International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:15, No:7, 2021

Adaptive Few-Shot Deep Metric Learning

Wentian Shi, Daming Shi, Maysam Orouskhani, Feng Tian

Abstract—Currently the most prevalent deep learning methods require a large amount of data for training, whereas few-shot learning tries to learn a model from limited data without extensive retraining. In this paper, we present a loss function based on triplet loss for solving few-shot problem using metric based learning. Instead of setting the margin distance in triplet loss as a constant number empirically, we propose an adaptive margin distance strategy to obtain the appropriate margin distance automatically. We implement the strategy in the deep siamese network for deep metric embedding, by utilizing an optimization approach by penalizing the worst case and rewarding the best. Our experiments on image recognition and co-segmentation model demonstrate that using our proposed triplet loss with adaptive margin distance can significantly improve the performance.

Keywords—Few-shot learning, triplet network, adaptive margin, deep learning.

I. Introduction

DEEP learning methods has achieved great success in recent years, which have being applied to various visual tasks like facial recognition [5], [26] and pedestrian recognition [7]. However, to train a good neural network model always requires a large amount of labelled data in numerous iterations, which can be quite time consuming and financially expensive. Additionally, it is not practically applicable to train a network for endangered objects like *Orcaella brevirostris*, due to insufficient data available. On the contrary, we as human are able to learn a new object like car, animal swiftly, given only one photo. Inspired by this capability of human's neural networks, few-shot learning approach was proposed [16], aiming to recognise new categories from limited few samples with the prior knowledge learned from the other datasets.

To recognise new categories from limited few samples, we can obtain the prior knowledge learned from the other datasets. As the training is based on few samples, few-shot learning is easily stuck in overfitting problem. To alleviate this problem, researchers propose Fine-Tune strategy models [22], [18], metric based models [20], [11], [24] and sequence based methods [17]. The metric based learning methods that employ distance measures to find the similarity between images have shown their superior performance. With the advent of modern deep neural networks, these approaches map the low-level visual features to an embedding space, where samples are clustered together if they belong to same class, while samples of different classes keep a large distance apart. As a result, a test sample can be easily classified using kNN. Matching networks [23] consider the relationship among the training examples and learn a classifier for any given support set. Relation networks [21] learn a deep distance metric instead of a fixed metric function. Prototype Network [20] proposes the prototype representation, calculated by mean point of each class in each episode training, and followed with a Softmax classifier to make the final prediction.

The key point of metric based methods is how to learn a good mapping function, making samples belonging to different classes in the embedding space keep as far as possible from others, while same-class samples are compact. Designing a proper loss function will contribute to learn such a good mapping function, and one of the most popular loss functions is the triplet loss [19]. A triplet is composed of three samples (an anchor, a positive and a negative data point). The distance between the anchor and the positive is smaller than that between the anchor and the negative, and usually we set a margin distance to value such a distance difference. Margin distance plays an important role in loss function, as a proper margin distance could make the embedding space more discriminative, which directly influences the final performance of whole algorithm. Apparently, a small margin distance makes most triplets fit the triplet criteria, so it speeds up the convergence of algorithm but outputs a bad classifier. Contrarily, a large margin distance may make the most selected triplets violate the rules, slowing down the convergence, and even making the algorithm rendering a failure of convergence, despite the good performance of classification. Many researchers set the margin distance empirically as a constant number, and conduct ablation study to compare the accuracy of their models with several margin distances for experiments. Usually the middle range margins are chosen. Unlike the empirical approach, in this paper we aim to learn this parameter automatically and adaptively, by taking a data-driven approach with a consideration of a problem related to zero-shot learning [8], [13]-[15], [3]. In few-shot learning, each task we use to train the model is different. More specifically, the distributions of the input data are not same. So, when dealing with different tasks, we may employ a constant margin to evaluate the distance difference in the triplets. To some extent, this problem is like the domain shift problem in zero-shot learning [8], [13]. To deal with this problem, [13] proposes a encoder-decoder model, whose core part is the constrain in its decoder. The model is able to re-construct the original visual feature, while backward influencing the mapping function in the encoder part. Reference [25] employs a large-margin criterion, making the margin distance relative to data itself, as it calculates the mean L-2 norm value of all samples in one mini-batch in

Wentian Shi and Maysam Orouskhani are with the College of Computer Science and Software Engineering, Shenzhen University, China (e-mail: shiwentian2018@ email.szu.edu.cn, maysam.orouskhani @szu.edu.cn).

