Analysis And Design Of Neural Networks
Homework 2

Abbas Nosrat
810199294

May 8, 2022

Contents

Abstract

Triplet loss is among the most popular losses for deep metric learning. This loss takes an anchor, a positive and a negative as input and aims to decrease the distance between the anchor and the positive, while increasing the distance between the anchor and the negative in the embedding space. However, this loss suffers from a few major downsides. Consider a case in which there are two triplets with the same anchor but different positives and negatives. The distance margin between both positive and negatives are the same. However, the first pair of positive and negative is closer to the anchor than the second.

$$\|p_1 - n_1\| = \|p_2 - n_2\|$$

$$\|p_1 - a\| < \|p_2 - a\|, \|n_1 - a\| < \|n_2 - a\|$$

In such cases, triplet loss considers the same value for both pairs and optimizes equally for both of them. To fix this issue, Circle loss gives weights to positives and negatives according to their distance from the anchor. In other words, when positive is close to the anchor, circle loss favors pushing the negative further and when the negative is far enough, the loss shifts focus towards drawing the positive closer.

$$(\alpha_n Sn — \alpha_p Sp)$$

where α_p and α_n independent weighting factors, allowing s_n and s_p to learn at different paces. Additionally, circle loss converges faster since it favors pairs which are not too far or close to the anchor due to the weighting factors. The proposed formula for this loss is

$$\mathcal{L}_{circle} = \log\left(1 + \sum_{j=1}^{L} \exp\left(\gamma \alpha_n^j \left(s_n^j - \Delta_n\right)\right) \sum_{i=1}^{K} \exp\left(\gamma \alpha_p^i \left(s_p^i - \Delta_p\right)\right)\right)$$

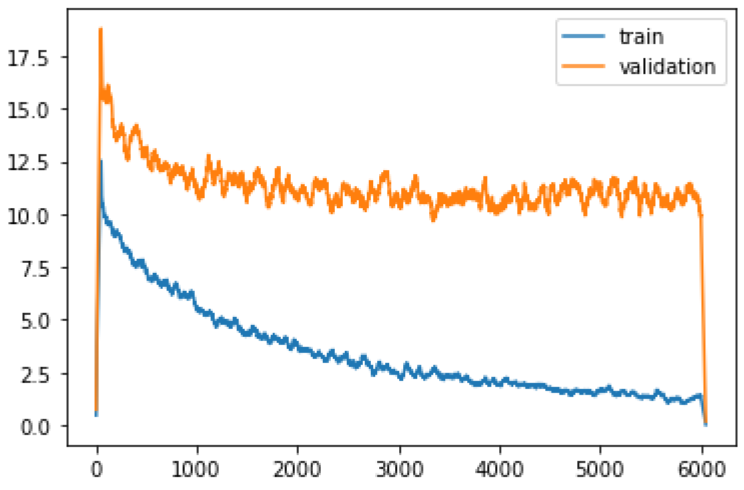
Part I

:

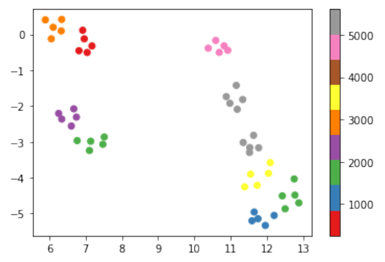
1. For implementation of this part, a pretrained ResNet18 model was chosen since it produced better results and took less time to train. Since the dataset is heavily imbalanced, A costume data loader was implemented. The following is the algorithm of the data loader:

```
Inintialize with user defined parameters (cpb, spc)
find classes with number of samples >= spc
randomly choose cpb from the found classes
choose spc samples from chosen classes
Convert to tensor
return
```

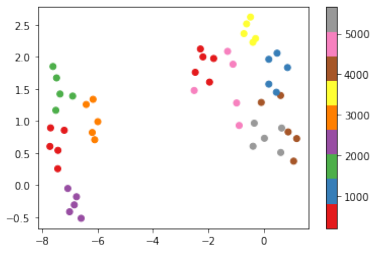
   spc stands for sample per class and cpb stands for class per batch. The training loop is a generic PyTorch loop with one difference. After ther forward pass, a miner finds hard examples for faster convergence and then passes the indices of pairs to the loss class.(Not sure if it made any difference since the forward pass of all losses takes the same parameters in PyTorch metric learning library). the model was trained for 6000 iterations and the loss plot and a umap of the embeddings for a few classes is brought in Figure 1



