

# A Conditional Triplet Loss for Few-shot Learning and its Application to Image Co-Segmentation

Daming Shi<sup>a</sup>, Maysam Orouskhani<sup>a,\*</sup> and Yasin Orouskhani<sup>b</sup>

<sup>a</sup>College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China

<sup>b</sup>Department of Machine Learning, Rahnama Corporation, Tehran, Iran

## ARTICLE INFO

### Keywords:

Conditional Triplet loss

Few-shot learning

Metric learning

Siamese network

Co-segmentation

## ABSTRACT


Few-shot learning tries to solve the problems that suffer the limited number of samples. In this paper we present a novel conditional Triplet loss for solving few-shot problems using deep metric learning. While the conventional Triplet loss suffers the limitation of random sampling of triplets which leads to slow convergence in training process, our proposed network tries to distinguish between samples so that it improves the training speed. Our main contributions are two-fold. (i) we propose a conditional Triplet loss to train a deep Triplet network for deep metric embedding. The proposed Triplet loss employs a penalty-reward technique to enhance the convergence of standard Triplet loss. (ii) we improve the performance of the existing image co-segmentation model by replacing the conventional loss function by our proposed conditional Triplet loss. To demonstrate the performance of the proposed network, experiments carry out on MNIST and CIFAR. Simulation results are evaluated by AUC and Recall (sensitivity) and indicate that the proposed conditional Triplet network achieves higher accuracy in comparison to state-of-the-arts.

## 1. Introduction

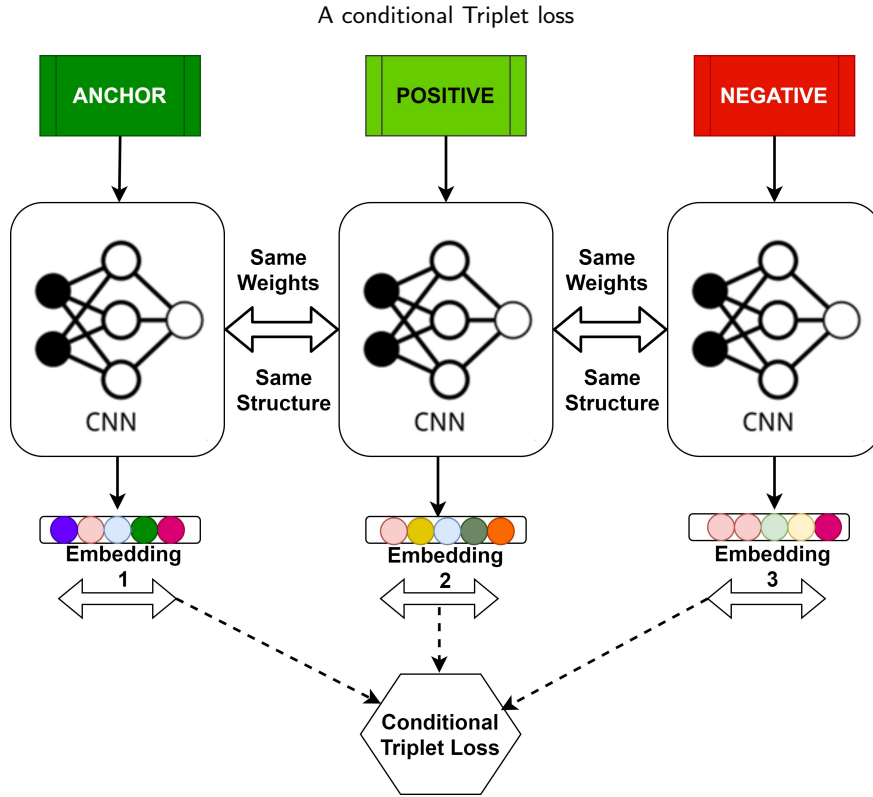
Whereas most modern machine learning algorithms such as deep neural networks require training on hundreds of samples, in some problems such a large number of examples is unavailable. Few-shot learning tries to solve these type of problems and train a computational model by using few number of examples. These algorithms need only few training example for each class and generalize the model to unfamiliar categories without extensive retraining. Therefore, the main goals of few-shot learning are to achieve a good generalization ability to new examples and reach high accuracy with very limited data. To solve few-shot problems, approaches such as metric learning algorithms are presented. Metric learning employs some distance measures to find the similarity between images and aim to learn to compare the inputs. The task of distance (similarity) metric learning is to learn a distance function over images in the different tasks of computer vision (Ge, 2018). With the advent of modern deep neural networks, metric learning are changing to the deep feature embeddings that better fit a simple distance function, such as Euclidean distance or cosine distance. Deep Siamese network, Matching, and Relation networks are examples of deep metric learning. The deep Siamese network (Koch, Zemel & Salakhutdinov, 2015) is composed of two twin networks and is given a pair of images. This network tries to learn the the similarity between given images. The network uses contrastive loss to calculate the distance between two generated feature embeddings, provided by the last layer, to determine the similarity rate of two images. If the calculated rate is less than the predefined threshold, it means that two input images belong to the same class. In addition to the deep Siamese network, Matching networks (Vinyals, Blundell, Lillicrap, Wierstra et al., 2016) consider the relationship among the training examples and learn a classifier for any given support set while Relation networks (Sung, Yang, Zhang, Xiang, Torr & Hospedales, 2018) learn a deep distance metric instead of a fixed metric function.

