Revisiting Loss Functions for Person Re-identification

Dustin Aganian(✉), Markus Eisenbach, Joachim Wagner, Daniel Seichter, and Horst-Michael Gross

Ilmenau University of Technology, 98693 Ilmenau, Germany
dustin.aganian@tu-ilmenau.de

Abstract. Appearance-based person re-identification is very challenging, i.a. due to changing illumination, image distortion, and differences in viewpoint. Therefore, it is crucial to learn an expressive feature embedding that compensates for changing environmental conditions. There are many loss functions available to achieve this goal. However, it is hard to judge which one is the best. In related work, the experiments are only performed on the same datasets, but the use of different setups and different training techniques compromises the comparability. Therefore, we compare the most widely used and most promising loss functions under identical conditions on three different setups. We provide insights into why some of the loss functions work better than others and what additional benefits they provide. We further propose sequential training as an additional training trick that improves the performance of most loss functions. In our conclusion, we provide guidance for future usage an d research regarding loss functions for appearance-based person re-identification. Source code is available (Source code: https://www.tu-ilmenau.de/neurob/data-sets-code/re-id-loss/).

Keywords: Person re-identification · Deep learning · Representation learning · Loss functions · Softmax loss · Triplet hard loss · Ring loss · Center loss · Additive angular margin loss · Circle loss

1 Introduction and Related Work

For appearance-based person re-identification (ReID), there are two strategies for learning a feature embedding using deep convolutional networks (see Fig. 1): First, the training can be formulated as a classification problem, where each person in the training set represents a separate class. By fitting the model this way, the penultimate layer forms a meaningful feature embedding that can subsequently be used for ReID. Second, the feature embedding can be learned directly using tuples of person images including a match and a mismatch. The objective is to create feature vectors being more similar to each other for matching pairs than for mismatches. In both strategies, the choice of a suitable loss function is crucial.

This work has received funding from the Carl Zeiss Foundation as part of the project E4SM under grant agreement no. P2017-01-005.

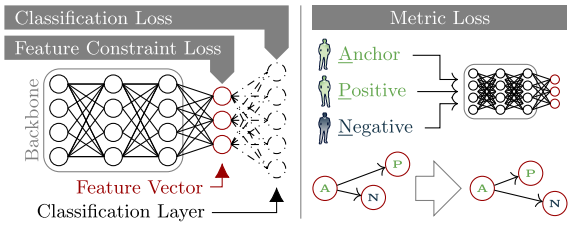

Fig. 1. Basic concepts for learning a feature embedding by applying different loss functions. Left: Formulation as classification problem. Right: Triplet-based training.

Figure 2 shows applicable loss functions, categorized into three types. The state of the art for ReID most often uses a combination of up to three loss functions simultaneously for training. Usually, one loss function from each category shown in Fig. 2 is selected. A popular baseline [10] composes softmax loss [1], triplet loss [11], and center loss [16]. Therefore, recent work mainly builds on these loss functions, despite there being more advanced loss functions available in each of the categories.

The single influence of these loss functions is rarely evaluated. In [6], the performance of softmax loss [1], a.k.a. ID loss, and its extensions multiplicative angular margin loss (MAML, A-Softmax, SphereFace) [8,9], additive cosine margin loss (ACML, CosFace) [13,15], and additive angular margin loss (AAML, ArcFace) [2] is compared. However, this comparison is done using a very weak baseline (see Table 1). There is also related work that evaluates the individual influence of loss functions for ReID, e.g., center loss [16] was used in [7] and ring

