Maintaining Difficulty: A Margin Scheduler for Triplet Loss in Siamese Networks Training

Roberto Sprengel Minozzo Tomchak
Departamento de Informática
Universidade Federal do Paraná
Curitiba, Brazil
robertotomchak@ufpr.br

Oge Marques
College of Engineering and Computer Science
Florida Atlantic University
Boca Raton, U.S.A
omarques@fau.edu

Lucas Garcia Pedroso
Departamento de Matemática
Universidade Federal do Paraná
Curitiba, Brazil
lucaspedroso@ufpr.br

Luiz Eduardo Oliveira
Departamento de Informática
Universidade Federal do Paraná
Curitiba, Brazil
luiz.oliveira@ufpr.br

Paulo Lisboa de Almeida
Departamento de Informática
Universidade Federal do Paraná
Curitiba, Brazil
paulorla@ufpr.br

***Abstract*—The Triplet Margin Ranking Loss is one of the most widely used loss functions in Siamese Networks for solving Distance Metric Learning (DML) problems. This loss function depends on a margin parameter μ, which defines the minimum distance that should separate positive and negative pairs during training. In this work, we show that, during training, the effective margin of many triplets often exceeds the predefined value of μ, provided that a sufficient number of triplets violating this margin is observed. This behavior indicates that fixing the margin throughout training may limit the learning process. Based on this observation, we propose a margin scheduler that adjusts the value of μ according to the proportion of easy triplets observed at each epoch, with the goal of maintaining training difficulty over time. We show that the proposed strategy leads to improved performance when compared to both a constant margin and a monotonically increasing margin scheme. Experimental results on four different datasets show consistent gains in verification performance. Our trained models and source code are available at github.com/robertotomchak/maintaining-difficulty.**

***Index Terms*—Siamese Network, Triplet Margin Loss, Scheduler**

I. Introduction

Distance Metric Learning (DML) tasks focus on training a model to transform samples into feature embeddings such that similar samples are mapped close to each other, while dissimilar samples are mapped further apart. This is a common approach for problems in which the goal is to assess whether two samples belong to the same class [1]. Typical applications include verifying whether two images belong to the same person [1], finding the closest match of an image within a database, as illustrated in Figure 1 [1], tracking objects by comparing them across multiple images [2], or transforming a classification problem into a comparison-based one by matching an instance against labeled samples [3].

This work has been supported by the Brazilian National Council for Scientific and Technological Development (CNPq) – Grants 444820/2024-8 and 405511/2022-1.

Fig. 1. To identify the person in the query image (left), the image is compared with all samples in a database, and the identity corresponding to the smallest distance is selected (right) as the answer. Images from LFW [4].

A common approach to train such models is to employ a loss function that takes triplets of samples into account [5]. These triplets are defined as $T = (A, P, N)$, where A is the anchor sample, P is a positive sample belonging to the same class as A, and N is a negative sample belonging to a different class. The objective is to enforce the distance between the embeddings of A and P to be smaller than the distance between the embeddings of A and N by a given margin, thus clustering similar samples while keeping dissimilar ones apart.

Formally, let $f(X)$ denote the embedding of sample X. The positive distance is defined as $\delta_+(T) = d(f(A), f(P))$ and the negative distance as $\delta_-(T) = d(f(A), f(N))$, where d is a distance function, typically the Euclidean distance [5]. The desired condition is $\delta_+(T) + \mu < \delta_-(T)$, where $\mu \geq 0$ denotes the desired margin. Triplets that do not satisfy this inequality must trigger an update in the model parameters. This behavior is commonly enforced using the Triplet Margin Ranking Loss [5], which is defined as $l(T) = \max(0, \mu + \delta_+(T) - \delta_-(T))$. Triplets that satisfy the margin constraint yield a zero loss and are referred to as easy, while those that violate the constraint, producing a positive loss, are referred to as hard [1], [5].

We define the effective margin of a triplet T as $\hat{\mu}(T) = \delta_-(T) - \delta_+(T)$. From the definitions above, a triplet is easy if and only if $\hat{\mu}(T) > \mu$, and hard otherwise. Consequently, triplets whose effective margin is greater than or equal to the desired margin μ yield a zero loss.