The most important element of a deep metric algorithm is loss function. Recently, a number of loss functions are presented to encourage samples from the same class to be closer, and pushing samples of different classes apart from each other. Generally, the loss functions are divided into pair-based and proxy-based losses. The pair-based losses calculate the pairwise distances between data in the embedding space. For example, Contrastive loss (Chopra, Hadsell & LeCun, 2005) minimizes the distance between a pair of images if their class labels are same and to separate them

\*Corresponding author

 dshi@szu.edu.cn (D. Shi); maysam.orouskhani@szu.edu.cn (M. Orouskhani)

ORCID(s):



**Figure 1:** Conditional Triplet Network. Contrary to the standard Triplet network, our model employs a conditional Triplet loss to find the distance between anchor, positive, and negative samples. The details are given in Section3.

otherwise. Other recently proposed losses in this category investigate a group of pairwise distances to handle relations between more than two data. (Kim, Seo, Laptev, Cho & Kwak, 2019; Sohn, 2016; Yu & Tao, 2019). Since pair-based losses take a tuple of data as input, the losses cause prohibitively high training complexity. As a result, it results in slow convergence of training process. Moreover, since we don't have enough information about the quality of samples, some tuples sampled from training data may deteriorate the quality of the embedding space.

The proxy-based losses provide proxies as a representative of a subset of training data to address the high complexity of pair-based losses. Since proxy-based losses use proxies and the number of proxies is clearly less than training data, the complexity of training process is absolutely reduced. However, they associate each data point only with proxies. Therefore, they can leverage only data-to-proxy relations (Kim, Kim, Cho & Kwak, 2020). Moreover, proxies estimation is another notable challenging issue in proxy-based losses.

In this paper, we propose a new pair-based loss function called conditional Triplet loss with penalty-reward approach. Our main goals are to address the limitations of standard Triplet loss and propose a conditional loss. The standard Triplet loss generally uses three sampling strategies to form the selective triplets of anchor-positive-negative including random, hard, and semi-hard sampling. Although the latter sampling can probably increase the network's accuracy, the selection of triplets where the negative is not closer to the anchor than the positive, suffers higher complexity than random selection. As a result, the training process takes longer and needs more time to achieve high accuracy. However, random sampling of triplets may lead to form the triplets that do not update any weights and thus do not contribute the training process. This may cause the network to get stuck in local minimum and slow convergence. To handle the limitation of random sampling in standard Triplet loss we categorize the available triplets into worst triplets, best triplets, and normal. We improve the standard Triplet loss by penalizing the worst case of triplets and giving reward to triplets those meet the best condition. Moreover, we employ our proposed conditional loss function to the image co-segmentation. Our contributions in this work are summarized as follows:

- (i) We propose a conditional Triplet loss using penalty-reward approach to calculate the distance of the embedding

features in a deep Triplet network for deep metric and its application to few-shot image recognition. We try to enhance the convergence of standard Triplet loss. In order to let the network have an appropriate insight of the informative triplets, the best and worst triplets are applied to the network with different adaptive loss function formulas. In this case, the network is penalized for the worst selected triplets and the best lead to rewarding the network.

(ii) We perform an ablation study to show the performance of our proposed conditional Triplet loss and compare with other algorithms. The simulation is conducted on MNIST and CIFAR-10 dataset and the results are indicated by mean and standard deviation of AUC and sensitivity. Moreover, to find out that there is a statistically significant difference between the obtained results by our network and previous works, we perform the Kruskal-Wallis and Mann-Whitney U test.

(iii) In addition to conduct the experiments for one-shot image recognition, we apply our proposed loss function to improve the performance of the existing image co-segmentation model. The existing co-segmentation Siamese network uses cosine distance as traditional loss for distance metric while we apply our conditional Triplet loss as main loss function in model to improve the segmentation accuracy.

## 2. Related Work

In this section, we categorize deep metric learning losses into proxy-based and pair-based losses. Then, we consider the current limitation of existing Triplet loss and mention how our proposed loss function differs from the standard one and handles the mentioned limitation. Section 2.2 reviews the standard Triplet loss and Triplet network.

### 2.1. Pair-based and Proxy-based Losses

The first pair-based loss function is the contrastive loss (Chopra et al., 2005; Hadsell, Chopra & LeCun, 2006). It minimizes the distance between pairs of images belonging to the same class, and maximizes the distance between pairs of images from different class. However, no relation between similar and dissimilar pairs is defined. One of the most popular loss functions is Triplet loss (Schroff, Kalenichenko & Philbin, 2015; Wang, Song, Leung, Rosenberg, Wang, Philbin, Chen & Wu, 2014). The loss is done using triplet of samples (an anchor, a positive and a negative data point) so that it makes the distance between features of the anchor and the positive sample smaller than the distance between the anchor and the negative sample. However, it requires a large batch size to achieve the promising performance. Recent pair-based losses utilize the higher order relations between data and reflect their hardness for further enhancement. As generalizations of Triplet loss, N-pair loss (Sohn, 2016) and Lifted structure loss (Oh Song, Xiang, Jegelka & Savarese, 2016) compare the anchor and positive samples with multiple negative samples, and pull the positive to the anchor and push the negatives away from the anchor while considering their hardness. Both methods sample the same number of data per negative class, thus may ignore informative examples during training. In contrast to the abovementioned loss functions, Ranked list loss (Wang, Hua, Kodirov, Hu, Garnier & Robertson, 2019b) takes into account all positive and negative data in a batch and tries to form distinct positive and negative sets. Multi-similarity loss (Wang, Han, Huang, Dong & Scott, 2019a) defines three complementary types of similarity to assigns a particular weight for each pair in a batch and aim to find and concentrate more on useful pairs. Pair-based losses use pairs or triplets of data during the training process. As the number of tuples increases polynomially with the number of training data, these methods suffer high complexity. Moreover, since we may sample some useless tuples from the training data, it deteriorates the performance and causes slow convergence (Wu, Manmatha, Smola & Krahenbuhl, 2017).

To address the complexity limitation of pair-based methods, some researchers use proxy-based losses. The first proxy-based loss is Proxy-NCA (Movshovitz-Attias, Toshev, Leung, Ioffe & Singh, 2017) which a single proxy is dedicated for each class in such a way that the positive pair to be close and negative pairs to be far from the proxies. As an extension of SoftMax loss, Soft Triple loss (Qian, Shang, Sun, Hu, Li & Jin, 2019) assigns multiple proxies to each class. By using proxies in N-pair loss, Manifold Proxy loss is introduced (Aziere & Todorovic, 2019). Instead of Euclidean distance, Manifold Proxy employs a manifold-aware distance so that it calculates the semantic distance in the embedding space.

