Milestone Report

1. Problem statement

Classification of objects in images is one of the core problems in Computer Vision. I'm going to assign an input image one label from a fixed set of categories.

The client can be companies who need to handle with large amounts of images. With image recognition, companies can easily organize their database because it allows for automatic classification of images in large quantities. This will help them monetize their visual content without investing countless hours for manual sorting and tagging.

2. Description of the dataset

The dataset for this project comes from https://www.kaggle.com/prasunroy/natural-images. The data consists of 6899 images from 8 distinct classes compiled from various sources. The classes include airplane, car, cat, dog, flower, fruit, motorbike, and person. But the images of different categories have different resolutions. In this part, I only used 100 images of car and 100 images of fruit to do classification, because their resolutions are the same (100x100).

3. Preprocess image data

All the images are put in the folder C:\Users\ww6848\Documents\Springboard\capstone project 2\natural_images\test. The images include two categories: car and fruit. All the titles for the images are read from a text file, for the category of car, the titles are from car_0000.jpg to car_0099.jpg; for the category of

fruit, the titles are from fruit_0000.jpg to fruit_0099.jpg. All the image data are read through imread function using cv2.

4. Deep learning method

Convolutional Neural Networks are similar to ordinary Neural Networks: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function on the last (fully-connected) layer. Convolutional neural networks make the explicit assumption that the inputs are images. Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. Here, the input images have dimensions 100x100x3. All the images are split into training images and testing images with test_size=0.3. For 200 images, 140 are for training, 60 are for testing.

Three main types of layers are used to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer. They will be stacked to form a full ConvNet architecture. The model has the architecture of [INPUT-CONV-RELU-CONV-RELU-POOL-FC]:

INPUT [100X100X3] will hold the raw pixel values of the image, in this case an image of width 100, height 100, and with three color channels R, G, B.

CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume.

RELU layer will apply an elementwise activation function, such as the max(0,x).

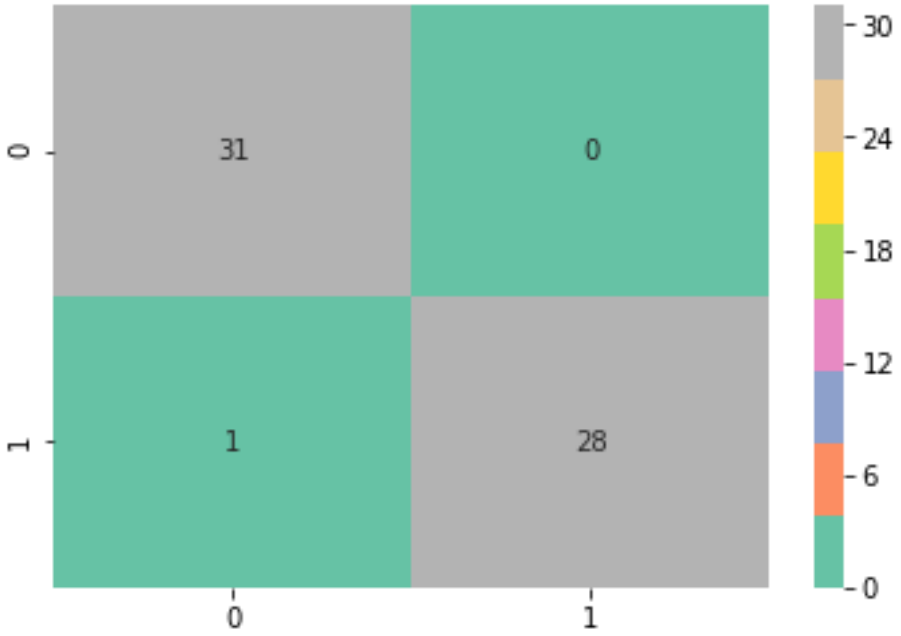
POOL layer will perform a downsampling operation along the spatial dimensions (width, height).

FC layer will compute the class scores, resulting in volume of size [1x1x2], where each of the 2 numbers correspond to a class score, among the 2 categories. As the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

In this way, ConvNets transform the original image layer by layer from the original pixel values to the final class scores.

After 100 epochs, the test accuracy reaches 0.98.

The heat map is shown as follows:


31
0
1
28
0
1
30
24
18
12
6
0