CSE 252C Assignment 2
Advanced Computer Vision
Spring 2019
Due Fri May 24, 4pm PST

Instructions :

- Attempt all questions.
- Please comment all your code adequately.
- Turn-in a PDF for the written report and a zip-folder for the code. The report should be called `LastName_FirstName.pdf` and similarly the zip folder should be `LastName_FirstName.zip`.
- Include all relevant information such as text answers, output images and main code snippets in the PDF.
- Include the following code in the zip file.
    1. `casia_train.py` and `faceNet.py` for Section C (the version that achieves best accuracy).
    2. `casia_train.py` and `faceNet.py` for Section D (the version that achieves best accuracy).
- Submit your homework by Gradescope. Remember to correctly select pages for each answer on Gradescope to allow proper grading.

In this homework, we will study face recognition, covering face verification, face alignment and training of face recognition networks.

A Using SphereFace [2] for Face Verification

In the first section, we will test a pretrained model of SphereFace on LFW [1] dataset.

1. Download the LFW dataset, which is provided on the course website in a `.zip` folder called `LFW.zip`. The dataset contains 6000 pairs of human face images with ground truth labels for whether they are from the same identity. Unzip the folder. **[0 points]**
2. Download the PyTorch code of SphereFace. The code is provided on course website as `sphereFace.zip`, which is modified based on the open source code from `https://github.com/clcarwin/sphereface_pytorch`. Unzip the file. **[0 points]**

Next answer the following questions:

3. Run `lfw_eval.py` and report the accuracy of SphereFace on LFW verification. **[5 points]**
4. Explain briefly how the following steps are performed when evaluating on LFW dataset:

(a) Given the features extracted from the network, what distance metric is used to measure the distance between two faces? (`lfw_eval.py`: Line 135) **[5 points]**

(b) How to set up the threshold to determine whether two faces are from the same human? How to compute the accuracy? (`lfw_eval.py`: Line 141 to 148) **[10 points]**

5. An important step before face recognition is face alignment, in which we warp and crop the image based on the location of facial landmarks.

   (a) Briefly describe how we warp and crop the image. (`lfw_eval.py`: Line 11-26) **[10 points]**

   (b) Instead of doing face alignment, crop an image patch of height 112 pixels and width 96 pixels at the center of the image and report the accuracy. **[5 points]**

B Using MTCNN for Detecting Face Landmarks

Instead of using provided facial landmarks, we will now use MTCNN [6] for detecting them.

1. Download the code folder `MTCNN.zip` from the course website, which is modified from `https://github.com/TropComplique/mtcnn-pytorch`. Unzip the folder under the same directory as the LFW dataset and SphereFace code. Run `lfw_landmark.py` to generate the facial landmarks. Include two example outputs in your report. **[5 points]**
2. Go to the SphereCode directory, run `lfw_eval.py` again by setting flag `alignmentMode` to be 2. Report the error using the predicted facial landmarks. **[5 points]**

Next, answer the following questions:

3. Is the result better than using the landmarks provided in the previous question? If not, how can you improve performance? **[10 points]**
4. What are the steps adopted by the method to achieve real-time speed? **[10 points]**
5. Briefly describe how non-maximal suppression (NMS) is implemented in this method. (`src/box_utils.py`: Line 5-68) **[5 points]**

C Training SphereFace [2] on CASIA Dataset [5]

You will now train a network on the CASIA dataset [5] and test on the LFW dataset [1]. In this section, the skeleton code for training is given.

1. Download the `CASIA-Webface.zip` dataset linked from the course webpage. Unzip the dataset under the same directory as the SphereFace code. **[0 points]**
2. Go to directory of SphereCode and open `faceNet.py`. Under class `CustomLienar` implement function ψ as shown in Eq. (7) in [2], which is as follows.

$$\psi(\theta) = (-1)^k \cos(m\theta) - 2k, \quad \theta \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}]$$

Under class `CustomLoss`, implement the loss function. **[15 points]**

3. Train the network using `casia_train.py`. You may try various hyperparameters like m, learning rate, batch size, training iterations and so on. Stop when you think the network behaves strangely (drop in accuracy, or loss stops decreasing). You may refer (and cite) any open source code (for example, `https://github.com/clcarwin/sphereface_pytorch`). Include the following in your report:

   (a) Curves for training loss and accuracy on CASIA, which have been saved in the `checkpoint` directory (you may smooth the curves to make them look better). **[10 points]**

   (b) Accuracy on the LFW dataset, evaluated using `lfw_eval.py`. You are expected to achieve accuracy higher than 90% on the LFW dataset. **[10 points]**

4. The architecture above is a 20-layer residual network as described in Table 2 of [2], but without batch normalization. Now add batch normalization after every convolutional and fully connected layer. Train the new network on CASIA dataset and test on LFW dataset. Following is a demonstration of a residual block with 128 filters and kernel size 3×3:

$$\begin{aligned} y &= \texttt{CONV}_{3\times3,128}(x) \\ y &= \texttt{BatchNorm}(y) \\ y &= \texttt{PReLU}(y) \\ y &= \texttt{CONV}_{3\times3,128}(y) \\ y &= \texttt{BatchNorm}(y) \\ y &= \texttt{PReLU}(y) \\ \texttt{OUT} &= x + y \end{aligned}$$

   Answer the questions below:

   (a) Draw the training curves for accuracy and loss on CASIA and compare to the curve without batch normalization. **[10 points]**

   (b) Report accuracy on the LFW dataset, evaluated using `lfw_eval.py`. **[10 points]**

   (c) Do you achieve better performance on LFW? If yes, explain how batch normalization helps. If not, try to explain why the results are worse. **[10 points]**

5. Randomly choose 10 identities from the CASIA dataset, forward pass all their images through the network and visualize the normalized features using tSNE [3]. You can use code from `https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html` for visualization. Try a few random samples and include the figure that you consider most illustrative of the method. Retain the identities chosen here since they will be used again for the next question. **[10 points]**

D Training CosFace [4] on CASIA Dataset [5]

In this section, you are required to implement CosFace [4] based on the code you use in the previous section, train it on CASIA dataset and test on LFW dataset.

1. Download `cosFace.zip` from the course webstie. Unzip the file. **[0 points]**
2. Again implement the function ψ and the loss function of CosFace under `CustomLinear` and `CustomLoss` in `faceNet.py`. You may check (and duly cite) any open source implementation for hints on improving the performance. **[15 points]**
3. Copy the 20-layer residual network with batch normalization you have written in the previous section to `faceNet.py`. Use that network structure for training and testing. **[0 points]**
4. Train the network on CASIA dataset. Again you are free to change any hyper parameters but report the hyper parameters that you think might influence the performance. Again draw the curve of loss and accuracy of training. Report the accuracy on LFW dataset. **[20 points]**
5. If you achieve better performance compared to SphereFace, well done! Can you provide a reason? If you do not outperform SphereFace, can you provide a cause? [1] **[10 points]**
6. Plot the tSNE visualization of the CosFace embedding for the same identities from the CASIA dataset used to visualize the SphereFace embedding in the previous question. **[10 points]**

References

[1] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[2] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[3] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[4] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5265–5274, 2018.

[5] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

[6] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[1] It is not necessary for you to achieve better performance, but if you do not, offer a reasonable explanation.