Since pair-based losses such as Triplet loss take a tuple of data as input, the losses cause prohibitively high training complexity. As a result, it results in slow convergence of training process. Moreover, since we don't have enough

information about the quality of samples, some tuples sampled from training data may deteriorate the quality of the embedding space. While Triplet loss suffers the high complexity and slow convergence, a proxy-based loss estimates a subset of training data (proxies) to capture the global structure of an embedding space so that it alleviates the high complexity of pair-based losses. However, since each data point is associated only with proxies, the rich data-to-data relations that are available for the pair-based methods are not accessible anymore. (Kim et al., 2020). Moreover, proxies estimation and how to estimate the proxies is another challenging issue in proxy-based losses. It means that incorrect estimation of proxies may reduce the network's performance and cause wrong distance calculation.

The proposed conditional Triplet loss in this paper belongs to the pair-based losses. The main difference between our proposed loss with standard Triplet loss is that we define an adaptive loss for different samples. While standard Triplet loss assigns a same formula for all samples, we provide informative samples from generated triplets. The large number of samples causes high complexity and leads to slow convergence. However, since we divide the samples into three different cases and an adaptive loss function is dedicated to each case, we may separately consider the effect of each triplets on the network's loss. As a result, it improves the convergence rate. In addition, contrary to some Triplet-based losses which employ hard and semi-hard sampling to generate the informative samples, we use random sampling to reduce the computational cost.

## 2.2. Review of Triplet Loss and Triplet Network

Although the basic deep convolutional Siamese network generally uses contrastive loss function to evaluate the performance of the network, it is shown that the Triplet loss increases the efficiency of the Siamese network and forms a new network entitled Triplet network (Hoffer & Ailon, 2015). A triple samples of input images as anchor, positive, and negative is formed and is given to the Triplet loss. Anchor and positive sample belong to the same class whereas negative sample has different label. Three samples go through the network and the corresponded features for each sample are determined by the last layer. In the embedding space, images from the same class should be close together and form well separated clusters. Therefore, the main goal is to make sure that two examples with the same label have their embeddings close together in the embedding space while two examples with different labels have their embeddings far away. It is notable that samples selection needs to be carefully done in such a way that the negative has to be farther away than the positive by some margin.

Let  $(x_i, y_i)$  be the  $i$ th sample in the training set  $D$  with  $N$  samples. The feature embedding of  $x_i$  is represented as  $\phi(x_i, \theta) \in R^d$ , where  $\theta$  is the learnable parameters of a differentiable deep networks,  $d$  is the dimension of embedding and  $y_i$  is the label of  $x_i$ .  $\phi(\cdot, \theta)$  is usually normalized into unit length for the training stability. During the neural network training, training samples forms the triplets, each of which  $T_z = (x_a, x_p, x_n)$  is consisted of an anchor sample  $x_a$ , a positive sample  $x_p$  and a negative sample  $x_n$ . The labels of the triplet  $T_z = (x_a^z, x_p^z, x_n^z)$  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 from each other. The optimization target of the triplet  $T$  is,

$$\text{loss}(T_z) = \text{Max} \left[ \text{dis}(x_a^z, x_p^z) - \text{dis}(x_a^z, x_n^z) + m, 0 \right] \quad (1)$$

Where  $\text{dis}$  is the Euclidean norm and  $m$  is the violate margin that plays a key role to sample selection in training a Triplet loss in deep metric learning. (Ge, 2018)

## 3. Triplet Loss with Penalty-Reward Approach

The majority of loss functions in deep metric learning treat all training samples equally with a fix margin. In a training process of a network by a given dataset with  $N$  samples, a Triplet loss generates  $O(N^3)$  triplets, which is infeasible to put all triplets into a single mini-batch. Although we randomly sample the triplets over the whole training set and it takes less time than other sampling methods, it increases the risk of slow convergence and stuck in local optima. Since all triplets are treated equally, we propose a conditional loss that fines the worst triplets by adding a penalty term and gives reward to triplets those satisfy the best condition. Other sampling methods such as hard and semi-hard find the particular negative samples, but take more computational time for generating the triplets. Consequently, in this paper we apply the random triplets sampling and try to improve its performance. (Ge, 2018)

### 3.1. Penalty-Reward Approach

In this paper we utilize the random sampling to create the triplets for each mini-batch. It is clear that different anchor, positive, and negative samples generate different cases. Thus in some triplets, negative sample is close to or far away from the anchor sample. Among generated triplets, two cases are more important than others and are not of equal value in terms of network's training. We call them the best and worst case. Since the Triplet network tends to use samples that increase the convergence after calculating the loss and the corresponded gradients, best triplets result in improving the network's convergence speed while worst triplets deteriorate the network's training ability and weights updating doesn't contribute the training process. However, network's training is based on all batch of triplets. Thus the training process must be performed according to all type of triplets and the network has to update its weights based on all triplets. As a result, the training loss should be increased for worst triplets to let the network find the better weights for them.

