Make a comparison study of Angular-based losses vs Metric-based ones on different datasets (MNIST, FLW, CIFAR-10)

ФИО: **Чжан Цзывэй**
Факультет: MEX-MAT *Курс:* Магистр 2 г.о.
Специальность: Математика и компьютерные науки
*Github:*Zzzkoala/LossFunction

2023/5/15

1 Angular-based losses and Metric-based losses

Angular-based losses, also known as angle-based losses, are a type of loss functions commonly used in various computer vision tasks such as face recognition and person re-identification. These losses aim to learn discriminative feature representations by considering the angular relationship between samples. One popular example of an angular-based loss is the ArcFace loss, which combines the softmax function with an additive angular margin. By incorporating the angular margin, the loss enforces larger angular separations between samples of different classes, leading to more robust and discriminative features.

On the other hand, metric-based losses focus on learning embeddings or feature representations that preserve the similarity relationships between samples. These losses aim to minimize the distance or dissimilarity between samples of the same class while maximizing the distance between samples from different classes. Commonly used metric-based losses include contrastive loss, triplet loss, and center loss. These loss functions help in creating compact and separable feature embeddings that are suitable for tasks like image retrieval and clustering.

The main difference between angular-based losses and metric-based losses lies in the way they model the relationship between samples. Angular-based losses consider the angle or angular separation between samples, focusing on the discriminative aspect of the features. They explicitly enforce larger angular separations between different classes, leading to well-separated clusters in the feature space. On the other hand, metric-based losses focus on the pairwise distance or similarity relationships between samples. They aim to pull samples from the same class closer together while pushing samples from different classes farther apart, promoting compact and distinct clusters in the feature space. In essence, angular-based losses emphasize angular separability, while metric-based losses emphasize distance-based separability.

2 Experimental Loss Function

In this experiment, ArcFace and CosFace will be used as representatives of Angular-based losses, and Triplet Loss and Contrastive Loss will be used as representatives of Metric-based losses. The ResNet-18 neural network model will be trained with MNIST, Omniglot, and CIFAR-10 datasets respectively to study the training results and explore which loss function is suitable for different datasets.

2.1 ArcFace and CosFace

ArcFace and CosFace are two popular methods in face recognition that improve the discriminative power of the feature embeddings.

ArcFace introduces an additive angular margin to the softmax loss function. It aims to enhance the intra-class compactness and inter-class separability. Given an input sample and its corresponding class label, ArcFace

calculates the angular margin based on the cosine similarity between the sample and the weights of the corresponding class. The angular margin is then added to the angle of the cosine similarity. The resulting angular value is used in the softmax function to compute the probability distribution over the classes. The angular margin enforces a larger angular separation between different classes, making the feature representations more discriminative.

The mathematical formula for ArcFace can be expressed as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(s\cdot\cos(\theta_{y_i}+m)))}{\sum_{j=1}^{C}\exp(s\cdot\cos(\theta_j))}$$

where N is the batch size, C is the number of classes, s is a scaling factor, θ_{y_i} is the angle between the input sample and the weight vector of its true class y_i, and m is the additive angular margin.

CosFace, on the other hand, modifies the softmax loss by adding a cosine margin directly to the cosine similarity. Similar to ArcFace, it enhances the inter-class separation. However, in CosFace, the angular margin is subtracted from the cosine similarity instead of being added. This effectively pushes the samples towards the decision boundary of their respective classes, resulting in sharper decision boundaries.

The mathematical formula for CosFace can be expressed as:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(s\cdot(\cos(\theta_{y_i})-m)))}{\sum_{j=1}^{C}\exp(s\cdot\cos(\theta_j))}$$

where the variables have the same meaning as in ArcFace.

Both ArcFace and CosFace are effective in improving the discriminative power of feature embeddings for face recognition tasks by incorporating angular margins. The choice between the two methods depends on the specific requirements and characteristics of the problem at hand.

2.2 Triplet Loss

Triplet Loss and Contrastive Loss are two commonly used metric-based loss functions in computer vision tasks such as image retrieval and clustering.

Triplet Loss focuses on learning embeddings that preserve the relative distances or similarities between samples. It aims to minimize the distance between an anchor sample and a positive sample (from the same class) while maximizing the distance between the anchor sample and a negative sample (from a different class). The loss encourages the anchor-positive pair to be closer together in the embedding space compared to the anchor-negative pair.

The mathematical formula for Triplet Loss can be expressed as follows:

$$L = \sum_{i=1}^{N}\max(0, d(a_i, p_i) - d(a_i, n_i) + \alpha)$$

where N is the batch size, a_i, p_i, and n_i represent the anchor, positive, and negative samples respectively, $d(\cdot,\cdot)$ is a distance function, and α is a margin that defines the minimum difference between the positive and negative distances.