One may incorrectly assume that, since easy triplets produce a zero loss, their effective margins remain unchanged over time. Under this assumption, triplets would tend to cluster around the desired margin μ, and there would be no incentive for the model to further increase the separation of easy triplets. However, we show that this intuition is incorrect. Despite yielding zero loss, the effective margins of easy triplets do increase during training. We argue that this behavior arises because updates triggered by hard triplets reshape the embedding space, indirectly affecting the distances of easy triplets.

Moreover, most existing works adopt a constant value of μ throughout training [1], [3], [5]. We show that gradually increasing the margin over time is beneficial. Based on this observation, we propose two margin schedulers: a linear scheduler, which increases the margin μ linearly over training epochs, and an adaptive scheduler, called Difficulty Adaptive Margin Scheduler (DAMS), which increases μ when the proportion of easy triplets reaches a predefined threshold. The adaptive scheduler can be interpreted as an attempt to maintain the difficulty level of the training triplets over time.

Experiments conducted on four well-known datasets from different domains, namely LFW, CelebA, CUB-200-2011, and Stanford Cars, show that networks trained using the proposed margin schedulers achieve better performance compared to a constant margin μ. These results indicate that the proposed approaches are not domain-specific. In summary, the main contributions of this work are as follows:

- We analyze how the effective margins of easy triplets evolve during training, showing that they are implicitly updated together with hard triplets, leading to improved embeddings for both types of triplets.
- We propose margin scheduling strategies that progressively increase the desired margin during training, enabling a larger number of effective updates and improving the overall training process.

The remainder of this paper is organized as follows. Section II reviews related work. Section III analyzes how the effective margin achieved during training often exceeds the predefined value of μ and uses this observation to motivate the proposed adaptive method. The experimental protocol is described in Section IV, and the experimental results are presented in Section V. Finally, conclusions and directions for future work are discussed in Section VI.

II. Related Works

Training networks for comparison-related tasks using loss functions that enforce a minimum margin μ between negative pairs is a common approach, adopted in works such as [1], [5]–[8]. Some studies have explored alternatives to using a constant margin in the triplet loss or in related loss functions, such as the contrastive loss. These works generally follow one of two main directions: assigning a different margin to each triplet or batch [9], [10] or modifying the margin value as training progresses [10]–[12].

An example of the first direction is the work of [9], which proposes an adaptive margin in the context of rating datasets. In this approach, the similarity between two samples is combined with their relative ratings to define the margin. Samples with similar ratings are associated with smaller margins, while larger rating differences lead to larger margins. Similar ideas are explored in [13], [14]. Although promising, these methods rely on additional information beyond class labels, such as explicit ratings between training pairs.

The authors of [11] propose a method in which the margin is increased during training. Their approach, called Learning Incremental Triplet Margin (LITM), starts from a small base margin and divides training into multiple stages. At each stage, the margin is increased by a predefined amount, which can vary across stages. Each margin increment is treated as a hyperparameter. In their experiments, three stages are used with margins set to $(4, 7, 10)$, although the paper does not provide clear guidelines on how these values should be selected in practice.

In [10], the authors introduce AdaFace, an adaptive loss function that adjusts the margin according to image quality. Since image quality is difficult to estimate directly, the method uses the norm of the image embedding within the batch as a proxy, under the assumption that higher norms correspond to higher-quality images.

More recently, [12] proposed an adaptive triplet loss that employs different margins for positive and negative pairs. The authors also highlight the difficulty of choosing fixed margin values and propose a temporal strategy in which margins are updated using an exponential moving average. In this method, the updated margin is computed as a weighted combination of the previous moving average and the mean pairwise distances observed in the current batch.

While existing approaches often focus on specific scenarios, such as exploiting additional dataset information or adapting the margin based on batch-level statistics, our work aims to provide a more general strategy. We propose updating the margin only when the learning problem becomes sufficiently easy, based on an explicit analysis of training difficulty across epochs.