Therefore, we first determine the worst and best cases in random triplets sampling. First, we describe the penalty function in the theory of optimization. In the optimization problems when we minimize a constrained problem, the goal is not only to find the optimal solutions but also to find the solutions that satisfy the constraints. Therefore, solutions that violate the constraints need to be given a penalty. For example, a general constrained minimization problem may be written as follows:

$$z = f(x), \quad \text{S.t: } h(x) > 0 \quad (2)$$

Where  $h(x)$  is the constraint that needs to be satisfied and  $f(x)$  is the objective function that needs to be optimized subject to the constraint. This problem can be solved by transforming to an unconstrained problem and adding a penalty function. A penalty method replaces a constrained optimization problem by a series of unconstrained problems whose solutions ideally converge to the solution of the original constrained problem. The unconstrained problems are formed by adding a term, called a penalty function, to the objective function that consists of a penalty parameter multiplied by a measure of violation of the constraints. The measure of violation is nonzero when the constraints are violated and is zero in the region where constraints are not violated. **For the abovementioned minimization problem, it can be transformed to the unconstrained problem as follows where  $f(x)$  is the objective function,  $h(x)$  is the penalty function, and  $\alpha$  is the penalty coefficient that determines the penalty scale:**

$$Z = f(x) + \alpha \times (h(x)) \quad (3)$$

For the Triplet loss function with random sampling the worst case scenario occurs when the positive sample is too far away from the anchor sample. In this case, we use the margin criterion to measure the distance between two samples. In other words, the worst triplets meet the following constraint:

$$\text{dis}(x_a^z, x_p^z) > \text{dis}(x_a^z, x_n^z) + m \quad (4)$$

So we penalize them by adding the penalty term. The loss function for the worst triplets,  $TW_z$  is formulated as follows:

$$\text{loss}(\mathcal{TW}_z) = \underbrace{\text{Max} \left[ \text{dis}(x_a^z, x_p^z) - \text{dis}(x_a^z, x_n^z) + m, 0 \right]}_{\text{loss}} + \alpha \times \underbrace{\left[ \frac{\text{dis}(x_a^z, x_p^z) + \text{dis}(x_a^z, x_n^z)}{2} \right]}_{\text{penalty}} \quad (5)$$

In contrast to the worst triplets, some triplets may produce the best case. As we consider the distance as:

$$\underbrace{\text{dis}(x_a^z, x_p^z) - \text{dis}(x_a^z, x_n^z) + m}_{\text{distance}}$$

the ideal case is when distance is less than zero in order to minimize the network's error. As it is clear, for *loss* minimization we have to minimize the *Max* between *distance* and 0. We need the *distance* equals to zero. However, zero value prevents the network to be trained well and network's weights aren't updated. So *distance* should be close to zero, but not absolute zero in order to allow the network to update the weights. As it is known, we have considered this as a small positive coefficient of margin ( $\epsilon = km$ ). On the other hand, if a very large *distance* is selected, the network is allowed to select the easy triplets where the distance between the positive sample and anchor is much less than the negative distance. Due to the large number of samples, very large *distance* results in slow convergence. Therefore, the best case appears when *distance* locates in a particular small range not very bigger than zero. To satisfy both goals simultaneously we consider the *distance* in the range of  $[\epsilon, 2\epsilon]$ . Other factors of epsilon such as  $3\epsilon$ ,  $4\epsilon$  may locate the *distance* in a wide range where causes divergence of the network.

Therefore, the best case appears when distance locates in a particular small range not very bigger than zero as follows:

$$\epsilon < \text{dis}(x_a^z, x_p^z) - \text{dis}(x_a^z, x_n^z) + m \leq 2\epsilon \quad (6)$$

We consider the  $\epsilon$  parameter as a small value and a factor of the margin:  $\epsilon = km$  where  $0 < k < 1$ . Therefore, we declare the best triplets those satisfy both constraints as:

$$\begin{cases} \text{dis}(x_a^a, x_p^z) - \text{dis}(x_a^z, x_n^z) > km - m \rightarrow \text{dis}(x_a^z, x_p^z) - \text{dis}(x_a^z, x_n^z) > m(k-1) \\ \text{dis}(x_a^z, x_p^z) - \text{dis}(x_a^z, x_n^z) < 2km - m \rightarrow \text{dis}(x_a^z, x_p^z) - \text{dis}(x_a^z, x_n^z) < m(2k-1) \end{cases} \quad (7)$$

Therefore, we reward the best triplets,  $T\mathcal{B}_z$  as,

$$\text{loss}(T\mathcal{B}_z) = \underbrace{\text{Max} \left[ \text{dis}(x_a^z, x_p^z) - \text{dis}(x_a^z, x_n^z) + m, 0 \right]}_{\text{loss}} - \alpha \times \underbrace{\left[ \frac{\text{dis}(x_a^z, x_n^z) - \text{dis}(x_a^z, x_p^z)}{2} \right]}_{\text{reward}} \quad (8)$$

Finally, the conditional triplet loss can be summarized as:

$$\text{Conditional Triplet Loss} = \begin{cases} \text{loss} + \text{penalty}, & \text{Worst Case} \\ \text{loss} - \text{reward}, & \text{Best Case} \\ \text{loss}, & \text{O.W} \end{cases}$$

## 4. Experimental Results

In this section, our method is evaluated and compared to standard Triplet network and deep Siamese network on the two benchmark datasets for image recognition. We also analyze the obtained results by two statistical tests to demonstrate the significance difference. Besides, we employ our proposed conditional Triplet loss for a pre-designed image segmentation model.

### 4.1. Image Recognition

- **Parameter Setting and Dataset:** The proposed conditional network is implemented by Keras (Craeymeersch, 2019). Also, all experiments are conducted using a 4-core PC, i7-6700 3.4GHz, with 8GB of RAM. We use a fix margin set to 0.2, random sampling, and train the network by 20000 iteration. As for dataset, we use the MNIST database (LeCun, Bottou, Bengio & Haffner, 1998) as a large database of handwritten digits that is commonly used for training various image processing systems. It contains 60,000 training images and 10,000 testing images. But as we are experimenting on few-shot learning approach, we have randomly sample and choose only 500 images of available dataset. The next dataset is CIFAR-10 (Krizhevsky, Hinton et al., 2009)



consisting of 60000 32x32 color images of 10 classes (of which 50000 are used for training only, and 10000 for test only).

