BME69500DL ECE695DL
Spring 2020

Homework 5

Deadline : Tuesday, April 7, 2020 , 11:00 am

1 Introduction

The main goal of this homework is to gain insights into the noise sensitivity of object detection with convolutional networks.

This homework requires a higher level of initiative and ingenuity on your part than has previously been the case.

Before embarking on this homework, do the following:

1. Review the Week 8 slides on "Object Detection and Localization with Deep Networks"
2. Download and install Version 1.1.0 of DLStudio. Note that if you '*pip install*' it directly from the 'pypi.org' website, you will not get the changes in the Examples directory of the distribution.
3. The new datasets that come with Version 1.1.0 of DLStudio add up to over 100MB in size, which is beyond the file-size limit at 'pypi.og'. So you will need to download those directly from the main API page for DLStudio. Download the main dataset archive and store it in the 'Examples' directory of the distribution. Subsequently, execute the following command in the Examples directory:

   ```
   tar xvf datasets_for_DLStudio.tar.gz
   ```

   This will create a subdirectory 'data' in the Examples directory and deposit all the datasets in it.
4. If you followed the previous instructions, you will find the following datasets in the 'Examples/data/' directory:

   ```
   PurdueShapes5-10000-train-noise-20.gz
   PurdueShapes5-10000-train-noise-50.gz
   PurdueShapes5-10000-train-noise-80.gz
   PurdueShapes5-1000-test-noise-20.gz
   PurdueShapes5-1000-test-noise-50.gz
   PurdueShapes5-1000-test-noise-80.gz
   ```

   Regarding the naming convention used for the archives, the numbers 20, 50, and 80 stand for the level of noise in the images. For example, 20 means 20% noise. The percentage level indicates the fraction of the color value range that is added as randomly generated noise to the pixels.

   The numbers 10000 and 1000 you see in the names of the files are for the number of images in each archive.
5. The six archives listed above are in addition to the following I used in the Week 8 Lecture PDF:

   ```
   PurdueShapes5-10000-train.gz
   PurdueShapes5-1000-test.gz
   PurdueShapes5-20-train.gz
   PurdueShapes5-20-test.gz
   ```

These are for the noiseless case (although, even here you will see the objects with significant distortion in how they appear in the images. There is just no randomly generated noise added to the images in these four archives.).

The archives with just 20 images are to make it easier for you to debug your code quickly.

6. After you have installed the datasets, it's time to become familiar with the actual code for object detection and localization. Do this by executing the following script in the `Examples` directory of the distribution:

   ```
   object_detection_and_localization.py
   ```

plots showing up on colab

   Make sure that you are running it for the training dataset `PurdueShapes5-10000-train.gz` and the testing dataset `PurdueShapes5-1000-test.gz`. Your results should be very similar to those shown in the slides for Week 8.

7. Next execute the following script in the `Examples` directory:

   ```
   noisy_object_detection_and_localization.py
   ```

   Start by running this script with the following training and testing datasets:

   ```
   PurdueShapes5-10000-train-noise-20.gz
   PurdueShapes5-1000-test-noise-20.gz
   ```

got 91 and 83

   You will notice that with 2 epochs of training, compared to the noiseless case, your classification accuracy will go down from 91% to around 83% and regression error will increase from less than a pixel to roughly 3 pixels.

8. Do the same thing as described in the previous step but with the following training and testing datasets:

   ```
   PurdueShapes5-10000-train-noise-50.gz
   PurdueShapes5-1000-test-noise-50.gz
   ```

   What do you make of the results you see?

   With that, you are ready to start working on the homework described in what follows.

2 Tasks

Task 1: As you will see from the results you get in the last step mentioned above, the LOADnet2 CNN fails to do any learning at all when the level of noise is at 50%. Nevertheless, when you look at the sample images displayed on your screen by the training program as it iterates through the data, it is not too difficult for a human to make out the shapes despite the presence of significant noise in the images.

To me that indicates that the neural network as defined in the LOADnet2 class simply does not have the power to get the job done when the noise level is significantly greater that 20%.

However, since the objects have simple shapes and also since they are quite discernible to the human eye even in the presence of noise, one would think that you need to either augment LOADnet2, or to perhaps preprocess the images to suppress the noise before they are processed by the CNN.

So the first task is for you to take stock of the situation so that you can make better decisions in the tasks that follow.

Task 2: Since the objects in the images remain discernible to the human eye, it is possible that you would get better results with LOADnet2 if you modified the dataloader class and inserted a smoothing step right at the beginning. Try different degrees of smoothing and see what happens to the results. Report your results that you get with different smoothing operators.

Task 3: The goal of this task is to explore the possibility that you could have a neural network learn the degree of noise in the images and that information could then be used to drive the downstream DL logic.

To elaborate, you have at your disposal 40,000 training images with noise levels of 0%, 20%, 50%, and 80%. Think of the noise level as a classification label. You could therefore devise a simple neural network that would correctly estimate the noise-level label of each image. Subsequently, this label could be used for deciding what CNN logic would be the most appropriate for each image.

Give this approach a try and report your results.

Task 4: It is possible that you will get better results for the 50% noise-level images by augmenting the logic in the LOADnet2 network. Give it a try, see what happens, and report your results.

3 Submission

- Make sure to submit your code in Python 3.x and not Python 2.x.
- Name your main Python file as hw05.py
- All homework to be submitted on-line through the *turnin* command on the min.ecn.purdue.edu[1][2] server.
- Log into min.ecn.purdue.edu server using your Purdue career account.
- From min.ecn.purdue.edu, go to the directory where your homework files are. e.g., if your files are at /home/hw1 directory then go to /home/hw1. Obviously, if you don't have your files on the server, move them there using, for example, the *scp* command.
- Type in the following command:

```
turnin -c ecedl -p hw<double digits hw#> <your files separated by a space>
```

- As an example, for this homework you will enter the command:

```
turnin -c ecedl -p hw05 hw05.py output.txt
```

- You should get a statement that says "Your files have been submitted to ecedl, hw05 for grading". You can verify your submission by typing:

```
turnin -c ecedl -p hw<double digits homework number> -v .
```

[1] If you are an undergraduate student, use the shay.ecn.purdue.edu server instead of the **min** server
[2] If you registered this class as the BME695 course then use the weldon.ecn.purdue.edu server