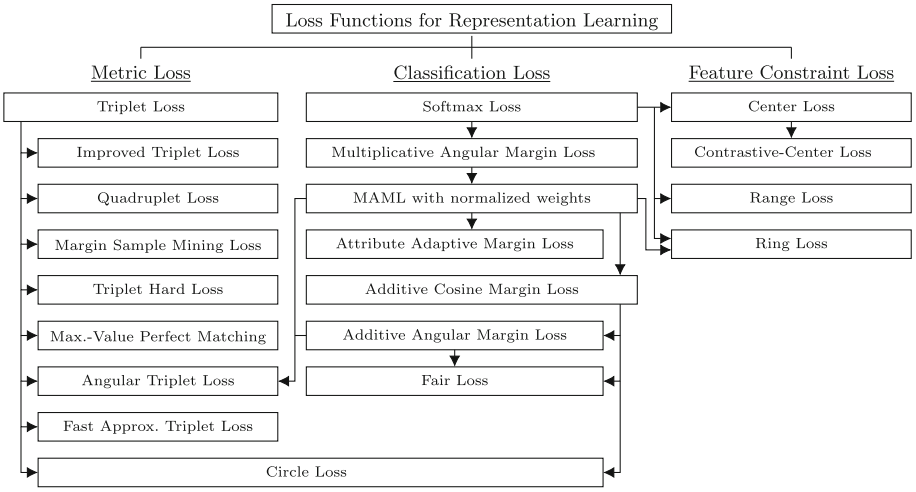

Fig. 2. Systematization of loss functions. Arrows show which loss functions improve previous ones. In our analysis, we incorporate loss functions of all three categories.

Table 1. Results of related work using a ResNet50 backbone on the Market-1501 dataset compared to our results using the same backbone with modern training techniques [10] and an identical setup for each loss function. A "—" signifies that the loss function was not evaluated on Market-1501 in the paper.

		Paper		Ours	
Loss	ReID	Rank 1	mAP	Rank 1	mAP
Center Loss	[7]	—	—	94.4	85.3
Ring Loss	[5]	—	—	94.7	86.1
Triplet Hard Loss	[4]	84.9	69.1	92.4	82.7
Softmax Loss, a.k.a. ID loss	[6]	71.0	46.3	93.1	82.0
MAML (with $\|\mathbf{w}_j\|_2 = 1$),	[6]	78.4	56.0		
a.k.a. A-Softmax (SphereFace)	[3]	94.4	83.6	95.2	87.9
	[12]	92.4	83.8		
ACML (CosFace)	[6]	77.8	56.2	95.4	87.7
AAML (ArcFace)	[6]	79.2	57.3	95.5	88.1
Circle Loss	[12]	94.2	84.9	95.5	88.4

loss [19] was used in [5], or introduced new loss functions, like circle loss [12], triplet hard loss [4], or other triplet loss extensions as in Fig. 2. All these papers share the same problem: They either compare with a weak baseline (mainly softmax loss, triplet loss, or MAML) or do not compare with other loss functions at all but use the loss functions in another context. Other loss functions (Attribute Adaptive Margin Loss, Viewpoint-aware Loss) need additional label information (attributes, camera IDs, etc.) and, thus, are not comparable. A second problem can be seen in Table 1: Even if they compare with the same baseline (MAML with $\|\mathbf{w}_j\|_2 = 1$), the results are not comparable at all. This is mainly due to different setups that include only subsets of new training techniques introduced in recent years that significantly improve results [10]. Table 1 also shows the performance of the single loss functions in our experiments on Market-1501 [18] using the same backbone as the papers above – a ResNet50 – and the training techniques of [10]. Our results are consistently better than the results reported in the respective papers. This confirms that likely some of the training techniques are missing in these papers or hyperparameters are not carefully tuned.

In order to find out, which loss functions of each category are most useful for ReID, we evaluate the effectiveness of solely applied loss functions on three different setups. Therefore, our contributions are three-fold:

1. Our evaluation is the first attempt towards a more fair comparison of the most promising and most often used loss functions for ReID under identical conditions on strong setups.
2. Due to different setups, we gain insights into why some of the loss functions work better than others and what additional benefits they provide.
3. We propose sequential training as an additional training trick that improves the performance of most loss functions.

2 Loss Functions

In our comparison, we include the typical baselines for the three categories of loss functions – triplet (hard) loss as metric loss, softmax loss as classification loss, and center loss as feature constraint loss – as well as additive angular margin loss as the most promising classification loss without adaptive margin and without side information based on benchmark results in ReID and face identification, ring loss as the most promising feature constraint loss based on benchmark results in face identification, and circle loss as the most promising loss with weighted similarities based on how far they are from the optimization goal. Circle loss can be formulated as metric loss or classification loss. We report the classification loss results since, in our experiments, this version clearly outperforms the metric loss version.