- **Network Structure:** Since in this case we are focusing more on the loss function and show the performance of our conditional Triplet loss, the network's architecture is not very relevant. A very simple network is taken as a few-shot learning network in image recognition as follows: three stacks of Convolutional/Pooling layers + one fully connected (No activation function in the final layer needs to be set to get full embedding). Three 2D Convolutional layers with kernel size of (7,7)-(3,3) and feature map dimensions of 128-128-256, a Relu non-linearity is applied between two consecutive layers, and *He – uniform* weights initialization. Moreover, we employ two 2D MaxPooling as pooling layer. As indicated, no activation function in the embedding layer, but a flatten layer to transform a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier.
- **Experiments:** This section presents the simulation results of running the proposed network and compares it with previous works. Training on dataset is conducted by Adam training algorithm with initial learning rate of 0.00006, total number of parameters 4688522, batch size of 10, and number of training iteration equals to 5000. Some random parameters such as random selection of samples and random value of the initial weights exist in this work. Therefore, to avoid randomness bias, we perform multiple runs of experiments. In each experiment we employ a different batch samples and then we calculate the average and standard deviation of the results for both performance measures.

Three input images are given to the network and the network produces the features embeddings for each image. Therefore, we have to compute the distance of features. If two images are from the same class, the distance should be “low” while “high” distance indicates that two inputs are from different classes. Thus we need a threshold to analyze the found distance and decide whether two inputs belong to the same class: if the distance is under the threshold then it's a “same” decision, if the distance is above the threshold then it's a “different” decision (Craeymeersch, 2019). Using too low threshold means we will have high precision but also too many false negatives while too high means too many false positives. To measure the performance of the proposed network, we employ two measures: AUC and Recall. The recall measure declares how many of the true positives were found as follows:

$$Recall = \frac{\text{Number of True Positives}}{\text{Number of True Positives} + \text{Number of False Negatives}} \quad (9)$$

Besides, an ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds and plots True Positive Rate (TPR = recall) vs False Positive Rate (FPR).

$$FPR = \frac{\text{Number of False Positives}}{\text{Number of False Positives} + \text{Number of True Negatives}} \quad (10)$$

AUC (Area under the ROC Curve) computes any point in an ROC. AUC measures the entire two-dimensional area (Integral) underneath the entire ROC curve from (0,0) to (1,1). It also determines the probability that the model ranks a random positive example more highly than a random negative example. In general, ROC is a probability curve while AUC represents degree of separability and indicates the ability of the model in distinguishing between different classes. In this paper, the threshold is chosen in such a way that the False Positive Rate is under  $10e-3$ .

After running the Siamese network with conditional Triplet loss, figure 2 illustrates that the AUC starts from 0.704 (before training) and finishes at 0.998 after training. Moreover, sensitivity reaches its maximum value at 87.2%. Comparing few tests images against a reference picture from each class clears that the distance between the test image vs another image of the same class is low and the distance between the tests image vs another class is high. For example, for the test image belongs to class 1, the distance to the sample in a same class is

equal to  $8.505e-3$  and distance to digits of other classes is more than  $5.91e-01$ .

Another performance measure to evaluate the effectiveness of the network is interclass distance which means that how “far” the generated embeddings from each class are from each other. As shown in figure 3, distances between classes moves from an average about 0.2 to 1.3. This clearly indicates that their respective embeddings are better produced by our network. It is notable that we run the network just for a limited iteration to show the efficiency of the proposed loss. It is clear that training the network over more iteration obtains higher AUC and sensitivity.

#### • Ablation Study:

In this section we conduct the simulation of the proposed Triplet for few-shot image recognition in two datasets: MNIST and CIFAR-10. Ablation study analyzes the effect of each contribution on the network’s performance separately and compares the results with Siamese convolutional network and standard Triplet network. Moreover, to avoid randomness bias, the results are indicated by mean and standard deviation of multiple runs. The simulation result and ablation study show the higher performance of the proposed Triplet loss. As it is mentioned in Table 1, for MNIST dataset Triplet loss with penalty and reward obtains 0.998 and 87.2% for AUC and recall respectively which outperforms the basic Triplet loss for both measures. Similarly, Table 2 indicates that our conditional Triplet loss achieves the highest efficiency compared to other methods. It achieves 0.9926 and 71.3% for AUC and Recall.

In addition to indicate the average and standard deviation of AUC and Recall, we have employed Kruskal–Wallis test to statistically analyze the results. The Kruskal–Wallis test is a nonparametric method for comparing two or more independent samples of equal or different sample sizes. A significant Kruskal–Wallis infers that at least one sample stochastically dominates the other (Helbig & Engelbrecht, 2013; Orouskhani, Shi & Cheng, 2020a). Thus we use this test to determine whether there is a statistically significant difference (by confidence level of 95%) between the results achieved by Siamese network, Triplet network, and conditional Triplet network for MNIST and CIFAR-10 dataset. Statistically significant difference between results is shown as bold in Table 3. The computed p value in Table 3 shows that for both AUC and Recall, there is a statistically significant difference between the results obtained by three algorithms in MNIST and CIFAR-10. While p value of the Kruskal–Wallis confirms a statistically significant difference, Mann–Whitney U tests determines which algorithms statistically achieve significant results (Helbig & Engelbrecht, 2013; Orouskhani, Shi & Orouskhani, 2020b). This test is also performed under confidence level of 95% and the results are indicated in Table 4-5. As shown, there is a statistically significant difference between all three networks for both measures of AUC and Recall in both datasets.

