A  Using SphereFace [1] for Face Verification

3. The accuracy of sphereFace on LFW dataset is 99.18%.

4.(a) We first normalized the features from the last fully connected layer to unit length and compute their dot product as the distance metric.

4.(b) We compute the accuracy using cross validation. There are 6000 pair of images in the test set. We evenly divide the test set into 10 parts so that each part contains 600 images. Every time we pick one part (600 images) for testing and the rest 9 parts (5400 images) for training. We find the threshold that achieves the highest accuracy for the 5400 images. And use that threshold to compute accuracy for the rest 600 images. This process repeats 10 times and each time we choose different 600 images for testing so that we have 10 accuracies. The final accuracy is the mean of the 10 accuracies.

5.(a) We align the faces by first computing an affine transformation to align the 5 facial landmarks, then warping the image using the computed affine transformation and cropping the center region of the warped image.

5.(b) The accuracy when cropping the center of the image is 87.48%.

B  Using MTCNN[5] for Detecting Face Landmarks

Figure 1: Two facial landmarks examples predicted by MTCNN[5].

1. Two example output can be found in Figure 1.

2. The accuracy of sphereFace on LFW dataset using the predicted landmarks is 98.45%.
Figure 2: From left to right, the curve of training accuracy and training loss for SphereFace[1], without and with batch normalization. The curves are smoothed by average 1000 data points. We can see batch normalization helps achieve lower training loss and higher training accuracy.

3. The results are not better than using the landmarks provided. The reason the results are worse is probably because the network is trained on the provided landmarks so that it will not generalize perfectly to the predicted landmarks. One way to improve the performance is to fine-tune the network on the images aligned using the predicted facial landmarks.

4. There are two reasons why it can achieve real time performance:

   (a) MTCNN uses a multi-scale cascade structure to do face detection and facial landmark localization so that they can keep the network of each cascade small while achieve good performance.

   (b) Only a small number of candidate proposals can be passed to the next level of cascade. Most of them are rejected either by the network or by non-maximal suppression. Therefore, the computational consumption is still small when we move to higher resolution.

5. For each bounding box, there is a confidence score. We sort the bounding box according to the confidence score. We first pick up the bounding box with the highest confidence score, and then remove all the bounding boxes in the list whose IoU with the picked bounding box is larger than 0.5. We repeat this process until all the bounding boxes are either picked up or removed. The picked up bounding boxes will be the results.

C Training SphereFace [1] on CASIA Dataset [4]

2. As you may have noticed. Directly implement the loss function as shown in Equation (7) of [1] will result in unstable training. In Appendix G of [1], the annealing training strategy is described. The open source implementation can be found in https://github.com/clcarwin/sphereface_pytorch.
3.(a) The training loss and accuracy curve of SphereFace can be found in Figure 2.

3.(b) The mean accuracy is 97.62%.

4.(a) The training loss and accuracy curve of SphereFace with batch normalization can be found in 2. We can see that after adding batch normalization, the training loss is smaller and the accuracy is higher.

4.(b) The mean accuracy is 97.83%, which is slightly higher than the results without batch normalization.

4.(c) Better performance can be achieved after adding batch normalization. There reasons why batch normalization can be beneficial include

   (a) It helps solve the vanishing gradient problem, which makes norm of the gradient to be relative stable even for very deep network.

   (b) It can be considered as a data augmentation. Since for each batch, there will be a different scale and offset, the same input image sent to network from different epochs will be scaled and offset differently. This may help network achieve better generalization ability.

5. The TSNE visualization is shown in Figure 3.


2. There is no special tricks. Follow the description in [3] and you should get good results.

4. The curves of training loss and accuracy are shown in Figure 4. The accuracy of cosFace on LFW dataset is 98.82%.

5. Better performance is achieved using cosFace. One reason why sphereFace is worse, as stated in [3], is because the margin in [1] is not consistent over all \( \theta \) values. The margin becomes
Figure 4: From left to right, the curve of training accuracy and training loss for CosFace[3], with batch normalization. The curves are smoothed by average 1000 data points.

Figure 5: TSNE visualization [2] of features extracted from CosFace [3].
smaller as \( \theta \) reduces. Therefore, for difficult classes with similar visual appearances, their margin can be small so that the loss function will be small. Besides, sphereFace is not stable and needs special tricks for training.

6. The TSNE visualization of features of CosFace is shown in Figure 5.

References


