University of Massachusetts Boston, Department of Computer Science

Term Project Phase I

Apply Neural Network to Real World Application (Crater)

CS480/CS697 Big Data Analytics

Hamidreza Mohebbi, Wei Ding

1. Educational Goals

- Pre-processing the Mars surface images.
- Build a neural network model using NASA's remote sensing data.

2. Project Description

Impacts and cratering on solar system objects is a fundamental and continuing process. Recognizing and distinguishing impact craters is a difficult task even for expert observers, because of their varied sizes and vast presence.

The main objective of the first phase of the term project is to train a neural network for crater images. You may use the handwritten digits recognition neural network as an example. You can find all the required files of this term project on this github repository: https://github.com/mohebbihr/cs697_term_project_phaseI

The following shows the general steps for doing the phase I of this project, you can find more information about each step on the rest of this document.

a. Download the dataset in zip format. You can find it in *crater_dataset* folder.
b. Extract the dataset.
c. Install the OpenCV library. You can find instructions on OpenCV Library section.
d. Please read the Preprocessing section and follow its instructions to preprocess dataset images.
e. Develop your Neural Network Model
f. Train the model
g. Extract the results
h. Analysis the results and write the report.

3. Dataset

Our dataset contains the *images* and ground-truth (*gt*) folders. The images folder includes two images from the surface of Mars planet in the pgm format (*tile3_24.pgm*, *tile3_25.pgm*). The size of these images are 1,700 by 1,700 pixels. Also, there are two sub-folders named tile3_24 and tile3_25 on this folder which contains extracted crater and non-crater regions from the .pgm files. The crater subfolder contains the positive examples (craters) in the jpg format. The non-crater sub-folder represents non-crater regions extracted from the .pgm file. The negative example (non-crater) extracted randomly using a uniform distribution. The negative samples with low standard deviation (less than 5) and with significant overlapped area with craters (more than 25%) selected for this project. The number of negative examples used was approximately the double of the positive ones.

The ground-truth files contains the location information of craters on Mars images. A domain expert manually marked 399 craters for images. The *gt* folder includes the *gt_tile3_24.csv* and *gt_tile_3_25.csv* files that represents ground-truth information for *tile3_24.pgm* and *tile3_25.pgm* images, respectively. Each file represents the position (x,y) and radius (r) of craters. You can use the following sample code to read the csv file in python.

```
import csv

with open('file.csv') as csvDataFile:
    csvReader = csv.reader(csvDataFile)
    for row in csvReader:
        print(row)
```

4. OpenCV Library

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. The library is cross-platform and free for use under the open-source BSD license.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye

movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

We will use this library for image pre-processing in this project. You may follow below instructions to install this library.

a. If you have pip on your computer you may skip this step. Otherwise please follow the instructions on this link (https://pip.pypa.io/en/stable/installing/) to install it.
b. After pip installation, you can use the following command on your terminal (Mac and Linux) or Command Prompt (Windows) to install OpenCV libarary.

```
pip install opencv-python
```

5. Preprocessing

In this part of the assignment we need to normalize the size of the input pictures to feed into the neural network on next step. The neural network in the sample code designed from the fixed size input images (28 x 28 pixels image). The crater and non-crater images are in different size and you need to resize all these image into 200 by 200 pixels images. You may choose one of the following methods:

a) Padding: you can add something like a photo frame using the cv2.copyMakeBorder function. The following code shows this method.

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
BLUE = [255,0,0]

img1 = cv2.imread('opencv_logo.png')
constant=
cv.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_CONSTANT,value=BLUE)
plt.subplot(236),plt.imshow(constant,'gray'),plt.title('CONSTANT')
plt.show()
```

b) MINMAX: when the type of normalization (normType = NORM_MINMAX) opencv normalize the source image in such a way that minimum value of destination image is alpha (input parameter) and max value is beta (input parameter). Opencv does this using only scales and shifts (i.e. adding constants and multiplying by constants).

```
import cv2 as cv

img = cv.imread(path)
img = cv.resize(img, (800, 800))
cv.normalize(img, img, 0, 255, cv.NORM_MINMAX)

cv.imshow('dst_rt', img)
cv.waitKey(0)
cv.destroyAllWindows()
```

Please follow these instructions for this step:

a. Create a directory named *normalized_images* under *images* directory of the dataset to store the results of this step.
b. Create a two sub-directory named crater and non-creater under normalized_images directory and save the results of this step.
c. Please select one of the Padding or MINMAX methods for pre-processing.
d. Create a python script named pre-processing.py and implement the selected method on previous step. This script should read images from the *images/tile3_24/* directory and save the results under *images/normalized_images/* directory.

6. Load the dataset

   Please follow below instructions to load the dataset.

   a. Please read and understand the *mnist_loader.py* script that you can find under *src* directory of sample neural network.
   b. Please write a Python script named crater_loader.py which contains a function named *load_crater_data_wrapper*. This function reads normalized images (previous step output) and create a list of tuples contain image data and its label. Please assign label 1 for crater images and 0 for non-crater images. Also, this function must select images randomly and make training and test sets (for instance, 70% training, 30% test.) In this phase of the project, both training and test sets are built from the *tile3_24.pgm* image. However, the order of crater and non-crater images in training and test sets are different (You select this order, randomly). The format of the output should be similar to the load_data_wrapper function in *mnist_loader.py* script.

7. Develop and train the model

   In this part you need to design your neural network for crater dataset. Please follow these instructions:

a. Please read and understand the *network.py* script (inside *src* directory of sample neural network).
b. Using the *network.py* script as your base implementation. Please develop your own model and save it in script named *crater_network.py*. Remember that the number of neurons on the last layer of your network should be two. The structure of neural network, the number of hidden layers is your choice. You may develop several models with different structure and compare their results together on your report to receive bonus points.
c. The evaluate function in *crater_network.py* is different from *network.py* file. This function calculate the number of true positive, false positive and false negative craters and calculates the detection, false and quality rates over test set. The evaluate function must return TP, FP, FN, detection, false and quality rates.

$$detection_{rate} = \frac{TP}{TP + FN} \, , false_{rate} = \frac{FP}{TP + FP}$$

$$quality_{rate} = \frac{TP}{TP + FP + FN}$$

d. Write a python script named *run_experiment.py* which its functionality is to load the dataset, create the crater network, train your model on the crater dataset, test it on test data and extract the results. This script must have a loop that iterate at least 10 times and repeat the above steps and finally report the average of TP, FP, FN, detection, false, and quality rates. Please draw the change of these measure over iterations as plot and put it on your report.

8. Evaluation

Please evaluate and analysis the results of the previous step and answer the following questions on your report.

a. Please determine which craters are difficult to recognize based on your analysis and try to explain it.
b. Please propose the ways to improve the problem that you mentioned on previous question.