Daming Shi is with the College of Computer Science and Software Engineering, Shenzhen University, China and MiddleSex University, UK (corresponding author, e-mail: shiwentian2018@ email.szu.edu.cn).

Feng Tian is with the Faculty of Science and Technology,Bournemouth University, UK (e-mail: ftian@bournemouth.ac.uk).

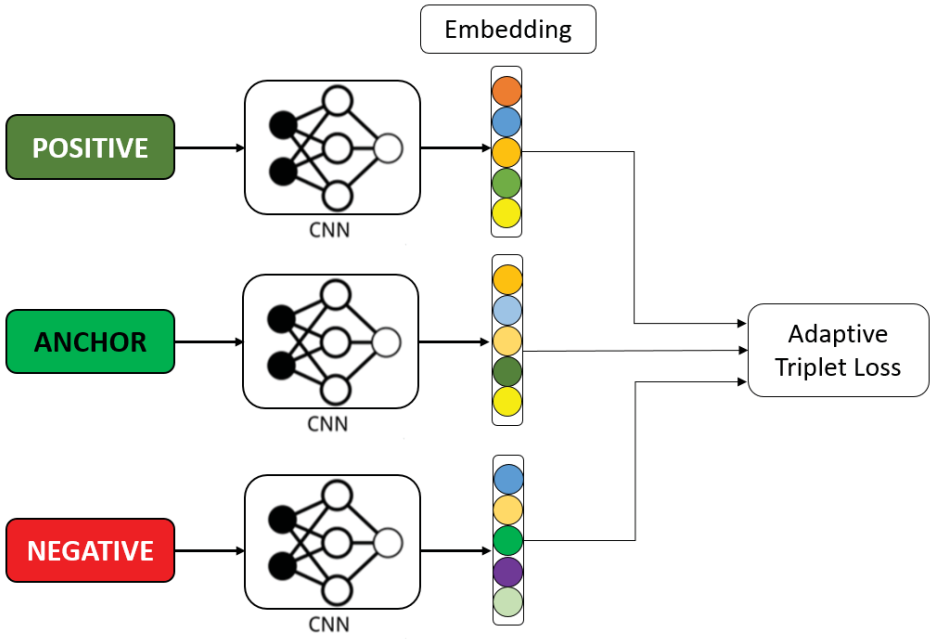

Fig. 1 Triplet Network: The three networks share same structure and parameters. Our proposed employs a conditional triplet loss and an adaptive margin strategy to compute the distance between anchor, positive, and negative samples

the embedding space. This strategy works and generates a better discriminative embedding space, but it is still not able to describe the difference between the distributions of each input task. We adopt the margin distance in [25] as the initial value, adding a variance item of each episode. The initial value is corrected with a Kullback–Leibler divergence to evaluate the distribution difference between episodes, so that the adaptive margin distance is related to the data distribution and obtained automatically in the training stage. Moreover, we propose a optimization strategy by penalizing the worst case of triplets and rewarding those that meet the best condition. We employ our proposed triplet loss function to image-recognition and image co-segmentation. Our contributions in this work are summarized as follows: (i) we propose an adaptive margin distance strategy to improve the performance of triplet loss, powering it with a penalty/reward optimization approach in training stage. (ii) We evaluate the performance of our methods in image recognition and co-segmentation by replacing the standard loss function with our proposed triplet loss. The experiments results demonstrate an obvious performance gain.

II. Related Works

A. Few Shot Learning

The meta few-shot learning aims to train a classifier template from tasks and swiftly adapts to new tasks which are formed of samples from different classes. Usually few-shot learning contains two stages. Given a dataset $D = (\mathbf{x}_i, y_i)\}_{i=1}^{N}, y_i \in C$, where $\mathbf{x}_i$ is the input images, $C = y_1, ...y_K$ is the classes and N is the number of samples. In training stage, we set training dataset $D_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$ from C_{train} classes to train the classifier. In testing phase, we use the remaining samples to construct a support set D_S with the same form of training tasks and a query set D_Q. $D_S = \{(\mathbf{x}_j, y_j)\}_{j=1}^{S}$ from classes C_{test}, $y_j \in C_{test}$ and $C_{train} \cap C_{test} = \oslash$. Then we predict the labels of query set $D_q = \{(\mathbf{x}_j)\}_{j=S+1}^{S+Q}$, where Q is the number of queries. Conventionally,it is called C-way K-shot learning if C classes are selected for the support set, and K samples selected from each classes in C, with $S = K \times C$.