III. Proposed Method

As discussed in Section I, updates to the embedding space affect both easy and hard triplets. Consider Figure 2, which shows the median value of $\hat{\mu}$ (effective margin) computed over all training triplets across 100 training epochs. This experiment follows the protocol described in Section IV, using the LFW dataset and a fixed margin $\mu = 0.3$. If the embeddings of easy triplets remained unchanged during training, one would expect the median effective margin to saturate close to 0.3. Instead, as training progresses, the median effective margin continues to increase well beyond this value, reaching approximately 1.1 in the final epoch.

This provides evidence that the effective margins of triplets keep increasing during training, even after they become easy. We observed similar behavior across all other datasets evaluated in this work. The histograms shown in Figure 3 further support this observation, indicating that the distribution of effective margins progressively shifts to values larger than μ as training advances.

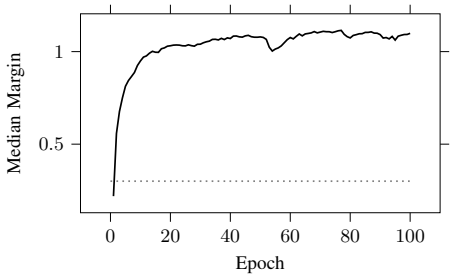
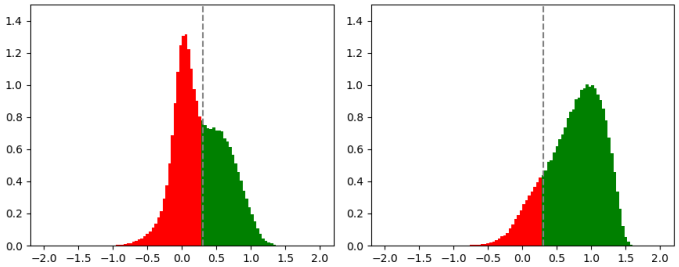
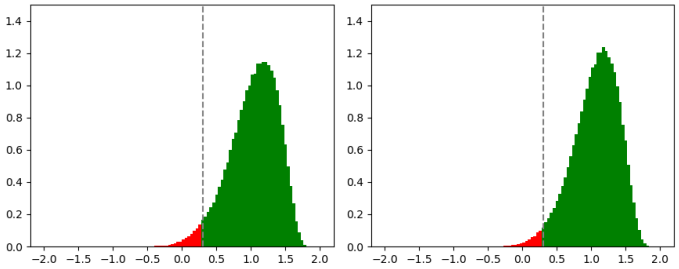


Fig. 2. Median effective margin of triplets at each training epoch on the LFW dataset.



(a) Epoch 1

(b) Epoch 5



(c) Epoch 50

(d) Epoch 100

Fig. 3. Histogram of effective margins across training epochs using the LFW dataset. Values below $\mu = 0.3$ correspond to hard triplets (red), while values above correspond to easy triplets (green).

Our hypothesis is that, while the model enforces the desired margin μ for hard triplets, it learns increasingly discriminative features that also improve the separation of easy triplets. For instance, consider a face recognition model. Early in training, the model may rely on simple attributes, such as hair color, which are sufficient to separate some triplets. As training progresses and the model is forced to handle harder triplets, such as faces with similar hair color, it must learn more detailed features, such as eye color or facial structure. As these features are learned, triplets that already exhibited large effective margins can become even better separated.

However, using a small margin μ may lead to a rapidly decreasing number of hard triplets, causing the model to focus on a limited subset of the data. This subset may include a disproportionate number of outliers or noisy samples, potentially hindering further improvements [12]. As an example, Figure 4 shows the proportion of easy triplets per epoch under the same experimental protocol. This proportion grows rapidly, causing the number of hard triplets to decrease quickly. When 80% of triplets are easy, which occurs at epoch 4, the training loss decreases by an average of 2.3% per epoch. In contrast, when 95% of triplets are easy, which occurs at epoch 22, the average decrease drops to 0.61% per epoch.

