - 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.**

References

[1] Texture and Color Tutorial. URL https://engineering.purdue.edu/kak/Tutorials/TextureAndColor.pdf.

[2] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

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

[4] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[5] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.