B. Triplet Loss and Triplet Network

Before introducing triplet network, we firstly come to Siamese neural network [12]. The Siamese network is composed of two paralleled sub-networks which share the same structure, weights and parameters. Given a pair of inputs, the Siamese network firstly maps them into an embedding space to obtain two corresponding embedding vectors, and then outputs the similarity of these inputs by a contravasive loss. If the similarity is smaller than a pre-set threshold, the two inputs are considered as same label, and vice versa.

Compared to Siamese network, the triplet network [10] implements a triplet loss to increase the performance and efficiency. A triplet is formed of an anchor sample, a negative and a positive, where the anchor and the positive are selected from the same class, while the anchor and the negative are from different. Fig. 1 shows the structure of the basic triplet network, in which we replace the standard triplet loss with our triplet loss implemented with our adaptive margin. Like Siamese network, the triplet network also maps triplets into an embedding space, where the anchor and positive samples should be near, while the anchor and negative are supposed to be far from each other.

The embedding feature vector of the input $\mathbf{x}_i$ is $f(\mathbf{x}_i) \in R^d$, where d is the dimension of embedding vector. In the training

stage, we select triplets $T = (\mathbf{x}^a, \mathbf{x}^p, \mathbf{x}^n)$ which consist of an anchor $\mathbf{x}^a$, a positive sample $\mathbf{x}^p$ and a negative sample $\mathbf{x}^n$. The labels of the triplet T satisfy $y^a = y^p \neq y^n$. Triplet loss aims to pull samples belonging to the same class into nearby points on a manifold surface, and push samples with different labels apart. The standard triplet loss is:

$$\mathrm{L}_{standard} = \frac{1}{N_t}\sum_{i=1}^{N_t}[\|(f(\mathbf{x}_i^a) - f(\mathbf{x}_i^p))\|_2^2 - \|(f(\mathbf{x}_i^a) - f(\mathbf{x}_i^n))\|_2^2 + m] \quad (1)$$

where N_t is the number of triplets and m is the margin distance.

III. Methodology

A. Adaptive Loss

We choose the triplet loss aforementioned in (1) as our adaptive margin function. Apparently, different anchor, negative and positive samples will create different inputs for the selection of triplets, so the distances between the anchor and the negative/positive samples may vary. To handle this variation, here we define an adaptive loss as:

$$L_{adaptive} = \frac{1}{N_t}\sum_{i=1}^{N_t}[\|(\mathbf{x}_i^a - \mathbf{x}_i^p)\|_2^2 - \|(\mathbf{x}_i^a - \mathbf{x}_i^n)\|_2^2 + m_a] \quad (2)$$

We use the adaptive margin with randomly initialized model parameters at the beginning of training. We set the initial margin value as m_e:

$$m_e = \frac{1}{N_b}\sum_{i=1}^{N_b}\|f(\mathbf{x}_i)\|_2^2 \quad (3)$$

where N_b is the number of all embedding samples in a mini-batch and m_e is the average of L_2-norm of the samples in the mini-batch. Then we update the value by:

$$m_a = m_e + \alpha s \pm \beta A \quad (4)$$

where s is the item for the variance in the embedding space, α is the balanced parameter, A is a penalty/reward item and β is the corresponding balanced item. The penalty/reward item fines the worst triplets by adding a penalty term and rewards triplets that satisfy the best condition, which will be introduced later.

For the variance parameter s, we take the intra-class and inter-class divergence of input data into consideration. For the sake of simplicity, we take L2-norm of all samples for calculating. Denoting N_c as the number of classes selected from $\{1, ..., C_{train}\}$, C_{N_c} as the set of class indices, c_k as the k-th element in C_{N_c}, D_{c_k} as the subset of all training samples with $y_i = c_k$. For each class, denoting S_k and Q_k as the subsets of support set and query set that are randomly selected from D_{c_k} respectively, and $S_k \cap Q_k \oslash$. The center of each class $\mathbf{c}_k = \frac{1}{|S_k|}\sum_{(\mathbf{x}_i, y_i)\in S_k}\mathbf{x}_i$. The intra-class divergence should be:

$$S_{intra} = \sum_{i=1}^{N_c}\frac{1}{|S_k|}\sum_{j=1}^{S_k}\|f(\mathbf{x}_{ij}) - \mathbf{c}_i)\|_2^2$$

where $\mathbf{x}_{ij}$is the j-th sample in i-th class. For the inter-class divergence, instead of using all samples to calculate a complex similarity matrix, we use their center as their representations. Therefore, the inter-class divergence is:

$$S_{inter} = \frac{1}{N_c}\sum_{i=1}^{N_c}\sum_{j=1}^{N_c}\|\mathbf{c}_i - \mathbf{c}_j\|_2^2$$