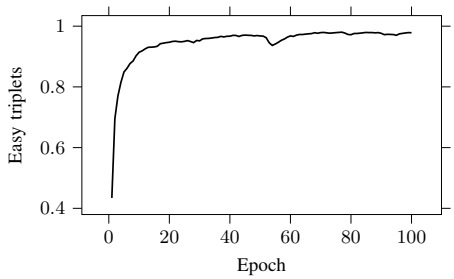


Fig. 4. Proportion of easy triplets per epoch on the LFW dataset.

These observations indicate that while a small margin is beneficial during the early stages of training, its effectiveness diminishes as the number of easy triplets becomes too large. To address this issue, we propose dynamically adjusting the margin μ in order to maintain a sufficient number of hard triplets throughout training. Specifically, we evaluate two margin scheduling strategies: a Linear scheduler and an Adaptive scheduler. The Linear scheduler starts from an initial margin μ_0 and increases it linearly at each epoch by a constant increment s_l.

The Adaptive scheduler also starts from an initial margin μ_0. Whenever the proportion of easy triplets in a given epoch exceeds a predefined threshold t, the margin is increased by a fixed increment s_a. This threshold t can be interpreted as a target level of task difficulty. We refer to this adaptive strategy as Difficulty Adaptive Margin Scheduler (DAMS).

IV. Experimental Protocol

In this section, we present the experimental protocol adopted in this work. Since our goal is to evaluate the behavior of the proposed margin schedulers in comparison to a constant margin strategy, we do not focus on large models or extensive data preprocessing to maximize absolute performance. Instead, we employ a relatively small model (Section IV-C) and use the datasets as provided.

A. Datasets

We evaluate our approach on the following datasets: LFW, CelebA, CUB-200-2011, and Stanford Cars. LFW consists of face images from multiple individuals and presents a high imbalance in the number of samples per class. Approximately 70% of identities have only one image, while some individuals

have more than 200 images. CelebFaces Attributes (CelebA) is a similar dataset composed exclusively of celebrity faces. In contrast to LFW, CelebA is more balanced, with more than 92% of identities having at least five images. CUB-200-2011 is a fine-grained dataset of bird images, where each class corresponds to a distinct bird species. Finally, Stanford Cars contains images of cars, where classes are typically defined by the tuple (manufacturer, model, year).

We split the datasets following protocols commonly adopted in the literature, ensuring that the classes in the training and test sets are disjoint. CUB-200-2011 and Stanford Cars are split evenly, with 50% of the classes used for training and 50% for testing [15]–[17]. LFW is split according to its recommended protocol, resulting in approximately 70% of the classes for training and 30% for testing [4]. For CelebA, we adopt the same splitting strategy used for LFW. Table I summarizes the datasets used in our experiments. Throughout this work, we refer to a class as a single identity, such as a person or a bird species, which may contain multiple images. Classes containing only one image are removed, as they cannot be used to form valid triplets or positive pairs.

Dataset	Train		Test	
	# Classes	# Images	# Classes	# Images
LFW	1,184	6,671	496	2,493
CelebA	7,094	141,859	3,039	60,696
CUB-200-2011	100	5,899	100	5,897
Stanford Cars	98	8,109	98	8,076

TABLE I
TRAIN AND TEST SPLITS OF THE DATASETS USED IN THE EXPERIMENTS.

B. Metrics

We evaluate performance using two metrics: the Area under the Receiver Operating Characteristic Curve (AUC-ROC) and Recall@k, with $k \in \{1, 2, 4, 8\}$. The AUC-ROC metric is computed using pairs of samples, with the goal of determining whether two samples belong to the same class [1]. Recall@k is a common metric in DML problems [16], which measures whether a sample can be correctly identified among its k nearest neighbors.

To compute the AUC-ROC, we generate pairs in the test set as follows. For each class c_k, with $k = 1, \ldots, K$, we randomly sample one positive pair (A_k, P_k), where both samples belong to class c_k and $A_k \neq P_k$, and one negative pair (A_k, N_k), where N_k does not belong to class c_k. This procedure ensures a balanced set of positive and negative pairs while covering all classes. The total number of pairs is therefore twice the number of classes, making it proportional to the dataset size.