2.1 Metric Loss

Most metric loss functions train a feature vector $\mathbf{x}$ directly with triplets of an anchor $\mathbf{x}^a$, a positive $\mathbf{x}^p$, and a negative $\mathbf{x}^n$ [4,11]. As shown in Fig. 1, the objective is to make the distances of matches $d(\mathbf{x}^a, \mathbf{x}^p)$ much smaller than the distances of mismatches $d(\mathbf{x}^a, \mathbf{x}^n)$. The metric loss functions differ in how the triplets are compiled and how deviations from the objective are penalized.

Triplet Hard Loss (THL) [4] builds triplets of P random classes and K images per class for each mini-batch. Then, for each person i and anchor a, the hardest positive $\hat{p}_i^a$ and hardest negative $\hat{n}_i^a$ triplets of this batch that violate the objective the most are selected. Thus, the loss L_{THL} is calculated as in Eq. 1 with either a hard [11] or a soft margin [4].

$$L_{\text{THL}} = \frac{1}{PK} \sum_{i=1}^{P} \sum_{a=1}^{K} f(\hat{p}_i^a, \hat{n}_i^a) \tag{1}$$

$$f_{Hard}(\hat{p}_i^a, \hat{n}_i^a) = \max\left(\hat{p}_i^a + m - \hat{n}_i^a, 0\right) \qquad \hat{p}_i^a = \max_{p=1\ldots K} \|\mathbf{x}_i^a - \mathbf{x}_i^p\|_2$$

$$f_{Soft}(\hat{p}_i^a, \hat{n}_i^a) = \ln\left(1 + e^{\hat{p}_i^a - \hat{n}_i^a}\right) \qquad \hat{n}_i^a = \min_{\substack{j=1\ldots P \\ n=1\ldots K \\ j \neq i}} \|\mathbf{x}_i^a - \mathbf{x}_j^n\|_2$$

2.2 Classification Loss

When using a classification loss, a classification layer with each person in the training set as a separate class is added after the feature vector $\mathbf{x}$ during training (see Fig. 1). The general equation for a mini-batch of size N and K classes, with y_i being the class label of the i-th mini-batch sample, is as follows:

$$L_{\text{Class}} = \frac{1}{N} \sum_{i=1}^{N} -\log\left(\frac{e^{f_p(\mathbf{x}_i, \mathbf{w}_{y_i})}}{e^{f_p(\mathbf{x}_i, \mathbf{w}_{y_i})} + \sum\limits_{\substack{k=1\ldots K \\ k \neq y_i}} e^{f_n(\mathbf{x}_i, \mathbf{w}_k)}}\right) \tag{2}$$

Softmax Loss (SL) [1], a.k.a. ID loss, applies a cross entropy loss on a softmax output layer. Therefore, f_p and f_n in Eq. 2 are defined as:

$$f_p^{\mathrm{SL}}(\mathbf{x}_i, \mathbf{w}_{y_i}) = \mathbf{x}_i \cdot \mathbf{w}_{y_i} + b_{y_i} \qquad f_n^{\mathrm{SL}}(\mathbf{x}_i, \mathbf{w}_k) = \mathbf{x}_i \cdot \mathbf{w}_k + b_k \tag{3}$$

Additive Angular Margin Loss (AAML) [2] is an extension of SL. It normalizes the feature ($\|\mathbf{x}_i\|_2 = \gamma$) and weight vectors ($\|\mathbf{w}_j\|_2 = 1$), removes the bias ($b_j = 0$), and adds a decision margin m regarding the angle θ_{x_i,w_j} in order to increase the inter-class distance and decrease the inner-class variance. Therefore, f_p and f_n in Eq. 2 are defined as:

$$\begin{aligned} f_p^{\mathrm{AAML}}(\mathbf{x}_i, \mathbf{w}_{y_i}) &= \gamma \cos(\theta_{\mathbf{x}_i,\mathbf{w}_{y_i}} + m) \\ f_n^{\mathrm{AAML}}(\mathbf{x}_i, \mathbf{w}_k) &= \gamma \cos(\theta_{\mathbf{x}_i,\mathbf{w}_k}) \\ \text{with } \cos(\theta_{\mathbf{x}_i,\mathbf{w}_j}) &= s_{\mathbf{x}_i,\mathbf{w}_j} = \frac{\mathbf{x}_i \cdot \mathbf{w}_j}{\|\mathbf{x}_i\|_2 \cdot \|\mathbf{w}_j\|_2} \end{aligned} \tag{4}$$