The total data divergence $S_{total} = S_{intra} + S_{inter}$

To better measure the difference of data distribution in embedding space in different tasks, we utilize the KL divergence to measure the variance difference in adjacent iteration.

First, we sort all samples in ascending order, we measure the KL divergence between two adjacent iterations, and used as a corrected parameter for S_{total}. For example, the KL divergence for n-th iteration and (n+1)-th iteration is:

$$KL_n^{n+1} = \frac{1}{N_c|S_k|}\sum_{i=1}^{N_c|S_k|}\|f(\mathbf{x}_i^{n+1})\|_2^2 \log\frac{\|f(\mathbf{x}_i^{n+1})\|_2^2}{\|f(\mathbf{x}_i^n)\|_2^2} \quad (5)$$

So we obtain

$$s_{n+1} = S_{total}^{n+1} \times (1 - \frac{KL_{n+1}^n + KL_n^{n+1}}{2}) \quad (6)$$

Like EM algorithm[6], we update s in each iteration, and We can obtain our final adaptive margin distance m_a after it converges.

B. The Penalty-Reward

Given $O(N^3)$ triplets, it is infeasible to put all of them into a single mini-batch. So we need to sample the triplets over the entire training set. Some sampling methods such as hard and semi-hard[19] find the particular negative samples which, but are more computationally expensive for creating the triplets. Consequently. In this paper we apply the random triplets sampling approach [9] and try to accelerate the convergence and reduce the risk of being stuck in local optima.

Among all triplets we generate, we pick up two cases: the worst and the best, as they are more important than others. The best triplets contribute to speed up the convergence while the worst possibly may weaken the ability of the training and result in poorly trained network. Hence, we increase the training loss for worst triplets to enable the network to find the better weights for them.

Firstly, we determine the worst and best cases in random triplets sampling. We can solve the penalty/reward function through optimization - find the solution while satisfying the constraints. For example, a general constrained minimization problem may be written as follows:

$$z = f(x), \quad \text{s.t: } h(x) > 0 \quad (7)$$

where $h(x)$ is the constraint and $f(x)$ is the objective function that needs to be optimized. This constrained problem

can be solved by transforming it to an unconstrained problem, with addition of a penalty function. This approach replaces a constrained optimization problem by a series of unconstrained problems whose solutions ideally converge to the solution of the original constrained problem. The aforementioned minimization problem, it can be transformed to the unconstrained problem as:

$$z = f(x) + \beta \times (h(x)) \tag{8}$$

where α is the balanced parameter. For the triplet loss function, the worst case is when the positive samples are too far away from the anchor, i.e.

$$\|f(\mathbf{x}_a) - f(\mathbf{x}_p)\|_2^2 > \|f(\mathbf{x}_a) - f(\mathbf{x}_n)\|_2^2 + m \tag{9}$$

So we penalize those samples by adding a penalty term:

$$A_{penalty} = \beta \times \left[\frac{\|f(\mathbf{x}_a) - f(\mathbf{x}_p)\|_2^2 + \|f(\mathbf{x}_a) - f(\mathbf{x}_n)\|_2^2}{2} \right] \tag{10}$$

Apart from the worst cases, some triplets may produce the best cases. The best cases meet the constrain as:

$$\|f(\mathbf{x}_a) - f(\mathbf{x}_p)\|_2^2 - \|f(\mathbf{x}_a) - f(\mathbf{x}_n)\|_2^2 + m < 0$$