To compute Recall@k, each sample a in the test set is temporarily removed from the dataset. We then retrieve the k nearest samples to a among the remaining test samples and verify whether at least one of them belongs to the same class as a.

C. Model and Training

We use a Convolutional Neural Network (CNN) pretrained on the ImageNet [18] dataset, which is a common practice in Siamese network training [3], [16]. Specifically, we adopt EfficientNetB0 [19] as the backbone network. Following [1], the backbone is followed by three fully connected layers: two hidden layers with 512 and 256 neurons, respectively, and an output layer with 128 neurons, followed by a L_2-normalization layer [1]. As a result, the generated embedding vectors are 128-dimensional and live in a hypersphere of radius one and centered at the origin.

All models are trained using the Adam optimizer with a learning rate of 0.001. Training is performed for 100 epochs with a batch size of 64 images. In-triplet hard negative mining, defined in [5], is employed in all experiments.

D. Tested Margin Schedulers

We evaluate the two proposed margin scheduling strategies. For the Linear scheduler, we initialize the margin with $\mu_0 = 0.0$ and increase it linearly by adding $s_l = 0.01$ at the end of each epoch. This results in a final margin of 1.0 at epoch 100. For the proposed DAMS scheduler, we also use $\mu_0 = 0.0$ and a step size of $s_a = 0.01$. If the margin were increased at every epoch, this would make the adaptive scheduler equivalent to the linear one, allowing for a fair comparison between the two strategies.

We set the threshold for the adaptive scheduler to $t = 0.95$, based on the observations in Section III, where the training loss was shown to decrease significantly less after this proportion of easy triplets was reached. Finally, we compare both schedulers against a constant margin baseline using a fixed margin value of $\mu = 0.3$.

V. EXPERIMENTS AND RESULTS

The experiments presented in this section are reported as the average of three executions. We first present the results considering all datasets in Section V-A. Then, in Section V-B, we perform a brief hyperparameter analysis to evaluate the influence of the proposed parameters, considering the LFW dataset.

A. Comparison of Margin Schedulers

Table II presents the results obtained on each dataset. We start by analyzing the linear scheduler, which, despite not achieving the best overall performance, outperformed the constant-margin approach in most scenarios. These results indicate that updating the margin over time is beneficial, which is consistent with previous works, such as [11], [12].

Our proposed DAMS adaptive method achieved the best results in most scenarios, with the linear scheduler outperforming it only in terms of AUC-ROC on the Stanford Cars dataset. This result suggests that increasing the desired margin μ only when the problem becomes sufficiently simple can be more effective than updating it at a constant rate.

In Figure 5, we show the margin evolution throughout training for all datasets. In the CelebA dataset, the DAMS scheduler exhibits a slow margin increase, with the curve becoming nearly horizontal around epoch 60 at approximately $\mu = 0.5$. This suggests that the linear scheduler likely increased the margin too aggressively, which may explain its