**Table 1**

Comparison of the results obtained by the conditional Triplet loss with previous works. Dataset: MNIST

Method	AUC	Recall (Sensitivity)
Siamese Net	$0.994 \pm 0.1$	$74.2 \pm 0.095\%$
Triplet Net with standard Triplet loss	$0.996 \pm 0.08$	$77.1 \pm 0.1\%$
Triplet Net with Triplet loss with penalty	$0.9981 \pm 0.077$	$80.8 \pm 0.09\%$
Triplet Net with Triplet loss with penalty+reward	<b><math>0.9985 \pm 0.079</math></b>	<b><math>86.1 \pm 0.08\%</math></b>

**Table 2**

Comparison of the results obtained by the conditional Triplet loss with previous works. Dataset: CIFAR-10

Method	AUC	Recall (Sensitivity)
Siamese Net	$0.988 \pm 0.05$	$63.6 \pm 0.003\%$
Triplet Net with standard Triplet loss	$0.990 \pm 0.03$	$65.1 \pm 0.03\%$
Triplet Net with Triplet loss with penalty	$0.9921 \pm 0.03$	$69.8 \pm 0.03\%$
Triplet Net with Triplet loss with penalty+reward	<b><math>0.9926 \pm 0.035</math></b>	<b><math>71.3 \pm 0.05\%</math></b>



**Table 3**  
p values of Kruskal-Wallis test

Dataset	AUC	Recall (Sensitivity)
MNIST	0.015	0.0082
CIFAR-10	0.048	0.029

**Table 4**  
Results of Mann–Whitney U test for MNIST dataset

Measure	Network	Siamese net	Triplet net	conditional Triplet net
AUC	Siamese net	n/a		
AUC	Triplet net	0.031	n/a	
AUC	conditional Triplet net	0.062	0.033	n/a
Recall	Siamese net	n/a		
Recall	Triplet net	0.011	n/a	
Recall	conditional Triplet net	0.0074	0.0091	n/a

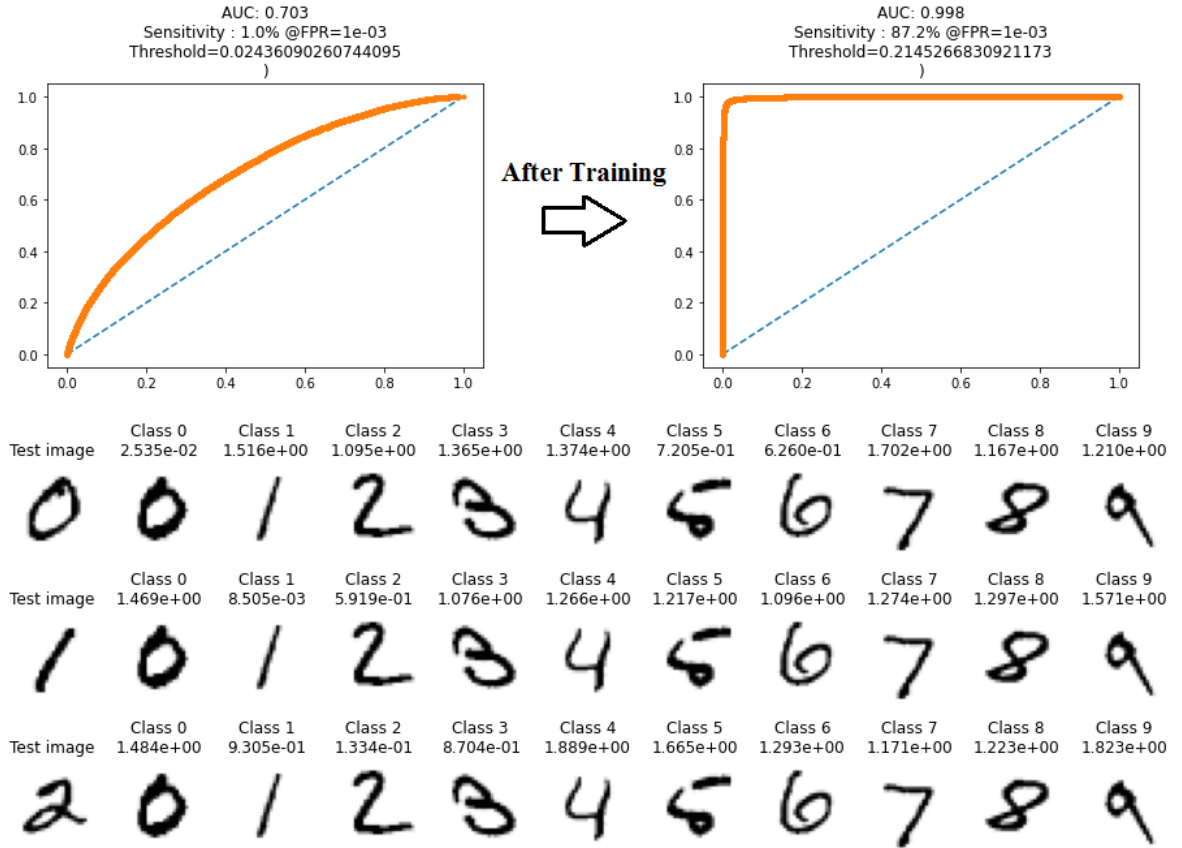
**Table 5**  
Results of Mann–Whitney U test for CIFAR-10 dataset

Measure	Network	Siamese net	Triplet net	conditional Triplet net
AUC	Siamese net	n/a		
AUC	Triplet net	0.014	n/a	
AUC	conditional Triplet net	0.021	0.040	n/a
Recall	Siamese net	n/a		
Recall	Triplet net	0.0081	n/a	
Recall	conditional Triplet net	0.00063	0.0079	n/a