. This constrain however means weights will not be updated, preventing the network from being trained. A large distance however will slow down the convergence of network. Therefore, we tune distance so that it is within a particular small-range:

$$\varepsilon < \|f(\mathbf{x}_a) - f(\mathbf{x}_p)\|_2^2 - \|f(\mathbf{x}_a) - f(\mathbf{x}_n)\|_2^2 + m \leq 2\varepsilon \tag{11}$$

where $\epsilon = km$ and $0 < k < 1$. Therefore, the best triplets satisfy both constraints below:

$$\|f(\mathbf{x}_a) - f(\mathbf{x}_p)\|_2^2 - \|f(\mathbf{x}_a) - f(\mathbf{x}_n)\|_2^2 > km - m \tag{12}$$

$$\|f(\mathbf{x}_a) - f(\mathbf{x}_p)\|_2^2 - \|f(\mathbf{x}_a) - f(\mathbf{x}_n)\|_2^2 < 2km - m \tag{13}$$

The reward item A_{reward} is :

$$A_{reward} = \beta \times \left[\frac{\|\mathbf{x}_a - \mathbf{x}_n\|_2^2 - \|\mathbf{x}_a - \mathbf{x}_p\|_2^2}{2} \right] \tag{14}$$

Finally, the adaptive margin can be summarized as:

$$\begin{cases} mean + \alpha s + \beta A_{penalty}, & \text{Worst Case} \\ mean + \alpha s - \beta A_{reward}, & \text{Best Case} \\ mean + \alpha s, & \text{O.W} \end{cases}$$

IV. Experimental

A. Image Recognition

Parameter Setting: The experiments are conducted on FasionMINIST dataset, that is a large database, containing 10 different commodities, and used for various image processing tasks. FasionMINIST consists of a training set of 60,000 examples and a test set of 10,000 examples. For our experiments on few-shot learning, we randomly sample and choose 500 images from FasionMINIST. Our work is implemented using Keras [4] in a 8-core PC, i7-6700 3.4 GHz with 8GB RAM to train the network with 20,000 of iterations.

Network Structure: As we focus more on margin distance for loss function, we take a simple network for our few-shot learning in image recognition: three stacks of Convolutional-Pool layers, plus one fully connected layer.

Experiments: Three images are put into the network, which then maps them into the embedding space. In the embedding space, the images from same class, i.e. the anchor and positive pair, should be near to each other with a small distance, while the distance between the anchor and the negative is large. Therefore, we set a threshold for this determination, as in class [4]. Low threshold may produce false negatives while a high one may result in false positives. This is a "ROC curve problem", and we take "Area Under Curve (AUC)" metric to evaluate our performance. We evaluate the recall after we have chosen appropriate threshold which making the false positive rate is under 10e-3. As seen in Fig. 2, the AUC started from 0.721 and finished at 0.961. Comparing test images with pictures from other classes, clearly we can see that the distance between the test image and pictures in same class is much smaller than that between the test image and pictures from other class.

Apart from AUC, we also use the interlace distance in the embedding space to evaluate the effectiveness of our network. As shown in Fig. 3, the average distance between classes grows from 0.5 to 1.5. This clearly demonstrates the effective embedding of our proposed approach.

We conducted the simulation of the proposed triplet for few-shot image recognition in FashionMNIST dataset and compared with the basic triplet loss. In Table I, our proposed triplet loss outperforms the basic triplet loss for both AUC and recall.

TABLE I
Comparison of the Proposed Triplet Loss with Adaptive Margin with the Basic Triplet

Method	AUC	Recall
Standard Triplet loss	0.997	77.7%
Triplet loss with penalty+reward	**0.998**	**87.3**%

B. Image Segmentation

We also employ our methods to CoSegNet for image co-segmentation using CMU-Cornell iCoseg dataset [2]. CoSegNet [1] is composed of a siamese network and a decision network, trained by end-to-end strategy. CogSeNet uses the standard triplet loss to calculate the loss of siamese net based on the pairs of images (positive, negative). We utilize an extra image as anchor, so that we can calculate the three features and use our proposed triplet loss to train the siamese net. Fig. 4 illustrates the CoSegNet model modified by our contribution. Fig. 5 shows the segmentation results using our proposed triplet loss function. Obviously the results are satisfactory and demonstrates the effectiveness of our approach.