TABLE II
METRICS RESULTS WITH EACH METHOD IN THE DATASETS

Dataset	Margin	AUC	Recall@1	Recall@2	Recall@4	Recall@8
LFW	Constant	0.956 ±0.01	0.273 ±0.04	0.385 ±0.04	0.500 ±0.04	0.620 ±0.04
	DAMS	**0.966 ±0.00**	**0.395 ±0.02**	**0.500 ±0.02**	**0.605 ±0.01**	**0.703 ±0.01**
	Linear	0.963 ±0.00	0.364 ±0.02	0.476 ±0.02	0.578 ±0.02	0.679 ±0.02
CelebA	Constant	0.953 ± 0.00	0.137 ± 0.00	0.211 ± 0.01	0.303 ± 0.01	0.411 ± 0.01
	DAMS	**0.956 ± 0.00**	**0.217 ± 0.02**	**0.309 ± 0.02**	**0.411 ± 0.02**	**0.519 ± 0.02**
	Linear	0.940 ± 0.00	0.101 ± 0.00	0.161 ± 0.00	0.243 ± 0.00	0.347 ± 0.00
CUB-200-2011	Constant	0.897 ±0.02	0.270 ±0.03	0.406 ±0.02	0.550 ±0.02	0.683 ±0.03
	DAMS	**0.899 ±0.02**	**0.379 ±0.01**	**0.515 ±0.00**	**0.642 ±0.00**	**0.755 ±0.00**
	Linear	0.894 ±0.00	0.334 ±0.01	0.475 ±0.02	0.613 ±0.01	0.734 ±0.01
Stanford Cars	Constant	0.933 ±0.00	0.329 ±0.04	0.474 ±0.05	0.621 ±0.05	0.742 ±0.04
	DAMS	0.941 ±0.01	**0.439 ±0.02**	**0.576 ±0.02**	**0.699 ±0.01**	**0.796 ±0.01**
	Linear	**0.953 ±0.01**	0.432 ±0.01	0.572 ±0.01	0.698 ±0.01	0.800 ±0.00
Average	Constant	0.928	0.856	0.242	0.355	0.476
	DAMS	**0.937**	**0.865**	**0.346**	**0.464**	**0.577**
	Linear	0.934	0.861	0.292	0.402	0.516

poorer performance on this dataset, being the only case where the constant-margin method outperformed it.

We hypothesize that the linear method probably increased the margin too aggressively. This is corroborated by the large number of classes in the CelebA dataset, with approximately 3,000 identities in the test set, which may require a lower margin to fit all embeddings in the 128-dimensional space. In contrast, the Stanford Cars and CUB-200-2011 datasets contain fewer classes, around 100 in the test set, and can therefore accommodate higher margins, which may explain the higher final margins achieved by the DAMS scheduler.

The results on LFW align with this analysis, as its number of classes is closer to that of CelebA, resulting in a similar final margin. Importantly, the proposed DAMS scheduler is able to adapt to these differences automatically, achieving appropriate margins across datasets while using the same hyperparameter configuration.

B. Hyperparameter Analysis

As discussed in Section III, the DAMS scheduler has three hyperparameters: the initial margin μ_0, the threshold of easy triplets t, and the step size s_a. The optimal values for these parameters are not obvious. To analyze their influence, we evaluate different combinations using the LFW dataset, following the experimental protocol described in Section IV. To do this, we test the following values:

- $\mu_0 \in \{0.0, 0.3\}$.
- $t \in \{50\%, 80\%, 85\%, 90\%, 95\%, 99\%\}$.
- $s_a \in \{0.01, 0.02, 0.05, 0.10\}$.

This results in $2 \cdot 6 \cdot 4 = 48$ configurations. We focus on the Recall@1 and AUC-ROC metrics. Table III reports the ten best and ten worst configurations, sorted in decreasing order of Recall@1.

As can be observed in Table III, the best configurations are predominantly associated with high values of t, while the worst results are obtained with lower values. Only three of the top-performing configurations use $t < 0.95$, and none use $t < 0.8$. This suggests that training benefits from maintaining a small proportion of hard triplets, indicating that the dataset contains limited noise or that noisy samples do not significantly affect the loss. This behavior is consistent with the analysis presented in Section III.