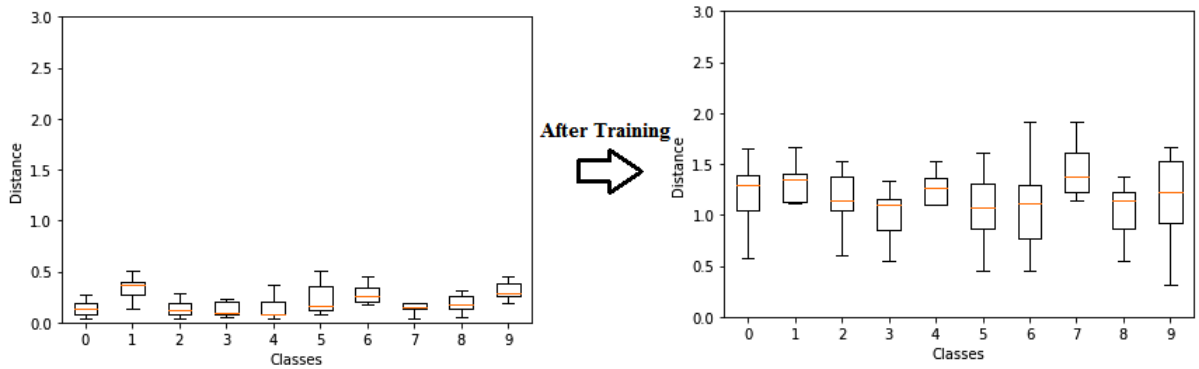
## 4.2. Image Segmentation

In this section we employ our conditional Triplet loss to CoSegNet, a pre-designed model, for image co-segmentation. CoSegNet (Banerjee et al., 2019) as a novel deep convolution neural network based end-to-end co-segmentation model is composed of a Siamese network and decision network. Although this model reflects a good performance compared to state-of-the-art methods on challenging co-segmentation datasets, experiments show that using our conditional Triplet loss improves the model's efficiency. This model uses three different losses for training the whole structure. In the CoSegNet, the standard Triplet loss is employed to calculate the loss of Siamese net based on the pairs of images (positive, negative). However, we utilize an extra image as anchor and pass it through the network. Thus we calculate the features of three passed samples and use our conditional Triplet loss to train the Siamese net. Figure 4 illustrates the CoSegNet model modified by our contribution. We conduct our proposed model to the segmentation of CMU-Cornell iCoseg dataset. (Batra, Kowdle, Parikh, Luo & Chen, 2010)

We evaluate the network's output by using two measures. The first one is Precision, which is the percentage of correctly segmented pixels of both the foreground and the background and the second one is Jaccard Index, which is the intersection over union of the co-segmentation result and the ground truth common foreground segment. As it is shown in Table 6, using conditional Triplet loss improves the performance of CoSegNet and it results in achieving the highest performance for both metrics. We conduct the segmentation of iCoseg dataset by using four losses including: Cosine, triplet, penalty-based triplet, and triplet with penalty and reward approach (our proposed conditional loss). Table 6

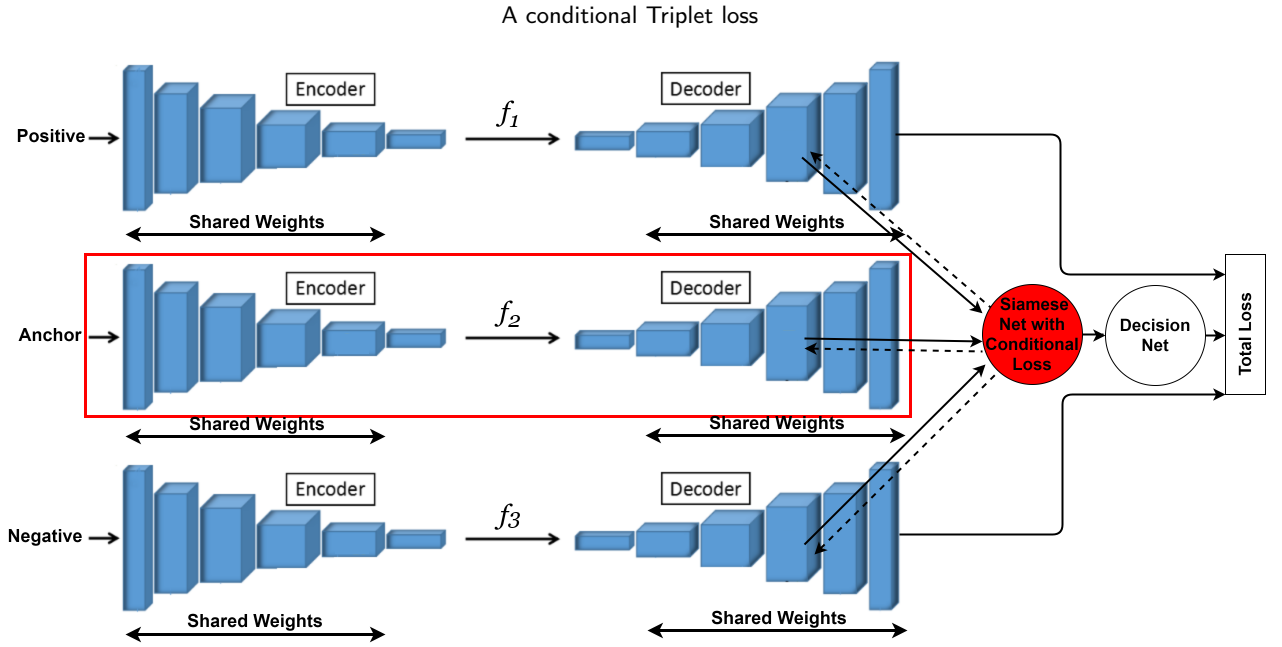


**Figure 2:** Top: AUC and sensitivity of the proposed Triplet loss. Down: Similarity and distance between three classes 0, 1, and 2 with other classes for one-shot image recognition.