We apply the Precision and Jaccard Index, which are the intersection over union of the co-segmentation result and the

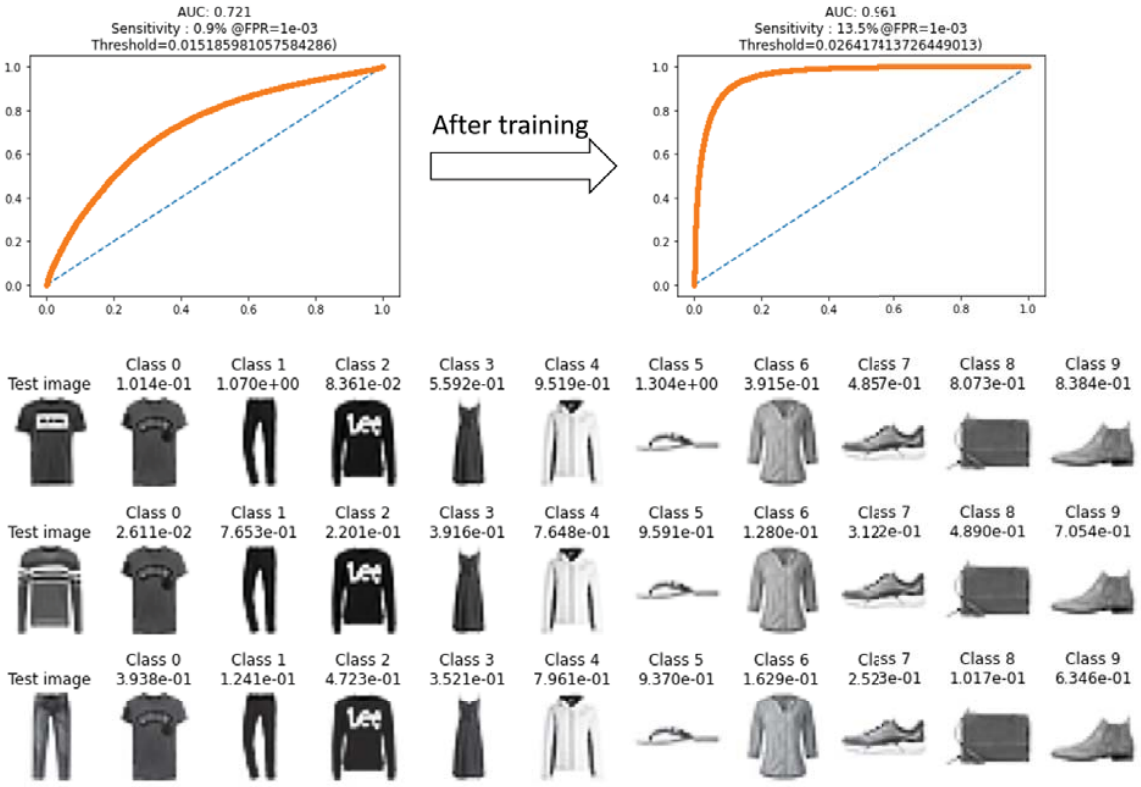

Fig. 2 Top: AUC and sensitivity of the proposed triplet loss. Down: Similarity and distance between three classes T-shirt, Trouser, and Pullover with other classes for one-shot image recognition

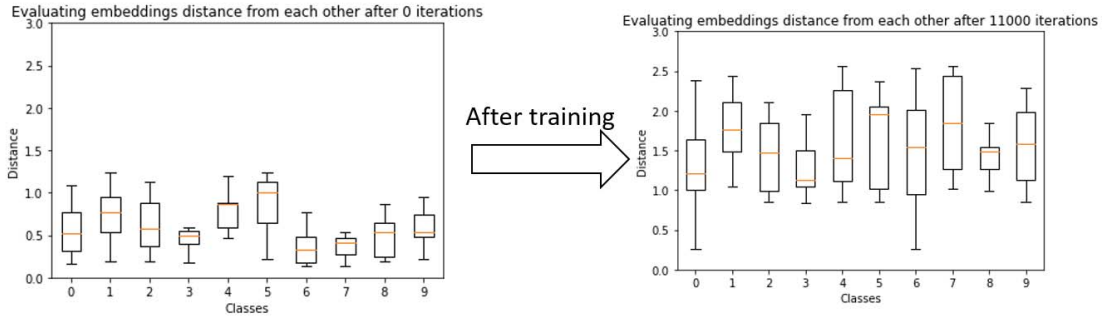

Fig. 3 Distance in embedding space after 11000 iterations

TABLE II
COMPARISON OF PRECISION AND JACCARD INDEX

Method	Precision	Jaccard Index
Standard Triplet loss	69.06	0.61
Triplet loss with penalty+reward	**73.3**	**0.67**

ground truth common foreground segment, to evaluate the performance. Table II is the results. Apparently, using our adaptive triplet loss improves the performance, with the better performance for both metrics being achieved. It is notable that the simulation for all of loss functions have been done using the same parameters, number of iterations, and pre-trained weights. It is expected that more iterations could improve the segmentation results.