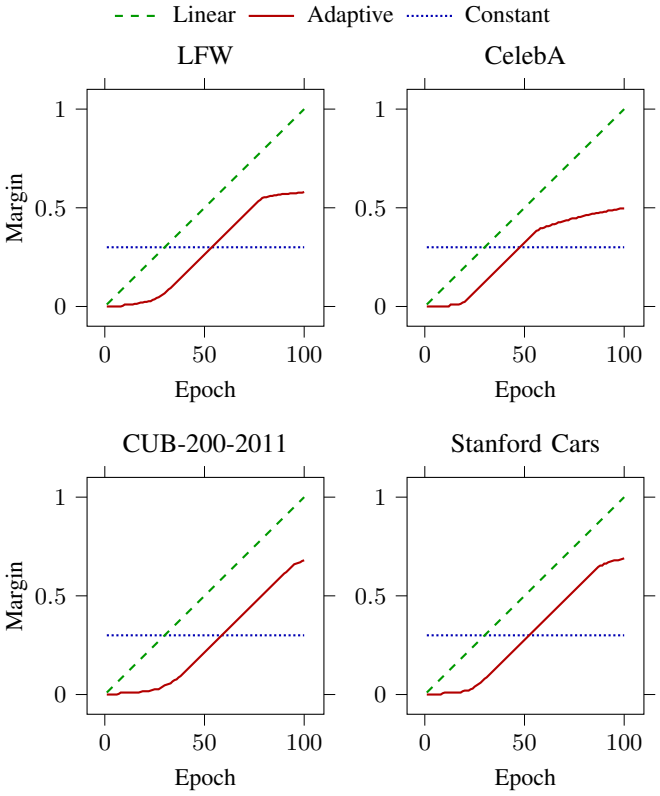


Fig. 5. Margin μ evolution for each method across datasets.

No clear trend is observed regarding the step size s_a, indicating that the proposed method is relatively robust to this parameter within the tested range. Finally, these results should be interpreted with caution, as the analysis is limited to the LFW dataset. A more comprehensive investigation across additional datasets is left for future work.

TABLE III
RESULTS FOR DIFFERENT PARAMETERS IN THE ADAPTIVE METHOD

μ_0	t	s_a	Recall@1	AUC
Best Configurations				
0.0	99%	0.02	0.411 ± 0.02	0.967 ± 0.00
0.0	95%	0.01	0.403 ± 0.01	0.963 ± 0.01
0.0	99%	0.05	0.387 ± 0.03	0.965 ± 0.01
0.0	95%	0.02	0.383 ± 0.00	0.964 ± 0.00
0.0	99%	0.01	0.380 ± 0.02	0.966 ± 0.00
0.0	90%	0.01	0.373 ± 0.04	0.966 ± 0.00
0.0	95%	0.05	0.366 ± 0.02	0.964 ± 0.00
0.0	99%	0.1	0.359 ± 0.03	0.968 ± 0.01
0.0	80%	0.01	0.351 ± 0.01	0.963 ± 0.00
0.0	85%	0.01	0.345 ± 0.01	0.958 ± 0.00
Worst Configurations				
0.3	50%	0.01	0.180 ± 0.01	0.914 ± 0.01
0.3	80%	0.05	0.176 ± 0.01	0.919 ± 0.00
0.0	80%	0.1	0.173 ± 0.00	0.916 ± 0.01
0.3	80%	0.1	0.159 ± 0.02	0.914 ± 0.01
0.0	50%	0.02	0.136 ± 0.01	0.841 ± 0.02
0.3	50%	0.02	0.108 ± 0.00	0.846 ± 0.02
0.0	50%	0.05	0.106 ± 0.01	0.817 ± 0.01
0.3	50%	0.05	0.099 ± 0.03	0.807 ± 0.01
0.0	50%	0.1	0.085 ± 0.02	0.801 ± 0.02
0.3	50%	0.1	0.072 ± 0.02	0.801 ± 0.01

VI. CONCLUSIONS

In this work, we show that even when easy triplets satisfy the desired margin μ in the Triplet Margin Ranking Loss and therefore produce zero loss, the embedding updates driven by hard triplets substantially influence these easy triplets. As a result, their effective margins increase over the course of training. This observation highlights a discrepancy between the fixed target margin enforced by the loss function and the evolving geometry of the embedding space.

Motivated by this behavior, we propose the Difficulty Adaptive Margin Scheduler (DAMS), which adapts the desired margin μ during training. Instead of increasing the margin according to a predefined schedule, DAMS relies on the proportion of easy triplets observed at each epoch, increasing μ only when the learning problem becomes sufficiently easy.

Results on four datasets covering distinct fine-grained and face recognition tasks show that DAMS consistently outperforms both a constant-margin strategy and a monotonically increasing margin scheduler. The same set of hyperparameters was used across all datasets, suggesting that the proposed method generalizes well without dataset-specific tuning. While we do not claim that DAMS is universally optimal for all metric learning scenarios, these results indicate that adapting the margin based on training difficulty is an effective strategy.