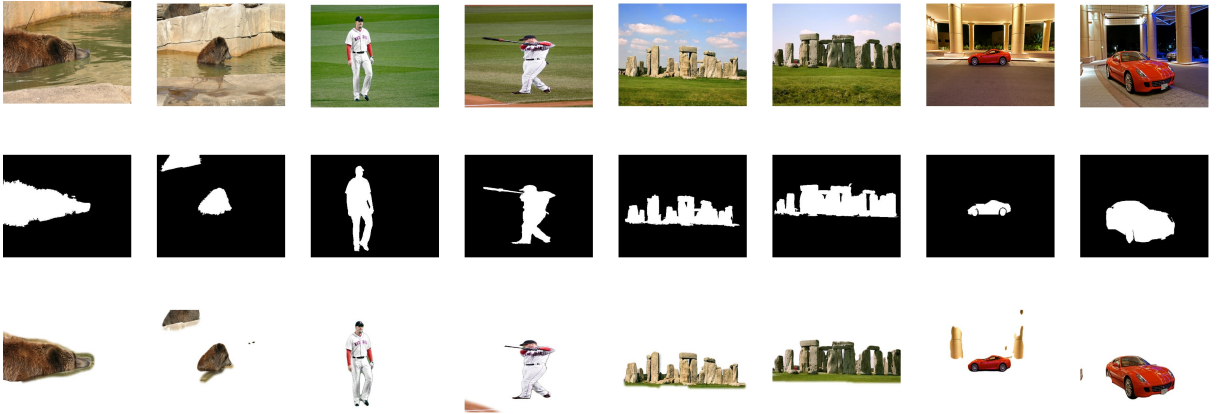


**Figure 3:** Evaluating embeddings distance from each other after 20000 iterations.

indicates that the conditional loss outperforms other cases for Precision and Jaccard index at 73.5 and 0.66 respectively. It is notable that the simulation for all loss functions have been done by the same parameters, equal number of iteration, and pre-trained weights have been used. As main goal is to show the performance of using different losses, we have used the limited number of iteration. It is clear that the more number of iteration results in improving the segmentation results.



**Figure 4:** Illustration of the modified CoSegNet, inspired by Banerjee, Hati, Chaudhuri & Velmurugan (2019). Our contribution is shown by red colors.



**Figure 5:** Illustration of co-segmentation of an image pair. First row shows the original image, second row is for the corresponding ground truth, and the segmented outputs are shown in the last row.

## 5. Conclusions

A novel conditional loss function for deep metric learning is presented. First, the worst and best triplets in random triplets sampling are selected. The worst triplets are given a penalty term whereas a reward is inserted to the loss of the best triplets. This helps the network to achieve higher convergence. Our method is evaluated on the task of few-shot image recognition in MNIST and CIFAR-10 dataset, where it outperforms the standard Triplet loss. Moreover, we employed our proposed conditional loss to improve the efficiency of the CoSegNet. The modified image co-segmentation model shows a better generalization and higher performance for both Precision and Jaccard index.

## 6. Acknowledgement

This work is supported by Ministry of Science and Technology China (MOST) Major Program on New Generation of Artificial 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.

**Table 6**

Comparison of Precision and Jaccard index of the conditional Triplet loss with other losses on the MSRC dataset.

Method	Precision	Jaccard Index
Cosine distance loss	68.2	0.57
Standard Triplet loss	69.06	0.61
Triplet loss with penalty	72.8	0.65
Triplet loss with penalty+reward	<b>73.2</b>	<b>0.66</b>

JCY20190808153619413. This work is also supported by the National Engineering Laboratory for Big Data System Computing Technology, China.

## References

- Aziere, N., & Todorovic, S. (2019). Ensemble deep manifold similarity learning using hard proxies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7299–7307).
- Banerjee, S., Hati, A., Chaudhuri, S., & Velumuran, R. (2019). Cosegnet: Image co-segmentation using a conditional siamese convolutional network. In *IJCAI* (pp. 673–679).
- Batra, D., Kowdle, A., Parikh, D., Luo, J., & Chen, T. (2010). icoseg: Interactive co-segmentation with intelligent scribble guidance. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 3169–3176). IEEE.
- Chopra, S., Hadsell, R., & LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (pp. 539–546). IEEE volume 1.
- Craeymeersch, E. (2019). One-shot learning, siamese networks and triplet loss with keras. <https://github.com/CrimyTheBold/tripletloss/>.
- Ge, W. (2018). Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 269–285).
- Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (pp. 1735–1742). IEEE volume 2.
- Helbig, M., & Engelbrecht, A. P. (2013). Dynamic multi-objective optimization using pso. In *Metaheuristics for Dynamic Optimization* (pp. 147–188). Springer.
- Hoffer, E., & Ailon, N. (2015). Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition* (pp. 84–92). Springer.
- Kim, S., Kim, D., Cho, M., & Kwak, S. (2020). Proxy anchor loss for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 3238–3247).
- Kim, S., Seo, M., Laptev, I., Cho, M., & Kwak, S. (2019). Deep metric learning beyond binary supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2288–2297).
- Koch, G., Zemel, R., & Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*. Lille volume 2.
- Krizhevsky, A., Hinton, G. et al. (2009). Learning multiple layers of features from tiny images, .
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 2278–2324.
- Movshovitz-Attias, Y., Toshev, A., Leung, T. K., Ioffe, S., & Singh, S. (2017). No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 360–368).
- Oh Song, H., Xiang, Y., Jegelka, S., & Savarese, S. (2016). Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4004–4012).
- Orouskhani, M., Shi, D., & Cheng, X. (2020a). A fuzzy adaptive dynamic nsga-ii with fuzzy-based borda ranking method and its application to multimedia data analysis. *IEEE Transactions on Fuzzy Systems*, .
- Orouskhani, M., Shi, D., & Orouskhani, Y. (2020b). Multi-objective evolutionary clustering with complex networks. *Expert Systems with Applications*, (p. 113916).
- Qian, Q., Shang, L., Sun, B., Hu, J., Li, H., & Jin, R. (2019). Softtriplet loss: Deep metric learning without triplet sampling. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 6450–6458).
- Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815–823).
- Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems* (pp. 1857–1865).
- Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., & Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1199–1208).
- Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D. et al. (2016). Matching networks for one shot learning. In *Advances in neural information processing systems* (pp. 3630–3638).

- Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., & Wu, Y. (2014). Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1386–1393).
- Wang, X., Han, X., Huang, W., Dong, D., & Scott, M. R. (2019a). Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5022–5030).
- Wang, X., Hua, Y., Kodirov, E., Hu, G., Garnier, R., & Robertson, N. M. (2019b). Ranked list loss for deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5207–5216).
- Wu, C.-Y., Manmatha, R., Smola, A. J., & Krahenbuhl, P. (2017). Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2840–2848).
- Yu, B., & Tao, D. (2019). Deep metric learning with tuple margin loss. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 6490–6499).