CS 512 - Assignment 4

Due by 10/30/2018

In this assignment you need to implement a basic Convolutional Neural Network for classification. Your implementation needs to use a GPU framework. Specifically, Keras or TensorFlow. If you do not have access to GPU on your computer or elsewhere you can create a free account on google colab: https://colab.research.google.com/notebooks/welcome.ipynb

Task

You have learned how to build a CNN using either Keras or TensorFlow. You will now train and evaluate a CNN to classify images of numbers in the MNIST dataset as either even or odd.

Note, images in the MNIST dataset are labeled by the corresponding integer. For example, a handwritten image of the digit '7' is labeled as the integer 7. Hence, you will have to map the integer labeling (10 classes) of the images to a binary labeling (even/odd).

Make sure that your program can save and load weights so that training can proceed with previous results. You will be required to log and plot some metrics as a function of training and evaluation iterations.

Deliverables 1: Custom CNN

1. Train your CNN using the following configuration:
    a. Use the MNIST train set: 55,000 samples
    b. 2 Convolutional and Pooling layers
        i. Layer 1: 32 filters of kernel size 5x5
        ii. Layer 2: 64 filters of kernel size 5x5
        iii. Pooling in both layers should downsample by a factor of 2
    c. Dropout rate of 40%
    d. Compute cross entropy loss
    e. Use gradient descent optimization and a learning rate of 0.001
    f. Train for 5 epochs
2. Report the training loss and accuracy
    a. As your model trains log the loss and accuracy for each step/iteration and plot the curve
    b. Also report the loss and accuracy value of the final training step

3. Report the evaluation loss, accuracy, precision, and recall using MNIST test set (10,000 samples)
    a. For each epoch trained, compute the above-listed metrics and plot the curve
    b. Also report the loss, accuracy, recall, and precision of the last epoch

Deliverable 2: Parameter Tuning

Evaluate different variations of the basic network as described below and measure performance. In your report, discuss your variations, compare the results you obtain and attempt to draw conclusions. You may evaluate the following aspects

- Changing the network architecture (e.g. number of layers and/or organization of layers)
- Changing the receptive field and stride parameters
- Changing optimizer and loss function (e.g. Adam optimizer)
- Changing various parameters (e.g. dropout, learning rate, number of filters, number of epochs)
- Adding batch and layer normalization
- Using different weight initializers (e.g. Xavier, He)
- Using features from a pretrained model (e.g. VGG16)

Deliverable 3: Application

Write a program to use your pretrained custom CNN (Deliverable 1). The program should do the following:

1. Accept as input an image of a handwritten digit. Assume each image contains one digit
2. Using OpenCV do some basic image preprocessing to prepare the image for your CNN
    a. Resize the image to fit your model's image size requirement
    b. Transform grayscale image to a binary image. Consider using the *GaussianBlur()* and *adaptiveThreshold()*, or any other type of binary thresholding that performs well.
    c. Display the original image and binary image in two separate windows.
3. Using your CNN classify the binary image.
4. Your program should continuously request the path to an image file, process the image, and output to the console that class (even/odd) of the image.
5. Program should terminate when 'q' or ESC key is entered.