Circle Loss (CirL) [12] implements a circular optimization goal. Therefore, f_p and f_n in Eq. 2 are defined as:

$$\begin{aligned} f_p^{\mathrm{CirL}}(\mathbf{x}_i, \mathbf{w}_{y_i}) &= \gamma \cdot \max(-(s_{\mathbf{x}_i,\mathbf{w}_{y_i}} - 1) + m, 0) \cdot (s_{\mathbf{x}_i,\mathbf{w}_{y_i}} - 1 + m) \\ f_n^{\mathrm{CirL}}(\mathbf{x}_i, \mathbf{w}_k) &= \gamma \cdot \max(s_{\mathbf{x}_i,\mathbf{w}_k} + m, 0) \cdot (s_{\mathbf{x}_i,\mathbf{w}_k} - m) \end{aligned} \tag{5}$$

2.3 Feature Constraint Loss

Loss functions of this type restrict the feature vector in a classification setting. Usually, it is scaled by a weighting factor λ and added to the classification loss.

Center Loss (CenL) [16] forces the model to learn feature vectors with low distances to their respective class centers $\mathbf{c}_{y_i}$. Therefore, the inner-class distance is reduced. For a mini-batch of size N the loss L_{CenL} is defined as:

$$L_{\mathrm{CenL}} = \frac{\lambda}{2} \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{c}_{y_i}||_2^2 \tag{6}$$

Ring Loss (RL) [19] forces the model to keep the feature vector on a hypersphere with a radius R, which leads to more robust feature vectors. The loss L_{RL} with the trainable parameter R is defined as:

$$L_{\mathrm{RL}} = \frac{\lambda}{2N} \sum_{i=1}^{N} (||\mathbf{x}_i||_2 - R)^2 \tag{7}$$

3 Experiments

In our experiments, we compare the performance of the six loss functions described above under identical conditions on three strong setups. In addition, we analyze the costs of a setup that follows best practices in real-world applications, the complexity of training, the influence of different initializations in a sequential training setting, and the generalization abilities.

3.1 Setup

In order to achieve comparability with most state-of-the-art ReID papers, we use a ResNet50 as backbone, pre-trained on ImageNet unless otherwise stated (PyTorch weights as in state of the art). For optimization, we either used SGD with momentum of 0.9 or Adam with standard parameters. The extensive hyperparameter search for each loss function included the learning rate, batch size, weight decay, batch composition, as well as scaling/weighting factors and the margin if applicable. All experiments were conducted using TensorFlow 2.

For assessing the ReID performance, we report the mean Average Precision (mAP) and the rank-1 accuracy (rank 1) in percentage values. We applied the single query protocol of [18] by running the evaluation code of [4]. The similarities of the feature vectors were calculated with cosine similarity in favor of the Euclidean distance since results are better regardless of the loss function used. For metric losses, it does not make a significant difference to the results, but for all other loss functions, the results always improve by a few percentage points.

Baseline Setups

In our experiments, we compare the models trained with the different loss functions on three baseline setups: The strong baseline as described in [10], the Multi Granular Network (MGN) architecture as described in [14], and a simple baseline, that we derived from [10]. In all setups, we use ResNet50 as architectural backbone. We trained and tested on the Market-1501 [18] dataset.[1]

Modifications for Simple Baseline: With the simple baseline, we follow two strategies: First, we intend to leave some room for improvements by the loss functions and, therefore, deliberately abandon some of the training techniques described in [10] that are not essential. Namely, these are label smoothing, a change of the last stride in the backbone, random erasing augmentation, and a batch normalization neck.

Second, we want this training setup to follow best practices in real-world applications. These include the use of validation data to avoid optimizing on the test data, and the use of side data to learn a better generalization. We used DukeMTMC-reID and CUHK03-NP as side data. The validation set is obtained by splitting off 10% of the training data. For this, we randomly drew from the 904 persons with the most samples. This resulted in 36,823 training samples incorporating 2,220 different persons for our simple baseline setup. We further employ an additional fully connected layer after the last ResNet block. This can reduce the size of the feature vector, which is particularly relevant for some applications with real-time and data-storage constraints. In the following, we simply refer to this setup as baseline.

