FaceNet Traing and Recognition Reproduction

1552746 崔鹤洁 2018.6.4

1. FaceNet Training

- **Dataset Used**
  - Training Set: Test Set of VGGFace
  - Test Set: LFW
- **Alian the Dataset Using mtcnn**

```
# align vgg dataset
python align_dataset_mtcnn.py ../../vgg ../../align_vgg --image_size 160 --
gpu_memory_fraction 0.8
# align lfw dataset
python align_dataset_mtcnn.py ../../lfw ../../align_lfw --image_size 160 --
gpu_memory_fraction 0.8
```

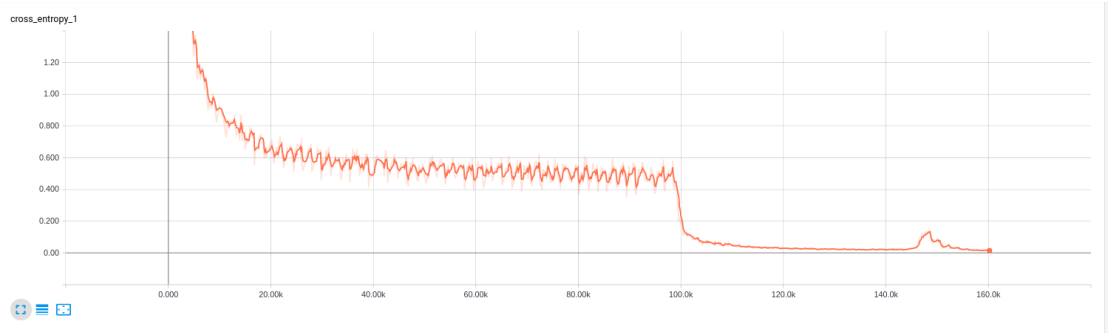
- **Training Command Line (transcirpt and parameters settings)**

```
python src/train_softmax.py \
--logs_base_dir /media/kanxuan/Data/facenet/logs/ \
--models_base_dir /media/kanxuan/Data/facenet/models/ \
--data_dir /media/kanxuan/Data/facenet/align_vgg/ \
--image_size 160 \
--model_def models.inception_resnet_v1 \
--lfw_dir /media/kanxuan/Data/facenet/align_lfw/ \
--optimizer ADAM \
--learning_rate -1 \
--max_nrof_epochs 500 \
--batch_size 90 \
--keep_probability 0.4 \
--random_flip \
--use_fixed_image_standardization \
--learning_rate_schedule_file
data/learning_rate_schedule_classifier_vggface2.txt \
--weight_decay 5e-4 \
--embedding_size 512 \
--lfw_distance_metric 1 \
--lfw_use_flipped_images \
--lfw_subtract_mean \
--validation_set_split_ratio 0.01 \
--validate_every_n_epochs 5 \
--gpu_memory_fraction 0.8 \
--epoch_size 1000
```

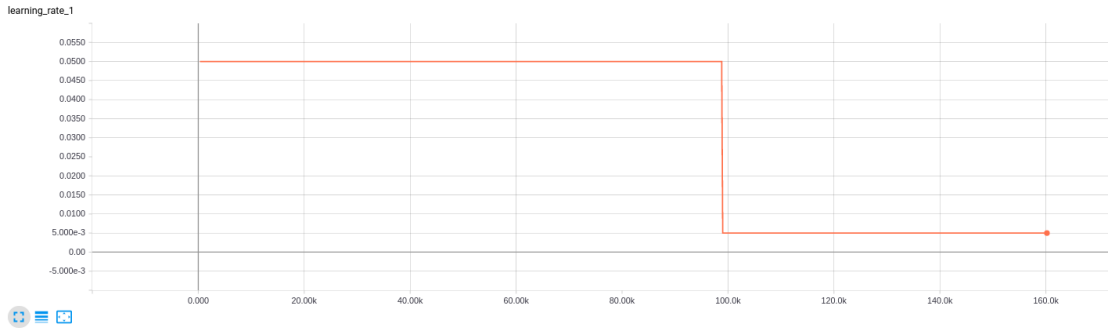
- **Start Tensorboard**

```
tensorboard - logdir /media/cuihejie/Data/facenet/logs/20180603-154515
```

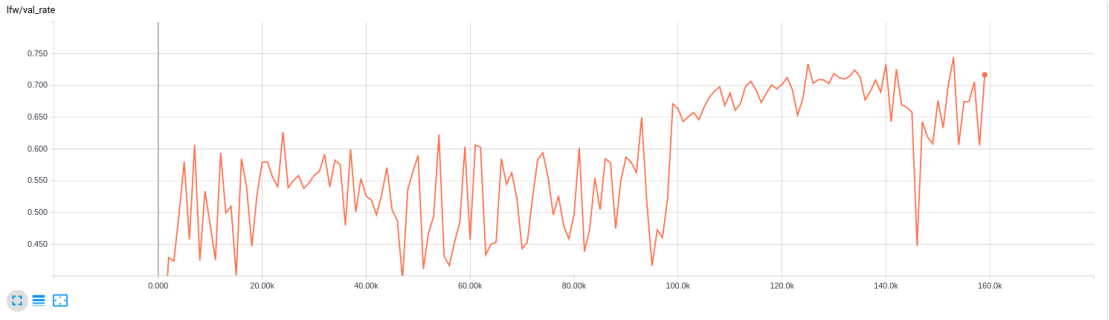
- **Accuracy and Total Loss Graphs from Tensorboard**
  - **cross_entropy_1**