3 Design of experiments

3.1 Design of Loss Functions

This experiment uses pytorch-metric-learning to implement the required loss function calculations, where each function requires specific incoming parameters to be implemented, some of the important parameters and the implementation process are listed below:

```
# ArcFaceLoss
losses.ArcFaceLoss(num_classes, embedding_size, margin=28.6, scale=64, **kwargs)
```

num classes: The number of classes in your training dataset.

embedding size: The size of the embeddings that you pass into the loss function. For example, if your batch size is 128 and your network outputs 512 dimensional embeddings, then set embedding size to 512.

margin: The angular margin penalty in degrees. In the above equation, m = radians(margin). The paper uses 0.5 radians, which is 28.6 degrees.

scale: This is s in the above equation. The paper uses 64.

```
# CosFaceLoss
losses.CosFaceLoss(num_classes, embedding_size, margin=0.35, scale=64, **kwargs)
```

num classes: The number of classes in your training dataset.

embedding size: The size of the embeddings that you pass into the loss function. For example, if your batch size is 128 and your network outputs 512 dimensional embeddings, then set embedding size to 512.

margin: The cosine margin penalty (m in the above equation). The paper used values between 0.25 and 0.45.

scale: This is s in the above equation. The paper uses 64.

```
# TripletLoss
losses.TripletMarginLoss(margin=0.05,
                         swap=False,
                         smooth_loss=False,
                         triplets_per_anchor="all",
                         **kwargs)
```

margin: The desired difference between the anchor-positive distance and the anchor-negative distance. This is m in the above equation.

swap: Use the positive-negative distance instead of anchor-negative distance, if it violates the margin more.

smooth loss: Use the log-exp version of the triplet loss

triplets per anchor: The number of triplets per element to sample within a batch. Can be an integer or the string "all". For example, if your batch size is 128, and triplets per anchor is 100, then 12800 triplets will be sampled. If triplets per anchor is "all", then all possible triplets in the batch will be used.

3.2 t-SNE Dimensionality reduction visualisation

At the end of each cross-training session, the t-SNE technique is used to downscale the test set classification results into a three-dimensional space, providing a more visual representation of the relationships between the categories in the dataset and the relationships between the data within each category.

The following t-SNE visualisation code for the MNIST - CosFace experiment was selected as an example:

```
embeddings = torch.cat(embeddings, dim=0).cpu().numpy()
# Splice together the embedding vectors and convert to a numpy array
labels = torch.cat(labels, dim=0).cpu().numpy()
# stitch together labels and convert to numpy arrays
tsne = TSNE(n_components=3, perplexity=30, n_iter=1000)
# Create a TSNE object and set the parameters
embeddings_3d = tsne.fit_transform(embeddings)
# use the TSNE object to downscale the embedding vector to get the coordinates in 3D
```

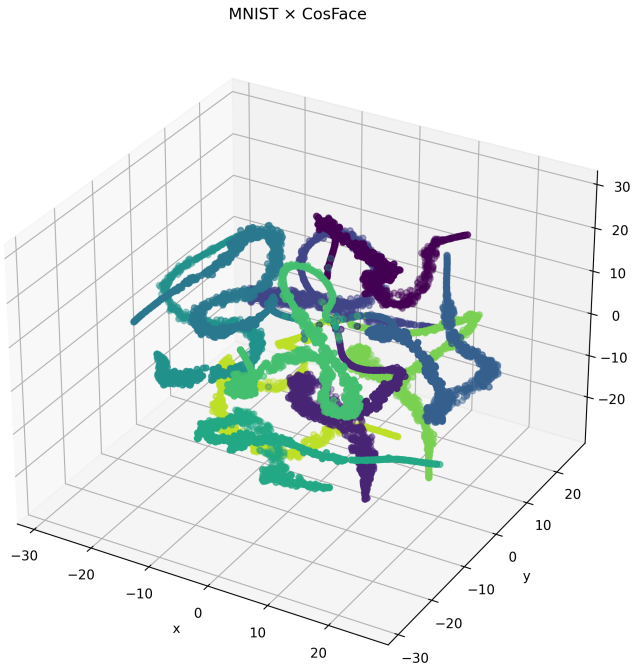

Figure 1: MNIST-CosFace

4 Experimental results

4.1 Generalization capabilities

The MNIST, FLW, and CIFAR-10 datasets were used to cross-train the ResNet-18 model with four loss functions, ArcFace Loss, CosFace Loss, Triplet Loss, respectively, for 10 epochs, and the final loss and accuracy achieved on the respective validation sets were printed as follows:

Table 1: Loss

	MNIST	FLW	CIFAR-10
ArcFace	0.4283	32.8566	12.1838
CosFace	1764.6043	1832.4358	1805.1597
Triplet	0.0500	0.0500	0.0500

Table 2: Acc

	MNIST	FLW	CIFAR-10
ArcFace	0.9939	0.4841	0.7079
CosFace	0.9935	0.4444	0.6470
Triplet	0.2003	0.0000	0.1000

4.2 Reduced dimensional display results by t-SNE

The following will show the 3D distribution of the test set data in each dataset after the training is completed using the t-SNE technique of dimensionality reduction:

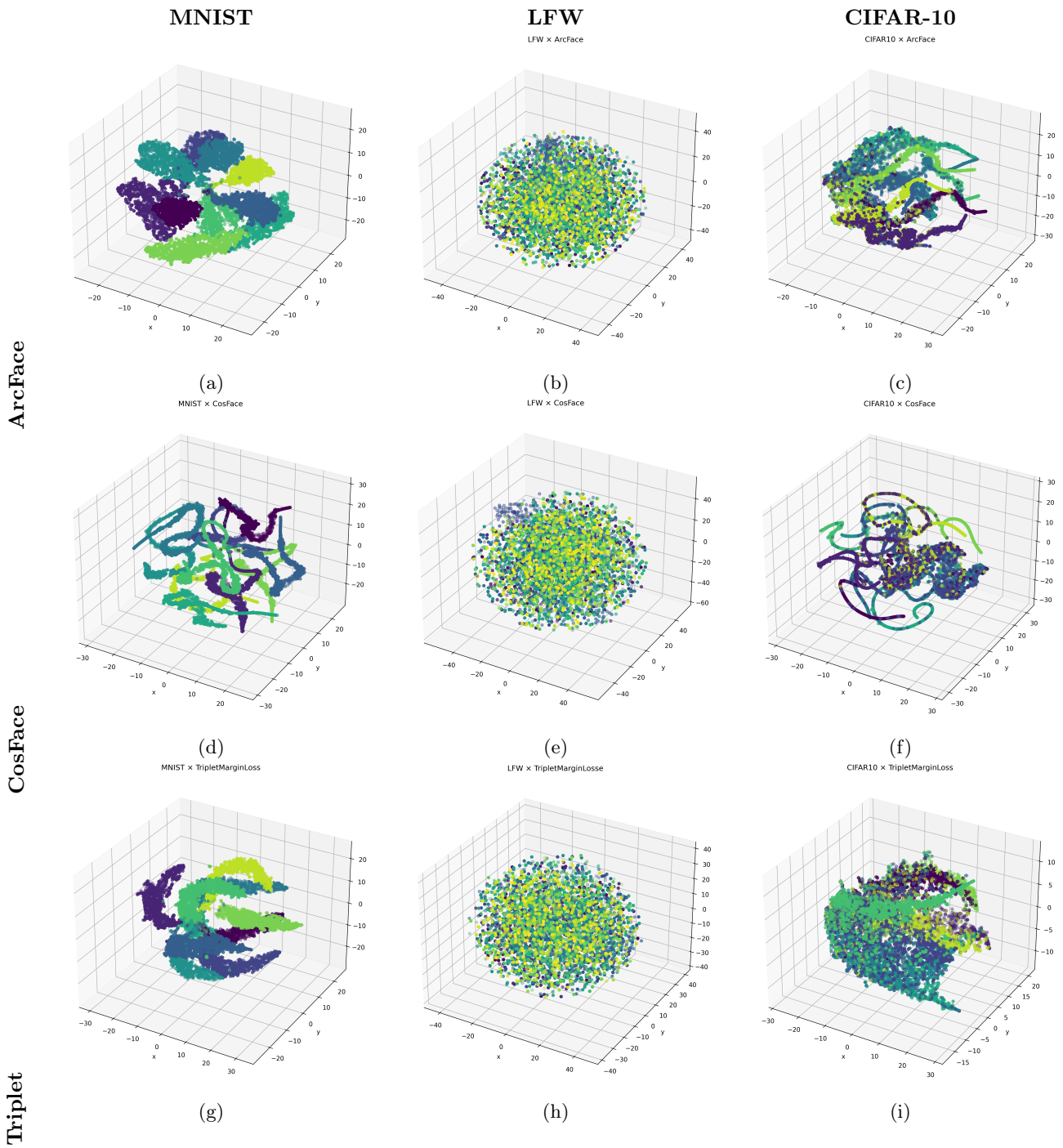

Figure 2: Cross-sectional experimental data graph

5 Analysis and Conclusion

If we analyze the model in terms of its generalization ability, MNIST, FLW, and CIFAR-10 are all suitable for using ArcFace Loss as a loss function under the conditions of the ResNet-18 neural network model.

If we analyze from the perspective of the t-SNE technique to downscale the distribution map of the dataset, under the condition of the ResNet-18 neural network model, MNIST, CIFAR-10 datasets can also be trained using Triplet Loss as the loss function.

It is worth noting that the LFW dataset as a face dataset should have been better suited to use Triplet Loss as the loss function, but as can be seen from the distribution plot after the data has been downscaled: The LFW Face dataset lacks fine-grained identity annotations. It only provides information on whether a pair of images belongs to the same person or not. The Triplet Loss heavily relies on accurate and specific labels to optimize the embedding space effectively. Without more detailed labels, it becomes challenging to construct informative triplets that can guide the learning process and encourage better discrimination between face identities.