[1] We also evaluated the performance on DukeMTMC-reID, and CUHK03-NP, but since the results are very similar and due to space restrictions, we decided in favor of reporting only the Market-1501 results in this paper.

3.2 Costs of a Setup for Real-World Applications

While using these best practices in our baseline setup brings us closer to a real-world application, this choice comes with a cost that diminishes the test results on the typical publicly available datasets in ReID. Therefore, we only use these techniques for comparison on our first baseline. For the strong baseline and MGN architecture, we follow the typical protocol from the state of the art in order to ensure our results are comparable.

In the following, these costs are described numerically using an example. In this example, we have trained the baseline with THL and SL. Our experiment reveals that if we add the training data of DukeMTMC-reID and CUHK03-NP as side data to the training and not just use the Market-1501 training data, the test result on the Market-1501 test data worsens by about 2 percentage points (p.p.) for the mAP and by about 1 p.p. for rank 1. If we split off validation data from the training data, then the mAP worsens by another 3 p.p. and rank 1 by about 2 p.p. These declines were expected since when using side data, the network no longer overspecializes for the test data on the dataset and when using validation, the training data is somewhat reduced and the hyperparameters are no longer optimized on the test data. Furthermore, the use of the additional fully connected layer also results in a decrease in performance by slightly under 2 p.p. for mAP and 1 p.p. for rank 1. However, it is not possible to avoid this if smaller feature vectors are needed on the target system. Likewise, as also stated in [17], this shows that an additional fully connected layer, which is still often used in current works (e.g. in MGN [14]), does not necessarily have to be advantageous.

3.3 Complexity of Training

All loss functions achieve bad results with randomly initialized weights. Therefore, transferring ImageNet weights is crucial for representation learning in ReID. In the following, we report challenges we have faced during training:

Easy to Train: We did not face any difficulties in learning a feature embedding for ReID using *SL*. Similarly, it is easy to train with a constraint loss (*RL, CenL*). We found that a good choice of the loss weighting factor λ is important. The larger the feature vector is, the smaller λ has to be chosen since, at the beginning of the training, the ℓ_2 norm of a large feature vector is huge, resulting in a huge constraint loss. Otherwise, the constraint loss would dominate the overall loss, which is composed of SL and the constraint loss. For *THL*, in our experiments, it was crucial to apply some kind of learning rate warm-up. Except for that, there is no difficulty.

Hard to Train: Contrarily to SL, while it is easy and straightforward to implement the SL extensions *AAML* and *CirL*, these advanced loss functions are quite difficult to handle. When initializing with ImageNet weights, the trained models performed poorly if training converged at all. In order to overcome the problems of having a margin right from the beginning of the training, we first pre-trained with SL on ReID data, transferred the weights, and then started training with

the SL extension. We refer to this approach as sequential training. Figure 5 gives evidence that pre-training on ReID data simplifies the problem to be solved (see Sect. 3.5).

3.4 Experimental Comparison

For each loss function included in our analysis, we performed an extensive hyperparameter search to ensure a fair comparison. Figure 3 shows the individual results[2] using the best hyperparameter combinations for all tested network architectures. The best models were determined based on the largest mAP, as the mAP is closer to a realistic use case, as not only the simplest example from many gallery matches is assessed in the evaluation, as it is the case for rank 1.

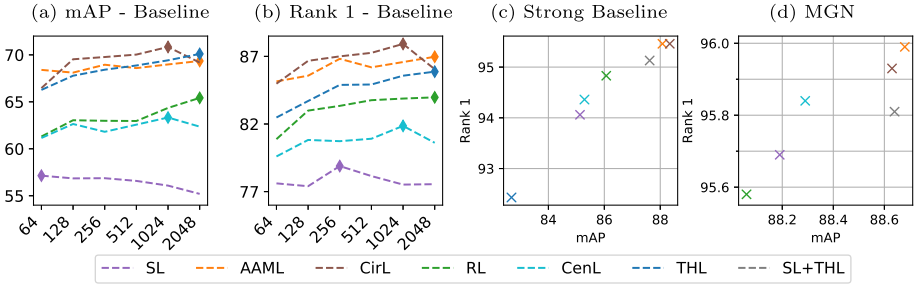

Fig. 3. (a, b) mAP and rank 1 baseline results for different feature vector sizes trained with the best hyperparameters for each of the loss functions. (c) mAP and rank 1 results for our strong baseline and (d) the MGN architecture for each of the loss functions. All results are reported on the Market-1501 [18] test set.