V. CONCLUSIONS

An adaptive margin strategy and a penalty/reward optimization function have been proposed to improve standard triplet loss function for metric learning. After calculating the data-relative margin in one mini-batch, the worst and best triplets in random triplets sampling are selected. The worst triplets are given a penalty term whereas the best a reward, which makes our model benefit from a better convergence speed. Moreover, our method is evaluated on image recognition task and image co-segmentation, which shows a better performance.

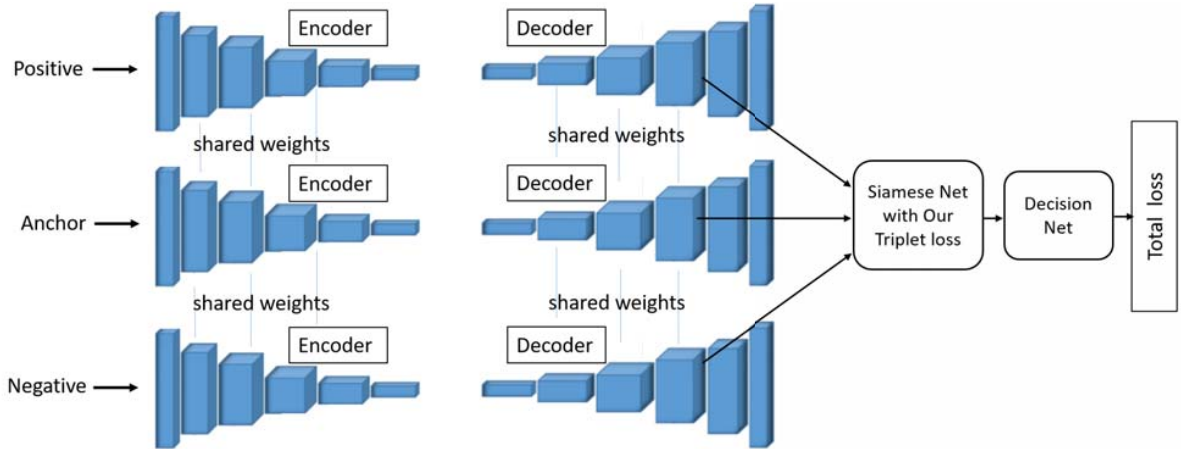

Fig. 4 Illustration of the modified CoSegNet, inspired by [1]. Our contribution is shown in red colors

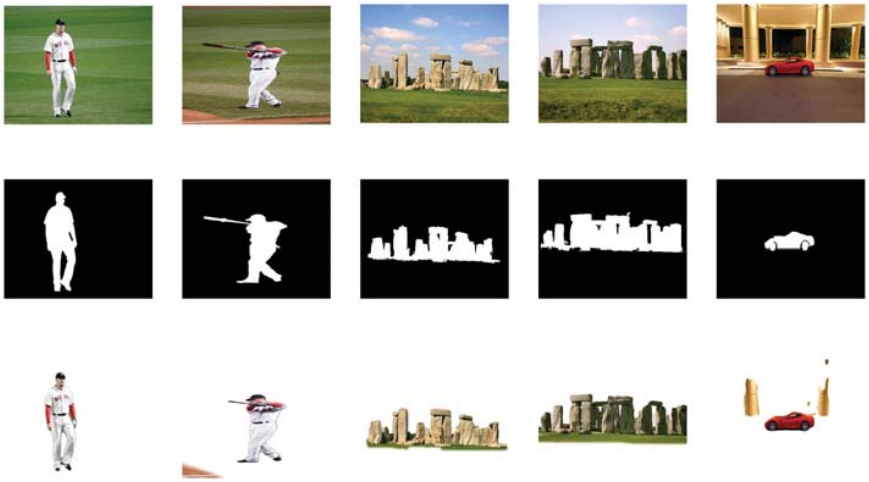

Fig. 5 Illustration of co-segmentation. The first row are the original images, the second row are the corresponding ground truth, and the segmented results are shown in the last row

ACKNOWLEDGMENTS

This work is supported by Ministry of Science and Technology China(MOST) Major Program on New Generation of Artifcial Intelligence 2030 No. 2018AAA0102200. This work is also supported by Natural Science Foundation China (NSFC) Major Project No. 61827814 and Shenzhen Innovation Council of Science and Technology Project No.JCY20190808153619413. This work is also supported by the National Engineering Laboratory for Big Data System Computing Technology,China.