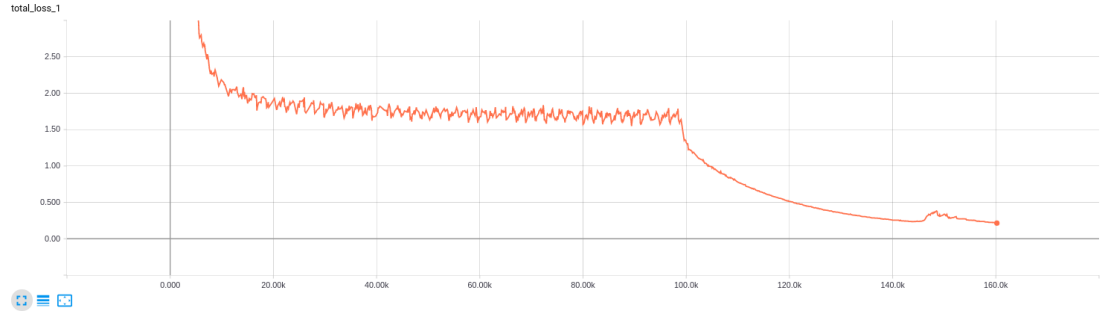
  - **learning_rate_1**



  - **lfw/val_rate**

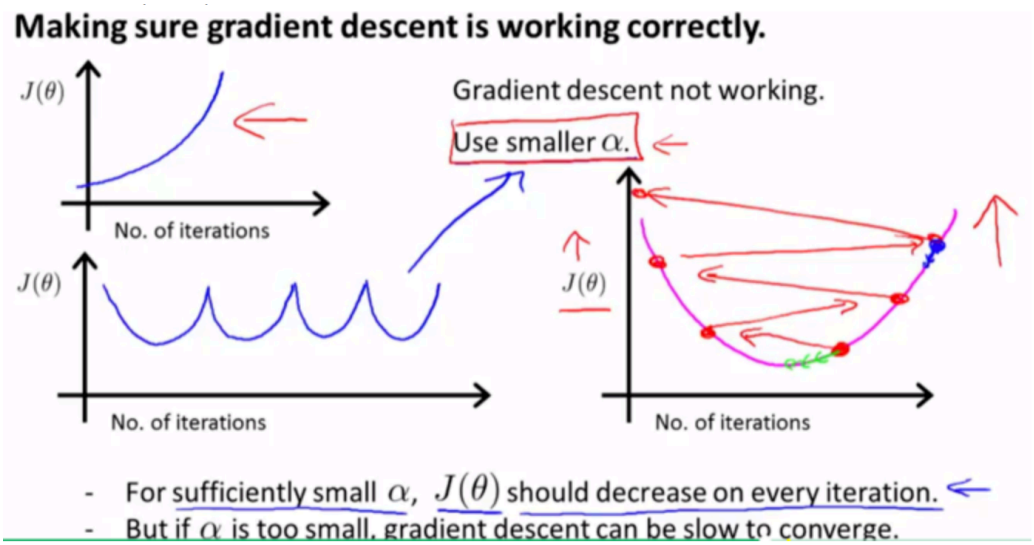


  - **total_loss_1**



  - **The Best Accuracy:** 0.9745
  - **Explaination**

- From the graphes, we can see the loss maintains steady for a while and then suddenly dropped when batch size is 100k. This is because that learning rate decreases suddenly at that time.
- For why it maintains for a period, I guess that when the learning rate is high, the gradient descent not working. It beats back and forth between two values, but it's hard to reach the lowest point, as it shows in the right picture.



- At the back of the curve, there shows signs of over-fitting. The loss was falling while the accuracy was not rising with it.

2.Face Recognition

- **Register Dataset**

  The register dataset contains five individuals, and there are 2 images for each individual. Totally there are 10 images.

  The individuals includes:

  - Clint_Eastwood
  - Laura_Linney
  - Vladimir_Putin
  - Yao_Ming
  - Zhang_Ziyi

- **Test Samples**

  The test samples contains two parts, 10 images (2 for each individuals) comes from the five individuals of the register dataset and 2 images are from individuals which do not appear in the register dataset. Totally there are 12 images.

- **Test Results**

  - **For the 10 images coming from the five individuals of register dataset:**

    The prediction labels and probabilities are as follows:

```
0  Clint Eastwood: 0.798
1  Clint Eastwood: 0.805
2  Laura Linney: 0.870
3  Laura Linney: 0.801
4  Vladimir Putin: 0.715
5  Vladimir Putin: 0.799
6  Yao Ming: 0.844
7  Yao Ming: 0.776
8  Zhang Ziyi: 0.964
9  Zhang Ziyi: 0.957
Accuracy: 1.000
```

The accuracy rate is calculated by comparing the predict labels and true labels.

- **For the two images from individuals which do not appear in the register dataset:**

  We know that if a image does not appear in a dataset, its probabilities for all classes are low. I set an threshold to judge whether an image is in the register dataset. For example, when the threshold is 0.6, it means if the probabilities for all classes are lower than 0.6, we can conclude the image is not from the register dataset. Then simply outputs "image_name.png is not in the register dataset!".

```
Calculating features for images...
Abdel_Madi_Shabneh_0001.png is not in the register dataset!
Billy_Edelin_0001.png is not in the register dataset!
```

- **The Implementation Ideas**

  The total implementation ideas can be roughly summarized into 4 steps:

  - Use the facenet model trained in the first task to generate feature vectors for the training dataset;
  - Use those generated feature vectors to train a SVM classify model;
  - Use the facenet model trained in the first task to generate feature vectors for the testing dataset;
  - Use the trained SVM model to classify the testing dataset.

3. Environment

```
Ubuntu: 16.04
Tensorflow: 1.6
Python: 3.6
CUDA: 9.0
GPU: GTX 1080 Titan
```