Baseline: We evaluated six different feature vector sizes for each of the loss functions. Figure 3 (a) and (b) show the results for each loss function. The best result is highlighted by a diamond. Based on mAP, CirL performs best, followed by THL and AAML. Next up are the feature constraint losses RL and CenL, which were both used together with SL, whereas SL alone scores significantly worse. When comparing the results over different feature vector sizes, it becomes apparent that the largest feature vector (2048) does not always provide the best result, e.g., for CirL, the best result is achieved with 1024, SL works best with 64. Often, in more complex architectures, such as the local-feature-focused MGN, additional fully connected layers are added in different heads after the last ResNet block. Our results show that depending on the loss function that is applied on such a head with an additional fully connected layer, the feature vector size might be adjusted to achieve better results.

[2] The standard deviation for 8 training runs of the best loss function in each case is $\sigma_{mAP} = 0.226\%$, $\sigma_{r1} = 0.418\%$ for the baseline setup, $\sigma_{mAP} = 0.079\%$, $\sigma_{r1} = 0.114\%$ for the strong baseline, and $\sigma_{mAP} = 0.046\%$, $\sigma_{r1} = 0.098\%$ for MGN.

Strong Baseline, MGN: To get a better comparison of the loss functions, we also tested them on two appropriate state-of-the-art architectures, the strong baseline of [10] and the MGN architecture [14]. The results of these experiments can be found on Fig. 3 (c) and (d). Here, we also trained SL together with THL (SL + THL), as this is a common approach. The best loss functions on these two architectures are AAML and CirL, followed by SL+THL. For the strong baseline, RL, CenL, and SL follow next, and THL scored the worst. On the MGN architecture, all loss functions, with the exception of THL,[3] produce results in a very close range of values. In this case, SL + THL corresponds to the original MGN setup proposed in [14]. For THL, it was applied on all heads, and for all other loss functions, the THL heads were removed and SL was extended with a constraint loss or replaced with a classification loss.

Findings: In general, all experiments have shown that AAML and CirL achieve the best results and far surpass the still often used SL. The comparable performance of AAML and CirL also shows the necessity to always compare with strong reference approaches on a strong setup when introducing a new loss function, since the postulated superiority of CirL for ReID in [12] is only due to a weaker reference approach.

It can be seen that all the loss functions perform differently depending on the architecture. For the baseline, which corresponds to a typical architecture a metric loss is applied to, THL is on a par with AAML and CirL. On the strong baseline, however, THL is among the weakest loss functions. Mainly this is due to training techniques in the strong baseline and MGN that help the classification loss functions to learn features better suited for cosine similarity, but do not benefit (Euclidean-distance-based) metric loss functions equally strong.

Furthermore, RL beats CenL on the strong baseline. However for MGN it is the other way around. This shows that the architecture plays a decisive role when choosing a loss function. It also shows that when developing new loss functions, a comparison with existing ones should always be done on various setups.

3.5 Generalization Ability and Sequential Training

Two aspects will be examined in this section. First, the generalization abilities of the loss functions are investigated by using validation data. Second, the sequential training of loss functions is examined in more detail. These investigations are performed on our baseline setup, since it follows best practices as described in Sect. 3.1 and Sect. 3.2. Especially generalization abilities are a quality that is in great demand in real-world applications and is often disregarded in the state of the art with the typical approach of hyperspecialization of training methods on public datasets. Fig. 4 shows the test and validation results of the best models from Fig. 3 (a) and (b). Furthermore, the results from our sequential training experiments are shown. The models with ImageNet in the index were initialized with weights from an ImageNet training and the models with SL in the index

[3] Results for THL (mAP: 79.38% and rank 1: 90.80%) are omitted in Fig. 3(d) for visualization purposes.