REFERENCES

[1] S. Banerjee, A. Hati, S. Chaudhuri, and R. Velmuru-ga, " Cosegnet: Image co-segmentation using a conditional siamese convolutional network," in *IJCAI,* pages 673–679,2019.

[2] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen, "icoseg: Interactive co-segmentation with intelligent scribble guidance," *In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* pages 3169–3176.IEEE, 2010.

[3] S. Changpinyo, W.-L. Chao, and F. Sha., "Predicting visual exemplars of unseen classes for zero-shot learning," *IEEE international conference on computer vision,* pages 3476–3485, 2017.

[4] E. Craeymeersch., "One-shot learning, siamese networks and triplet loss with keras," https://github.com/CrimyTheBold/tripletloss/, 2019.

[5] C. Ding and D. Tao, "Trunk-branch ensemble convolutional neural networks for video-based face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 40(4):1002–1014, 2018.

[6] C. B. Do and S. Batzoglou, "What is the expectation maximization algorithm?" *Nature biotechnology* 26(8):897–899, 2008.

[7] A. Dominguez-Sanchez, M. Cazorla, and S. Cogdell. "Orts-Escolano. Pedestrian movement direction recognition using convolutional neural networks," *IEEE Transactions on Intelligent Transportation Systems,* 18(12):3540–3548, 2017.

[8] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong. "Transductive multi-view zero-shot learning." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(11):2332– 2345, 2015.

[9] W. Ge."Deep metric learning with hierarchical triplet loss." *European Conference on Computer Vision (ECCV)*, pages 269–285, 2018.

[10] E. Hoffer and N. Ailon. "Deep metric learning using triplet network". In International Workshop on Similarity-Based Pattern Recognition, pages 84–92. Springer, 2015.

[11] L. Karlinsky, J. Shtok, S. Harary, E. Schwartz, A. Aides, R. Feris, R. Giryes, and A. M. Bronstein. "Repmet: Representative-based metric learning for classification and few-shot object detection." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5192–5201, 2019.

[12] G. Koch, R. Zemel, and R. Salakhutdinov. "Siamese neural networks for one-shot image recognition". *In ICML deep learning workshop,* volume 2. Lille, 2015.

[13] E. Kodirov, T. Xiang, and S. Gong. "Semantic autoencoder for zero-shot learning." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pages 4447– 4456, 2017.
[14] C. H. Lampert, H. Nickisch, and S. Harmeling. "Learning to detect unseen object classes by between-class attribute transfer." *IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958, 2009.
[15] Y. Li, D. Wang, H. Hu, Y. Lin, and Y. Zhuang. "Zero-shot recognition using dual visual-semantic mapping paths". *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pages 3279–3287, 2017.
[16] Li Fe-Fei, Fergus, and Perona. "A bayesian approach to unsupervised one-shot learning of object categories." *In Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1134–1141 vol.2, 2003.
[17] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. "A simple neural attentive meta-learner". 2017.
[18] P. M¨uller. "Model-agnostic meta-learning (maml) for fast adaptation of deep networks."
[19] F. Schroff, D. Kalenichenko, and J. Philbin. "Facenet: A unified embedding for face recognition and clustering." *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pages 815–823, 2015.
[20] J. Snell, K. Swersky, and R. S. Zemel. "Prototypical networks for few-shot learning". 2017.
[21] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales."Learning to compare: Relation network for few-shot learning." *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pages 1199– 1208, 2018.
[22] T.Munkhdalai and H. Yu. "Meta networks." *In 2017 Proceedings of machine learning research,* pages 2554–2563, 2017.
[23] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. "Matching networks for one shot learning." *In Advances in neural information processing systems,* pages 3630–3638, 2016.
[24] X. Wang, F. Yu, R. Wang, T. Darrell, and J. E. Gonzalez. "Tafe-net: Task-aware feature embeddings for low shot learning." *In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* 2020.
[25] Y. Wang, X.-M. Wu, Q. Li, J. Gu, W. Xiang, L. Zhang, and V. O. Li. "Large margin few-shot learning." 2018.
[26] M. Zhu, D. Shi, M. Zheng, and M. Sadiq. "Robust facial landmark detection via occlusion-adaptive deep networks." *In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* pages 3481–3491, 2019.