(a) training and validation loss(the lines at the beginning and the end are caused the smoothing filter)



(b) UMAP of training embeddings

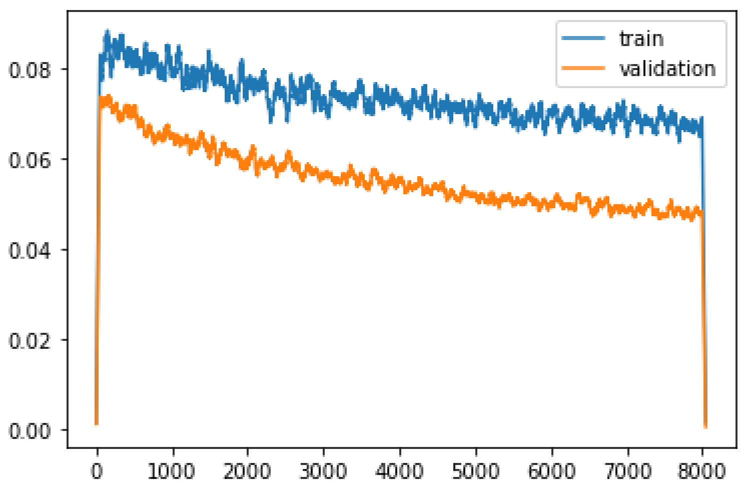


(c) UMAP for validation embeddings

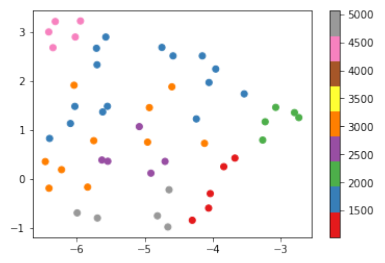
Figure 1: Training results for circle loss

There is an unusual distance between training and validation loss both with pretrained and randomly initialized networks. It could be due to lack of a miner for validation loss. According to the documentation, without miner indices, the loss is evaluated for all possible triplets in the batch. This hypotheses was not tested due to lack of time. In contrast with the behavior of validation loss, the embeddings were good and a nice separation can be seen in the embedding space. The embedding vector size was 128 and the embeddings were normalized with L2 normalization since the same procedure was done in FaceNet model.

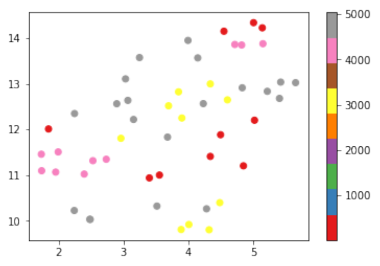
2. The exact same process as the previous section with a change in loss function. The results are demonstrated in Figure 2



(a) training and validation loss(the lines at the beginning and the end are caused the smoothing filter)



(b) UMAP of training embeddings

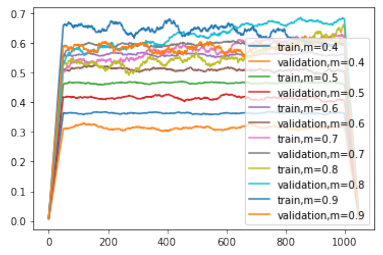


(c) UMAP for validation embeddings

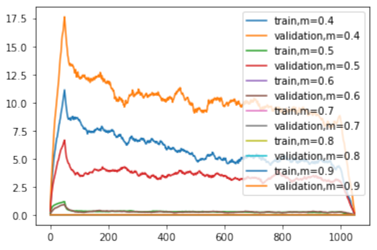
Figure 2: Training results for circle loss

Although the model has not converged, the further training was not possible since it was done on a public server and others had to use it as well.

3. Each iteration lasted almost for the same amount for both loss functions. However, triplet loss clearly needs more iterations to converge and thus circle loss is faster to train. The quality of embeddings for circle loss if far better than triplet loss and circle loss had better generalization dispite what the training logs demonstrate. To have a visual sense of sensitivity over margin parameter, the model was trained for 1000 iterations and each time the margin was increased by 0.1 starting from 0.4 and ending on 1. Figure 3 demonstrates the results.



(a) Triplet loss



(b) circle loss

Figure 3: Sensitivity over margin parameter for circle loss and triplet loss

As demonstrated in Figure 3, Although there is no change in the behavior of both losses, circle loss is far more sensitive to changes in the margin parameter. Another observation can be made from training log figures is that circle loss converges with more stability than triplet loss. The smoothing filter for both losses was a moving average with size of 50 which is just an array of ones with size of 50, convolved in the loss array.