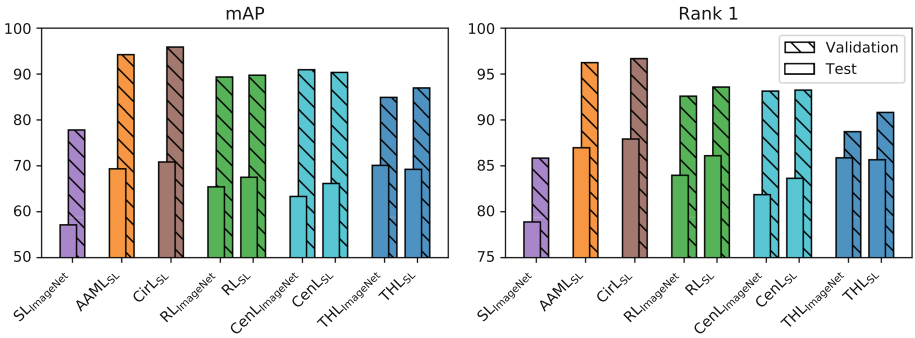

Fig. 4. Comparison of validation and test results. The validation split is taken from the training set of Market-1501, DukeMTMC-reID, and CUHK03-NP. The test data are from Market-1501.

were initialized with weights from a previous ImageNet-initialized training with SL on ReID data.

Generalization: First, we examine the difference between validation and test results, and thus the generalization abilities of the loss functions. As mentioned in Sect. 3.1, we split the training data into a training and a validation set. Images in the training and validation set are disjoint but comprise the same persons. In contrast, the test set contains different persons. Therefore, on the validation set, we measure the model's ability to distinguish known persons, while on the test set, we measure the generalization ability to unknown data. CirL and AAML achieve the best validation results by a wide margin. The constraint losses, THL, and lastly SL follow in descending order. When comparing the mAP test results, THL is about as good as AAML and CirL. This shows that THL is less likely to memorize the classes in the training data and more likely to achieve good generalization ability. This is a reasonable explanation why THL is used in addition to SL (SL + THL) in current approaches, since according to these results, the additional use of a metric loss in addition to a classification loss induces the network to develop better generalization capabilities. This is also an important conclusion for practical applications where the generalization abilities of a network should be maximized.

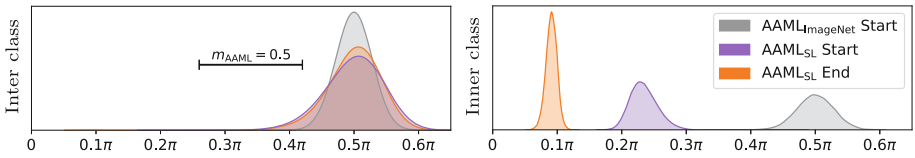

Fig. 5. Different initializations influence the distributions of inner-class angles $\theta_{x_i,w_{y_i}}$ and inter-class angles θ_{x_i,w_j} (training data).

Sequential Training: In order to train with AAML and CirL successfully, we had to initialize the models with weights from a previous training with SL on the ReID data. This approach is not described in the state of the art, but this is the easiest way to achieve similar results to the state of the art when training with these classification loss functions that introduce a margin. For a closer look at why this sequential training works so well, Fig. 5 shows the inner-class and inter-class angles when training with AAML[4]. As can be seen, in a typical initialization with ImageNet weights (gray), the inner- and inter-class angles are distributed in the same range of values and, thus, very large errors occur in the beginning of the training. If weights from a pre-training with SL on ReID data (purple) are used for initialization, then already at the beginning many inner-class angles are by m smaller than the inter-class angles, so the problem becomes simpler and the training easier to handle. Thus, right from the start the loss functions mainly focus on hard positives and hard negatives that violate the margin constraint.

Since this has led to many benefits for AAML and CirL, we further examined whether other loss functions that complicate the optimization target also benefit from such an approach. The results of these sequential training experiments are shown in Fig. 4. As it turns out, RL and CenL benefit from this approach. The optimizer has an easier problem to solve right at the beginning of the training. Thus, the training simplifies, which results in a better generalization. This can be seen in Fig. 4 when comparing the validation and test results with ($\mathrm{RL_{SL}}$, $\mathrm{CenL_{SL}}$) and without sequential training ($\mathrm{RL_{ImageNet}}$, $\mathrm{CenL_{ImageNet}}$). While validation results are similar, test results are better with sequential training.

