CSE474/574: Introduction to Machine Learning(Fall 2017)

Instructor: Sargur N. Srihari
Teaching Assistants: Jun Chu, Tianhang Zheng, Mengdi Huai

Project 4: Introduction to Deep Learning

Due Date: Code, Report (Softcopy, Hardcopy) Wed, Dec 6, 2017 (11:59pm)

1 Overview

In this assignment, your task is to implement a convolution neural network to determine whether the person in a portrait image is wearing glasses or not. We will use the Celeb dataset which has more than 200k celebrity images in total. In this task you are to use a publicly available convolutional neural network package, train it on the Celeb images, tune hyperparameters and apply regularization to improve performance.

1.1 CelebFaces Attributes Dataset (CelebA)

The CelebFaces Attributes Dataset (CelebA)[1] is a large-scale face attributes dataset with 202,599 celebrity images, each in color with an original definition of 178×218 as in Figure 1.

You can download the dataset from http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html. After downloading the data files, we will use the "Eyeglasses" column as labels in the file Anno/list_attr_celeba.txt, which indicate whether the person in the picture is wearing glasses or not. As model inputs, you can find the original celebrity images in the img_align_celeba directory.

1.2 Convolutional Neural Network

To implement the convolutional neural networks, you could use existing publicly available packages such as Tensorflow, Caffe2, PyTorch and so on.

1.3 Evaluation

Evaluate the accuracy of CNNs on the CelebA data set (test data only), to distinguish between two classes - wearing glasses and not wearing glasses. Evaluate your solution on a separate validation dataset using the classification error rate

$$E = \frac{N_{\text{wrong}}}{N_{\text{V}}},$$

Figure 1: Overview of the Celeb dataset

where N_{wrong} is the number of misclassification and N_{V} is the size of the validation dataset. Under the 1-of-K coding scheme, the input will be classified as

$$C = \arg\max_i y_i.$$

2 Plan of Work

The following steps are the rough guidance and suggestions that you can follow to achieve better evaluation performance on the problem. Some of the parameters in the steps are not pre-determined (such as the resolutions you choose and the sizes of the dataset) so that certain variations should appear among individuals/groups. We do not expect to see those working in different groups to pick up exact same values.

Achieving all following steps requires a lot of work and is not required. You can choose some part of it. As always, the more complete the experiments and project report are, the more preferable it is.

1. **Extract feature values and labels from the data**: Download the CelebA dataset from the Internet and process the original data file into Numpy arrays that contains the feature vectors and a Numpy array that contains the labels.

2. **Reduce the resolution of the original images**

3. **Reduce the size of training set.** For example, take out a small part of the entire set for training.

4. **Data Partition**: Partition the picked-out CelebA dataset into a training set and a testing set. You will use this partition and train your model on the training set.
5. **Apply dropout or other regularization methods.**
6. **Train model parameter**: For a given group of hyper-parameters such as dropout rates, the number of layers and the number of nodes in each layer, train the model parameters on the training set.
7. **Tune hyper-parameters**: Validate the classfication performance of your model on the validation set. Change your hyper-parameters and repeat the previous. Try to find what values those hyper-parameters should take to give better performance on the testing set.
8. **Retrain the model using higher resolutions.** Does that improve the performance?
9. **Use bigger sizes of the training set.** Does that improve the performance?
10. **Use even bigger sizes of the training set by data augmentation.** Does that improve the performance?

3 Deliverables

There are two parts in your submission:

1. Report

   The report describes your implementations and results using graphs, tables, etc. Write a concise project report, which includes a description of how you implement the models and tune the parameters. Your report should be edited in PDF format. Additional grading considerations will include the performance of the training, creativity in paramter tuning and the clarity and flow of your report. Highlight the innovative parts and do not include what is already in the project description. You should also include the printed out results from your code in your report.

   Submission:

   Submit the PDF on a CSE student server with the following script:

   `submit_cse474 proj4.pdf` for undergraduates.

   `submit_cse574 proj4.pdf` for graduates.

   In addition to the PDF version of the report, you also need to hand in the hard copy version by the due date or else your project **will not be graded**.

2. Code

   The code for your implementations. Code in Python is the only accepted one for this project. You can submit multiple files, but the name of the entrance file should be main.py. All Python code files should be packed in a ZIP file named `proj4code.zip`. After extracting the ZIP file and executing command `python main.py` in the first level directory, it should be able to generate all the results and plots you used in your report and print them out in a clear manner.

   Submission:

   Submit the Python code on a CSE student server with the following script:

`submit_cse474 proj4code.zip` for undergraduates.

`submit_cse574 proj4code.zip` for graduates.

4 Due Date and Time

The due date is Wed, Dec 6, 2017 (11:59pm) for both online submission and hardcopy submission.

Appendix 1 CNN

For the convolutional neural network, you can use packages from online. Therefore, in this project description, we will not go into detail of the implementation. This could be done using several open source deep learning libraries. Here we provide a few examples:

1. Tensorflow: https://www.tensorflow.org/tutorials/. The "Convolutional Neural Networks" section.
2. Caffe2: https://caffe2.ai/docs/tutorials/. The "Convolutional Neural Networks" example part.
3. PyTorch: http://pytorch.org/tutorials/. The "Convolutional Neural Networks" example part.

Appendix 2 Dropout

Dropout is a regularization technique for reducing overfitting in neural networks. We can effectively remove units by multiplying its output value by zero. This is implemented almost in all the aforementioned publicly available libraries' tutorial examples. You can simply follow their implementations.

Appendix 3 Data Augmentation

The best way to make a ML model to generalize better is to train it on more data. However, in practice, the amount of data is limited. We can get around the problem by creating fake data. For some ML tasks it is straightforward to create fake data. We can generate new samples just by transforming inputs. The following are some examples:

1. Translating the images a few pixels.
2. Rotating, horizontal reflecting and scaling.
3. Injecting noise into the input of a neural network.
4. Random crops on the original images.

In order to save disk or memory space, you can implement on-the-fly augmentation, which means generating augmented images during mini-batch gradient descent and disgard them right after they are used.

References

[1] Liu, Ziwei; Luo, Ping; Wang, Xiaogang; Tang, Xiaoou. "Deep Learning Face Attributes in the Wild", Proceedings of International Conference on Computer Vision (ICCV) (2015).