We hypothesize that the improved performance of DAMS over linear scheduling arises from its ability to prevent over-constraining the embedding space. Differently from methods that increase the margin independently of the data distribution, DAMS implicitly accounts for factors such as the number of classes and embedding dimensionality, reducing the risk of enforcing margins incompatible with the embedding-space.

Nevertheless, DAMS has some limitations. First, because margin updates depend on reaching a target proportion of easy triplets, the maximum achievable margin growth is inherently bounded for a given step size. In scenarios where faster margin expansion is beneficial, such as the Stanford Cars dataset, this behavior may slow convergence. Second, the method introduces three hyperparameters that require selection. However, we showed a group of hyperparameters that seem to work well in a range of scenarios ($t = 0.95, \mu_0 = 0$ and $s_a = 0.01$).

In future works, we will better investigate these hyperparameters. Also, the graphs in Figure 5 show that the margin tends to saturate in the DAMS method at some point, which indicates we might be able to use it as a stopping criterion.

REFERENCES

[1] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[2] M. M. Ribas, H. B. Mendes, L. E. de Oliveira, L. A. Zanlorensi, and P. L. de Almeida, "Using deep neural networks to quantify parking dwell time," in *ICMLA*. IEEE, 2024, pp. 1504–1509.

[3] Y. Li, C. P. Chen, and T. Zhang, "A survey on siamese network: Methodologies, applications, and opportunities," *IEEE Transactions on artificial intelligence*, vol. 3, no. 6, pp. 994–1014, 2022.

[4] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[5] D. P. Vassileios Balntas, Edgar Riba and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, September 2016, pp. 119.1–119.11.

[6] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *CVPR*, 2014, pp. 1386–1393.

[7] F. Boutros, N. Damer, F. Kirchbuchner, and A. Kuijper, "Self-restrained triplet loss for accurate masked face recognition," *Pattern Recognition*, vol. 124, p. 108473, 2022.

[8] Z. Zhang, B. Sun, J. Liang, H. He, and D. Qiao, "Low-resolution driver face recognition based on super-resolution and triplet loss," *Scientific Reports*, vol. 15, no. 1, p. 42299, 2025.

[9] M. L. Ha and V. Blanz, "Deep ranking with adaptive margin triplet loss.(2021)," *arXiv preprint arXiv:2107.06187*, 2021.

[10] M. Kim, A. K. Jain, and X. Liu, "Adaface: Quality adaptive margin for face recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 18 750–18 759.

[11] Y. Zhang, Q. Zhong, L. Ma, D. Xie, and S. Pu, "Learning incremental triplet margin for person re-identification," in *AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 9243–9250.

[12] J. Moonrinta, M. Dailey, and M. Ekpanyapong, "Improving noisy face embeddings using time series transformers and adaptive triplet loss," *IEEE Access*, vol. 13, pp. 174 254–174 269, 2025.

[13] D. Semedo and J. Magalhães, "Cross-modal subspace learning with scheduled adaptive margin constraints," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 75–83.

[14] J. Moonrinta, M. N. Dailey, and M. Ekpanyapong, "Improving noisy face embeddings using time series transformers and adaptive triplet loss," *IEEE Access*, 2025.

[15] R. Ji, J. Li, and L. Zhang, "Siamese self-supervised learning for fine-grained visual classification," *Computer Vision and Image Understanding*, vol. 229, p. 103658, 2023.

[16] T.-T. Do, T. Tran, I. Reid, V. Kumar, T. Hoang, and G. Carneiro, "A theoretically sound upper bound on the triplet loss for improving the efficiency of metric learning," in *CVPR*, 2019, pp. 10 404–10 413.

[17] B. Harwood, V. Kumar BG, G. Carneiro, I. Reid, and T. Drummond, "Smart mining for deep metric learning," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2821–2829.

[18] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[19] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.