On the other hand, the THL results worsen with sequential training. Most likely, many class-specific aspects of the training data are learned in the pre-training with SL. Thus, there is no way to learn an equally good generalization as in the less biased initialization with ImageNet weights. This is confirmed by the larger gap between validation and test results for $\mathrm{THL_{SL}}$ in comparison to $\mathrm{THL_{ImageNet}}$.

4 Conclusion

Our analysis is the first step towards a fairer comparison of loss functions for ReID. We compared the most widely used and most promising loss functions on three different setups. We confirmed the superiority of the softmax loss extension additive angular margin loss (AAML) and circle loss (CirL). We also analyzed the effect of sequential training, which is a necessity for AAML and CirL to perform well but also benefits constraint loss functions by improving the generalization ability. Furthermore, we observed that the performance of the loss functions strongly depends on the architecture and the training techniques used, which needs to be taken into consideration for loss function selection. Thus, a decisive ranking of all loss functions is not feasible as it would change depending on

[4] The angles of AAML are shown instead of those of CirL, because here the margin of AAML to be learned can be better visualized.

the training setup. Modern training techniques seem to benefit classification and constraint loss functions but do not improve the performance of (Euclidean-distance-based) metric loss functions, like triplet hard loss (THL), equally strong. However, THL tends to generalize better than classification and constraint loss functions. Therefore, it is understandable why metric loss functions are still in use as a complement to classification loss functions. Furthermore, we examined what costs are to be expected on a public test dataset when typical best practices are used for a real-world application.

As a consequence, for future work, we suggest to benchmark new loss functions on different setups, including a setup following best practices for real-world applications, and to compare against strong reference approaches. Based on our findings, we propose to always try sequential training instead of immediately starting training with ImageNet weights, as this is beneficial in most cases and allows stronger constraints to be enforced by loss functions. We also suggest to finally replace the softmax loss with a more advanced classification loss function.

References

1. Bridle, J.S.: Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In: NIPS (1990)
2. Deng, J., et al.: ArcFace: additive angular margin loss for deep face recognition. In: CVPR (2019)
3. Fan, X., et al.: SphereReID: deep hypersphere manifold embedding for person re-identification. In: JVCIR (2019)
4. Hermans, A., et al.: In defense of the triplet loss for person re-identification. arXiv (2017)
5. Jia, J., et al.: Frustratingly easy person re-identification: generalizing person re-id in practice. In: BMVC (2019)
6. Jie, S., et al.: A new discriminative feature learning for person re-identification using additive angular margin softmax loss. In: UCET (2019)
7. Jin, H., et al.: Deep person re-identification with improved embedding and efficient training. In: IJCB (2017)
8. Liu, W., et al.: Large-margin softmax loss for convolutional neural networks. In: ICML (2016)
9. Liu, W., et al.: SphereFace: deep hypersphere embedding for face recognition. In: CVPR (2017)
10. Luo, H., et al.: Bag of tricks and a strong baseline for deep person re-identification. In: CVPRW (2019)
11. Schroff, F., et al.: FaceNet: a unified embedding for face recognition and clustering. In: CVPR (2015)
12. Sun, Y., et al.: Circle loss: a unified perspective of pair similarity optimization. In: CVPR (2020)
13. Wang, F., et al.: Additive margin softmax for face verification. SPL **25**(7), 926–930 (2018)
14. Wang, G., et al.: Learning discriminative features with multiple granularities for person re-identification. In: ICM (2018)
15. Wang, H., et al.: CosFace: large margin cosine loss for deep face recognition. In: CVPR (2018)

16. Wen, Y., Zhang, K., Li, Z., Qiao, Yu.: A discriminative feature learning approach for deep face recognition. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 499–515. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46478-7_31
17. Xiong, F., et al.: Good practices on building effective CNN baseline model for person re-identification. In: ICGIP (2019)
18. Zheng, L., et al.: Scalable person re-identification: a benchmark. In: ICCV (2015)
19. Zheng, Y., et al.: Ring loss: convex feature normalization for face recognition. In: CVPR (2018)