EE488B Experiment Report

20180467 Lee Seungyong

The purpose of the experiment is to train a face recognition model with a low EER (equal error rate) through a given dataset, VGGFace2 and Korean Faces (EE488 course self-made). EER is often used as a performance metric in face recognition task. The lower the EER value, the better the performance.

Model Architecture

Since the number of parameters of the model was limited, I use relatively lightweight model. I create my own model by referring to the **ThinResNet** model (Chung et al., 2020). The overall structure is the same as ResNet50, and the number of parameters is adjusted by adjusting the number of residual blocks and the channel size of each residual block. Its architecture is shown in the chart below.

ThinResNet34 has about 4.9M parameters, ThinResNet50 has about 6M parameters, and ThinResNet50_V2 has about 12.4M parameters. Also, the size of the output embedding vector of the model is 512 dimensions.

Layer name	Output size	ResNet50	ThinResnet34	ThinResNet50	ThinResNet50_V2
conv1	112×112	7×7, 64, stride 2			
maxpool & layer1	56×56	3×3, max pool, stride 2			
		$\begin{pmatrix}1\times1 & 64\\ 3\times3 & 64\\ 1\times1 & 256\end{pmatrix}\times3$	$\begin{pmatrix}1\times1 & 48\\ 3\times3 & 48\\ 1\times1 & 192\end{pmatrix}\times2$	$\begin{pmatrix}1\times1 & 48\\ 3\times3 & 48\\ 1\times1 & 192\end{pmatrix}\times3$	$\begin{pmatrix}1\times1 & 48\\ 3\times3 & 48\\ 1\times1 & 192\end{pmatrix}\times3$
layer2	28×28	$\begin{pmatrix}1\times1 & 128\\ 3\times3 & 128\\ 1\times1 & 512\end{pmatrix}\times4$	$\begin{pmatrix}1\times1 & 96\\ 3\times3 & 96\\ 1\times1 & 384\end{pmatrix}\times3$	$\begin{pmatrix}1\times1 & 96\\ 3\times3 & 96\\ 1\times1 & 384\end{pmatrix}\times4$	$\begin{pmatrix}1\times1 & 96\\ 3\times3 & 96\\ 1\times1 & 384\end{pmatrix}\times4$
layer3	14×14	$\begin{pmatrix}1\times1 & 256\\ 3\times3 & 256\\ 1\times1 & 1024\end{pmatrix}\times6$	$\begin{pmatrix}1\times1 & 128\\ 3\times3 & 128\\ 1\times1 & 512\end{pmatrix}\times3$	$\begin{pmatrix}1\times1 & 128\\ 3\times3 & 128\\ 1\times1 & 512\end{pmatrix}\times6$	$\begin{pmatrix}1\times1 & 192\\ 3\times3 & 192\\ 1\times1 & 768\end{pmatrix}\times6$
layer4	7×7	$\begin{pmatrix}1\times1 & 512\\ 3\times3 & 512\\ 1\times1 & 2048\end{pmatrix}\times3$	$\begin{pmatrix}1\times1 & 256\\ 3\times3 & 256\\ 1\times1 & 1024\end{pmatrix}\times3$	$\begin{pmatrix}1\times1 & 256\\ 3\times3 & 256\\ 1\times1 & 1024\end{pmatrix}\times3$	$\begin{pmatrix}1\times1 & 384\\ 3\times3 & 384\\ 1\times1 & 1536\end{pmatrix}\times3$
	1×1	Average pool, 512-d fc			

Loss Function

In the experiment, two loss functions, **AM-softmax loss** and **Angular prototypical loss** are used. AM-softmax loss is classification-based loss that we learned in course.

Angular prototypical loss is one of the metric-based loss which use cosine-based similarity metric with learnable scale and bias. **M** is a hyperparameter for this loss. For each mini-batch, there are M-1 images per class in a support set and 1 image per class in a query set.

Implementation Detail

Dataset

There are 3,141,890 images of 8,631 identities in **VGGFace2** training dataset. This dataset is used for pretraining my model. The **Korean Faces** dataset is an EE488 course self-made dataset with 83,219 images of 1,229 identities in the training dataset. The pretrained model is finetuned by Korean Faces training dataset to make a good performance on the Korean Faces validation/test dataset.

Training

My implementation is based on the PyTorch famework and trained on the EElab25 machine using GTX 1070 GPU.

For **pertraining**, I use VGGFace2 training dataset. The models are trained for 60 epochs. For each epoch, I randomly sample a maximum of 200 images from each of the 8,631 identities to reduce class imbalance. I use Adam optimizer (β1=0.9, β2=0.999, ε=1e-08) with initial learning rate of 0.001. For learning rate scheduler, I use cosine annealing warm restarts (cycle=10epochs, minimum lr=0.00001). I use AM-softmax loss with scale of 30 and margin of 0.1 with batch size of 200. The training takes approximately 3 days for each model.

For **finetuning**, I use Korean Faces training dataset. The models are trained for 50 epochs. I use Adam optimizer (β1=0.9, β2=0.999, ε=1e-08) with initial learning rate of 0.0005, which is half of the initial learning rate of the pretraining task. For learning rate scheduler, I use cosine annealing warm restarts (cycle=10epochs, minimum lr=0.00001). I use Angular prototypical loss with various M (2, 3, and 4) with batch size of 200 and 250 to find the best model. The training takes approximately 5 hours for each model.

Data augmentation

For each the training above, I use cropping as image augmentation by **cropping** random portion from 0.8 to 1 of input image and resize it to a given size. I also tried **color jittering** as augmentation for experiments. It randomly changes the brightness, contrast, and saturation of an image by the factor which is chosen uniformly from 0.3 to 1.7.

Evaluation

The evaluation metric is EER in the validation set and EER in the test set. EER in the test set was tested only on certain models with good performance. The lowest validation EER among 50 epochs is marked below charts.

	Hyper Parameter	ThinResNet34	ThinResNet50	ThinResNet50_V2
Angular prototypical loss (batch size=200)	M=2	4.02349	3.89570 (test EER: 4.983)	3.85381 (test EER: 4.757)
	M=3	4.16798	3.92711 (test EER: 4.994)	3.89570
	M=4	4.28317	4.04231	3.94806

The performance of the complex model with a large number of parameters was better, and the performance was the best when M=2. It can be explained that when M=2, it is similar to 1 shot learning and is the most similar to our validation and test environment.

Batch size	200	250
ThinResNet50 (M=2)	3.94436	3.88153
ThinResNet50_V2 (M=2)	3.85381	3.76251 **(test EER: 4.671)**

Due to GPU memory limitations, it's impossible to experiment with many larger batch sizes, but models show better performance on larger batch sizes. It can be explained by the ability to sample harder negatives within the batch.

Data augmentation	None	Random crop	Color jitter	Random crop & color jitter
ThinResNet50_V2 (M=2, batch size=250)	4.17125	3.76251	4.20835	3.79513

I observe that random cropping is in effectively but color jittering is not. It can be explained that random cropping helps to avoid overfitting but color jittering rather distorts the information of the original image and hinders learning.

Reference

[1] Chung et al., 2020 Chung, J. S., Huh, J., Mun, S., Lee, M., Heo, H. -S., & Choe, S., et al. (2020). In defence of metric learning for speaker recognition. In Proc. INTERSPEECH 2020